Algorithms for Reconstruction of Undersampled Atomic Force Microscopy Images

Oxvig, Christian Schou

DOI (link to publication from Publisher):
10.5278/VBN.PHD.ENGSCI.00158

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

ALGORITHMS FOR RECONSTRUCTION OF UNDERSAMPLED ATOMIC FORCE MICROSCOPY IMAGES

BY
CHRISTIAN SCHOU OXVIG

DISSERTATION SUBMITTED 2017

AALBORG UNIVERSITY
DENMARK

Algorithms for Reconstruction of Undersampled Atomic Force Microscopy Images

Ph.D. Dissertation
Christian Schou Oxvig

AALBORG UNIVERSITY
DENMARK

April 24, 2017

Dissertation submitted: April 24, 2017

PhD supervisor: Professor Torben Larsen
Department of Electronic Systems
Aalborg University

Assistant PhD supervisor: Associate Professor Thomas Arildsen
Department of Electronic Systems
Aalborg University

PhD committee: Associate Professor Zheng-Hua Tan (chairman)
Department of Electronic Systems
Aalborg University

Senior Principal Research Scientist Petros T. Boufounos
Mitsubishi Electric Research Laboratories

Professor Lars Kai Hansen
Department of Applied Mathematics and Computer Science
Technical University of Denmark

PhD Series: Technical Faculty of IT and Design, Aalborg University

ISSN (online): 2446-1628
ISBN (online): 978-87-7112-951-9

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

An electronic version of this thesis is available at doi:10.5278/vbn.phd.engsci.00158.

Printed in Denmark by Rosendahls, 2017

The work presented in this thesis has been supported by 1) The Danish Council for Independent Research (DFF/FTP) for project number 1335-00278B/12-134971, and 2) by the Danish e-Infrastructure Cooperation (DeIC) for project number DeIC2013.12.23.

Abstract

An atomic force microscope is capable of measuring the topography of a surface on a nano meter scale by moving a cantilever with a tiny tip at its end across the surface. The flexibility in the choice of the scan pattern, that the cantilever follows, allows for undersampling of the surface, i.e. measuring only a part of the surface. In order to display an image of the entire surface, algorithms for reconstructing the missing parts are required. Such algorithms provide a description of the computations needed to estimate the topography of the entire surface being imaged. If the time it takes to do the computations is small compared to the time saved by only acquiring a part of the surface, a speed-up in the entire imaging process is achievable. An image acquisition speed-up may potentially enable applications such as surface "pre-viewing", video-rate imaging, or super resolution imaging.

In this thesis, we describe and discuss algorithms for reconstructing undersampled atomic force microscopy images or other similar types of high dimensional signals. In particular, our contributions relate to the design of algorithms that not only exploit sparsity but *structured sparsity*. That is, algorithms that exploit the structure in prior information about an image to guide the reconstruction of the image. We show that such structure may be used to improve the estimate of the full topography surface compared to the estimate obtained by algorithms relying only on sparsity.

The main body of this thesis is a collection of individual scientific papers and a tech report which are preceded by an introductory part that summarises the topic and provides perspective on our results and contributions. In the introductory part, we carefully outline our results related to the design of weighted iterative thresholding algorithms and the design of a weighted prior for approximate message passing algorithms. Furthermore, we address the issue of efficient handling of entrywise squared transforms for the high-dimensional image reconstruction setting. Since our proposed algorithms are primarily evaluated empirically through simulations, a major part of this thesis is devoted to the description of our work and results on ensuring correctness and reproducibility of our computational results. Having accounted for this important aspect, we conclude the introductory part of the thesis with a comparison and discussion of the capability of our proposed reconstruction algorithms compared to baseline algorithms.

Resumé

Atomar kraftmikroskopi kan med nanometeropløsning aftaste en overflades topografi ved at føre en mikroskopisk nål henover overfladen. Fleksibiliteten i valget af nålens scannemønster giver mulighed for at underaftaste overfladen, dvs. kun måle en mindre del af overfladen. Det er herefter nødvendigt at bruge algoritmer til at genopbygge de manglede dele af overfladen for at kunne fremvise et billede af den samlede overflade. Sådanne algoritmer beskriver de beregninger, der er nødvendige for at kunne estimere topografien af den samlede overflade. Det er således muligt at fremskynde det samlede optageforløb, hvis tidsforbruget til udførelsen af beregningerne er lille sammenlignet med tiden sparet ved kun at aftaste dele af overfalden. Derved vil anvendelser som "overfladeeksemplificering", videooptagelser eller superopløsningsoptagelser potentielt muliggøres.

I denne afhandling beskrives og diskuteres algoritmer til genopbygning af underaftastede atomar kraftmikroskopibilleder og lignende højdimensionelle signaler. Afhandlingens bidrag relaterer sig især til udformning af algoritmer, hvori ikke alene tyndt besatte egenskaber, men strukturerede tyndt besatte egenskaber udnyttes. Det vil sige algoritmer, hvori strukturen i forudgående information om et billede bruges som rettesnor i billedgenopbygningen. Udnyttelsen af denne struktur viser sig at give anledning til et forbedret estimat af overfladetopografien sammenlignet med et estimat opnået ved brug af algoritmer, der alene er baseret på tyndt besatte egenskaber.

Afhandlingens hoveddel er en samling af individuelle videnskabelige artikler samt en teknisk rapport. Forud for denne hoveddel præsenteres en introduktionsdel, der opsummerer afhandlingens emne og giver perspektiv på dens resultater og bidrag. I afhandlingens introduktionsdel gives en sammenfatning af udformningen af vægtede iterative tærskelalgoritmer samt udformningen af en vægtet prior til tilnærmet meddelelsesudvekslingsalgoritmer. Derudover adresseres problemet med effektiv håndtering af indgangsvist kvadrerede transformationer til højdimensionel billedgenopbygning. Da de foreslåede algoritmer primært evalueres empirisk via simuleringer, er en betydelig del af denne afhandling viet til en beskrivelse af det udførte arbejde med sikring af de beregningsmæssige resultaters korrekthed og reproducerbarhed. Efter fremstillingen af dette vigtige forhold afrundes afhandlingens introduktionsdel med en diskussion af egnetheden af de foreslåede algoritmer sammenlignet med deres udgangspunkt.

Contents

Preface

In the beginning of 2013, I was finishing my master's studies. Despite having already spent five years at the university, I felt a strong desire to further improve my skills in science, engineering, and research. Thus, I was, and still am, much grateful to Professor Torben Larsen for encouraging me to apply for a position as PhD student at the Department of Electronic Systems, Aalborg University. That became the beginning of my endeavours into the fuzzy world of PhD studies. It has been a journey with astounding ups and downs that eventually has come together as the present thesis.

In studying Chapters 1 through 9 of this thesis, the reader should be aware that lists of acronyms, symbols, figures, and tables are available towards the end of Part I. Most of the figures in the thesis convey subtle details that may not be easily identifiable using a printed copy of the thesis. I encourage the reader to use the magnification options available when studying an electronic copy of this thesis[1].

In the sciences, researchers build on each other's ideas and give appropriate credit by citing the works detailing those ideas. Unfortunately, the scientific community has yet to establish appropriate ways to give credit to those who build the tools used in research. My work would not have been possible without the contributions to open science tools done by numerous people. Thus, I would like to express my gratitude to all who have taken part in and continue to contribute to the open scientific software community. The simulation experiments presented in this thesis would not have been possible without the existence of the SciPy stack projects and Project Jupyter. Also my thanks goes to those who have improved the state of open data visualisation. In particular, the work on ColorBrewer [1] and the work on the "coolwarm" colour map [2] have enabled me to clearly visualise my data. A special thanks also goes to Christian Rankl of Keysight Technologies who have kindly provided the AFM images[2] on which my work is based.

I would like to express my sincere gratitude to my supervisors Professor Torben Larsen and Associate Professor Thomas Arildsen for their ever constructive feedback and invaluable support. I want to thank Professor Jan Østergaard and Associate Professor Tobias Lindstrøm Jensen for their contributions to numerous discussions that have helped me improve my work. An equally grateful thanks goes to Patrick Steffen Pedersen who has provided much appreciated feedback on many parts of my work, in particular my scientific software. I would also like to thank Professor Florent Krzakala and Dr. Eric Tramel for welcoming me so warmly to Paris during my stay abroad. First and foremost, though, I would like to express my sincerest gratitude to my friends and my family without whose endless love and support, I would not have made it to this moment where I can write these final words of my thesis.

Christian Schou Oxvig
Aarhus, April 2017

[1] An electronic copy of this thesis is available at doi:10.5278/vbn.phd.engsci.00158

[2] All AFM images displayed in this thesis are derivatives of the "Atomic Force Microscopy Images of Cell Specimens" and "Atomic Force Microscopy Images of Various Specimens" both by Christian Rankl, Keysight Technologies. They are licensed under CC-BY 4.0, available at doi:10.5281/zenodo.17573 and doi:10.5281/zenodo.60434, and provided as-is without warranty of any kind.

List of Manuscripts

This thesis is based on the following manuscripts:

A T. Arildsen, C. S. Oxvig, P. S. Pedersen, J. Østergaard, and T. Larsen, "Reconstruction Algorithms in Undersampled AFM Imaging", *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 1, pp. 31–46, Feb. 2016. doi:10.1109/JSTSP.2015.2500363

B C. S. Oxvig, T. Arildsen, and T. Larsen, "Structure Assisted Compressed Sensing Reconstruction of Undersampled AFM Images", *Ultramicroscopy*, vol. 172, pp. 1–9, Jan. 2017. doi:10.1016/j.ultramic.2016.09.011

C C. S. Oxvig, T Arildsen, and T. Larsen, "Entrywise Squared Transforms for High Dimensional Signal Reconstruction via Generalized Approximate Message Passing", submitted to *IEEE Transactions on Computational Imaging*.

D C. S. Oxvig, P. S. Pedersen, T. Arildsen, J. Østergaard, and T. Larsen, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images", *Journal of Open Research Software*, vol. 2, no. 1, p. e29, Oct. 2014. doi:10.5334/jors.bk

E P. S. Pedersen, C. S. Oxvig, J. Østergaard, and T. Larsen, "Validating Function Arguments in Python Signal Processing Applications," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 106–113. `http://conference.scipy.org/proceedings/scipy2016/patrick_pedersen.html`

F C. S. Oxvig, T. Arildsen, and T. Larsen, "Storing Reproducible Results from Computational Experiments using Scientific Python Packages", in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 45–50. `http://conference.scipy.org/proceedings/scipy2016/christian_oxvig.html`

G C. S. Oxvig, T. Arildsen, and T. Larsen, "Generalized Approximate Message Passing: Relations and Derivations", Department of Electronic Systems, Aalborg University, Tech. Rep., Apr. 2017. doi:10.5278/VBN.GAMPTechReport.

Part I

Background, Overview & Perspectives

1 Introduction & Outline

Atomic force microscopy (AFM) is a type of scanning probe microscopy (SPM) which allows for acquiring images at sub-nanometer scale [3], [4]. It has applications in several areas of imaging, e.g. cell imaging [5] and materials imaging [6]. Additionally, due to its physics, the AFM instrument also has applications in materials manipulation such as lithography [7].

One major concern in using the AFM is the extent of the imaging time. Using traditional AFM imaging methods, the imaging time is on the order of seconds to minutes or even longer [8], [9], [10]. Not only are such wait times annoying to the practitioner, they also prevent or limit the use of AFM in intriguing applications such as video-rate imaging [11] or various studies of biological processes [12], [13] which require a minimum of interaction between the AFM and the sample. Thus, if the AFM imaging process is accelerated, it may potentially enable such new applications in addition to new imaging features such as image previewing [14] and super resolution imaging [15], [16].

In the past decade, significant time and effort has been invested in research in the field of compressed sensing (CS) [17], [18], [19], [20], [21]. The theories of CS provide a framework for reconstructing undersampled signals, i.e. one may for instance recover a complete image from a partial acquisition of it. Recovering the undersampled signal comes at the cost of using computationally demanding reconstruction algorithms. However, given the availability of high performance compute platforms and the maturity of the theoretical foundations of CS, it seems reasonable to ask if such "undersample and reconstruct" methods may be used in AFM imaging? Is it possible to speed up the overall AFM imaging process by significantly reducing the acquisition time through undersampling while only introducing a relatively small time consumption due to the reconstruction process? These are the questions that motivate the work presented in this thesis.

1.1 Main Hypothesis & Aims

When reconstructing an undersampled AFM image, one generally obtains an estimate or approximation of the true underlying image. That is, the reconstruction is not perfect and one must somehow judge the quality of the reconstruction. Independently of the choice of quality indicator, the more measurements that are acquired, the better the reconstruction generally gets. However, the choice of reconstruction algorithm also plays an eminent role in getting the best possible reconstruction. Traditional CS reconstruction algorithms rely on so-called sparse models. Informally, these methods exploit the availability of a concise general description of the signal to reconstruct. However, we ask ourselves if it is possible to obtain more accurate reconstructions by using algorithms tailored specifically for the undersampled AFM image reconstruction problem? Thus, the work presented in this thesis revolves around the following hypothesis:

Main Hypothesis
For a fixed number of measurements, undersampled AFM images may be reconstructed to superior quality using algorithms that exploit statistical information in a structured sparse model of the AFM images compared to algorithms that only rely on a sparse model.

Throughout our work on exploring this main hypothesis, we have identified several research areas of particular interest. Obviously models of AFM images and the actual reconstruction algorithms play an important role. However, in comparing our proposed

reconstruction algorithms to baseline alternatives, the choice of reconstruction quality indicator becomes important. Also, our proposed AFM tailored reconstruction algorithms are of little value if they are not readily available to the AFM practitioner. That is, reusable and high quality implementations of the algorithms must be available and it must be feasible to run those algorithms on standard computing platforms. Finally, since several of our algorithm design and test methods are empirical in nature, i.e. they are based on large simulation studies, correctness and reproducibility of our computational results become crucial. Consequently, our overall aims in our work have been to:

1. Establish models of AFM images that may be used in structure exploiting reconstruction algorithms.
2. Design structure exploiting reconstruction algorithms specifically tailored for the undersampled AFM image reconstruction problem and evaluate their performance relative to the more general baseline reconstruction algorithms.
3. Identify critical elements in comparing the quality of reconstructions of undersampled AFM images.
4. Ensure that the proposed algorithms have sufficiently low computational- and memory requirements to make it feasible to implement them on standard computation platforms.
5. Provide high quality and well tested reference implementations of our proposed algorithms that are readily available to the AFM practitioner.
6. Take actions towards ensuring the correctness and reproducibility of our computational results.

1.2 Organisation & Main Contributions

The present thesis is organised as a collection of individual manuscripts with an extended contiguous introductory part. That is, Part I provides a contiguous narrative on the background, overview, and perspectives on our exploration of the main hypothesis and the aims stated in Section 1.1 whereas Part II is a collection of individual manuscripts and datasets that each give full disclosure on one or more of our main results and contributions. Figure 1.1 illustrates this organisation.

The remainder of Part I is organised as follows: In the background Chapters 2 and 3, we formally introduce the undersampled AFM image reconstruction problem and its mathematical foundations. Our proposed AFM tailored reconstruction algorithms are presented in Chapters 4 and 5. The actions we have taken towards ensuring correctness and reproducibility of our results are detailed in Chapter 6. A discussion of the assessment of the quality of reconstructed AFM images is presented in Chapter 7 along with the results from a large simulation study of the performance of our proposed reconstruction algorithms compared to a range of baseline algorithms. In Chapter 8, we discuss our overall results whereas our conclusions are stated in Chapter 9.

Our main contributions are presented throughout the introductory Part I, Chapters 2 – 6. They are highlighted with a vertical green bar and include references to the works in Part II that give their full disclosure. Specifically, towards the end of Chapter 2, we present our proposed structured model of AFM images from Paper B. Initially, our work focused on using this model with iterative thresholding reconstruction algorithms. This line of work is presented in Chapter 4 based on our results from Paper B. Having established results on the use of our proposed model in iterative thresholding algorithms, we directed our attention to applying our proposed model in the more general and adaptable approximate message passing algorithms. That line of work is presented in Chapter 5 and includes our results on efficient implementations (in terms of computational and memory requirements) of the

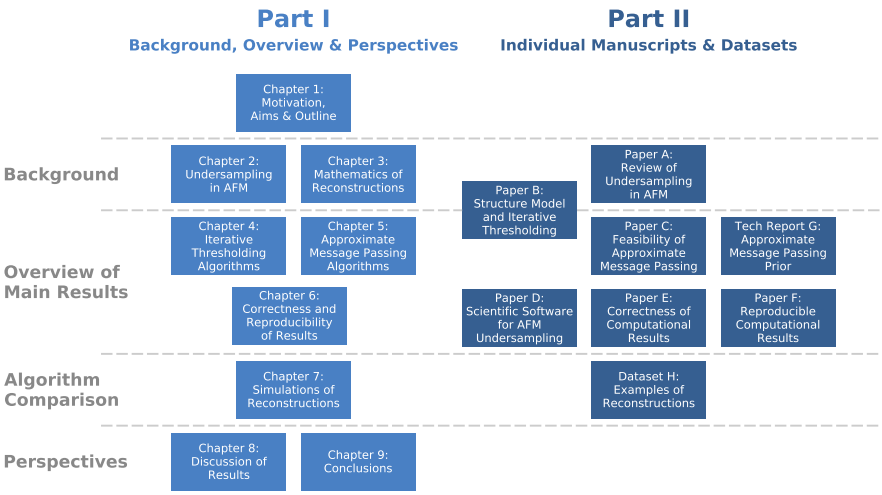


Figure 1.1: Illustration of the organisation of this thesis. Part I provides a contiguous introduction to our line of work whereas Part II features the individual manuscripts and datasets detailing our individual results and contributions. The chapters and individual manuscripts are grouped based on the overall work they relate to, i.e. whether they provide background on the undersampled AFM reconstruction problem, describe our main results, support our algorithm comparison, or provide perspective on the entirety of our work.

involved transforms from Paper C as well as our results on a general weighted prior and an AFM tailored weighted Bernoulli-Laplace prior from Tech Report G. Well tested and fully documented implementations of our proposed reconstruction algorithms are available in the Magni Python package which we introduce in Chapter 6 with more details in Paper D. Throughout the entire course of our work on reconstruction algorithms, we have constantly refined our actions towards ensuring correctness and reproducibility of our computational results. As a consequence of this, we have proposed ways to handle this matter in the typical scientific Python workflow. These results are presented in Chapter 6 based on their description in Paper E and Paper F.

2 Undersampling of Atomic Force Microscopy Images

Motivated by the potential gains in using undersampling in AFM discussed in Chapter 1, we now provide a brief introduction to the basics of the AFM mechanics with an emphasis on the image formation and the undersampling potential in AFM. More details about the AFM mechanics, images formation, and undersampling potential are given in Paper A.

2.1 AFM Mechanics

In an atomic force microscope, a cantilever with a tiny tip at its end is moved across the surface of a sample to be imaged. Due to its interaction with the surface, the cantilever gets deflected. A laser beam is reflected off the cantilever and onto a photo detector in order to record the cantilever deflection. The recorded deflection is then used in a feedback control loop such that the feedback signal resembles the sample surface as the cantilever is moved across it [3], [4]. This principle of operation is illustrated in Figure 2.1.

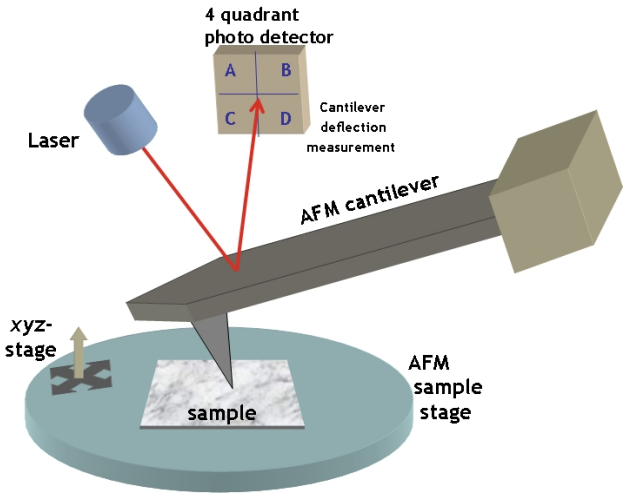


Figure 2.1: "Typical atomic force microscope (AFM) setup: The deflection of a microfabricated cantilever with a sharp tip is measured by reflecting a laser beam off the backside of the cantilever while it is scanning over the surface of the sample." by yashvant. The illustration is part of the The Opensource Handbook of Nanoscience and Nanotechnology and is licensed under CC-BY 2.5.

The specifics of the feedback control loop are tied to the choice of AFM operation mode, e.g. contact mode or tapping mode [3]. Even though all these AFM subtleties are of great importance to the practitioner, in this thesis we simply assume that the AFM is configured to allow for the use of arbitrary continuous sampling patterns when imaging the sample topography. That is, we assume that the cantilever tip may follow any continuous trajectory across the sample surface.

2.2 Image Formation

Traditionally, an image of an entire sample surface is created by moving the cantilever tip across the surface in a raster scan pattern as illustrated in Figure 2.2 [4]. The surface is then overlayed by a uniform pixel grid and a topography level relative to a reference level from the raster scan sampling is assigned to each pixel[1]. When shown on a computer screen, a color map is used to visualise the relative topography levels as shown in e.g. Figure 3.1.

The cantilever is moved relative to the surface using a piezoelectric element. A significant drift in the position of the cantilever may occur in the piezoelectric element [4], [23] which may lead to translation, rotation, and stretch distortions of the resulting image. In this thesis, we assume that such drift is mitigated by the either using a closed loop mode (position feedback control in the AFM) and/or appropriate image post processing. Furthermore, if the sample surface is not perpendicular to the cantilever tip, a tilt in the resulting image occurs. In this thesis, we assume that such a tilt may be accurately modelled by a plane which may be subtracted from the image when it is displayed.

The physics of the piezoelectric element as well as the need for slowly approaching the surface with the cantilever tip makes it difficult to quickly move the cantilever between different locations on the surface [24], [25]. It is, thus, generally a necessity that a continuous sampling pattern is used in the imaging process, though some studies indicate that it may be feasible to use piece-wise continuous sampling patterns which rely on the cantilever tip being raised from the surface, moved to another path segment, and lowered to the surface again [25].

2.3 Undersampling Potential

The flexibility in the choice of sampling pattern allows for spatially undersampling the image in the sense of only covering parts of the image pixel grid. Examples of such undersampling strategies are shown in Figure 2.2. Undersampling may speed-up the AFM imaging process since the distance travelled by the cantilever tip (which is proportional to the imaging time for a fixed cantilever speed) is smaller than the full raster scan distance. This leads to our definition of the AFM undersampling ratio in Definition 1. At first, the factor of two in the definition of δ may seem odd. However, when using the raster scan sampling pattern the entire surface is covered twice even though only one of the resulting sets of pixels (blue or red in Figure 2.2) is usually displayed. Traditionally, this has been the way to handle scan direction specific artefacts and distortions, i.e. by making sure that the entire surface is scanned using a left-to-right scan (or, equivalently, a right-to-left scan). In an undersampling setting, we assume that one may mitigate such artefacts and distortions as part of the reconstruction.

Definition 1 (from [22, (Paper A)], [27, (Paper B)])

The AFM undersampling ratio is

$$\delta = \frac{L}{L_{\text{ref}}} = \frac{L}{2\text{hw}}\,, \tag{2.1}$$

where L is the sampling path length (e.g. the length of one of the green curves in Figure 2.2) and $L_{\text{ref}} = 2\text{hw}$ is the raster sampling baseline path length, i.e. the length of all the blue and red line segments in Figure 2.2 approximated by two times the height times the width of the surface.

[1] More details about such strategies for assignment of topography levels are given in [22, (Paper A)]. Since these details are of less importance to the topic of this thesis, we simply assume that it is done in a reasonable way.

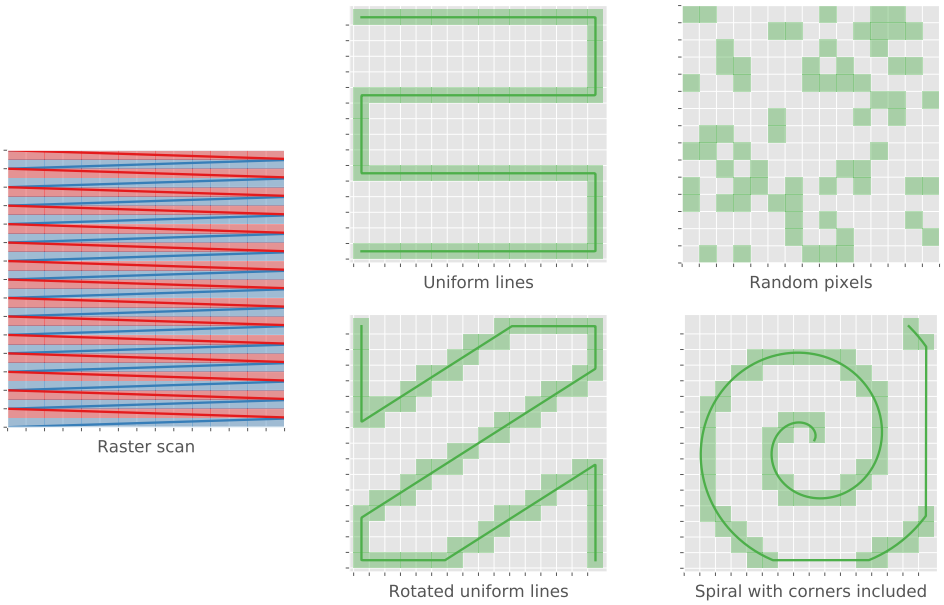


Figure 2.2: Illustration of AFM sampling patterns mapped to a 16-by-16 pixel grid. Solid lines mark the path followed by the cantilever tip. The shaded pixels are the ones covered by the sampling pattern. The raster scan sampling pattern covers all pixels twice: First in the left-to-right direction (blue) and then in the right-to-left direction (red). The uniform lines and uniform rotated lines sampling patterns are simple approaches to sub-sampling the traditional raster scan pattern. The random pixels sampling pattern is not easily implementable on the AFM since it requires defining some continuous path through all the selected pixels. When using the spiral sampling pattern from [26] the "corners" of the rectangular area can only be included by either moving the probe along the edges of the rectangular area (as illustrated) or by following the spiral pattern outside of the rectangular area. The non-raster scan sampling patterns are based on an undersampling ratio of $\delta = 0.15$ (see Definition 1).

The catch in using such undersampling techniques is the need for post processing in order to reconstruct the full image, i.e. fill in the missing pixels (e.g. the grey ones in Figure 2.2). The use of spatial undersampling techniques in combination with reconstruction algorithms is, however, an option for AFM and other SPM modalities. Specifically, the use of spatial undersampling in microscopy imaging applications has already been explored in a number of previous works including in AFM [11], [12], [23], [24], [25], [28], [29], [30] in scanning electron microscopy (SEM) [31], [32], and in electron tomography [33]. We discuss the reconstruction methods used in these works once we have formally introduced the mathematical framework defining the reconstruction of the undersampled AFM images in Chapter 3.

3 Mathematical Background on Reconstruction of Undersampled Images

An undersampled AFM image may be reconstructed in several ways. Our focus is on using methods related to the theory of compressed sensing (CS) [17], [18], [19], [20], [21]. Towards that end, we now introduce the mathematical background on this reconstruction problem and discuss several practicalities in making it tractable for high-dimensional image reconstruction. Finally, we introduce the main idea on which our AFM tailored CS image reconstruction algorithms to be presented in the succeeding chapter are based. More details about the mathematical background on the undersampled AFM image reconstruction problem are given in Paper A and Paper B.

3.1 The Reconstruction Problem

From a mathematical point of view, we consider a rectangular AFM image described by a matrix $\mathbf{X} \in \mathbb{R}^{\mathrm{h}\times\mathrm{w}}$. By stacking the columns of $\mathbf{X}$, we may also represent the image by the vector $\mathbf{x} \in \mathbb{R}^{p\times 1}$ with $p = \mathrm{h}\cdot\mathrm{w}$. The spatial undersampling of AFM images, discussed in Section 2.3, is assumed to be a linear process described by the sampling operator $\mathbf{\Phi} \in \mathbb{R}^{m\times p}$ with $m \leq p$ (usually, $m \ll p$). Applying this sampling operator on $\mathbf{x}$, we have

$$\mathbf{z} = \mathbf{\Phi}\mathbf{x}\,, \tag{3.1}$$

where $\mathbf{z} \in \mathbb{R}^{m\times 1}$ is the vector of (noiseless) measurements of the undersampled AFM image, e.g. the vector with entries corresponding to the green pixels in one of the sub-figures of Figure 2.2. If we consider the measurement process to be corrupted by an additive noise $\mathbf{e} \in \mathbb{R}^{m\times 1}$, e.g. an additive white Gaussian noise (AWGN), we have

$$\mathbf{y} = \mathbf{z} + \mathbf{e} \tag{3.2}$$

$$= \mathbf{\Phi}\mathbf{x} + \mathbf{e}\,, \tag{3.3}$$

where $\mathbf{y} \in \mathbb{R}^{m\times 1}$ is the vector of noisy measurements of the undersampled AFM image. Given $\mathbf{y}$ and knowledge about the measurement matrix $\mathbf{\Phi}$, we are now interested in reconstructing $\mathbf{x}$, i.e. estimating the full AFM image $\mathbf{x}$ by assigning topography values to the grey pixels in the sub-figures of Figure 2.2, based on the noisy measurements $\mathbf{y}$ (the corresponding green pixels in the sub-figures of Figure 2.2). This is in general, in the undersampling setting, an nontrivial problem, since there are more entries in the image vector $\mathbf{x}$ than there are measurements, i.e. $m < p$. Compressed sensing does, however, offer the theoretical foundations for solving this undersampling problem provided that the information in the image is captured by some succinct representation, e.g. if $\mathbf{x}$ is *sparse* (i.e. most of its entries are zero) in some dictionary matrix [19]. Informally, the idea is to use some additional a priori information to make up for the lack of measurements. Thus, if for some dictionary matrix, $\mathbf{\Psi} \in \mathbb{C}^{p\times n}$ with $p \leq n$, we have

$$\mathbf{x} = \mathbf{\Psi}\boldsymbol{\alpha}\,, \tag{3.4}$$

for some *sparse or otherwise structured* $\boldsymbol{\alpha} \in \mathbb{C}^{n\times 1}$ then under certain assumptions it is possible to recover $\mathbf{x}$ from $\mathbf{y}$ [19], [20]. By defining the system matrix $\mathbf{A} = \mathbf{\Phi}\mathbf{\Psi} \in \mathbb{C}^{m\times n}$, we may state the set of equations defining our reconstruction problem as

$$\mathbf{y} = \mathbf{A}\boldsymbol{\alpha} + \mathbf{e} \tag{3.5}$$

$$= \mathbf{\Phi}\mathbf{\Psi}\boldsymbol{\alpha} + \mathbf{e} \tag{3.6}$$

$$\mathbf{x} = \mathbf{\Psi}\boldsymbol{\alpha}\,. \tag{3.7}$$

As discussed in Section 3.4, there exists numerous methods to obtain an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ from $\mathbf{y}$. One popular CS related method for solving this reconstruction problem is by use of the least absolute shrinkage and selection operator (LASSO) [34], [35], [36], i.e. solving the convex optimisation problem

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2 + \beta||\boldsymbol{\alpha}||_1 \tag{3.8}$$

or the equivalent formulation (for a given β there exists an ϵ such that the two problems have the same solution [37])

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} ||\boldsymbol{\alpha}||_1 \quad \text{s.t.} \quad ||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2 \leq \epsilon\,. \tag{3.9}$$

The CS theory gives several guarantees in terms of how well $\hat{\boldsymbol{\alpha}}$ approximates $\boldsymbol{\alpha}$. These guarantees are generally based the application of incoherence [19], [38], restricted isometry property (RIP) [38], [39], or state evolution [36], [40]. Though all of these are interesting theoretical elements, in this thesis we focus on empirical evaluation of the reconstruction algorithm performance. Our reasons for this choice are twofold

- The performance predicted by such theoretical bounds is oftentimes worse than what may be observed empirically [41], [42]. Thus, it may be possible to reconstruct undersampled AFM images in settings which are not covered by the theoretical guarantees.
- It is difficult, if not impossible, to verify if the assumptions (e.g. certain distributions on the entries in the system matrix and coefficient vector, bounds on the sparsity, or certain types of measurement noise [19], [21], [43]) behind the theoretical bounds are fulfilled in practical applications [44] such as the AFM image reconstruction problem. In particular, the AFM sampling process does not easily permit the application of the theoretical guarantees as discussed in Section 3.2.

3.1.1 The Dimensionality Problem

The CS related methods for solving the undersampled AFM image reconstruction problem, e.g. split-Bregman [45] or Douglas-Rachford splitting for the LASSO optimisation problem in (3.8), the iterative thresholding methods [41] discussed in Chapter 4, or the approximate message passing [36] algorithms discussed in Chapter 5 generally are first order methods that rely on the iterative applications of matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$ (the complex conjugate transpose of $\mathbf{A}$). Two problems arise from the application of such matrix-vector products

- They require on the order of $\mathcal{O}(mn)$ floating point operations which makes them infeasible (or at least rather slow) for large problem sizes (i.e. large p, n) and/or high undersampling ratios (i.e. $\delta \approx 1$) [42] (see also the profilings reported on in Paper C with the details available at doi:10.5278/240710282).
- The memory requirement for storing $\mathbf{A}$ scales poorly with the problem size. Consider for instance a situation in which a 256-by-256 pixels AFM image is spatially undersampled with $\delta = 0.5$ ($\mathbf{A} \in \mathbb{R}^{0.5\cdot 256^2 \times 256^2}$). In this case, it would require $8 \cdot 0.5 \cdot (256^2)^2/1024^3 = 16\,\text{GiB}$ RAM to store $\mathbf{A}$ in double precision.

The poor scaling in terms of both floating point operations and memory requirements in implementing these matrix-vector products makes it a necessity to replace such direct matrix-vector products with implicit or otherwise fast and memory efficient implementations. Options for practically feasible implementations of $\mathbf{A}$ include the use of *fast transforms* (e.g. a fast Fourier transform (FFT) for Fourier based transforms implementable using $\mathcal{O}(n\log_2(n))$ operations [46]) or possibly the use of a 2D separable transform, when

$\mathbf{A}$ involves matrices defined by Kronecker products [47] (see also the presentation in Paper C). An overview of the floating point operations and memory scaling of these methods for the AFM spatial undersampling problem is given in Table 3.1.

	Matrix-vector product	Separable transform	Fast transform
Floating point operations	$\mathcal{O}(mn)$	$\mathcal{O}(m+\sqrt{n}n)$	$\mathcal{O}(m+n\log_2(n))$
Memory requirement	$\mathcal{O}(mn)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

Table 3.1: Scaling of floating point operations and memory requirements with reconstruction problem size for the AFM spatial undersampling problem when the reconstruction method is based on the application of $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$. The $\mathcal{O}$ denotes Big-O notation (see e.g. [48]). In the computation of the floating point operations for both the separable and fast transform, we have assumed that the sampling operator may be implemented using m operations (see Section 3.2) such that only the dictionary is implemented using the specified method. For the fast transform implementation, we assume an in-memory operation. The details about the scaling of the separable transform may be found in Paper C.

3.2 The Sampling Operator

As detailed in Section 2.2, the AFM image formation amounts to assigning relative topography levels to a pixel grid. When using spatial undersampling as detailed in Section 2.3, topography values are only assigned to a subset of the pixels on the grid. Such a sampling operation may be modelled by a matrix that extracts only that subset of the pixels from the image vector $\mathbf{x}$. That is, $\mathbf{\Phi} = \mathbf{I}_\Omega$ is a row sub-sampled identity matrix $\mathbf{I}$ with Ω being the set of indices of rows corresponding to the pixels to extract. Thus, we have $\mathbf{I}_\Omega \in \mathbb{R}^{m\times p}$ with $|\Omega| = m$. The vector of extracted pixels is then $\mathbf{z} = \mathbf{\Phi}\mathbf{x} = \mathbf{I}_\Omega\mathbf{x}$. Applying $\mathbf{\Phi}^{\mathsf{T}}$ to $\mathbf{z}$ results in a vector similar to $\mathbf{x}$ but with the pixels corresponding to the non-indexed entries set to zero.

Matrix-vector products involving such $\mathbf{\Phi}$ and $\mathbf{\Phi}^{\mathsf{T}}$ reduce to the problem of "looking up" the entries in the vector corresponding to the set Ω which allows for an efficient implementation. One only needs to keep track of the m indices and may thus implement the sampling operation by implementing the m extractions from, or insertions into, the vector.

An important note to make at this point is that the AFM physics lead to a sparse and structured (due to the requirement of using a continuous sampling pattern) sampling operator $\mathbf{\Phi}$. Contrary to this sparse and structured sampling operator is the dense random sampling operators (e.g. a measurement matrix with independent and identically distributed (i.i.d.) zero mean Gaussian entries or randomly sub-sampled Fourier matrices) used to derive many of the theoretical results in the CS theory [19], [38], [49], thus making it difficult to use such theoretical results in relation to the AFM image reconstruction problem. The frameworks of structured random matrices [43] and structurally random matrices (SRM) [50] provide some theoretical guarantees for sub-sampled structured transforms, e.g. when using the AFM sampling operator in combination with a structured dictionary such as a Fourier dictionary. However, these results still depend on the sub-sampling being random. That is, for the theoretical results to be applicable to the AFM undersampling problem, one must use random sampling as illustrated in Figure 2.2 which is, unfortunately, not easily implementable on the AFM.

3.3 The Dictionary

Traditional choices of dictionaries for CS image reconstruction problems include orthonormal bases (with $p = n$) such as the cosine transform or some of the wavelet transforms

[51]. However, it is also possible to use learning methods to adapt the dictionary to the application at hand [52], [53], [54]. Some dictionary learning approaches as well as oversampling of the cosine transform or using wavelet frames lead to overcomplete and redundant dictionaries ($p < n$) which may also be used in CS methods [54], [55].

The dictionary learning approaches, though intriguing in terms of the potentially achievable reconstruction performance, are of limited interest with large problem sizes due to their computational complexity. The learned dictionaries generally do not have any structure that allow for efficient implementations [54]. Thus, one must resort to using direct matrix-vector multiplication in implementing transforms based on learned dictionaries. Finally, the use of learning methods entails the need for training data. AFM specific training data is not easily available due to the time usage and other costs associated with the data collection. All of this makes dictionary learning approaches less appealing in connection to the undersampled AFM image reconstruction problem.

This leaves the discrete wavelet [56] and cosine transforms [57] as more obvious choices. They generally have *fast transforms* allowing for implicit implementations using $\mathcal{O}(n \log_2(n))$ floating point operations [52], [54]. The wavelets used in the JPEG2000 image compression standard [58] generally provide better compression of natural images compared to the cosine transform used in the JPEG image compression standard [59] and, thus, yields a more (approximately) sparse $\boldsymbol{\alpha}$. Furthermore, wavelet dictionaries have been successfully applied in some CS imaging applications when combined with dense random sampling [60], [61]. However, a discrete wavelet transform (DWT) matrix is generally sparse whereas the discrete cosine transform (DCT) matrix is dense. The matrix product of a sparse sampling operator $\boldsymbol{\Phi}$ (as the AFM sampling operator detailed in Section 3.2) and a sparse dictionary $\boldsymbol{\Psi}$ generally yields a sparse system matrix $\mathbf{A}$ which may be problematic in an undersampling setting. Intuitively, if $\mathbf{A}$ is too sparse, the sparse elements of $\boldsymbol{\Psi}$ are unable to interpolate between the sparse sampling points extracted by $\boldsymbol{\Phi}$. This need for a dense dictionary when used with sparse sampling is captured in the incoherent sampling requirement in the CS theory [19] as further detailed in [22, (Paper A)]. Thus, one must balance the ability of the dictionary to provide a sparse representation of the signal of interest against its incoherence with the sampling matrix. A large set of empirical evaluations of the AFM image reconstruction performance when using both DWT and DCT dictionaries are presented in [22, (Paper A)]. The results are highly dependent on the choice of sampling pattern. However, the best possible results are obtained using the dense DCT. Finally, since AFM images tend to have repetitive patterns as well as larger smooth regions, it is reasonable to expect that the DCT compresses such images well.

The above discussion suggests that the DCT dictionary provides a reasonable balance between reconstruction performance and ease of implementation in terms of both computational and memory requirements. Thus, in this thesis, we focus on the use of the orthonormal DCT as the dictionary of choice in the undersampled AFM image reconstruction problem. At this point we note that the simulation results in [22, (Paper A)] show that an overcomplete DCT may yield a small gain in reconstruction performance compared to the orthonormal DCT. However, this comes at increased computational complexity.

The 2D separable orthogonal DCT transform may be implemented using two 1D DCTs (one on the columns and one on the rows) [47] which may in turn be implemented using an FFT based method [62]. Thus, the number of floating point operations in implementing the 2D transform reduces to $2\sqrt{n}\mathcal{O}(\sqrt{n}\log_2(\sqrt{n}) = \mathcal{O}(n\log_2(\sqrt{n}))$[1] which is slightly less than for the more general fast transform result given in Table 3.1. If an even faster transform is needed, specialised 2D DCT algorithms exist [63].

[1]Strictly speaking, the result is $\mathcal{O}(n\log(n))$ since in Big-O notation, one only considers the scaling behaviour up to a constant. However, in a practical application of our results, such constants have a significant impact which is why we have included these extra details in the results.

3.3.1 DCT Coefficient Structure in AFM Images

The CS theory allows for not only assuming *sparsity* but *structured sparsity* on $\boldsymbol{\alpha}$ which may be exploited to further improve the capabilities of reconstruction algorithms [64], [65].

A few examples[2] of the placement of the 10 % largest DCT coefficients of typical AFM images are shown in Figure 3.1. With the exception of the last image, all of these examples share a common low-frequency structure with a dispersion of the coefficients from the low-frequency area towards the high-frequency area. The last image is a calibration grid (and thus not a natural AFM image) which has a periodic structure. If the image matrix $\mathbf{X}$ is transposed, its DCT domain is also transposed which suggests a general symmetry in the DCT coefficients around the diagonal. These observations along with similar observations based on other natural AFM images is the motivation behind our proposed model of the DCT coefficient structure in AFM images detailed in Main Contribution 1. As is seen from the examples in Figure 3.1, if, using some reconstruction algorithm, this structure model in combination with a sparsity assumption makes it possible to identify the 10 % largest (or more) DCT coefficients and their values, accurate reconstructions of AFM images are obtainable.

Main Contribution 1 (AFM DCT Support Structure from [27, (Paper B)])

We consider a model of the DCT coefficients that is characterised by:

1. *A smooth transition from a peak value in the low-frequency area to a small constant value in the high-frequency area. Examining the full DCT domain suggests an exponentially decaying transition.*
2. *A sufficient number of degrees of freedom to model the dispersion in the coefficients from low-frequencies to high-frequencies while still imposing symmetry around the diagonal (in the model of the coefficients).*

Specifically, we propose to model the dispersion in the DCT coefficient values using a Gaussian function f of the form:

$$f(\mathbf{z}) = a \cdot \exp\left(-(\mathbf{z}-\mathbf{b})^{\mathsf{T}}\mathbf{C}^{-1}(\mathbf{z}-\mathbf{b})\right) \tag{3.10}$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b \\ b \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix}$$

$$\mathbf{C}^{-1} = \mathbf{R}\begin{bmatrix} 1/c_1 & 0 \\ 0 & 1/c_2 \end{bmatrix}\mathbf{R}^{\mathsf{T}},$$

where:

- z_1, z_2 *are the two pixel coordinates.*
- a *is a user specified parameter that determines the flatness of the model relative to the data.*
- b *is the "mean" value positioned on the* $z_1 = z_2$ *diagonal.*
- $\mathbf{C}$ *is the "covariance" matrix rotated to ensure symmetry around the diagonal.*

We use the quadratic form $(\mathbf{z}-\mathbf{b})^{\mathsf{T}}\mathbf{C}^{-1}(\mathbf{z}-\mathbf{b})$ *to model the shape of the spread. Choosing both entries in* $\mathbf{b}$ *to be equal forces the peak value to be on the diagonal. Combined with*

[2] More examples for all the 17 AFM images shown in Figure 7.1 are included in the "Extra Figures" supplementary material available at doi:10.5278/252861471.

the fixed rotation matrix $\mathbf{R}$ *that aligns the principal axes with the diagonals, this forces the desired symmetry.*

More details are given in [27, (Paper B)].

3.4 Reconstruction Algorithms in General

One general approach to obtaining an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ from the undersampled noisy measurements $\mathbf{y}$ is by directly estimating values of the missing pixels [66]. Interpolation techniques such as nearest neighbour, linear, or cubic interpolation [12] [67], [68] based on e.g. Delaunay triangulation provides one way to apply this strategy. Another way is to minimise the total variation (TV) of the reconstructed image subject a least squares (LS) constraint [69], [70], [71], e.g. solving the optimisation problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathrm{tv}(\mathbf{x}) \quad \text{s.t.} \quad ||\mathbf{y} - \mathbf{\Phi}\mathbf{x}||_2^2 \leq \epsilon\,. \tag{3.11}$$

More details about these approaches are given in [22, (Paper A)]. Examples of studies on undersampled microscopy image reconstruction based on this approach include [23], [24], [12], [29] (interpolation based), as well as [11], [28], [31], [32], [33] (TV based).

A different approach to obtaining the estimate $\hat{\mathbf{x}}$ is to use the CS theory presented in Section 3.1 - 3.3. A vast selection of CS reconstruction algorithms exist. Popular choices include ℓ_1-norm minimisation approaches such as the LASSO in (3.8) [34] (also known as basis pursuit denoising (BPDN) [72]), greedy algorithms [41], and approximate message passing (AMP) based methods [36], [73]. Examples of studies on undersampled microscopy image reconstruction based on the CS approach include [25], [28], [29], [32], [33] (ℓ_1-norm minimisation based) as well as [30] (greedy matching pursuit based).

The main difference between the two approaches (direct estimation and CS methods) is the introduction of the dictionary in the CS based methods. In the interpolation and TV[3] based methods one attempts to use structure in the image domain to guide the reconstruction of the AFM image, e.g. the presence of larger smooth areas in typical AFM images which motivates the use of interpolation as well as TV based methods. In the CS based methods one instead attempts to use structure in the dictionary domain, e.g. in the DCT domain structure discussed in Section 3.3.1. In our proposed AFM image reconstruction algorithms presented in Chapters 4 and 5, we attempt to improve the reconstruction performance by exploiting the model of the DCT domain structure presented in Main Contribution 1.

[3] As detailed in [22, (Paper A)], one may interpret TV methods in a CS context by expressing the TV operator as a dictionary. In the AFM undersampling problem, we like to make a distinction between methods that directly attempts to "fill-in the blanks" in the image domain and methods that address the reconstruction problem in another domain, e.g. the DCT domain. We reserve the CS context for the latter type of methods.

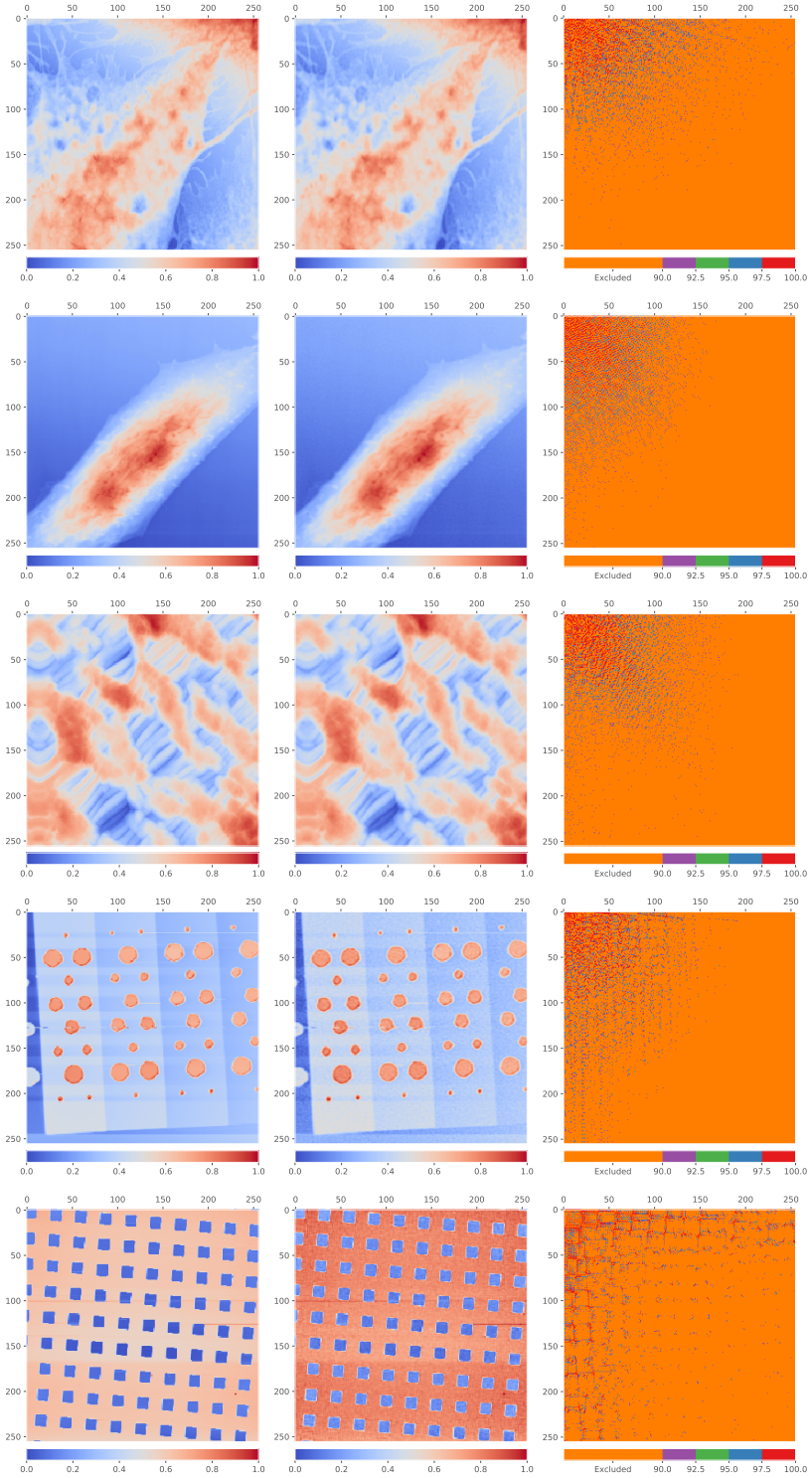


Figure 3.1: Examples of structure in the DCT coefficients of typical AFM images (a subset of the images shown in Figure 7.1). Each row displays for an image: The ground truth image (left), an approximation based on the 10 % largest DCT coefficients (mid), and the structure of the 10 % largest DCT coefficients (right). The right column figures show the placement of the 10 % largest coefficients (divided into four intervals) in the DCT domain. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

4 Iterative Thresholding Reconstruction Algorithms

With an outset in the mathematical background provided in Chapter 3, we now present our first proposed type of reconstruction algorithms which exploit structured sparsity to solve the undersampled AFM image reconstruction problem. These are iterative thresholding algorithms which allow us to use the structure model of AFM image DCT coefficients presented in Main Contribution 1. We introduce and motivate the basic iterative thresholding algorithms before presenting our proposed modifications and discussing various implementation details. More details about the use of iterative thresholding algorithms for the undersampled AFM image reconstruction problem may be found in Paper B.

4.1 The Greedy Idea and Iterative Thresholding Algorithms

Iterative thresholding reconstruction algorithms belong to a larger class of so-called *greedy* reconstruction algorithms [41]. The greedy algorithms are based on the idea of choosing locally optimal solutions in each iteration, hence the term *greedy*. In the CS setting, one is typically trying to reconstruct a vector $\boldsymbol{\alpha}$ which is assumed to be sparse. That is, the $\ell_0 = |\text{supp}(\boldsymbol{\alpha})|$ pseudo-norm which counts the size of the support of $\boldsymbol{\alpha}$ (the number of non-zero entries) is assumed small. Thus, one may be looking to solve the optimisation problem

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\text{argmin}} \, ||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2 \quad \text{subject to} \quad ||\boldsymbol{\alpha}||_0 \leq k \tag{4.1}$$

in order to find a k-sparse solution with the minimum mean squared error. Solving this problem directly is unfortunately NP-hard [20]. However, the iterative hard thresholding (IHT) algorithm is capable of finding a *locally optimal* solution to an approximation to this optimisation problem [74], [75]. This represents the general greedy idea: Obtain a computationally tractable reconstruction algorithm by accepting only finding a locally optimal solution.

Specifically, the iterative thresholding algorithms are characterised by having a single step in each iteration that forces an assumed sparsity constraint, yet still allows for the support set to change between iterations [41]. The general iterative thresholding scheme is stated in Algorithm 1. In this scheme, η_t is a scalar non-linear threshold operator that acts independently on each of the entries in its vector argument and, in general, may be iteration dependent. Furthermore, κ is a step-size (relaxation) parameter. The procedure in Algorithm 1 may be interpreted as a gradient descent step in solving the unconstrained optimisation problem minimise $||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2$ followed by a thresholding operation in order to force sparsity in the solution [75], e.g. simply setting the k smallest (in absolute value) entries to zero as is done in IHT [74].

The step-size parameter κ may be fixed to some sufficiently small value to ensure convergence [41]. It may be chosen adaptively to optimally solve the unconstrained optimisation problem minimise $||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2$ provided that the correct support has been found [75]. Or it is may be fixed to the worst case empirically tuned value from [44] which is a more practically applicable method.

Algorithm 1 Iterative Thresholding (based on [41], [76])

initialise: $\hat{\boldsymbol{\alpha}}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{y}$
for $t = 1 \dots T_{\max}$ **do**
  $\mathbf{c}_t = \mathbf{A}^{\mathsf{T}}\mathbf{r}_{t\text{-}1}$
  $\hat{\boldsymbol{\alpha}}_t = \eta_t(\hat{\boldsymbol{\alpha}}_{t\text{-}1} + \kappa \cdot \mathbf{c}_t)$
  $\mathbf{r}_t = \mathbf{y} - \mathbf{A}\hat{\boldsymbol{\alpha}}_t$
  if stop criterion is met **then**
    break
  end if
end for

4.2 Weighted Threshold Operators

The threshold operator η_t in Algorithm 1 may be chosen to reflect an assumed structure on the signal being reconstructed. For sparse vectors, one may use the hard or soft threshold operators

$$\text{Hard [74]: } \eta_t^{\mathrm{H}}(x) = x \cdot \mathbb{1}_{|x|>t} \tag{4.2}$$

$$\text{Soft [77]: } \eta_t^{\mathrm{S}}(x) = \operatorname{sgn}(x)(|x| - t)_+ \,, \tag{4.3}$$

where $\mathbb{1}_{|x|>t}$ is the indicator of $|x| > t$ for some threshold level t and $(|x| - t)_+ = (|x| - t) \cdot \mathbb{1}_{(|x|-t)>0}$. That is, the hard threshold operator forces all entries smaller in absolute value than t to zero, whereas the soft threshold operators forces all entries smaller than t to zero and shrinks the remaining entries towards zero by t. When using the hard threshold operator in Algorithm 1, the resulting algorithm is known as IHT [74]. Similarly, when using the soft threshold operator, the resulting algorithm is known as iterative soft thresholding (IST) [78] or the iterative shrinkage-thresholding algorithm (ISTA) [79]. IHT forces sparsity in the solution by greedily attempting to solve the optimisation problem in (4.1) for a fixed sparsity k. Similarly, it can be shown that for a fixed threshold level $t = \beta$, IST attempts to solve the LASSO optimisation problem in (3.8) [80].

When a structured sparsity is assumed on the solution, the threshold operator may be chosen to reflect that assumption, e.g. if the solution is assumed to lie on one of several subspaces of a Hilbert space the threshold operator may be chosen to reflect that [81]. Though not strictly based on the union of subspaces models for structured sparsity [41], [64], [65], we have used this idea of designing a threshold operator for the assumed structure on the solution in our proposed weighted threshold operators presented in Main Contribution 2.

Main Contribution 2 (Weighted Threshold Operators from [27, (Paper B)])

Based on the idea of adapting the thresholding operators in model based CS [65], we propose the following two weighted thresholding operators:

$$\textit{Weighted hard: } \eta_t^{\mathrm{wH}}(x) = x\mathbb{1}_{|wx|>t} \tag{4.4}$$

$$\textit{Weighted soft: } \eta_t^{\mathrm{wS}}(x) = \frac{1}{w}\operatorname{sgn}(x)(|wx| - t)_+ \tag{4.5}$$

where $w \in \mathbb{R}_+$ is a weight applied to the coefficient before thresholding.

More details are given in [27, (Paper B)].

4.3 Iterative Thresholding for the AFM Image Reconstruction Problem

The weighted threshold operators presented in Main Contribution 2 may be used with any dictionary to promote an arbitrary structure described by a vector of weights $\mathbf{w} \in \mathbb{R}_+^{n\times 1}$. In particular, by obtaining a matrix of DCT domain weights from the application of the DCT coefficient model in Main Contribution 1 and stacking its columns similarly to way $\mathbf{x}$ is obtained from $\mathbf{X}$, a vector of weights for the AFM undersampling problem is obtained. This use of the DCT coefficient structure model in Main Contribution 1 with the weighted threshold operators presented in Main Contribution 2 defines our first proposed type of reconstruction algorithms tailored for the undersampled AFM reconstruction problem. A comparison of this method to other reconstruction methods is given in Chapter 7.

When using the weighted hard threshold operator in (4.4) in Algorithm 1, we denote the resulting algorithm weighted iterative hard thresholding (w-IHT). Similarly, when using the weighted soft threshold operator in (4.5), we denote the resulting algorithm weighted iterative soft thresholding (w-IST). Whereas the IHT and IST algorithms exploit sparsity in $\boldsymbol{\alpha}$, the w-IHT and w-IST algorithms exploit structured sparsity in $\boldsymbol{\alpha}$. That is, using $\mathbf{w}$, one may mark some coefficients as being more likely to be included in the solution than others while still forcing an overall sparse solution. Note that in the weighted threshold operators in (4.4) and (4.5), only the choice of coefficients to threshold is influenced by the weights - the values of the coefficients are not. Also note that when using w-IST, one must ensure $w_j \neq 0, \forall j$ to avoid a division by zero.

4.3.1 Threshold Level

When using iterative thresholding (Algorithm 1) with any of the threshold operators in (4.2), (4.3), (4.4), (4.5), one must choose a threshold level t. The threshold level may be fixed at e.g. $t = \beta$ (as in the LASSO optimisation problem) if the application suggests such a fixed value. Otherwise, it may be chosen adaptively, e.g. by using the false alarm rate heuristic from [44].

In our evaluation of the iterative threshold algorithms for the undersampled AFM image reconstruction problem presented in Chapter 7, we adaptively, in each iteration, choose t such that the sparsity level $\rho = \frac{k}{m}$ is fixed. That is, $t = |\boldsymbol{\alpha}|_{(k)}$ where $|\boldsymbol{\alpha}|_{(k)}$ is the k'th largest (in absolute value) element in $\boldsymbol{\alpha}$. In line with our empirical approach to evaluating the performance of reconstruction algorithms for the undersampled AFM image reconstruction problem, we then evaluate the performance of the algorithm by testing several values of ρ for a given δ.

4.3.2 Stop Criteria

A stop criterion may be used to terminate the iterations in Algorithm 1 when the estimate $\hat{\boldsymbol{\alpha}}$ is sufficiently close to the true solution. A few such stop criteria are detailed in the generalized approximate message passing (GAMP) tech report (Tech Report G). We have empirically found that the residual measurements ratio stop criterion proposed in [44] works well when attempting to solve the AFM undersampled image reconstruction problem. That is, the iterations in Algorithm 1 should be stopped when

$$||\mathbf{r}_t||_2 < \epsilon \cdot ||\mathbf{y}||_2 \,, \tag{4.6}$$

for some tolerance $\epsilon \in \mathbb{R}_+$.

5 Generalized Approximate Message Passing Reconstruction Algorithms

Our second proposed type of reconstruction algorithms that exploit structured sparsity to solve the AFM undersampling problem are based on the approximate message passing (AMP) algorithm [36], [40], [73], [82]. These are iterative algorithms with a Bayesian probabilistic interpretation of the undersampling problem, they attempt to solve. This interpretation allows us to define and use a probabilistic model of AFM images with the AMP algorithms. We introduce and motivate the AMP and generalized approximate message passing (GAMP) algorithms before presenting our proposed modifications and discussing various implementation details. More details may be found in Paper C and Tech Report G.

5.1 The AMP and GAMP Algorithms

From a practical perspective, the approximate message passing (AMP) algorithm [36], [40], [73], [82] and the generalized approximate message passing (GAMP) algorithm [83], [84] are of interest since they may be used to find Bayes optimal solutions to undersampling problems involving application specific probabilistic models. Specifically, based on an application specfic prior (input channel) $p(\boldsymbol{\alpha};\boldsymbol{\theta}_I)$ parameterised by $\boldsymbol{\theta}_I$ and an application specific measurement model (output channel) $p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_o)$ parameterised by $\boldsymbol{\theta}_o$, the GAMP algorithm may be used to find minimum mean squared error (MMSE) and maximum a posteriori (MAP) estimates of $\boldsymbol{\alpha}$. An AWGN output channel is assumed in the AMP algorithm whereas the GAMP algorithm allows for other and more general measurement models to be used. From a theoretical perspective, the AMP algorithms are of interest in solving undersampling problems since the state evolution (SE) formalism [36], [40], [85] may be used to show that these algorithms are optimal (in the large system limit $n \to \infty$ and under certain conditions) in the sense of reaching the theoretical reconstruction boundaries described by the precise undersampling theorems [49]. The basics of AMP/GAMP as well as several implementation details are summarised in Paper C and further elaborated on in Tech Report G.

The AMP algorithm is stated in Algorithm 2 whereas the GAMP algorithm for finding MMSE estimates is stated in Algorithm 3. In the AMP algorithm, η_t is a threshold function (with a first derivative η_t') and $\langle\cdot\rangle$ denotes averaging. In the GAMP algorithm, $|\mathbf{A}|^{\circ 2}$ is the entrywise absolute value square of $\mathbf{A}$. The output channel functions $f_{\bar{z}}$, $f_{\tilde{z}}$ and the input channel functions $f_{\bar{\alpha}}$, $f_{\tilde{\alpha}}$ are scalar conditional expectations that operate independently on each entry of their vector arguments according to

$$f_{\bar{z}}(v,o;y,\boldsymbol{\theta}_o) = \frac{1}{\mathcal{Z}_o}\int_z z p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)dz \tag{5.1}$$

$$f_{\tilde{z}}(v,o;y,\boldsymbol{\theta}_o) = \frac{1}{\mathcal{Z}_o}\int_z |z|^2 p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)dz - |f_{\bar{z}}(v,o;y,\boldsymbol{\theta}_o)|^2 \tag{5.2}$$

$$f_{\bar{\alpha}}(s,r;\boldsymbol{\theta}_I) = \frac{1}{\mathcal{Z}_I}\int_\alpha \alpha p(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)d\alpha \tag{5.3}$$

$$f_{\tilde{\alpha}}(s,r;\boldsymbol{\theta}_I) = \frac{1}{\mathcal{Z}_I}\int_\alpha |\alpha|^2 p(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)d\alpha - |f_{\bar{\alpha}}(s,r;\boldsymbol{\theta}_I)|^2\,, \tag{5.4}$$

for

$$\mathcal{Z}_I = \int_\alpha p(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)d\alpha \tag{5.5}$$

$$\mathcal{Z}_o = \int_z p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)dz \tag{5.6}$$

$$\mathcal{N}(\alpha;r,s) = \frac{1}{\sqrt{2\pi s}}\exp\left(-\frac{1}{2}\frac{(\alpha-r)^2}{s}\right) \tag{5.7}$$

$$\mathcal{N}(z;o,v) = \frac{1}{\sqrt{2\pi v}}\exp\left(-\frac{1}{2}\frac{(z-o)^2}{v}\right), \tag{5.8}$$

where $\mathcal{N}(x;\bar{\theta},\tilde{\theta})$ denotes a Gaussian distribution on x with mean $\bar{\theta}$ and variance $\tilde{\theta}$. Furthermore, the GAMP initialisations are the conditional expectations and variances

$$\mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}] = \int_{\boldsymbol{\alpha}} \boldsymbol{\alpha} p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I)d\boldsymbol{\alpha} \tag{5.9}$$

$$\mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha}) = \int_{\boldsymbol{\alpha}} |\boldsymbol{\alpha}|^2 p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I)d\boldsymbol{\alpha} - |\mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]|^2. \tag{5.10}$$

Algorithm 2 AMP [36], [40].

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}_n$, $\boldsymbol{\chi}_0 = \mathbf{0}_m$
for $t = 1$ **to** $T_{\max}$ **do**
  $\bar{\boldsymbol{\alpha}}_t = \eta_t(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^{\mathsf{H}}\boldsymbol{\chi}_{t\text{-}1})$
  $\boldsymbol{\chi}_t = \mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_t + \frac{n}{m}\langle\eta_t'(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^{\mathsf{H}}\boldsymbol{\chi}_{t\text{-}1})\rangle\boldsymbol{\chi}_{t\text{-}1}$
  if stop criterion is met **then**
    break
  end if
end for

Algorithm 3 MMSE GAMP [83]. Note that we use $\circ$ to denote entrywise multiplication of vectors and $\oslash$ to denote entrywise division of vectors.

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\tilde{\boldsymbol{\alpha}}_0 = \mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})$, $\mathbf{q}_0 = \mathbf{0}_m$
for $t = 1$ **to** $T_{\max}$ **do**
  $\mathbf{v}_t = |\mathbf{A}|^{\circ 2}\tilde{\boldsymbol{\alpha}}_{t\text{-}1}$ # Factor side / output updates
  $\mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \mathbf{v}_t \circ \mathbf{q}_{t\text{-}1}$
  $\bar{\mathbf{z}}_t = f_{\bar{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\tilde{\mathbf{z}}_t = f_{\tilde{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\mathbf{q}_t = (\bar{\mathbf{z}}_t - \mathbf{o}_t) \oslash \mathbf{v}_t$
  $\mathbf{u}_t = (\mathbf{v}_t - \tilde{\mathbf{z}}_t) \oslash (\mathbf{v}_t \circ \mathbf{v}_t)$
  $\mathbf{s}_t = \mathbf{1}_n \oslash ((|\mathbf{A}|^{\circ 2})^{\mathsf{T}}\mathbf{u}_t)$ # Variable side / input updates
  $\mathbf{r}_t = \bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{s}_t \circ \mathbf{A}^{\mathsf{H}}\mathbf{q}_t$
  $\bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  $\tilde{\boldsymbol{\alpha}}_t = f_{\tilde{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  if stop criterion is met **then**
    break
  end if
end for

At first, AMP (Algorithm 2) and GAMP (Algorithm 3) may seem quite different. However, as detailed in [86] and further elaborated on in the Tech Report G, the GAMP algorithm reduces to the AMP algorithm under the assumption of an AWGN output channel and a few simplifications of the GAMP states. When used to find MMSE estimates, the threshold operator η_t in the AMP algorithm corresponds to the GAMP channel function f_α [73], [87]. However, for MAP estimates, e.g. solving the LASSO problem from a probabilistic interpretation, the threshold operator may be chosen to the soft threshold operator in (4.3) [36], [40]. Note that when used to find MAP estimates, the GAMP channel functions have different definitions and are not conditional expectations but maximisation problems [83], [84]. Comparing the soft threshold based AMP (Algorithm 2) to IST (Algorithm 1 with a soft threshold operator), the only difference between the two is the extra momentum term, $\frac{n}{m}\langle\eta_t'(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^{\mathsf{H}}\boldsymbol{\chi}_{t\text{-}1})\rangle\boldsymbol{\chi}_{t\text{-}1}$, in the so-called Onsager corrected residual $\boldsymbol{\chi}$ which is needed to achieve the theoretical convergence guarantees of AMP compared to IST [36], [40].

5.2 A Weighted Prior

The GAMP channel functions are required to be separable, i.e.

$$p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I) = \prod_j p(\alpha_j|[\boldsymbol{\theta}_I]_j) \tag{5.11}$$

$$p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_o) = \prod_i p(y_i|z_i;[\boldsymbol{\theta}_o]_i)\,. \tag{5.12}$$

Examples of input channel functions[1] that satisfy this requirement are i.i.d. channels such as the Bernoulli-Gaussian channel [82], [88], Bernoulli-Gaussian mixture models [42], elastic net priors [89], or non-i.i.d. models such as Markov tree priors [61]. A particularly popular prior is the i.i.d. general sparse prior consisting of a Bernoulli component plus an arbitrary distribution φ [42], [73], [89]

$$p(\alpha_j;\boldsymbol{\theta}_I) = (1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j)\,, \tag{5.13}$$

where $\tau = \frac{k}{n}$ is the signal density and δ_{Dirac} is the Dirac delta function.

For reconstruction of undersampled signals based on structured sparsity, we propose to use the general weighted sparse (GWS) input channel detailed in Main Contribution 3. In an implementation of the GWS input channel, one may separate the channel updates related to the φ distribution from the rest of the channel updates. That is, it is easy to reuse the GWS channel updates with different φ as detailed in Tech Report G.

Main Contribution 3 (GWS input channel from Tech Report G)

If we assume that $\boldsymbol{\alpha}$ is not only sparse but structured sparse in the sense that some of the coefficient values in $\boldsymbol{\alpha}$ are more likely to be zero than others, we may consider an independent but non-identical general weighted sparse (GWS) input channel, i.e.

$$p(\alpha_j;\boldsymbol{\theta}_I) = (1-w_j\tau)\delta_{Dirac}(\alpha_j) + w_j\tau\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j)\,, \tag{5.14}$$

where $\tau \in [0;1]$ models the overall signal density and the $w_j \in [0;1]$, $j = 1,\ldots,n$ are individual weights that model the belief about the sparsity of the individual coefficients. An expectation maximization (EM) update of τ is

$$\tau^{t+1} = \frac{\sum_{j=1}^n \pi_j^w(r_j, s_j, \boldsymbol{\theta}_I^t)}{\sum_{j=1}^n w_j}\,. \tag{5.15}$$

[1] We are mostly interested in the input channel function which allows us to define a probabilistic model on $\boldsymbol{\alpha}$, e.g. a model of AFM images in the DCT domain. However, since the separability constraint is the same for both in- and output channels, all our proposed input channels may also be modified to model the measurement process described by the GAMP output channel.

In order for the GAMP updates to be stable, the GAMP posterior must remain a proper density which requires that $w_j\tau^{t+1} \in [0;1]$, $\forall j$. *If the requirement on the choice of weights is* $w_j \in [0;1]$, $\forall j$, *then one must also require that* $\tau^{t+1} \in [0;1]$. *This requirement may be satisfied in at least two different ways*

1. *We may force* $\tau^{t+1} = 1$ *whenever* $\tau^{t+1} > 1$. *This may be interpreted as forcing the prior belief on the support probabilities. Note that since* τ *models the overall average sparsity of the signal, forcing* $\tau \leq 1$ *has the effect of forcing the average sparsity in the next GAMP iteration to be no larger than the average of the weights.*

2. *We may adjust the weights towards* $w_j = 1$, $\forall j$ *according to some scheme detailing the weights update. This strategy allows for the weighted model to adapt towards a non-weighted model, if the data suggest such a change.*

More details are given in Tech Report G.

5.3 GAMP for the AFM Image Reconstruction Problem

When implementing the GAMP algorithm for solving the undersampled AFM image reconstruction problem there are several practical issues that must be addressed. First of all, one must decide on the type of estimator to use, i.e. a MMSE or MAP estimator. The rigorous theoretical convergence guarantees for AMP/GAMP in the large system limit ($n \to \infty$) are based on the assumption that $\mathbf{A}$ is a random matrix with i.i.d., zero-mean sub-Gaussian entries [90], [91]. For finite n, the probability of deviation from the SE decreases exponentially in n [92], i.e. finite problem sizes on the order of thousands of pixels are likely not causing convergence issues. A potentially more critical element is the fundamental difference between the random $\mathbf{A}$ used in the convergence analysis and the structured and non-random $\mathbf{A}$ used in the undersampled AFM image reconstruction problem as detailed in Chapter 3. Empirical evidence suggests that MMSE GAMP also converges for randomly sub-sampled structured transforms [42], [93]. As discussed in Section 3.2, it is not easy to implement random sampling on the AFM. Thus, one can hope for this convergence with randomly sub-sampled structured matrices to generalise to the non-random structured AFM sampling patterns. For that reason and due to the computational tractability of our proposed MMSE AFM input channel (detailed in Main Contribution 5), we focus on the MMSE GAMP. That being said, it would be interesting, as a future research task, to investigate in depth the differences between MMSE and MAP GAMP for the undersampled AFM imaging problem.

Computational tractability is a major concern when using GAMP in high-dimensional applications such as image reconstruction. As discussed in Section 3.1.1, fast transform are needed in implementing the matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$. For the GAMP algorithm this requirement extends to the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^{\mathsf{T}}$. We present our proposed solution to this problem for the AFM setting in Section 5.3.1. Computing the conditional expectations in (5.1)-(5.4) may in general entail numerical integration which may easily make the GAMP channel evaluations computationally intractable. Fortunately, for some in- and output channels, analytic solutions to the integrals exist. We discuss such computationally tractable channels for the AFM undersampling problem in Sections 5.3.2 and 5.3.3. The problem of estimating or learning the channel parameters is discussed in Section 5.3.4 whereas Section 5.3.5 provides a short discussion of the choice of stop criterion. The resulting GAMP algorithm when combining all these elements is our second proposed type of reconstruction algorithms tailored for the undersampled AFM reconstruction problem. A comparison of this GAMP method (and an AMP method) to other reconstruction methods is given in Chapter 7.

5.3.1 The Squared Transform and Sum Approximations

The direct application of the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^{\mathsf{T}}$ in Algorithm 3 easily becomes infeasible as discussed in Section 3.1.1. One approach to this problem is to use sum approximations (also known as uniform variance) in the GAMP algorithm. Essentially, one replaces the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^{\mathsf{T}}$ with scaled sums of the vectors. The scaling factor is then either based on the variance of the assumed random $\mathbf{A}$ as in the sum approximation by Krzakala et al. [73] or it is based on the Frobenius norm of $\mathbf{A}$ as in the sum approximation by Rangan [84]. The differences and similarities of these sum approximations are further discussed in Tech Report G. As is empirically shown in Paper C, the choice of the scaling factor in the sum approximations is critical for the performance of GAMP. A mismatched scaling factor may severely degrade the performance of the algorithm. Thus, to use Rangan's sum approximation, one needs to know the Frobenius norm of $\mathbf{A}$ or be able to accurately estimate it since directly computing it may be infeasible if one has to store the full $\mathbf{A}$ in memory. Fortunately, it turns out that for sub-sampled Kronecker products, it is possible to find an efficient (in terms of computational and memory requirements) way to directly implement the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^{\mathsf{T}}$, thus, avoiding the sum approximations altogether. Such sub-sampled Kronecker products are widely applicable in high-dimensional reconstruction problems when combining the theories of structured random matrices [43] or SRMs [50] with Kronecker CS [94]. One particularly interesting application is the sub-sampled DCT used in the undersampled AFM image reconstruction problem, i.e.

$$\mathbf{A} = \mathbf{D}_\Omega \mathbf{\Psi}_{\text{i2DDCT}} \tag{5.16}$$

$$\mathbf{A} = \mathbf{D}_\Omega (\mathbf{\Psi}_{\text{iDCT}} \otimes \mathbf{\Psi}_{\text{iDCT}}), \tag{5.17}$$

with $\mathbf{D}_\Omega = \mathbf{\Phi} = \mathbf{I}_\Omega$, the sampling operator discussed in Section 3.2 and $\mathbf{\Psi}_{\text{i2DDCT}}$, the 2D inverse DCT dictionary discussed in Section 3.3, i.e. $\mathbf{\Psi}_{\text{i2DDCT}} = \mathbf{\Psi}_{\text{iDCT}} \otimes \mathbf{\Psi}_{\text{iDCT}}$ (a Kronecker product) for $\mathbf{\Psi}_{\text{iDCT}}$ a 1D inverse DCT matrix. Our theorems on Hadamard powers of such sub-sampled Kronecker products are presented in Main Contribution 4. In these theorems, we use the following notation from Paper C (see e.g. [95] for a full introduction). The Kronecker product of two matrices $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $\mathbf{B} \in \mathbb{C}^{k \times l}$, denoted by $\mathbf{A} \otimes \mathbf{B} \in \mathbb{C}^{mk \times nl}$, is

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}, \tag{5.18}$$

where a_{ij} denotes the ij-th entry of $\mathbf{A}$. The Hadamard product (the entrywise product) of two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{m \times n}$, denoted by $\mathbf{A} \circ \mathbf{B} \in \mathbb{C}^{m \times n}$, is

$$\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} a_{11} \cdot b_{11} & \cdots & a_{1n} \cdot b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot b_{m1} & \cdots & a_{mn} \cdot b_{mn} \end{bmatrix}. \tag{5.19}$$

The p-th Hadamard power of a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ with $p \in \mathbb{Z}$, denoted by $\mathbf{A}^{\circ p} \in \mathbb{C}^{m \times n}$, is

$$\mathbf{A}^{\circ p} = \begin{bmatrix} a_{11}^p & \cdots & a_{1n}^p \\ \vdots & \ddots & \vdots \\ a_{m1}^p & \cdots & a_{mn}^p \end{bmatrix}. \tag{5.20}$$

That is, it is the Hadamard product with itself p times.

Main Contribution 4 (Hadamard powers of sub-sampled Kronecker products theorems from Paper C)

Theorem 1

Let $\mathbf{D} = diag(d_1, \dots, d_n) \in \mathbb{C}^{n\times n}$ *be any diagonal matrix with diagonal entries* $d_1, \dots, d_n$. *Furthermore, let* $\mathbf{D}_\Omega \in \mathbb{C}^{m\times n}$ *be a matrix created from* $\mathbf{D}$ *by taking only the rows indexed by the set of indices* Ω *with* $|\Omega| = m \leq n$. *That is, the rows not indexed by* Ω *are removed from* $\mathbf{D}$. *Now, take any matrix* $\mathbf{A} \in \mathbb{C}^{n\times l}$ *and* $p \in \mathbb{Z}$. *Then,*

$$(\mathbf{D}_\Omega \mathbf{A})^{\circ p} = \mathbf{D}_\Omega^{\circ p} \mathbf{A}^{\circ p}\,, \tag{5.21}$$

where $\mathbf{D}_\Omega \mathbf{A} \in \mathbb{C}^{m\times l}$ *denotes the matrix product of* $\mathbf{D}_\Omega$ *and* $\mathbf{A}$.

Theorem 2

For any $\mathbf{A}_1 \in \mathbb{C}^{m_1\times n_1}, \dots, \mathbf{A}_s \in \mathbb{C}^{m_s\times n_s}$, *and any* $p \in \mathbb{Z}$,

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p} = \mathbf{A}_1^{\circ p} \otimes \cdots \otimes \mathbf{A}_s^{\circ p}\,. \tag{5.22}$$

Theorem 3

Let $\mathbf{A}_1 \in \mathbb{C}^{m_1\times n_1}, \dots, \mathbf{A}_s \in \mathbb{C}^{m_s\times n_s}$, $\mathbf{B}_1 \in \mathbb{C}^{n_1\times k_1}, \dots, \mathbf{B}_s \in \mathbb{C}^{n_s\times k_s}$, *and* $\mathbf{E}_s = \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s, \mathbf{F}_s = \mathbf{B}_1 \otimes \cdots \otimes \mathbf{B}_s$. *Then for any* $p \in \mathbb{Z}$,

$$(\mathbf{E}_s \mathbf{F}_s)^{\circ p} = (\mathbf{A}_1\mathbf{B}_1)^{\circ p} \otimes \cdots \otimes (\mathbf{A}_s\mathbf{B}_s)^{\circ p}\,. \tag{5.23}$$

Theorem 4

Let $\mathbf{A}_1 \in \mathbb{C}^{m_1\times n_1}, \mathbf{A}_2 \in \mathbb{C}^{n_1\times n_2}, \dots, \mathbf{A}_s \in \mathbb{C}^{n_{s-1}\times n_s}$, $\mathbf{B}_1 \in \mathbb{C}^{l_1\times q_1}, \mathbf{B}_2 \in \mathbb{C}^{q_1\times q_2}, \dots, \mathbf{B}_s \in \mathbb{C}^{q_{s-1}\times q_s}$, *and* $\mathbf{E}_1 = \mathbf{A}_1 \otimes \mathbf{B}_1, \dots, \mathbf{E}_s = \mathbf{A}_s \otimes \mathbf{B}_s$. *Then for any* $p \in \mathbb{Z}$,

$$(\mathbf{E}_1 \dots \mathbf{E}_s)^{\circ p} = (\mathbf{A}_1 \dots \mathbf{A}_s)^{\circ p} \otimes (\mathbf{B}_1 \dots \mathbf{B}_s)^{\circ p}\,. \tag{5.24}$$

Any result regarding Hadamard powers of Kronecker products of complex matrices also holds for the moduli of the matrices, e.g. for Theorem 2, we have

$$|(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p}| = |(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)|^{\circ p} \tag{5.25}$$
$$= |\mathbf{A}_1|^{\circ p} \otimes \cdots \otimes |\mathbf{A}_s|^{\circ p}\,. \tag{5.26}$$

Additionally, since $(\mathbf{D}_\Omega \mathbf{A})^{\circ p}$ *from Theorem 1 also only consists of products of entries of the matrices, we have the similar result*

$$|(\mathbf{D}_\Omega \mathbf{A})^{\circ p}| = |\mathbf{D}_\Omega|^{\circ p} |\mathbf{A}|^{\circ p} \tag{5.27}$$

More details as well as proofs of the theorems are given in Paper C.

Using Theorems 1 and 2, we may express $|\mathbf{A}|^{\circ 2}$ based on the $\mathbf{A}$ in (5.17) as

$$|\mathbf{A}|^{\circ 2} = |\mathbf{D}_\Omega|^{\circ 2} |\mathbf{\Psi}_{\text{i2DDCT}}|^{\circ 2} \tag{5.28}$$
$$= |\mathbf{D}_\Omega|^{\circ 2} (|\mathbf{\Psi}_{\text{iDCT}}|^{\circ 2} \otimes |\mathbf{\Psi}_{\text{iDCT}}|^{\circ 2}) \tag{5.29}$$
$$= \mathbf{D}_\Omega (\mathbf{\Psi}_{\text{iDCT}}^{\circ 2} \otimes \mathbf{\Psi}_{\text{iDCT}}^{\circ 2})\,, \tag{5.30}$$

where in (5.30), we have used that $\mathbf{D}_\Omega$ in the AFM setting is a sub-sampled identity matrix and that the DCT transform is a real transform. Thus, the $|\mathbf{A}|^{\circ 2}$ in (5.30) is a sub-sampled separable transform that is significantly more computationally tractable for

large problem sizes than the application of the direct matrix-vector products. The impact of this difference is summarised in Table 3.1.

5.3.2 Choice of Input Channel

For the undersampled AFM image reconstruction problem, we suggest to use our proposed GWS input channel presented in Main Contribution 3. The weights in the GWS input channel may then be used to exploit the DCT coefficient structure from Main Contribution 1. The coefficient structure is used in a similar way to its usage in our proposed w-IST and w-IHT algorithms presented in Chapter 4. However, in addition to modelling the structure between the DCT coefficients, the GWS input channel also allows for exploiting knowledge about the distribution of the coefficients through the choice of φ in (5.14). In choosing φ, we consider the empirical distribution of the DCT coefficients of the 17 AFM images depicted in Figure 7.1. This distribution is shown in Figure 5.1. Also shown in the figure are examples of a Gaussian distribution and a Laplace distribution. These example distributions are not necessarily optimal in terms of being "best" fits to the empirical distribution of the DCT coefficients. They are merely included to exemplify the trade-off between capturing the heavy tails and capturing the sharp peak of the empirical distribution. The example Gaussian distribution reveals that a Gaussian distribution is unable to model both the heavy tails of the empirical distribution and its peak around zero at the same time. In comparison, the example Laplace distribution is significantly better, though still not ideal, at capturing the shape of the empirical distribution. If the example Laplace distribution is to more accurately capture the peak, it still comes at the expense of not capturing the heavy tails. However, when combined with a Bernoulli component, the Laplace distribution seems to be a reasonable choice of distribution for the AFM setting: The Laplace component captures the heavy tails whereas the Bernoulli component somewhat compensates for the lack of density around the peak at zero. That is, we consider a weighted sparse Bernoulli-Laplace (wBL) prior with signal density τ, Laplace mean μ, and Laplace rate parameter $\lambda > 0$, ($\boldsymbol{\theta}_I = [\tau, \mu, \lambda]^T$), i.e.

$$p(\alpha; \boldsymbol{\theta}_I) = (1 - w_j\tau)\delta_{\text{Dirac}}(\alpha) + w_j\tau\frac{\lambda}{2}\exp(-\lambda|\alpha - \mu|) \ . \tag{5.31}$$

Note that by disregarding the structure among the DCT coefficients, i.e. choosing $w_j = 1, \forall j$, the weighted sparse Bernoulli-Laplace prior in (5.31) reduces to a sparse Bernoulli-Laplace prior based on (5.13). Our results for the MMSE GAMP channel updates for the weighted sparse Bernoulli-Laplace prior in (5.31) are presented in Main Contribution 5.

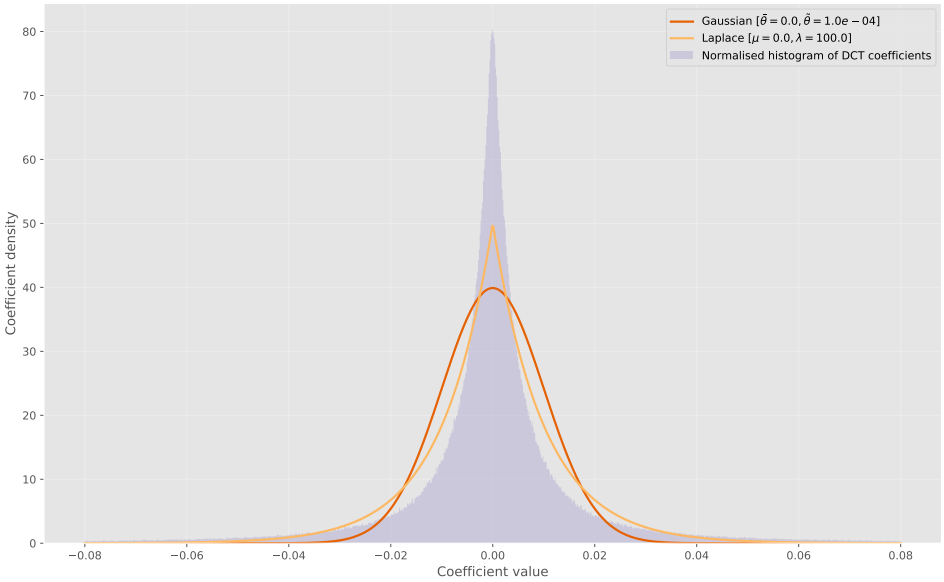


Figure 5.1: Distribution of the DCT coefficients in typical AFM images. The normalised histogram is based on the DCT coefficients of the 17 de-tilted AFM images shown in Figure 7.1. For comparison, the histogram is overlaid with the PDF of a Gaussian distribution with mean $\bar{\theta} = 0.0$, variance $\tilde{\theta} = 10^{-4}$, and the PDF of a Laplace distribution with mean $\mu = 0.0$, rate $\lambda = 100$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

Main Contribution 5 (Weighted Bernoulli-Laplace Input Channel updates from Tech Report G)

For the weighted Bernoulli-Laplace Input Channel in (5.31), the following channel updates may be used

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^w(r_j, s_j, [\boldsymbol{\theta}_I]_j) \Bigg\{ \mu_j + \frac{\mathcal{Z}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \Bigg\} \tag{5.32}$$

$$\begin{aligned} f_{\tilde{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = 2\mu f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) + \pi_j^w(r_j, s_j, [\boldsymbol{\theta}_I]_j) \Bigg\{ & -\mu_j^2 \\ & + \frac{\mathcal{Z}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left[s_j \left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \left(\frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} - \frac{\underline{r}_j}{\sqrt{s_j}} \right) \right) + \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right)^2 \right] \\ & + \frac{\bar{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left[s_j \left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \left(\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} + \frac{\bar{r}_j}{\sqrt{s_j}} \right) \right) + \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right)^2 \right] \Bigg\} \\ & - f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j)^2 , \end{aligned} \tag{5.33}$$

for

$$\mathcal{Z}_{\varphi_j} = \underline{\mathcal{Z}}_{I_j} + \bar{\mathcal{Z}}_{I_j} \tag{5.34}$$

$$\underline{\mathcal{Z}}_{I_j} = \frac{\lambda_j}{2} \exp\left(\frac{1}{2}\lambda_j^2 s_j + \check{r}_j \lambda_j\right) \Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right) \tag{5.35}$$

$$\bar{\mathcal{Z}}_{I_j} = \frac{\lambda_j}{2} \exp\left(\frac{1}{2}\lambda_j^2 s_j - \check{r}_j \lambda_j\right) \Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right) \tag{5.36}$$

$$\check{r}_j = r_j - \mu_j \tag{5.37}$$

$$\underline{r}_j = \check{r}_j + \lambda_j s_j \tag{5.38}$$

$$\bar{r}_j = \check{r}_j - \lambda_j s_j \,, \tag{5.39}$$

and with $\Phi_{\mathcal{N}}(\check{x}) = \int_{-\infty}^{\check{x}} \phi_{\mathcal{N}}(t)\,dt = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\check{x}} \exp\left(-\frac{t^2}{2}\right) dt$ *the cumulative distribution function (CDF) of a standard normal distribution with PDF* $\phi_{\mathcal{N}}(\check{x}) = \mathcal{N}(\check{x}, 0, 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\check{x}^2}{2}\right)$. *Furthermore,* $\pi_j^w(r_j, s_j, [\boldsymbol{\theta}_I]_j) = \frac{w_j \tau \mathcal{Z}_{\varphi_j}}{(1 - w_j\tau)\mathcal{N}(0; r_j, s_j) + w_j \tau \mathcal{Z}_{\varphi_j}}$ *for* $\mathcal{Z}_{\varphi_j} = \int_{\alpha_j} \varphi(\alpha_j; [\boldsymbol{\theta}_I]_j) \mathcal{N}(\alpha_j; r_j, s_j) d\alpha_j$ *and* $\varphi_{\alpha_j|\mathbf{y}; s_j, r_j, [\boldsymbol{\theta}_I]_j}(\alpha_j; [\boldsymbol{\theta}_I]_j) = \frac{\mathcal{N}(\alpha_j; r_j, s_j)\varphi(\alpha_j; [\boldsymbol{\theta}_I]_j)}{\mathcal{Z}_{\varphi_j}}$.

The related Laplace component EM updates are

$$\lambda^{t+1} = \frac{\tau^{t+1} \sum_{j=1}^n w_j}{\sum_{j=1}^n \pi_j^w(r_j, s_j, \boldsymbol{\theta}_I^t) \left(\frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) - \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) \right)} \tag{5.40}$$

$$\mu^{t+1} = \frac{\sum_{j=1}^n \pi_j^w(r_j, s_j, \boldsymbol{\theta}_I^t) \left(\mu^t + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \right)}{\tau^{t+1} \sum_{j=1}^n w_j} \,. \tag{5.41}$$

More details are given in Tech Report G.

5.3.3 Choice of Output Channel

The AFM measurement process is subject to several potential impairments from stochastic noise sources that depend on the measurements system [96], [97], [98]. The GAMP output channel may be used to model such stochastic sources subject to the separability constraint in (5.12). Since our focus has primarily been on making the GAMP algorithm computationally tractable for high dimensional problems as well as designing an AFM specific prior, we have not considered advanced output channels that model the noise sources in the AFM measurement process. We merely restrict our attention to the "default fallback" AWGN GAMP output channel. That is, we consider an AWGN output channel with noise variance σ^2 ($\boldsymbol{\theta}_o = [\sigma^2]$)

$$p(y|z; \boldsymbol{\theta}_o) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-z)^2}{2\sigma^2}\right) , \tag{5.42}$$

which has channel functions [83], [84]

$$f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{vy + \sigma^2 o}{\sigma^2 + v} \tag{5.43}$$

$$f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{\sigma^2 v}{\sigma^2 + v} \tag{5.44}$$

and a corresponding EM update of the noise variance [73]

$$(\sigma^2)^{t+1} = \frac{\sum_i \frac{(y_i - o_i^{t+1})^2}{\left(1 + \frac{1}{(\sigma^2)^t} v_i^{t+1}\right)^2}}{\sum_i \frac{1}{1 + \frac{1}{(\sigma^2)^t} v_i^{t+1}}} \,. \tag{5.45}$$

5.3.4 Parameter Learning and Initialisation

If the in- and output channel parameters $\boldsymbol{\theta}_I$ and $\boldsymbol{\theta}_o$ are known, one may fix their values[2] when using the GAMP algorithm. If the channel parameters are unknown a priori or if a more general channel model (e.g. a Gaussian mixture [42]) is used, one may learn the GAMP channel parameter values as part of the reconstruction, e.g. via expectation maximization (EM) [42], [73]. EM is an iterative maximum likelihood (ML) parameter estimation algorithm [99]. For the GAMP channel parameter updates, the EM algorithm amounts to iterating [42]

$$\boldsymbol{\theta}_I^{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y}, \boldsymbol{\theta}_C^t}[\ln(p(\boldsymbol{\alpha}; \boldsymbol{\theta}))] \tag{5.46}$$

$$\boldsymbol{\theta}_o^{t+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, \mathbb{E}_{\mathbf{z}|\mathbf{y}, \boldsymbol{\theta}_C^t}[\ln(p(\mathbf{y}|\mathbf{z}; \boldsymbol{\theta}))] \,, \tag{5.47}$$

where $\boldsymbol{\theta}_C = [\boldsymbol{\theta}_I^T, \boldsymbol{\theta}_o^T]^T$ is the vector of all channel parameters. The posteriors used in the expectations in (5.46) and (5.47) are approximated by the GAMP posteriors and a partial update procedure (similar the expectation conditional maximization algorithm [100]) in which one parameter is updated at a time is used [42]. More details about these EM for GAMP methods are given in Tech Report G. The resulting EM updates for our proposed wBL input channel, presented in Main Contribution 5, are given in (5.40) and (5.41). The AWGN output channel noise variance EM update is given in (5.45).

Since the EM algorithm is only guaranteed to find a local maximum [99], proper initialisation becomes important. In our evaluation of the GAMP algorithms for solving the undersampled AFM imaging problem presented in Chapter 7, we initialise the GAMP channel parameters using an offline ML estimate as detailed in Section 7.3.1. A more generic initialisation strategy is described in Tech Report G.

5.3.5 Stop Criteria

A stop criterion may be used to terminate the iterations in Algorithms 2 and 3 when the estimate $\hat{\boldsymbol{\alpha}}$ is sufficiently close to the true solution. A few such stop criteria are detailed in Tech Report G. We have empirically found that the normalised mean squared error stop criterion proposed in [42] as well as the residual measurements ratio stop criterion proposed in [44] both work well when attempting to solve the undersampled AFM image reconstructed problem.

[2] Our simulations have revealed that fixing the AWGN noise variance results in poor convergence of the GAMP algorithm. In a practical setup, a certain "slack" is needed for the algorithm to converge.

6 Correctness and Reproducibilty of Results

Our main method for assessing the performance of our proposed reconstruction algorithms from Chapters 4 and 5 is computer simulations. Such an empirical approach to performance evaluation entails the need for strict requirements to the assurance of correctness and reproducibility of the results. We now present our initiatives in terms of assuring the correctness and reproducibility of our results. First, we discuss the general need for correctness and reproducibility of computational results. Next, we introduce the Magni Python Package which includes our proposed tools for aiding in ensuring correct and reproducible computational experiments. Finally, we provide an overview of our actions taken in ensuring correctness and reproducibility of the results presented in this thesis. More details about Magni and our proposed tools may be found in Papers D, E, and F.

6.1 The Credibility Crisis

The use of scientific computations in academic research has increased significantly in the past decade and with it has followed an increased focus on transparency and reproducibility of computational experiments [101], [102], [103], [104]. Requirements for adhering to best practices for making research reproducible have been put forth [105], [106] and journals have started to require of authors to make publicly available the tools needed to reproduce their results [103], [107], [108]. This move towards more open and easily reproducible computational experiments is largely an answer to the so-called *credibility crisis* that has hit the computational sciences [103] due to lack of reproducibility of published results [109], [110], [111] and retractions from high-ranked journals stemming from incorrect computational results [112]. In order to address this credibility crisis in the computational sciences, more focus must be directed towards at least two critical elements in any computational experiment

1. The assurance of correctness of the computational results.
2. The inspectability of the methods and reproducibility of the computational results.

Throughout the course of the research program that led to the results reported on in this thesis, we have been adopting as many of the best practice policies in addressing these two issues as possible as they have been put forth by the research communities. Additionally, we have continuously been working on enabling such policies in the typical scientific Python workflow. Thus, in line with the general evolution in this area, our methods and actions taken towards ensuring correctness of results and making our research reproducible have evolved from publication to publication.

6.1.1 The Need for Assurrance of Correctness of Results

The results presented in this thesis are primarily based on simulations that involve random numbers. We essentially input random numbers (typically randomly sampled images) to our algorithms which, consequently, produce a random output in the sense that the output depends on the randomness in the input. We then evaluate the performance of our algorithms by running a large number of simulations in order to assess general patterns across all (random) results. By design this method is sensitive to errors in the experimental setup and the implementation of the algorithms. If any part of the experiment is wrongly implemented, the results of our experiments become truly *random*, i.e. they convey no meaningful patterns. One can hardly expect a large computational experiment

to be completely free of errors. However, it is reasonable to expect of the authors of the experiment to take actions to reduce the risk of any errors being of significant importance to the conclusions of the study.

Focusing on the use of proper design principles, code quality and testing, and general best practices in writing scientific software is one way to build trust in computational results [109], [113], [114], [115]. Another way is to use well designed computing workflows that systematically address potential flaws in the computational experiment [116], [117].

6.1.2 The Need for Reproducibility of Results

Reproducibility refers to the notion of being able to independently re-implement and repeat an experiment and its outcome from a description of it [103], [118]. A related notion is *replicability*, which refers to the notion of being able to repeat an experiment and its outcome using the same experimental setup as was used in the original experiment[1].

Without reproducibility (or at least replicability) in a computational experiment, it becomes difficult (if not impossible) to build on published results and identify and correct any errors in the experiment. Thus, for computational results to be useful for others, it is important that the description of the experiment that led to the results, i.e. the software or code as well as associated data, is transparent, complete, and easily available to others.

Extensive time and effort have been put into defining best practices and guidelines for making computational research reproducible and transparent, see e.g. [105], [106], [119], [120], [121], [122], [123]. As noted in several of these works, such guidelines are constantly evolving as more evidence on practical issues with reproducibility becomes available. Software for aiding in making computational results reproducible is evolving similarly with new tools frequently becoming available, see e.g. [121], [124], [125], [126].

6.2 The Magni Python Package

The Magni Python package is our take at a collection of tools for use in a scientific Python workflow for undersampling and reconstruction of AFM images. It has been designed for reuseability and with a focus on ensuring correctness of results as well as aiding in making computational results reproducible according to best practices as found in the literature discussed in Sections 6.1.1 and 6.1.2. In terms of asserting correctness of results, the main actions, we have taken, are

- All code is under version control using the Git version control system to automatically keep track of all changes to the code (see e.g. [127] for an introduction to Git version control for research software).
- The inclusion of extensive unit tests (test code included to test a "code unit", e.g. a part of a function, for correctness) and doc tests (tests based on the usage examples in the documentation of the code) that cover $\sim 90\,\%$ of the code paths in addition to 15 carefully designed end to end examples. All of these are routinely tested using a continuous integration system, i.e. all tests are automatically run every time a new set of changes is added using the version control system.
- All code consistently adheres to the PEP8 style guide[2] for making Python code easily readable. Variables are named consistently throughout the package and the maximum cyclomatic complexity [128], [129] has been kept below 10. These are all best practices for keeping code simple, easily comprehensible, and easily maintainable and, thus, hopefully, free of errors. Automatic tests for these style constraints are included in the test suite run by the continuous integration system.

[1] Some authors interchange this meaning of *reproducibility* and *replicability*. We use the definitions from [103], [118].

[2] The PEP8 style guide is available at `https://www.python.org/dev/peps/pep-0008/`.

- The entire package including its full API has been documented according to the Numpydoc standard[3] and the documentation is automatically built and served online[4].
- All code included in Magni releases have been reviewed by at least one other person than its author.
- An extensive function input validation system (detailed in Section 6.2.1) is used consistently throughout the package.

The `magni.reproducibility` subpackage provides tools that may be used to annotate results databases and track provenance in order to aid in making results reproducible. This subpackage is detailed in Section 6.2.2. More details about Magni may be found in [130, (Paper D)].

6.2.1 Input Validation

In implementing a large computational experiment, numerous variables are passed between functions, class constructors, class methods, etc. Typically, a function poses strict requirements for the types of its input variables, e.g. a variable must be a string or it must be an integer. Compliance to such requirements are not automatically tested in weakly typed languages like Python[5] which may lead to unexpected and erroneous results, e.g. if a string is supplied instead of an integer (in Python: »2 * 2 = 4« whereas »2 * '2' = '22'«). Additionally, scientific computations in Python typically involve the use of advanced types such as the NumPy ndarray [131] for storing matrices and vectors. The NumPy ndarray is not only itself a data type but also holds equally important information about the shape of the array, the type of its elements, etc. All of these define requirements on the input to functions.

In order to prevent erroneous results due to wrongly passed function input variables, we propose to use an online (at execution time) input validation framework based on the application-driven data types presented in Main Contribution 6. These application driven data types are specifically tailored for easily expressing the requirements for the numerical variables typically used in scientific computations, e.g. matrices and vectors. Our implementation of our proposed input validation framework in Python enforces the validation requirements by raising an exception if an input variable does not comply with the validation scheme. This input validation framework is used throughout Magni and we have found that it not only helps us identify potentially disastrous problems in our own code but also helps in identifying problems due to changes in the underlying software libraries that we make use of, i.e. NumPy and other packages from the Scipy stack.

Main Contribution 6 (Application-driven Data Types from [132, (Paper E)])

We suggest the concept of so-called application-driven data types as a signal processing data model for programming. These data types are intended for expressing the validation scheme of function arguments.

An application-driven data type is some "mental" intersection between math and computer science in scientific computing and signal processing in particular. For example, the set of real-valued matrices with dimensions m times n, $\mathbb{R}^{m \times n}$, is an example of an application-driven data type. If the user is able to test the validity of a function

[3] The Numpydoc standard is available at https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt.

[4] The Magni documentation is available at http://magni.readthedocs.io/en/latest/.

[5] The latest versions of Python (3.5/3.6) include a standard for specifying such requirements, though third party tools are still needed in order to enforce the requirements. See https://www.python.org/dev/peps/pep-0526/ for the details.

argument against this application-driven data type, there is no need for the user to consider the distinction between Python floats, numpy generics, numpy ndarrays, and so on.

In Python signal processing applications, there should be an application-driven data type representing the most general numerical value being able to assume any numerical value of any shape. This data type should be able to be restricted to less general data types by specifying the mathematical set, the range or domain of valid values, the number of dimensions, and/or the specific shape of the data type. The suggested validation schemes should be expressed in terms of the desired application-driven data type.

A reference implementation of this suggested validation strategy is made available by the open source Magni Python package [130] through the subpackage `magni.utils.validation`.

More details are given in [132, (Paper E)].

6.2.2 Storing Experiments Metadata

In making a computational experiment reproducible, one must create a complete record of its computational environment including details such as library versions, parameter values, and clear detailing of the exact code that is run [103], [109], [110]. Two strategies for tracking such provenance of a computational experiment seem to have found adoption in the scientific communities:

1. Using tools that track the computational workflow and store metadata detailing the experiment. Ideally such workflow details enable a later reproduction of the experiment. Examples of such tools include Sumatra [121], Madagascar [124], and ActivePapers [125].

2. Running the experiment in a virtual machine or container such as Docker [133]. Ideally, the virtual machine or container remains forever executable and, thus, allows for future reproduction of the experiment.

The container strategy seems compelling in the current situation where scientific code tends to *rot*, i.e. it eventually stops working as time passes due to incompatible changes in hardware platforms and the low-level software libraries [125], [133]. However, the container strategy has been criticised for being an unstable and non-transparent solution due to its black-box nature[6]. Thus, at least for now, the tools for implementing the container strategy do not seem to have matured sufficiently to be reliably used in science and research.

In our work, we have adopted a variant of the metadata storage strategy to making computational results reproducible. Specifically, we store metadata about a particular computational experiment along with its results. Such metadata include information about hardware and software libraries used as well as details about the input data, parameter choices, and the specific code that was run. Our proposed practical solution to this problem of storing metadata about a single computational experiment is detailed in Main Contribution 7.

Main Contribution 7 (A Pythonic Approach to Storing Reproducibility Metadata from [134, (Paper F)])

We propose creating a scientific Python package that may be imported in existing scientific Python scripts and may be used to store all relevant metadata for a computational experiment along with the results of that experiment in an HDF5 database. For the

[6] This criticism has been raised by several leading researches in the computational sciences. See e.g. the blog by C. Titus Brown http://ivory.idyll.org/blog/2017-pof-software-archivability.html or the blog by Gaël Varoquaux http://gael-varoquaux.info/programming/of-software-and-science-reproducible-science-what-why-and-how.html.

> *highest flexibility, we propose to store the metadata as separate HDF5 arrays and suggest using JavaScript Object Notation (JSON) [135] for serializing the metadata. This makes for a humanly readable representation. Furthermore, JSON is a standard format with bindings for most major programming languages.*
>
> *We argue that this idea of storing metadata along with results is an excellent solution. Having everything compiled into one standardized and open file format helps keep track of all the individual elements and makes it easy to share the full computational experiment including results and metadata.*
>
> *A reference implementation of the above suggested library design is available in the open source Magni Python package [130]. In particular, the subpackage* `magni.reproducibility` *is based on this suggested design.*
>
> More details are given in [134, (Paper F)].

6.3 Quality Assurance and Reproducibility in the Present Work

The size of our computational experiments entails the use of compute servers in a compute cluster for handling the computations. Thus, we use a two part workflow as illustrated in Figure 6.1. That is, in our simulations, we specify our experiment in a Python script which also includes the code needed for configuring the Python packages used in the experiment. This Python script is then executed on one or more compute servers in order to carry out the computations and store the results. Once the computations have finished an analysis and visualisation of the results is done on a workstation or laptop using Jupyter notebooks.

To the extent possible, we use consistent labelling of the data in all processing steps in our workflow in order to ensure that the final visualisation of the results corresponds to the specified experiment. Furthermore, in all steps, we use the best practices for ensuring correctness of results as discussed in Section 6.1.1. This includes adherence to style and workflow guidelines, the use of frequent assertions in the code to check the sanity of the intermediate results, the consistent usage of our proposed input validation framework detailed in Section 6.2.1, and comprehensive logging and inspection of intermediate states. Whenever possible, we also attempt to reproduce already published baseline results using our experimental setup. All our results are stored in HDF5 databases which are annotated according to our proposed metadata storage strategy for enhanced reproducibility of our results as discussed in Section 6.2.2. Finally, to the extent possible, all critical algorithm implementations and support routines used in the computations are included in Magni with its strict requirements for code quality as discussed in Section 6.2. Finally, all of our Python code builds on the well established scientific Python stack [136]. An overview of the relevant resources needed to reproduce the computational results presented in this thesis is given in Table 6.1.

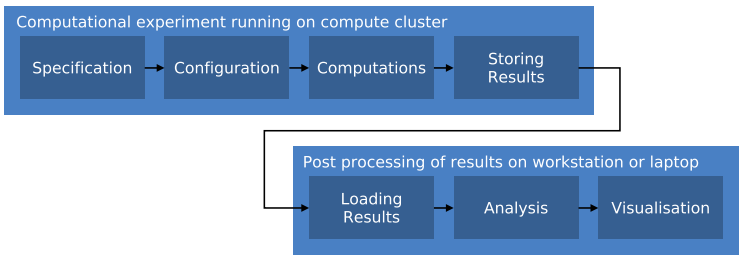


Figure 6.1: Workflow in the typical computational experiment presented in this thesis. An experimental part which includes the specification and configuration of the experiment as well as the computations and storage of results is run on a compute cluster. Afterwards the results are loaded on a workstation or laptop for analysis and visualisation.

Origin	Python Libraries	Input Data	Scripts and Code	Results
Paper A	Anaconda 2.3.0 (Python 2.7) Magni 1.3.0 PyWavelets 0.3.0 PyUNLocBoX v0.2.1-211-g585027a	doi:10.5281/zenodo.17573	doi:10.5281/zenodo.32959	doi:10.5281/zenodo.32958
Paper B	Anaconda 4.1.1 (Python 3.5) Magni 1.5.0 PyUNLocBoX v0.2.1-211-g585027a	doi:10.5281/zenodo.17573 doi:10.5281/zenodo.60434	doi:10.5281/zenodo.60512	doi:10.5281/zenodo.60512 doi:10.5278/vbn/projects/wISTwIHT
Paper C	Anaconda 4.1.1 (Python 3.5) Magni 1.6.0 PyWavelets 0.4.0	doi:10.5281/zenodo.160700	doi:10.5281/zenodo.165051	doi:10.5281/zenodo.165051 doi:10.5278/240710282
Paper F	Anaconda 2.4.1 (Python 3.5) Magni 1.5.0	n/a	doi:10.5278/VBN/MISC/MagniRE	doi:10.5278/VBN/MISC/MagniRE
Chapter 7	Anaconda 4.3.0 (Python 3.6) Magni 1.7.0 PyUNLocBoX v0.2.1-211-g585027a	doi:10.5281/zenodo.17573 doi:10.5281/zenodo.60434	doi:10.5281/zenodo.375833	doi:10.5281/zenodo.375833 doi:10.5278/252861471

Table 6.1: Overview of external resources which may be used to reproduce the computational results presented in this thesis. All releases of Magni are available at doi:10.5278/VBN/MISC/Magni. Anaconda installers are available at `https://repo.continuum.io/archive/index.html`. PyWavelets is available at `https://github.com/PyWavelets/pywt`. PyUNLocBoX is available at `https://github.com/epfl-lts2/pyunlocbox`. See the individual origins (papers or chapters) for further details.

7 Reconstructions of Undersampled AFM Images

Having introduced our proposed algorithms for solving the undersampled AFM image reconstruction problem in Chapters 4 and 5, we now present results from a large simulation study comparing our proposed algorithms to the baseline approaches discussed in Section 3.4. We present both extensive quantitative evaluations of reconstructions of undersampled AFM images and exemplify typical reconstructions. Before giving our simulation results, we outline our comparison criteria and specify our experimental setup.

7.1 Ground Truth Images

In our experiments, we evaluate a given reconstruction of an undersampled AFM image by comparing it to a reference *ground truth image*. Establishing such natural ground truth AFM images is not an easy task since they must themselves be acquired using AFM or a similar nano scale measurement processes. Hence, the assumed ground truth images are subject to measurement impairments and noise. Our reference AFM images have been acquired using traditional raster scan methods at slow scan speeds on calibrated AFM equipment. Thus, we expect and assume that measurement impairments and noise are at a minimum, the only exception being a possible tilt. We assume a small enough tilt to be able to accurately correct it using a plane fit method as detailed in [98]. However, keep in mind when assessing our results that some of the distortion in the reconstructions relative to the assumed ground truth images may in fact stem from noise removal or similar desirable properties of the reconstruction algorithms.

Our simulation experiments are based on the 17 AFM images of various specimens shown in Figure 7.1. For our purpose of comparing AFM images, the absolute topography levels are irrelevant. Thus, in our comparison, we scale and offset all pixel values to a floating point representation with values in the interval $[0, 1]$. That is, all images (reference images and reconstructions) are scaled such that the smallest pixel value is 0 and the largest pixel value is 1.

7.2 Reconstruction Quality Indicators

To quantitatively evaluate the quality of reconstructed AFM images compared to a reference, we use two quality indicators: peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). The PSNR is a mean squared error based quality indicator defined as (see e.g. [137])

$$\mathrm{PSNR}(\mathbf{X}, \hat{\mathbf{X}}) = 10 \log_{10} \left(\frac{P^2}{\sum_{k=1}^{\mathrm{h}} \sum_{l=1}^{\mathrm{w}} \left(\mathbf{X}_{kl} - \hat{\mathbf{X}}_{kl} \right)^2} \right) \tag{7.1}$$

where P is the pixel value dynamic range, i.e. $P = 1$ by our convention of considering images with pixel values in the interval $[0, 1]$.

The PSNR is a traditional image processing quality indicator - the higher a PSNR, the better an estimate $\hat{\mathbf{X}}$. However, the PSNR has been criticised for not accurately predicting the human perception of image quality [137]. The SSIM is an attempt at defining a quality indicator that more accurately predicts the human perception by considering local similarity expressed in terms of local means, variances and co-variances of smaller image patches [138], [139]. By computing such local structure in a moving window across an entire

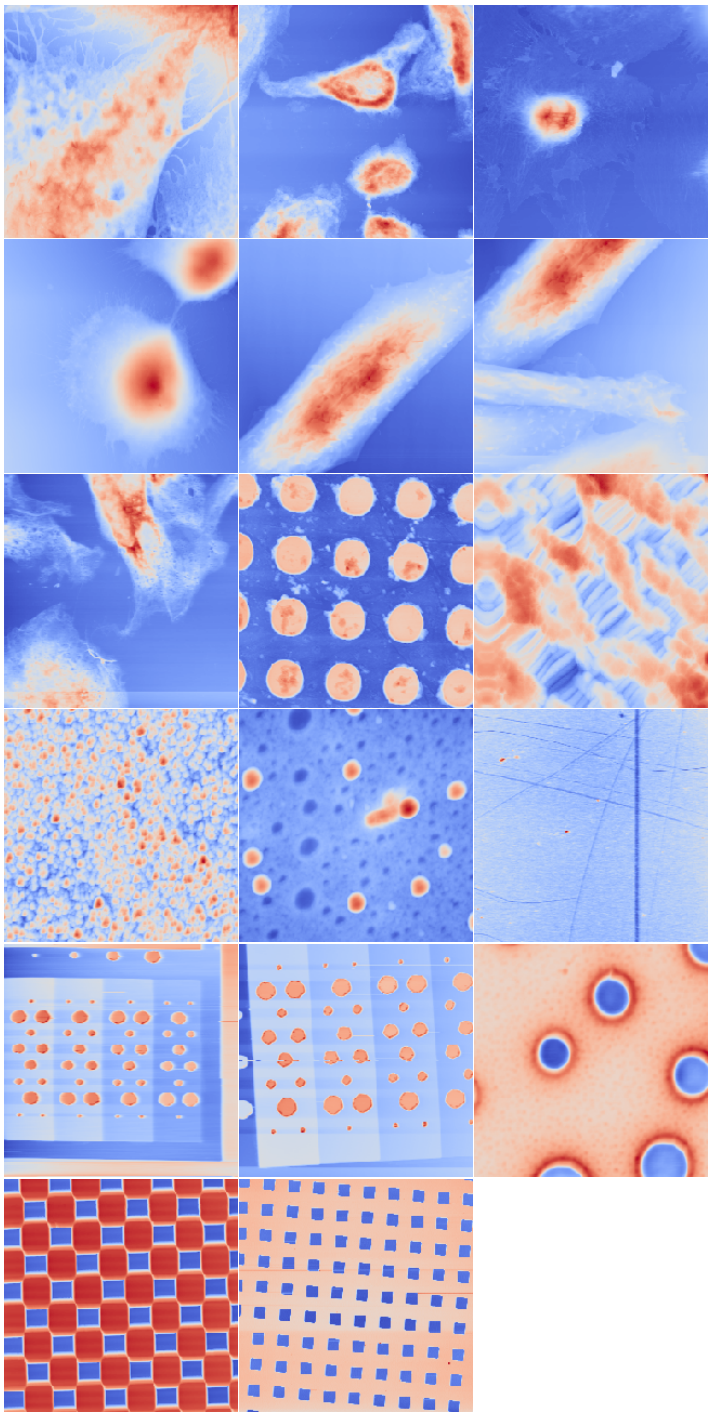


Figure 7.1: The 17 ground truth AFM images used in the reconstruction simulations. The images are from "Atomic Force Microscopy Images of Cell Specimens" and "Atomic Force Microscopy Images of Various Specimens" both by Christian Rankl, Keysight Technologies. They are licensed under CC-BY 4.0, available at doi:10.5281/zenodo.17573 and doi:10.5281/zenodo.60434, and provided as-is without warranty of any kind. All images have been de-tilted using a plane de-tilt method, i.e. for each image a least square fit of a plane to that image has been subtracted from the image using the method detailed in [98]. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

image, a SSIM map results. This SSIM map details the structural similarity throughout the image. To obtain a scalar index, one may compute the mean of the SSIM map. With a slight abuse of notation, in the sequel, we take SSIM to denote this mean SSIM map. Thus, our presented SSIM is the mean SSIM as detailed in equations (5) trough (17) in [138] with window size 7, $K_1 = 0.01$, $K_2 = 0.03$, $C_3 = \frac{C_2}{2}$, $\alpha = \beta = \gamma = 1$ in those equations. The $\mathrm{SSIM}(\mathbf{X}, \hat{\mathbf{X}})$ is then a scalar in the interval $[-1; 1]$ with 1 being a perfect match between $\mathbf{X}$ and $\hat{\mathbf{X}}$. SSIM has been used to assess image quality in a range of natural imaging applications including medical imaging (see [137] for an overview). Thus, we expect the SSIM to be a reasonable indicator of the quality of our reconstructed AFM images.

In assessing our results presented in Sections 7.3.2 and 7.3.3, it is beneficial to keep in mind that the SSIM tends to penalise structural problems in $\hat{\mathbf{X}}$ such as introduced artefacts whereas PSNR tends to penalise more (mathematically) systematic changes in $\hat{\mathbf{X}}$ such as blurring, i.e. a loss of details, or contrast changes.

7.3 Reconstruction Simulations

We now present a large simulation study designed to empirically determine the performance of the various reconstruction algorithms, we have presented for solving the undersampled AFM image reconstruction problem. In this simulation study, we test a large set of combinations of key choices in the reconstruction algorithms and record the resulting reconstruction performance for each combination. Specifically, in order to be able to compare the reconstructed images to the assumed ground truth images, we simulate the undersampling and reconstruction process and record the following performance indicators of the reconstructed image

- The PSNR of the reconstruction relative to the ground truth as defined in (7.1).
- The SSIM of the reconstruction relative to the ground truth as detailed in Section 7.2.
- The measured reconstruction time in seconds[1].

In addition to these scalar performance indicators, the full reconstructed images are saved. Finally, we also record the pixel undersampling ratio ι (see Definition 2). This allows us to compare the performance of the reconstruction algorithms versus both the AFM undersampling ratio δ (see Definition 1) and the more traditional image pixel undersampling ratio ι. As can be seen from Figure 2.2, the number of pixels that are included in the sampling may vary significantly between sampling patterns for a fixed δ. That is, when all pixels touched by the sampling path are included in the measurements, the number of measurements varies significantly with the choice of sampling pattern for a fixed sampling path length.

Definition 2

The pixel undersampling ratio is

$$\iota = \frac{m}{p}, \tag{7.2}$$

where p is the total number of pixels in the reconstructed AFM image and m is the number of measurements as detailed in Chapter 3.

[1]The degree to which the tested reconstruction algorithms have been optimised for execution speed varies significantly. Thus, this measured execution time should only be used as an indicator of order of the execution time, one can expect.

The details of our experimental setup are given in Section 7.3.1. A few typical examples of reconstructions are presented in Section 7.3.2. The full set of results comparing PSNR and SSIM for all tested combinations are presented in Section 7.3.3. A discussion of the results is given in Chapter 8. All the material needed to reproduce the results of our simulation experiments is listed in Table 6.1. This includes a Jupyter Notebook which reproduces all the results figures show in this thesis. In re-running the experiments, it suffices to download the required material listed in Table 6.1, make any auxiliary modules available on the PYTHONPATH, and then execute the main simulation script according to the documentation provided with it.

7.3.1 Experimental Setup

We simulate the undersampling and reconstruction of AFM images based on all combinations of

- 17 AFM images, i.e. the images shown in Figure 7.1.
- 4 different sampling patterns.
- 12 different reconstruction algorithms including our proposed algorithms as well as well established baseline algorithms.
- 25 different undersampling ratios, i.e. $\delta \in \{0.05, 0.0625, \ldots, 0.35\}$

Furthermore, for the algorithms based on iterative thresholding (Algorithm 1), we test 25 sparsity levels, i.e. $\rho \in \{0.05, 0.065, \ldots, 0.4\}$. The threshold level used in the iterative thresholding is then determined from ρ as detailed in Section 4.3.1.

For all reconstruction algorithms that rely on a dictionary, we use the DCT dictionary discussed in Section 3.3. The resulting system matrix in (5.17) is implemented using the fast transform approaches described in Sections 3.1.1, 3.2, and 3.3. The ground truth images all have a size of $\mathrm{h} = 256$ times $\mathrm{w} = 256$ pixels and the reconstructed images have the same size. The specifics of the sampling patterns and reconstruction algorithms used in the simulations are detailed below.

Sampling patterns

We consider the four sampling patterns illustrated in Figure 2.2, i.e. uniform lines, rotated uniform lines, random pixels, and spiral with corners. For a given sampling path length determined by δ, we then simulate the undersampling of the ground truth images by extracting pixels according to the following rules (which are illustrated in Figure 2.2)

> **Uniform lines:** The sampling path length is truncated such that all horizontal lines are fully sampled. The lines are uniformly distributed across the full image. The values of all pixels touched by the sampling path are included in $\mathbf{y}$.
>
> **Rotated uniform lines:** The sampling path length is truncated such that all rotated lines are fully sampled. The lines are uniformly distributed across the full image. The specific angle with which the lines are rotated depends on δ, i.e. the angle is chosen as detailed in Table 7.1 based on the value of δ in the table that is closest to the value of δ used in the simulation. The values of all pixels touched by the sampling path are included in $\mathbf{y}$.
>
> **Random pixels:** The pixel values in $\mathbf{y}$ are chosen uniformly at random from the full image. The number of included pixels are chosen such that $\iota = 2\delta$. As discussed in Chapter 2, such a sampling pattern is not easily implementable in AFM. However, since a significant number of CS results rely on such random sampling, we include the random pixels sampling approach for comparison.

Spiral with corners: The sampling path length is determined by δ. The pitch of the spiral is chosen such that the sample path ends at a distance from the centre corresponding to the distance from the centre to the corners of the image. The sampling is assumed to continue outside of the image area following the spiral pattern (which is different from what is illustrated in Figure 2.2). The values of all pixels within the image area touched by the sampling path are included in $\mathbf{y}$.

Undersampling ratio δ	0.10	0.15	0.20	0.25	0.30
Angle in radians	0.8216608	2.574812	0.6337494	2.503844	2.524497

Table 7.1: Angels used in rotated uniform lines sampling. When using an angle of zero radians, the rotated uniform lines sampling pattern corresponds to the uniform lines sampling pattern. As the angle is increased, the vertical lines rotate counterclockwise.

Reconstruction Algorithms

An overview of the 12 different reconstruction algorithms, which we consider in our simulations, is given in Table 7.2. The rest of this section is devoted to stating all the details about the configuration of these reconstruction algorithms.

For the algorithms which make use of weights, the weights have been chosen as follows. A jack-knife approach (see e.g [146]) is used in selecting the training images to which the model of the DCT coefficients in Main Contribution 1 is fitted. That is, the DCT spectra of all but the image being reconstructed is used in fitting the model in (3.10). A least squares fit is used for fitting the model as proposed in [27, (Paper B)]. That is, for fixed $a = 2.5 \cdot 10^{-3}$, we solve

$$\underset{b,c_1,c_2}{\text{minimise}} \sum_{\check{\mathbf{z}}} (|\check{\boldsymbol{\alpha}}(\check{\mathbf{z}})| - f(b, c_1, c_2; a, \check{\mathbf{z}}))^2 \tag{7.3}$$

using Powell's method (see e.g. [141]) with the initial guess $b = 0.005$, $c_1 = c_2 = 0.01$. Here $\check{\boldsymbol{\alpha}}(\check{\mathbf{z}})$ is a 2D representation of the average $\boldsymbol{\alpha}$ (the average DCT spectrum) indexed by $\check{\mathbf{z}} = [z_1, z_2]^\mathsf{T}$. In computing the fit, we use a re-scaling and offset of both image height, image width, and topography height to the interval $[0, 1]$. Having fitted the model f, we then construct a weights vector $\mathbf{w}$ with an ordering matching that of $\boldsymbol{\alpha}$ using the following algorithm class specific transformations

Iterative Thresholding: The fitted weights are scaled and offset to have values in the interval $[10^{-3}; 1]$ as suggested in [27, (Paper B)].

Weighted ℓ_1 minimization: The fitted weights are scaled and offset to have values in the interval $[10^{-3}; 1]$. Since the weights should, in general, be selected to relate inversely to the expected signal magnitudes [147], the inverses of these re-scaled weights are used.

GAMP with GWS prior: The fitted weights are scaled and offset to have values in the interval $[0.1, 0.99]$ which we have empirically found to result in stable reconstructions.

The weighted ℓ_1 LS reconstruction method [147][2] is based on solving

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\text{argmin}} \, ||\mathbf{W}\boldsymbol{\alpha}||_1 \quad \text{s.t.} \quad ||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2^2 \leq \epsilon \,, \tag{7.4}$$

[2] A re-weighting scheme is used in [147]. However, in order to guide the algorithm to a solution that is in line with our model, we fix the weights instead of iteratively updating them.

Name	Algorithm Specification	Fixed Parameter Values	Implementation
Linear Interpolation	Delaunay triangularisation via Qhull [140] + Barycentric interpolation (see e.g. [141])	n/a	`scipy.interpolate.griddata`
Cubic Interpolation	Delaunay triangularisation via Qhull [140] + Cubic Beizer curve interpolation via a Clough-Tocher scheme [142]	n/a	`scipy.interpolate.griddata`
Nearest Neighbour Interpolation	Nearest Neighbour via the kd-tree algorithm with sliding midpoint splitting [143]	n/a	`scipy.interpolate.griddata`
Total Variation LS	Solving (3.11) as detailed in [144] using Douglas-Rachford splitting [145]	$T_{\max} = 300$, $\kappa = 10^{-2}$, $\epsilon = 10^{-6} \cdot \|\|\mathbf{y}\|\|_2$	`pyunlocbox`
Weighted ℓ_1 LS	Solving (7.4) using Douglas-Rachford splitting [145], weights from solving (7.3)	$T_{\max} = 300$, $\kappa = 1.0$, $\epsilon = 10^{-6} \cdot \|\|\mathbf{y}\|\|_2$	`pyunlocbox`
ℓ_1 LS	Solving (3.9) using Douglas-Rachford splitting [145]	$T_{\max} = 300$, $\kappa = 1.0$, $\epsilon = 10^{-6} \cdot \|\|\mathbf{y}\|\|_2$	`pyunlocbox`
w-IST (Res/Meas)	Algorithm 1 with η_t from (4.5), weights from solving (7.3), and the stop criterion in (4.6)	$T_{\max} = 300$, $\kappa = 0.6$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.it`
IST (Res/Meas)	Algorithm 1 with η_t from (4.3) and the stop criterion in (4.6)	$T_{\max} = 300$, $\kappa = 0.6$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.it`
w-IHT (Res/Meas)	Algorithm 1 with η_t from (4.4), weights from solving (7.3), and the stop criterion in (4.6)	$T_{\max} = 300$, $\kappa = 0.6$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.it`
AWGN wBL GAMP EM (Res/Meas)	Algorithm 3 with $\|\mathbf{A}\|^{\circ 2}$ from (5.30), the wBL prior with EM in Main Contribution 5 with weights from solving (7.3), the AWGN output channel with EM from (5.43)-(5.45), and the stop criterion in (4.6)	$T_{\max} = 300$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.gamp`
AWGN iidBL GAMP EM (Res/Meas)	Algorithm 3 with $\|\mathbf{A}\|^{\circ 2}$ from (5.30), an i.i.d. BL prior with EM based on Main Contribution 5 with $w_j = 1, \forall j$, the AWGN output channel with EM from (5.43)-(5.45), and the stop criterion in (4.6)	$T_{\max} = 300$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.gamp`
DMM AMP M (Res/Meas)	Algorithm 2 with η_t from (4.3), the median threshold update from (7.11), and the stop criterion in (4.6)	$T_{\max} = 300$, $\epsilon = 10^{-6}$	`magni.cs.reconstruction.amp`

Table 7.2: Overview of algorithms tested in the reconstruction simulations. For the LS based algorithms, κ and $T_{\max}$ refers to the step-size and maximum number of iterations, respectively, used in the `pyunlocbox` implementation. See the text in Section 7.3.1 for details about the choice of weights, GAMP channel initialisation, and the AMP threshold level choice.

where $\mathbf{W} \in \mathbb{R}_+^{n \times n}$ is the diagonal matrix with the entries of $\mathbf{w}$ on its diagonal.

In order to align the feasibility constraint in the reconstruction algorithms that make use of iterative solvers, the LS optimisation methods use the feasibility constraint

$$||\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}||_2 < 10^{-6} \cdot ||\mathbf{y}||_2 \,, \tag{7.5}$$

which mimics the stop criterion constraint in (4.6) used in the iterative thresholding and AMP/GAMP methods.

In the implementation of the Douglas-Rachford splitting used in PyUNLocBoX to solve the LS optimisation problems, a relative tolerance stop criterion is used, i.e. the solution is accepted once the objective g (i.e. the ℓ_1-norm, weighted ℓ_1-norm, or TV criterion, depending on the algorithm) satisfies

$$\left| \frac{g(\boldsymbol{\alpha}_t) - g(\boldsymbol{\alpha}_{t\text{-}1})}{g(\boldsymbol{\alpha}_t)} \right| < 10^{-3} \,. \tag{7.6}$$

The GAMP in- and output channel parameters are estimated as part of the GAMP iteration using EM. Thus, they must be initialised in a reasonable way. It is our experience that the GAMP EM procedure performs the best when the parameters are initialised to values that allow for a "slack" in the algorithm. That is, the parameters should be initialised to include "most" solutions such that the algorithm may tune itself towards a specific solution. Towards that end, we initialise the AWGN output channel noise variance to

$$\sigma_0^2 = 1 \,, \tag{7.7}$$

whereas Bernoulli-Laplace input channel parameters are initialised to

$$\tau_0 = \frac{\delta \rho_{\mathrm{SE}}(\delta)}{2\frac{1}{n}\sum_{j=1}^{n} w_j} \tag{7.8}$$

$$\mu_0 = \mathrm{mdn}(\check{\boldsymbol{\alpha}}) \tag{7.9}$$

$$\lambda_0 = \frac{1}{10 \cdot \frac{1}{q}\sum_{l=1}^{q} |\check{\boldsymbol{\alpha}}_l - \mu_0|} \,, \tag{7.10}$$

where q is the number of elements in $\check{\boldsymbol{\alpha}}$, $\mathrm{mdn}(\cdot)$ denotes the median, and $\delta \cdot \rho_{\mathrm{SE}}(\delta)$ is the theoretical LASSO phase transition from [40]. Thus, the signal density initialisation is based on a "slacked" version of the theoretical LASSO phase transition and the Laplace parameters are initialised based on the Laplace distribution maximum likelihood values of $\check{\boldsymbol{\alpha}}$ - but with a slack on the rate parameter λ_0.

For the AMP algorithm, we use the median based threshold level update suggested in [36]. That is, we use an iteration specific threshold level $\theta\hat{\tau}_t$ where θ is a tuning parameter that we set to the minimax optimal value as detailed in [40] and

$$\hat{\tau}_t = \frac{1}{\Phi_{\mathcal{N}}^{-1}(0.75)} \cdot \mathrm{mdn}(\boldsymbol{\chi}_t) \,, \tag{7.11}$$

$$\tag{7.12}$$

where $\Phi_{\mathcal{N}}^{-1}(\cdot)$ is the inverse CDF of a zero-mean, unit variance Gaussian random variable. We initialise $\hat{\tau} = 1$.

For all our simulations we used the Anaconda[3] Python distribution[4] version 4.3.0 based on Python 3.6. Version 0.18.1 of SciPy was used for the interpolation implementations.

[3] The Anaconda Python distribution is freely available at http://www.continuum.io/anaconda

[4] A detailed list of all used Python packages and their version is stored as part of the annotations in the results database

For the implementation of the optimisation methods, we used a pre-release of PyUNLocBox[5] since the latest release does not include a proximal operator for TV. Release 1.7.0 of the Magni Python Package was used for the iterative thresholding and AMP/GAMP implementations. Our actions taken towards ensuring correctness and reproduciblity of the results are detailed along with Magni in Chapter 6. All computations where done using double precision floating point representations of decimal numbers. We used a compute server featuring two Intel Xeon E5-2697V2 CPUs and 384 GiB RAM and which was running Ubuntu 14.04.3 LTS.

7.3.2 Examples of Typical Reconstructions

Two sets of typical reconstructions of undersampled AFM images from our simulation study are depicted in Figures 7.2 and 7.3. These two figures are part of a larger set of examples based on all combinations of the four sampling patterns, four different undersampling ratios, and four different AFM images. A subset of this larger set of examples is displayed in Dataset H whereas the full set of examples is part of the "Extra Figures" supplementary material available at doi:10.5278/252861471. See also the reconstruction examples in [22, (Paper A)] and [27, (Paper B)].

The two figures (7.2 and 7.3) have been selected to highlight the typical reconstruction artefacts that occur when using a given combination of reconstruction algorithm and sampling pattern. Also, the figures serve to illustrate the strong visual difference that may exist between reconstructions of nearly the same PSNR/SSIM. This difference should be kept in mind when comparing the PSNR/SSIM results presented in Section 7.3.3. We note the following typical visual artefacts introduced by the different reconstruction algorithms (when reconstruction is successful):

Linear / cubic interpolation: Mild blurring and/or a perceived stretching of the reconstructed image.

Nearest neighbour interpolation: Pixelation.

TV LS: Smoothing of individual smaller areas in the image.

ℓ_1 **LS / Weighted** ℓ_1 **LS:** Introduction of noise that makes the image look "grainy".

IST / w-IST / w-IHT: Heavy blurring and/or expressions of the sampling pattern.

GAMP: Mild blurring and/or introduction of "grainy" noise.

AMP: Heavy blurring.

7.3.3 Reconstruction Performance Results

The reconstruction results constitute a six dimensional data set consisting of the choices of image, sampling pattern, reconstruction algorithm, undersampling ratio, reconstruction quality indicator, and reconstruction algorithm specific parameter (the sparsity level for the iterative thresholding algorithms). In order to reduce the dimensionality to the point where the results may be visualised, we average the PSNR/SSIM over the choice of image and maximise it over the reconstruction specific parameter (i.e. for the iterative thresholding methods, we pick only the sparsity levels that yield the highest PSNR/SSIM). This reduces the results to four dimensions. We then visualise the PSNR and SSIM results vs undersampling ratio in separate figures which leaves only handling the choices of sampling

[5]Specifically, we used the code from `https://github.com/epfl-lts2/pyunlocbox`, master branch, tag: v0.2.1-211-g585027a.

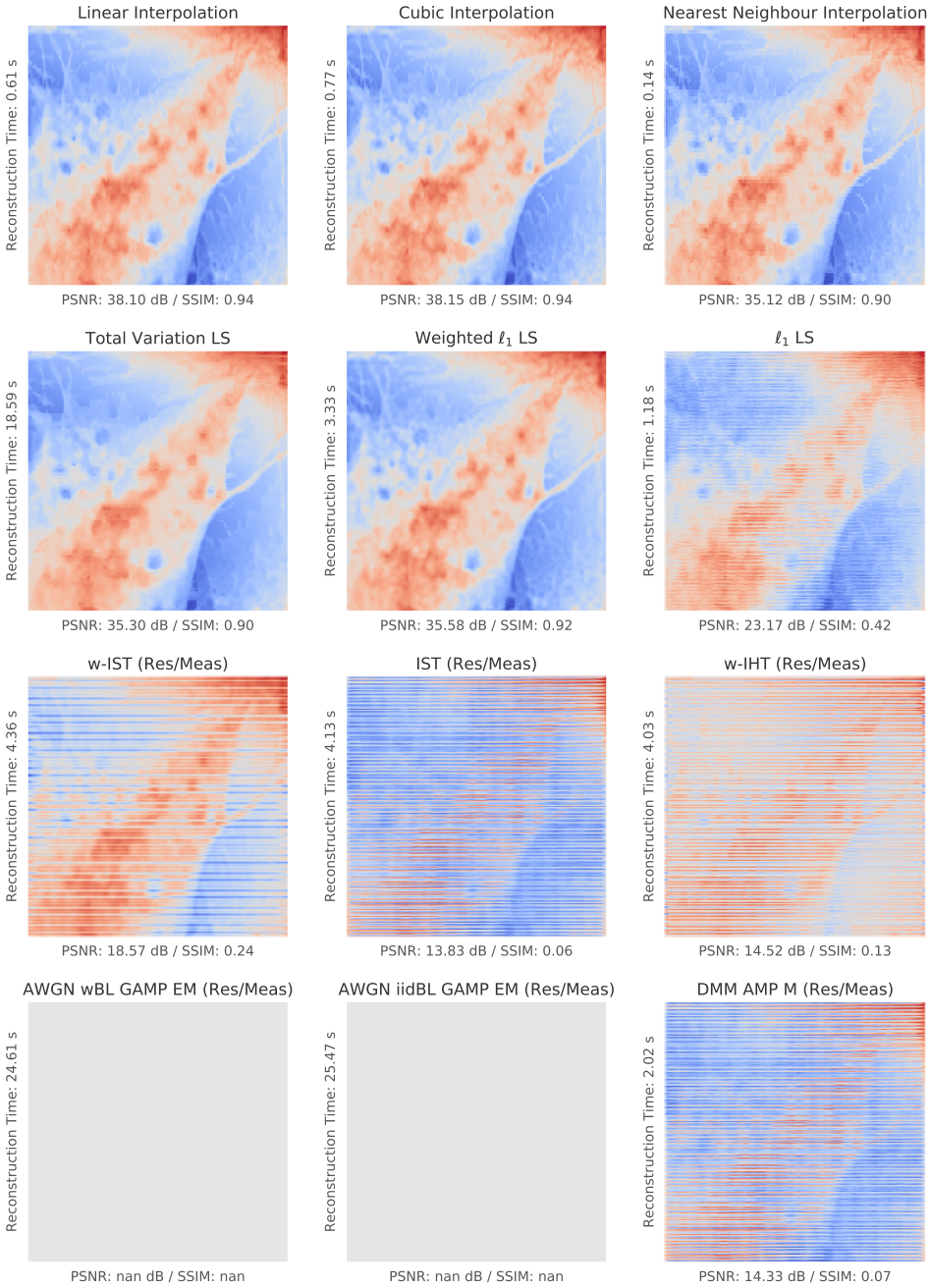


Figure 7.2: Typical reconstructions of the first AFM image shown in Figure 7.1 when using uniform line sampling and an undersampling ratio of $\delta = 0.15$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

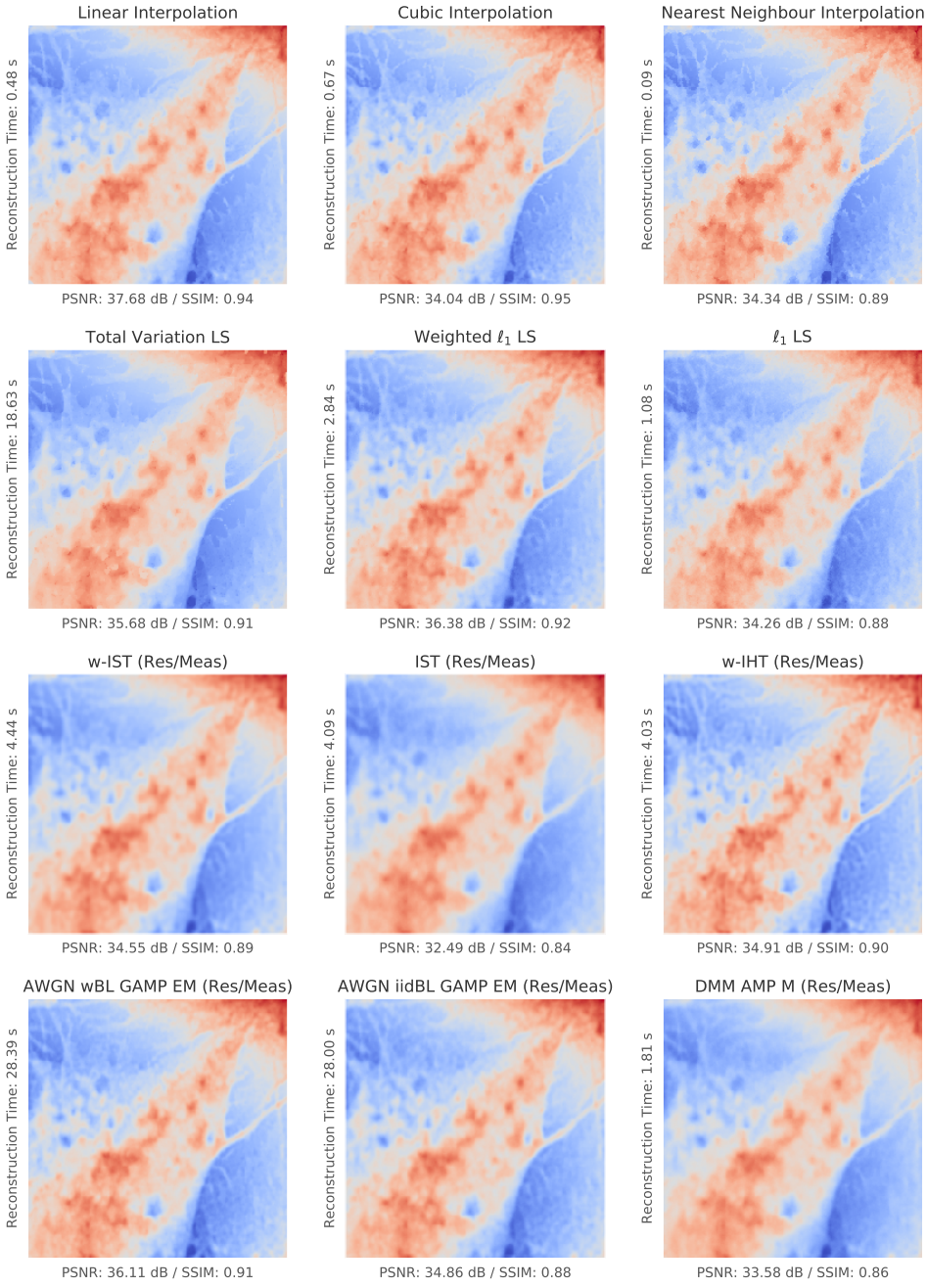


Figure 7.3: Typical reconstructions of the first AFM image shown in Figure 7.1 when using random pixels sampling and an undersampling ratio of $\delta = 0.15$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

pattern and reconstruction algorithm. In order to allow for easy comparison of both sampling patterns and reconstruction algorithms, we display two sets of figures: In Figures 7.5 and 7.6, we overlay the reconstruction algorithms and facet the sampling patterns, i.e. the PSNR/SSIM performance vs δ is displayed simultaneously for all algorithms in a sub-figure with separate sub-figures for each of the sampling patterns. This allows for easy comparison of the reconstruction algorithms. In Figures 7.7 and 7.8, we overlay the sampling patterns and facet the reconstruction algorithms, i.e. the PSNR/SSIM performance vs δ is displayed simultaneously for all sampling patterns in as sub-figure with separate sub-figures for each of the reconstruction algorithms which allows for easy comparison of the sampling patterns.

As discussed in Section 7.3, the ratio between the AFM undersampling ratio δ and the pixel undersampling ratio ι may vary significantly with the choice of sampling pattern. Thus, in order to also allow for an assessment of sampling pattern PSNR/SSIM vs ι performance, these results are shown in Figures 7.9 and 7.10, respectively. Additionally, the relations of ι vs δ for each of the sampling patterns is displayed in Figure 7.4.

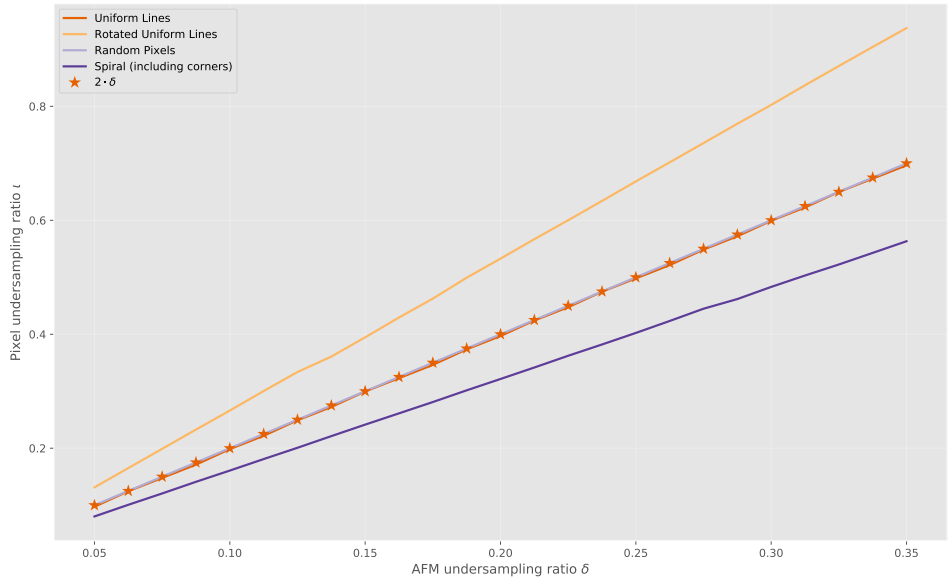


Figure 7.4: Comparison of sampling pattern pixel undersampling ratio (ι) vs AFM undersampling ratio (δ). The random pixels sampling has been designed to include pixels corresponding to $\iota = 2\delta$ by virtue of the definitions of δ and ι in Definitions 1 and 2, respectively. The uniform lines sampling matches this relationship closely with the exception of small deviations due to truncated lines. The rotated uniform lines sampling "overshoots" in terms of the number of included pixels whereas the spiral with included corners sampling "undershoots". The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

From Figures 7.5 and 7.6 we find that both in terms of PSNR and SSIM, the interpolation methods generally provide the best reconstructions. The results are more mixed for the CS algorithms depending on the sampling pattern and undersampling ratio. The AMP/GAMP algorithms generally fall behind the other methods, though. The optimisation approaches based on TV and weighted ℓ_1 generally perform well with PSNR/SSIM values nearly matching those of the interpolation methods for all sampling patterns. The w-IST algorithm performs particularly well with the random pixels sampling but degrades for low undersampling ratios with the structured sampling patterns.

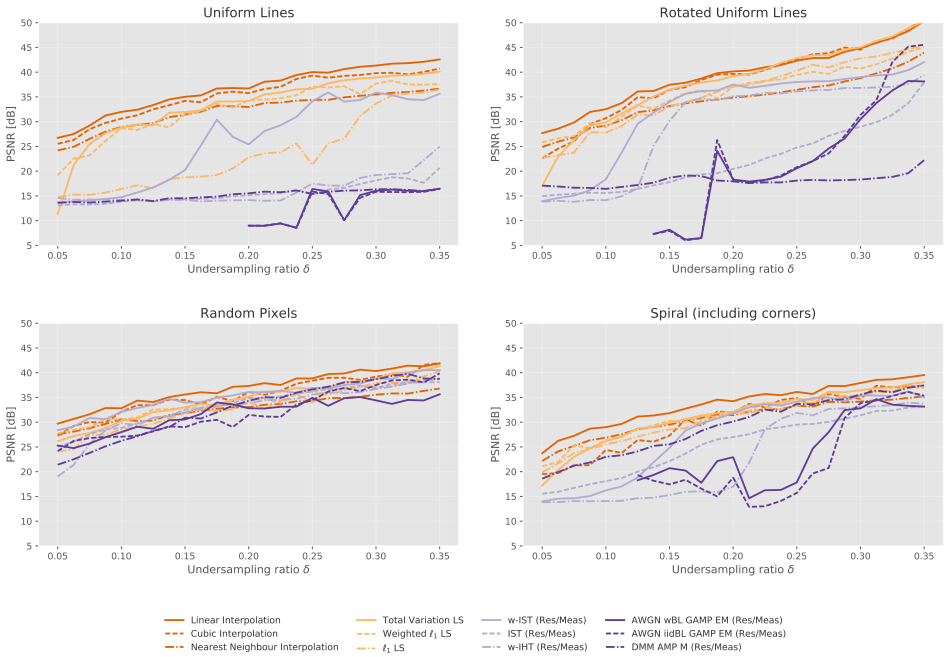


Figure 7.5: Comparison of reconstruction algorithms in terms of average PSNR versus AFM undersampling ratio (δ). The results for all the reconstruction algorithms are overlaid in a facet plot based on the sampling patterns. The corresponding SSIM results are shown in Figure 7.6. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

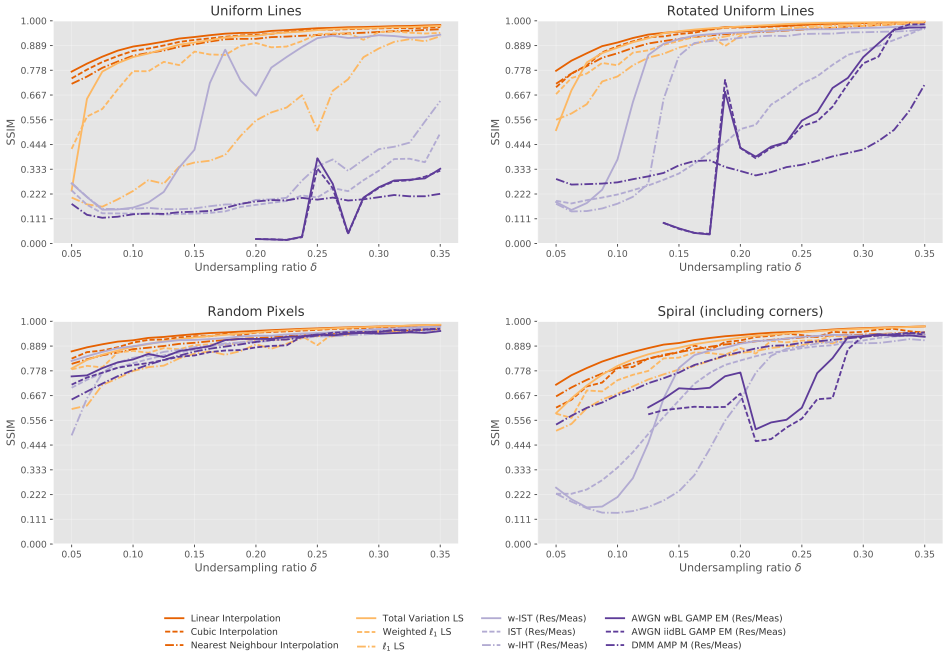


Figure 7.6: Comparison of reconstruction algorithms in terms of average SSIM versus AFM undersampling ratio (δ). The results for all the reconstruction algorithms are overlaid in a facet plot based on the sampling patterns. The corresponding PSNR results are shown in Figure 7.5. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

Looking at Figures 7.7 and 7.8, we find that the rotated lines sampling pattern gives excellent PSNR/SSIM performance when used with the interpolation and optimisation based reconstruction algorithms. However, comparing with Figures 7.9 and 7.10, it is clear that the results are more close when considering PSNR/SSIM versus pixel undersampling ratio rather than versus AFM undersampling ratio. Most interesting, though, from Figures 7.9 and 7.10 (and also partly 7.7 and 7.8) is that all the reconstruction algorithms perform the best when random pixel sampling is used - especially in the high undersampling setting, i.e. for small values of δ (or ι). This is, in particular, true for the CS reconstruction algorithms.

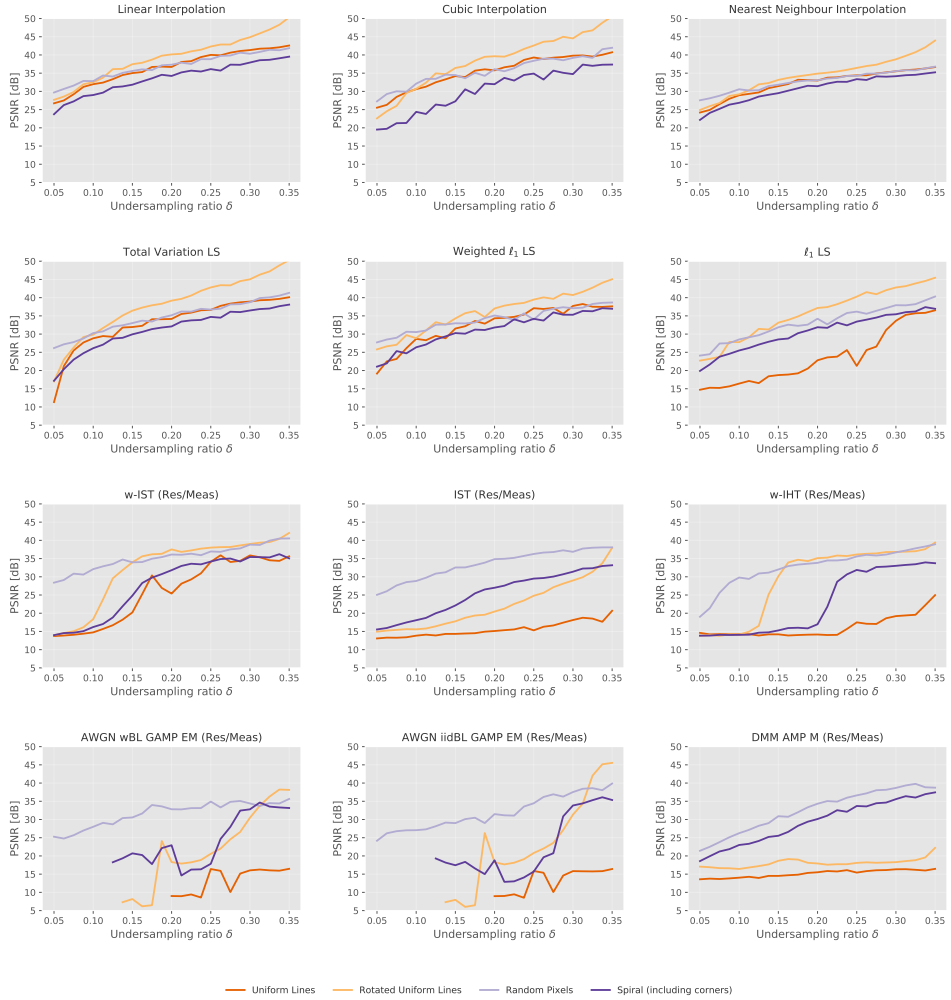


Figure 7.7: Comparison of sampling patterns in terms of average PSNR versus AFM undersampling ratio (δ). The results for all the sampling patterns are overlaid in a facet plot based on the reconstruction algorithms. The corresponding SSIM results are shown in Figure 7.8. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

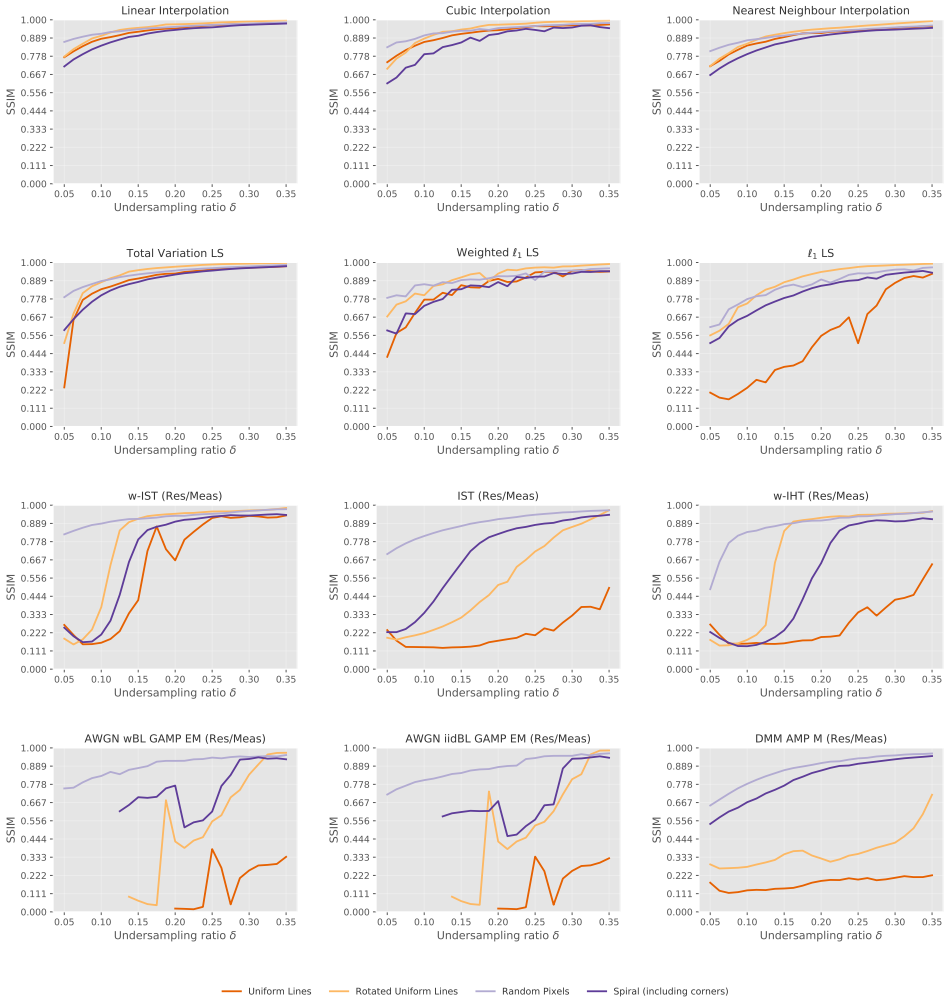


Figure 7.8: Comparison of sampling patterns in terms of average SSIM versus AFM undersampling ratio (δ). The results for all the sampling patterns are overlaid in a facet plot based on the reconstruction algorithms. The corresponding PSNR results are shown in Figure 7.7. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

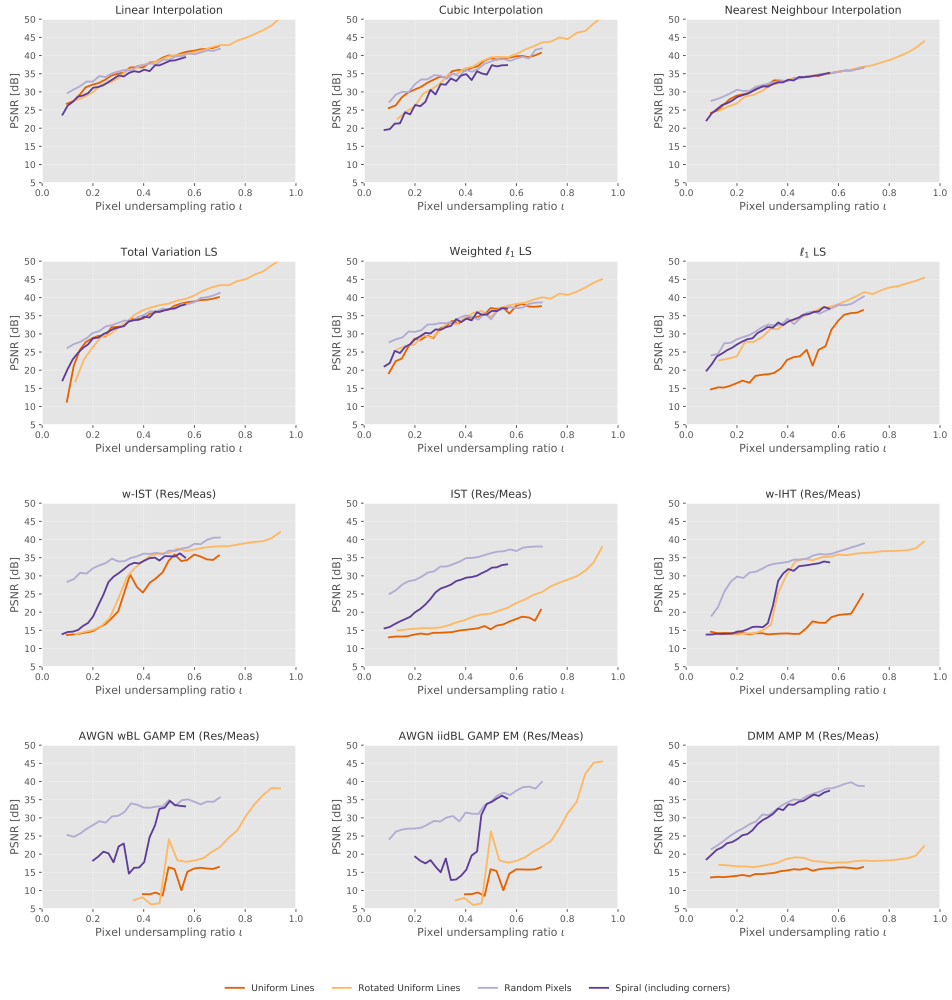


Figure 7.9: Comparison of sampling patterns in terms of average PSNR versus pixel undersampling ratio (ι). The results for all the sampling patterns are overlaid in a facet plot based on the reconstruction algorithms. The corresponding SSIM results are shown in Figure 7.10. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

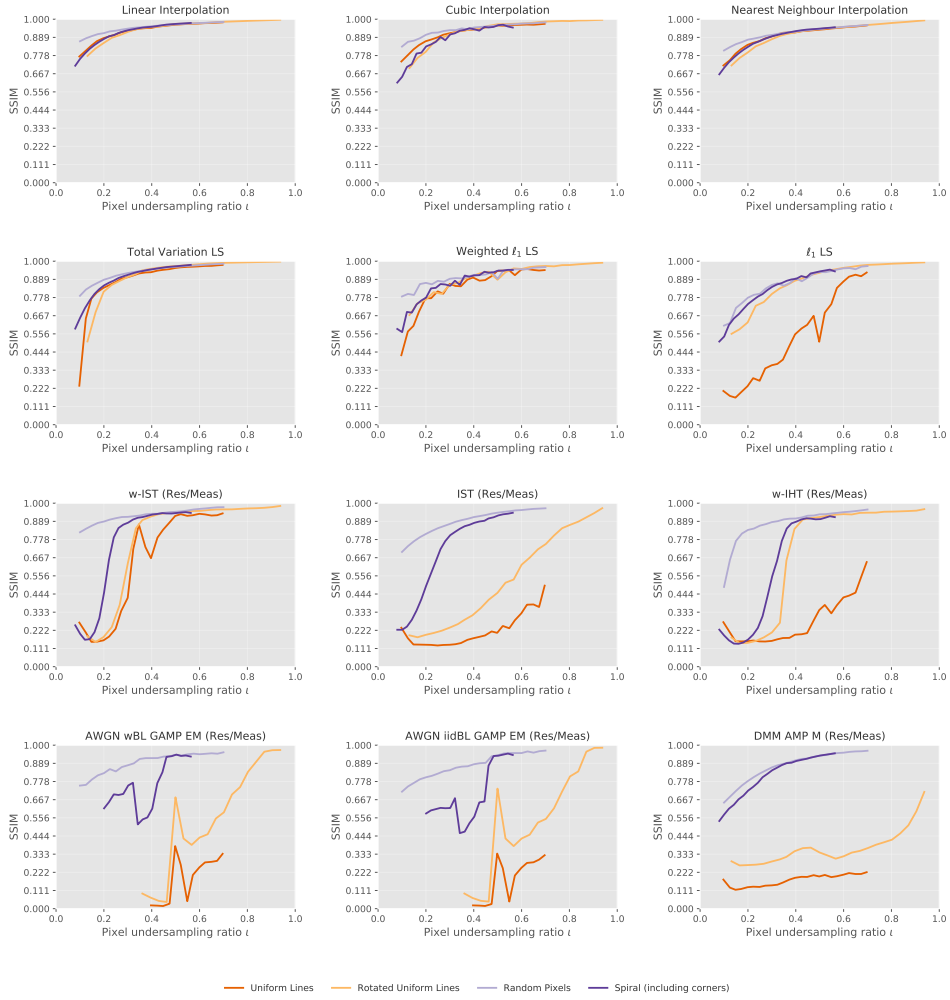


Figure 7.10: Comparison of sampling patterns in terms of average SSIM versus pixel undersampling ratio (ι). The results for all the sampling patterns are overlaid in a facet plot based on the reconstruction algorithms. The corresponding PSNR results are shown in Figure 7.9. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

8 Discussion

As outlined in Chapter 1, our overall goal is to obtain state-of-the-art reconstructions of undersampled AFM images using algorithms that exploit a structured sparse model of AFM images. Towards that end, we have presented two families of reconstruction algorithms that are applicable to high dimensional image reconstruction problems and allow for exploiting a structured sparse model of the images in the reconstruction: 1. Weighted iterative thresholding methods detailed in Chapter 4, and 2. Generalised approximate message passing using a general weighted sparse prior detailed in Chapter 5.

Results from a large simulation study show an improvement in reconstruction quality in terms of average PSNR and SSIM obtained using our proposed weighted sparse reconstruction methods when compared to their baseline equivalents. More surprisingly, our simulation results also show that these advanced CS based reconstruction methods are inferior to the more simple interpolation and TV methods. As is evident from Figures 7.5 and 7.6, none of our proposed methods provide higher average PSNR and SSIM than the interpolation and TV methods in any of our tested settings. The major cause for this somewhat unexpected result is likely to be found in the choice of sampling pattern. The difference in average PSNR and SSIM between the CS and interpolation methods is significantly smaller when using random pixels sampling than it is using any of the other more structured sampling patterns. This ultimately questions if CS based reconstruction methods are at all applicable to the undersampled AFM image reconstruction problem which is physically constrained to the use of (at least somewhat) structured sampling patterns. Even with algorithmic improvements such as the incorporation of more accurate noise and acquisition impairment models, the use of even more advanced structured models of the images, or the use of dictionary learning or parameter tuning strategies, it seems unlikely that the CS methods come out on top unless more random AFM applicable sampling patterns are used. Potentially, an AFM undersampling pattern may have to be designed specifically to match a certain choice of reconstruction algorithm. Such a design might also need to incorporate more sophisticated mappings of the continuous surface samples to the discrete grid used in the digital image reconstruction and representation.

A study on undersampling in SEM [31] report simulated reconstruction results similar to our AFM results. That is, interpolation and TV reconstruction methods are well suited for the undersampled SEM application. Also interesting to note from the results in [31] is that the interpolation methods break down when reconstructing SEM images from noisy measurements. This result suggests that the CS based reconstruction methods may have more merit in the SEM and AFM applications if the measurements are noisy. Our simulations are based on AFM images that are assumed noiseless but a practical AFM undersampling setup may produce images with significantly more measurement noise. This, however, needs further investigation on its own. In two other studies, CS based reconstruction methods are used together with a random line segment sampling pattern in an undersampled AFM application [25], [30]. The results reported in those studies are somewhat varying but for some images the simulated reconstruction quality in terms of PSNR come close to our simulation results using random pixel sampling. This result suggests that improvements in reconstruction quality may be obtained from using more advanced sampling patterns such as randomly chosen line segments [25], a double archimedian spiral [23], or lissajous patterns [14]. However, common to all of these patterns is the need for choosing pattern specific parameters. Unfortunately, there seems to be no clear recommendation towards such choices. An option is to empirically measure the performance for a large range of parameters. However, when combined with the choices of reconstruction algorithm parameters, the resulting parameter space to explore is of a high dimension and

likely difficult to comprehend, thus, making this option less appealing.

The use of undersampling in AFM applications first of all has to be useful to the practitioner. All reconstructions of undersampled AFM images suffer from some sort of impairments and artefacts stemming from the reconstruction process. Some of these artefacts, listed in Section 7.3.2, may be more critical to the practitioner than others. Image reconstruction quality indicators such as PSNR and SSIM do not necessarily express such differences in the importance of the introduced artefacts. As is evident from Figures 7.2, 7.3, and H.1 – H.10, different reconstructions with comparable PSNR or SSIM may have significantly different visual expressions. Thus, the premise that all artefacts are equally important, which underlies our comparison of algorithms based on PSNR and SSIM, may not accurately reflect all practical use cases. This should be kept in mind when assessing the comparisons of both reconstruction algorithms in Figures 7.5 and 7.6 and comparisons of sampling patterns in Figures 7.7, 7.8, 7.9, and 7.10.

Our comprehensive simulation study has been designed to serve as a guideline to practitioners interested in using undersampling in their AFM applications. We have tested a significant number of relevant undersampling settings and made all the results available for mining as detailed in Table 6.1. The simulations have been obtained using best practices for ensuring correctness and reproducibility of computational results. Though our proposed methods for incorporating these best practices into our scientific Python workflow significantly strengthens the trustworthiness our results, we do acknowledge that such empirical simulation studies are sensitive to errors in the experiments design and implementation. However, given all our actions towards ensuring correctness and reproducibility of our results, we believe our results to generalise. Furthermore, we believe that our methods and results are both inspectable and reproducible to a high extent.

9 Conclusions

We have shown that it is possible, for a fixed number of measurements, to use statistical information in a structured sparse model of AFM images to obtain superior (in terms of average PSNR and SSIM) reconstructions of undersampled AFM images when compared to CS sparsity only exploiting reconstruction methods, thus, confirming our main hypothesis from Chapter 1. However, our empirical performance evaluation also shows that baseline reconstruction methods such as interpolation or TV methods yield reconstructions with higher average PSNR and SSIM than the CS sparsity exploiting methods. We believe that the main reason for this somewhat surprising inferior performance of the CS algorithms is the discrepancy between the assumed random measurement model of the CS algorithms and the fixed structured measurements process used in AFM. That being said, our empirical performance evaluation also reveals that different reconstruction algorithms introduce visually different artefacts in the reconstructed images making it difficult to unconditionally recommend one reconstruction method over the others. When using undersampling in AFM, the practitioner must consider which reconstruction method best highlights the elements of interest in the reconstructed image.

The significant reconstruction performance issues, we observe when using fixed structured sampling patterns, suggest that future research in undersampling of AFM images should focus on a joint design of sampling pattern and reconstruction algorithm. Any such design likely also needs to include elements of a consistent mapping of the continuous surface measurements to the fixed grid used in the reconstruction process. Possibly, further improvements to reconstruction algorithms such as the incorporation of more accurate prior information or noise models may lead to improved reconstructions. However, given the solid, though still moderate, improvements we have seen with our proposed reconstruction methods, it seems that the sampling pattern design issue is by far the most critical element to investigate.

Our main contributions are associated with two main areas of research: 1. Practically applicable reconstruction methods that exploit structured signal sparsity, and 2. Methods and guidelines for implementing best practices to ensuring correctness and reproducibility of computational results in a scientific Python workflow. Our proposed structure exploiting reconstruction methods are suitable for use in more general high dimensional signal reconstruction problems and have sufficiently low computational and memory requirements to make them practically feasible. Our proposed input validation framework significantly reduces the risk of obtaining erroneous results due to falsely formatted input data. Finally, our proposed scheme for storing reproducibility metadata along with computational results aids in making computational results easily inspectable and reproducible.

List of Acronyms

AFM	atomic force microscopy	3–5, 7–9, 11–17, 19, 21, 23, 25–32, 34, 39–42, 46–51, 55–57, 60, 213–222
AMP	approximate message passing	16, 23–26, 44–46, 49, 60
AWGN	additive white Gaussian noise	11, 23, 25, 31, 32, 44, 45, 60
BPDN	basis pursuit denoising	16
CDF	cumulative distribution function	31, 45, 60
CS	compressed sensing	3, 11–16, 19, 27, 42, 49, 51, 55, 57
DCT	discrete cosine transform	14–17, 19, 21, 25, 27, 29, 30, 42, 43
DWT	discrete wavelet transform	14
EM	expectation maximization	25, 31, 32, 44, 45
GAMP	generalized approximate message passing	21, 23–27, 29, 31, 32, 43–47, 49, 213, 216, 220–222
GWS	general weighted sparse	25, 29, 43
IHT	iterative hard thresholding	19–21
i.i.d.	independent and identically distributed	13, 25, 26, 44
IST	iterative soft thresholding	20, 21, 25, 44, 46
ISTA	iterative shrinkage-thresholding algorithm	20
JSON	JavaScript Object Notation	37
LASSO	least absolute shrinkage and selection operator	12, 16, 20, 21, 25, 45
LS	least squares	16, 43–46
MAP	maximum a posteriori	23, 25, 26
ML	maximum likelihood	32
MMSE	minimum mean squared error	23–26, 29
PDF	probability density function	30, 31, 60
PSNR	peak signal-to-noise ratio	39, 41, 42, 46, 49–57
RIP	restricted isometry property	12
SE	state evolution	23, 26
SEM	scanning electron microscopy	9, 55
SPM	scanning probe microscopy	3, 9
SRM	structurally random matrices	13, 27
SSIM	structural similarity index measure	39, 41, 42, 46, 49–57
TV	total variation	16, 45, 46, 49, 55, 57
wBL	weighted sparse Bernoulli-Laplace	29, 32, 44
w-IHT	weighted iterative hard thresholding	21, 29, 44, 46
w-IST	weighted iterative soft thresholding	21, 29, 44, 46, 49

List of Symbols

$\mathbb{N}$	The natural numbers
$\mathbb{Z}$	The integers
$\mathbb{R}$	The real numbers
$\mathbb{R}_+$	The non-negative real numbers
$\mathbb{C}$	The complex numbers
$\mathrm{h} \in \mathbb{N}$	Image height in pixels
$\mathrm{w} \in \mathbb{N}$	Image width in pixels
$p \in \mathbb{N}$	Number of image pixels
$n \in \mathbb{N}$	Number of dictionary coefficients
$m \in \mathbb{N}$	Number of measurements
$k \in \mathbb{N}$	Number of non-zeros
$L \in \mathbb{R}$	AFM sampling path length
$L_{\mathrm{ref}} \in \mathbb{R}$	AFM raster reference path length
$\mathbf{X} \in \mathbb{R}^{\mathrm{w} \times \mathrm{h}}$	Image as a matrix
$\hat{\mathbf{X}} \in \mathbb{R}^{\mathrm{w} \times \mathrm{h}}$	Estimate of image as a matrix
$\mathbf{x} \in \mathbb{R}^{p \times 1}$	Image as a vector
$\hat{\mathbf{x}} \in \mathbb{R}^{p \times 1}$	Estimate of image as a vector
$\boldsymbol{\alpha} \in \mathbb{C}^{n \times 1}$	Dictionary coefficients
$\hat{\boldsymbol{\alpha}} \in \mathbb{C}^{n \times 1}$	Estimate of dictionary coefficients
$\mathbf{z} \in \mathbb{R}^{m \times 1}$	Noiseless measurements
$\mathbf{y} \in \mathbb{R}^{m \times 1}$	Noisy measurements
$\mathbf{e} \in \mathbb{R}^{m \times 1}$	Additive noise
$\mathbf{w} \in \mathbb{R}_+^{n \times 1}$	Weights vector
$\mathbf{W} \in \mathbb{R}_+^{n \times n}$	Weights (diagonal) matrix
$\boldsymbol{\Phi} \in \mathbb{R}^{m \times p}$	Measurement matrix
$\boldsymbol{\Psi} \in \mathbb{C}^{p \times n}$	Dictionary matrix
$\mathbf{A} \in \mathbb{C}^{m \times n}$	System matrix
$\mathbf{A}^{\mathsf{T}} \in \mathbb{C}^{n \times m}$	System matrix transposed
$\mathbf{A}^{\mathsf{H}} \in \mathbb{C}^{n \times m}$	System matrix conjugated transposed
$\mathbf{A}^{\circ 2} \in \mathbb{C}^{m \times n}$	Entrywise squared system matrix
$\lvert\mathbf{A}\rvert^{\circ 2} \in \mathbb{R}^{m \times n}$	Entrywise absolute value squared system matrix
$\mathbf{D} \in \mathbb{C}^{n \times n}$	Diagonal matrix
$\mathbf{D}_\Omega \in \mathbb{C}^{m \times n}$	Sub-sampled diagonal matrix
$\boldsymbol{\chi} \in \mathbb{R}^{m \times 1}$	AMP Onsager corrected residual
$\mathbf{0}_q \in \mathbb{R}^{q \times 1}$	Zero vector
$\mathbf{1}_q \in \mathbb{R}^{q \times 1}$	One vector
$\boldsymbol{\theta}$	Parameter vector
$\mathbf{I}$	Identity matrix
$\delta \in (0, 1)$	AFM undersampling ratio
$\rho \in (0, 1)$	Sparsity level
$\tau \in (0, 1)$	Signal density
$\iota \in (0, 1)$	Pixel undersampling ratio
$\kappa \in \mathbb{R}_+$	Step-size
$\epsilon \in \mathbb{R}_+$	Tolerance
$t \in \mathbb{N}$	Iteration number
$T_{\max} \in \mathbb{N}$	Max iterations
$\beta \in \mathbb{R}_+$	Regularisation parameter
$\sigma^2 \in \mathbb{R}_+$	AWGN noise power
$\bar{\theta} \in \mathbb{R}$	Gaussian distribution mean
$\tilde{\theta} \in \mathbb{R}_+$	Gaussian distribution variance
$\mu \in \mathbb{R}$	Laplace distribution mean
$\lambda \in \mathbb{R}_+$	Laplace distribution rate
$\mathcal{Z} \in \mathbb{R}$	Distribution normalisation const.
$\varphi(\cdot)$	Arbitrary PDF
$\mathcal{N}(\cdot)$	Gaussian PDF
$\phi_{\mathcal{N}}(\cdot)$	Standard normal PDF
$\Phi_{\mathcal{N}}(\cdot)$	Standard normal CDF
$\Phi_{\mathcal{N}}^{-1}(\cdot)$	Inverse standard normal CDF
$\mathbb{E}[\cdot]$	Expectation
$\mathrm{Var}(\cdot)$	Variance
$\delta_{\mathrm{Dirac}}(\cdot)$	Dirac delta function
$\eta_t(\cdot)$	Threshold operator
t	Threshold level
$\mathcal{O}(\cdot)$	Big-O notation

List of Figures

List of Tables

References

[1] M. Harrower and C. A. Brewer, "ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps," *The Cartographic Journal*, vol. 40, no. 1, pp. 27–37, Jun. 2003. doi:10.1179/000870403235002042

[2] K. Moreland, "Diverging Color Maps for Scientific Visualization," in *Advances in Visual Computing: 5th International Symposium, ISVC*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, Eds. Las Vegas, NV, USA,: Springer Berlin Heidelberg, Nov. 30 – Dec. 2 2009, vol. 5876, pp. 92–103, proceedings Part II. doi:10.1007/978-3-642-10520-3_9

[3] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, "A Tutorial on the Mechanisms, Dynamics, and Control of Atomic Force Microscopes," in *American Control Conference*, New York City, USA, Jul. 11-13, 2007, pp. 3488–3502. doi:10.1109/ACC.2007.4282300

[4] B. Bhushan and O. Marti, "Scanning Probe Microscopy – Principle of Operation, Instrumentation, and Probes," in *Springer Handbook of Nanotechnology*, B. Bhushan, Ed. Springer Berlin Heidelberg, 2010, ch. 21, pp. 573–617. doi:10.1007/978-3-642-02525-9_21

[5] D. P. Allison, N. P. Mortensen, C. J. Sullivan, and M. J. Doktycz, "Atomic force microscopy of biological samples," *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology*, vol. 2, no. 6, pp. 618–634, Nov. 2010. doi:10.1002/wnan.104

[6] J. Loos, "The Art of SPM: Scanning Probe Microscopy in Materials Science," *Advanced Materials*, vol. 17, no. 15, p. 1821–1833, Aug. 2005. doi:10.1002/adma.200500701

[7] Z. J. Davis, G. Abadal, O. Hansen, X. Borisé, N. Barniol, F. Pérez-Muranob, and A. Boisena, "AFM lithography of aluminum for fabrication of nanomechanical systems," *Ultramicroscopy*, vol. 97, no. 1-4, pp. 467–472, Nov. 2003. doi:10.1016/S0304-3991(03)00075-5

[8] Y. K. Yong, A. Bazaei, S. O. R. Moheimani, and F. Allgöwer, "Design and Control of a Novel Non-Raster Scan Pattern for Fast Scanning Probe Microscopy," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Kachsiung, Taiwan, Jul., 11 – 14, 2012, pp. 456–461. doi:10.1109/AIM.2012.6266062

[9] G. Agarwal and T. M. Nocera, *The Nanobiotechnology Handbook*. CRC Press, 2012, ch. Atomic Force Microscopy, pp. 369–392.

[10] P. K. Hansma, G. Schitter, G. E. Fantner, and C. Prater, "High-Speed Atomic Force Microscopy," *Science*, vol. 314, no. 5799, pp. 601–602, Oct. 2006. doi:10.1126/science.1133497

[11] B. Song, N. Xi, R. Yang, K. W. C. Lai, and C. Qu, "Video Rate Atomic Force Microscopy (AFM) Imaging using Compressive Sensing," in *11th IEEE International Conference on Nanotechnology*, Portland, Oregon, USA, Aug. 15-18, 2011, pp. 1056–1059. doi:10.1109/NANO.2011.6144587

[12] A. Chen, A. L. Bertozzi, P. D. Ashby, P. Getreuer, and Y. Lou, "Enhancement and Recovery in Atomic Force Microscopy Images," in *Excursions in Harmonic Analysis*, ser. Applied and Numerical Harmonic Analysis„ T. D. Andrews, R. Balan, J. J. Benedetto, W. Czaja, and K. A. Okoudjou, Eds. Birkhäuser Boston, 2013, vol. 2, pp. 311–332. doi:10.1007/978-0-8176-8379-5_16

[13] D. J. Müller and Y. F. Dufrêne, "Atomic force microscopy: a nanoscopic window on the cell surface," *Trends in Cell Biology*, vol. 21, no. 8, pp. 461–469, Aug. 2011. doi:10.1016/j.tcb.2011.04.008

[14] T. Tuma, J. Lygeros, V. Kartik, A. Sebastian, and A. Pantazi, "High-speed multiresolution scanning probe microscopy based on Lissajous scan trajectories," *Nanotechnology*, vol. 23, no. 18, p. 9, Apr. 2012. doi:10.1088/0957-4484/23/18/185501

[15] L. Zhu, W. Zhang, D. Elnatan, and B. Huang, "Faster STORM using compressed sensing," *Nature Methods*, vol. 9, no. 7, pp. 721–723, Jul. 2012. doi:10.1038/nmeth.1978

[16] P. Sen and S. Darabi, "Compressive Image Super-resolution," in *Forty Third Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 1 – 4, 2009, pp. 1235–1242. doi:10.1109/ACSSC.2009.5469968

[17] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006. doi:10.1109/TIT.2006.871582

[18] E. J. Candès, J. Romberg, and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction From Highly Incomplete Frequency Information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006. doi:10.1109/TIT.2005.862083

[19] E. J. Candès and M. B. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008. doi:10.1109/MSP.2007.914731

[20] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, Jul. 2007. doi:10.1109/MSP.2007.4286571

[21] Y. C. Eldar and G. Kutyniok, Eds., *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[22] T. Arildsen, C. S. Oxvig, P. S. Pedersen, J. Østergaard, and T. Larsen, "Reconstruction Algorithms in Undersampled AFM Imaging," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 1, pp. 31–46, Feb. 2016. doi:10.1109/JSTSP.2015.2500363

[23] T. R. Meyer, D. Ziegler, C. Brune, A. Chen, R. Farnham, N. Huynh, J.-M. Chang, A. L. Bertozzi, and P. D. Ashby, "Height drift correction in non-raster atomic force microscopy," *Ultramicroscopy*, vol. 137, pp. 48–54, Feb 2014. doi:10.1016/j.ultramic.2013.10.014

[24] P. I. Chang, P. Huang, J. Maeng, and S. B. Andersson, "Local raster scanning for high-speed imaging of biopolymers in atomic force microscopy," *Review of Scientific Instruments*, vol. 82, no. 6, pp. 1–8, Jun. 2011. doi:10.1063/1.3600558

[25] B. D. Maxwell and S. B. Andersson, "A Compressed Sensing Measurement Matrix For Atomic Force Microscopy," in *American Control Conference (ACC)*, Portland, Oregon, USA, Jun. 4 – 6, 2014, pp. 1631–1636. doi:10.1109/ACC.2014.6858710

[26] I. A. Mahmood, S. O. R. Moheimani, and B. Bhikkaji, "A New Scanning Method for Fast Atomic Force Microscopy," *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 203–216, Mar. 2011. doi:10.1109/TNANO.2009.2036844

[27] C. S. Oxvig, T. Arildsen, and T. Larsen, "Structure Assisted Compressed Sensing Reconstruction of Undersampled AFM Images," *Ultramicroscopy*, vol. 172, pp. 1–9, Jan. 2017. doi:10.1016/j.ultramic.2016.09.011

[28] S. B. Andersson and L. Y. Pao, "Non-Raster Sampling in Atomic Force Microscopy: A Compressed Sensing Approach," in *American Control Conference (ACC)*, Montréal, Canada, Jun. 27-29, 2012, pp. 2485–2490. doi:10.1109/ACC.2012.6315406

[29] T. L. Jensen, T. Arildsen, J. Østergaard, and T. Larsen, "Reconstruction of Undersampled Atomic Force Microscopy Images : Interpolation versus Basis Pursuit," in *International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, Kyoto, Japan, Dec. 2 – 5, 2013, p. 6. doi:10.1109/SITIS.2013.32

[30] Y. Luo and S. B. Andersson, "A Fast Image Reconstruction Algorithm For Compressed Sensing-Based Atomic Force Microscopy," in *American Control Conference (ACC)*, Chicago, Illinois, USA, Jul. 1 – 3, 2015, pp. 3503–3508. doi:10.1109/ACC.2015.7171873

[31] H. S. Anderson, J. Ilic-Helms, B. Rohrer, J. Wheeler, and K. Larson, "Sparse imaging for fast electron microscopy," in *Computational Imaging XI, Proceedings of the SPIE*, vol. 8657, Burlingame, California, USA, Feb. 3 2013, pp. (86 570C–1)–(86 570C–12). doi:10.1117/12.2008313

[32] P. Binev, W. Dahmen, R. DeVore, P. Lamby, D. Savu, and R. Sharpley, "Compressed Sensing and Electron Microscopy," in *Modeling Nanoscale Imaging in Electron Microscopy*, T. Vogt, W. Dahmen, and P. Binev, Eds. Springer, 2012, pp. 73–126. doi:10.1007/978-1-4614-2191-7_4

[33] R. Leary, Z. Saghi, P. A. Midgley, and D. J. Holland, "Compressed sensing electron tomography," *Ultramicroscopy*, vol. 131, pp. 70–91, Aug. 2013. doi:10.1016/j.ultramic.2013.03.019

[34] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[35] ——, "Regression shrinkage and selection via the lasso: a retrospective," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 73, no. 3, pp. 273–282, Jun. 2011. doi:10.1111/j.1467-9868.2011.00771.x

[36] A. Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, ch. 9, pp. 394–438.

[37] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast Image Recovery Using Variable Splitting and Constrained Optimization," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2345–2356, Sep. 2010. doi:10.1109/TIP.2010.2047910

[38] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, "Introduction to compressed sensing," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, ch. 1, pp. 1–64.

[39] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9–10, pp. 589–592, May 2008. doi:10.1016/j.crma.2008.03.014

[40] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, p. 18914–18919, Nov. 2009. doi:10.1073/pnas.0909892106

[41] T. Blumensath, M. E. Davies, and G. Rilling, "Greedy algorithms for compressed sensing," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, ch. 8, pp. 348–393.

[42] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013. doi:10.1109/TSP.2013.2272287

[43] H. Rauhut, "Compressive Sensing and Structured Random Matrices," in *Theoretical Foundations and Numerical Methods for Sparse Recovery*, ser. Radon Series on Computational and Applied Mathematics, M. Fornasier, Ed. De Gruyter, 2010, vol. 9, pp. 1–92.

[44] A. Maleki and D. L. Donoho, "Optimally Tuned Iterative Reconstruction Algorithms for Compressed Sensing," *IEEE Journal Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 330–341, Apr. 2010. doi:10.1109/JSTSP.2009.2039176

[45] T. Goldstein and S. Osher, "The Split Bregman Method for L1-Regularized Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, Apr. 2009. doi:10.1137/080725891

[46] K. R. Rao and P. C. Yip, Eds., *The Transform and Data Compression Handbook*, 1st ed. CRC Press, 2000.

[47] A. N. Akansu, R. A. Haddad, and P. R. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, 2nd ed. Academic Press, 2000.

[48] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

[49] D. L. Donoho and J. Tanner, "Precise Undersampling Theorems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 913–924, Jun. 2010. doi:10.1109/JPROC.2010.2045630

[50] T. T. Do, L. Gan, N. H. Nguyen, and T. D. Tran, "Fast and Efficient Compressive Sensing Using Structurally Random Matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 139–154, Jan. 2012. doi:10.1109/TSP.2011.2170977

[51] J. Romberg, "Imaging via Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, Mar. 2008. doi:10.1109/MSP.2007.914729

[52] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010. doi:10.1109/JPROC.2010.2040551

[53] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006. doi:10.1109/TSP.2006.881199

[54] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing.* Springer, 2010. doi:10.1007/978-1-4419-7011-4

[55] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, Jul. 2011. doi:10.1016/j.acha.2010.10.002

[56] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Elsevier, 2009.

[57] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan. 1974. doi:10.1109/T-C.1974.223784

[58] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 Still Image Compression Standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sep. 2001. doi:10.1109/79.952804

[59] G. K. Wallace, "The JPEG Picture Compression Standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. 18–34, Feb. 1992. doi:10.1109/30.125072

[60] L. He and L. Carin, "Exploiting Structure in Wavelet-Based Bayesian Compressive Sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 9, pp. 3488–3497, Sep. 2009. doi:10.1109/TSP.2009.2022003

[61] S. Som and P. Schniter, "Compressive Imaging Using Approximate Message Passing and a Markov-Tree Prior," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3439–3448, Jul. 2012. doi:10.1109/TSP.2012.2191780

[62] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 1, pp. 27–34, Feb. 1980. doi:10.1109/TASSP.1980.1163351

[63] G. Bi, G. Li, K.-K. Ma, and T. C. Tan, "On the Computation of Two-Dimensional DCT," *IEEE Transactions on Signal Processing*, vol. 48, no. 4, pp. 1171–1183, Apr. 2000. doi:10.1109/78.827550

[64] M. F. Duarte and Y. C. Eldar, "Structured Compressed Sensing: From Theory to Applications," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053–4085, Sep. 2011. doi:10.1109/TSP.2011.2161982

[65] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-Based Compressive Sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, Apr. 2010. doi:10.1109/TIT.2010.2040894

[66] L. P. Yaroslavsky, G. Shabat, B. G. Salomon, I. A. Ideses, and B. Fishbain, "Nonuniform sampling, image recovery from sparse data and the discrete sampling theorem," *J. Opt. Soc. Am. A*, vol. 26, no. 3, pp. 566–575, Mar. 2009. doi:10.1364/JOSAA.26.000566

[67] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer Berlin Heidelberg, 1997. doi:10.1007/978-3-662-03427-9

[68] S. K. Lodha and R. Franke, "Scattered data techniques for surfaces," in *Scientific Visualization Conference*, Dagstuhl, Germany, Jun. 9 – 13, 1997, pp. 1–42.

[69] P. L. Combettes and J.-C. Pesquet, "Image Restoration Subject to a Total Variation Constraint," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1213–1222, Sep. 2004. doi:10.1109/TIP.2004.832922

[70] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods," *Numerical Algorithms*, vol. 53, no. 1, pp. 67–92, Jan. 2010. doi:10.1007/s11075-009-9310-3

[71] D. Needell and R. Ward, "Stable Image Reconstruction Using Total Variation Minimization," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 1035–1058, Jun. 2013. doi:10.1137/120868281

[72] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit," *Siam Review*, vol. 43, no. 1, pp. 129–159, Mar. 2001. doi:10.1137/S003614450037906X

[73] F. Krzakala, M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices," *Journal of Statistical Mechanics: Theory and Experiment*, vol. P08009, pp. 1–57, Aug. 2012. doi:10.1088/1742-5468/2012/08/P08009

[74] T. Blumensath and M. E. Davies, "Iterative Thresholding for Sparse Approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, Sep. 2008. doi:10.1007/s00041-008-9035-z

[75] ——, "Normalized Iterative Hard Thresholding: Guaranteed Stability and Performance," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 298–309, Apr. 2010. doi:10.1109/JSTSP.2010.2042411

[76] K. K. Herrity, A. C. Gilbert, and J. A. Tropp, "Sparse Approximation via Iterative Thresholding," in *IEEE International Conference on Acoustics, Speech and Signal Proccesing (ICASSP)*, Toulouse, France, May 14 – 19, 2006, pp. III–624 – III–627. doi:10.1109/ICASSP.2006.1660731

[77] D. L. Donoho, "De-Noising by Soft-Thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, May 1995. doi:10.1109/18.382009

[78] I. Daubechies, M. Defrise, and C. D. Mol, "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004. doi:10.1002/cpa.20042

[79] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Mar. 2009. doi:10.1137/080716542

[80] M. A. T. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak, "Majorization–Minimization Algorithms for Wavelet-Based Image Restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2980–2991, Dec. 2007. doi:10.1109/TIP.2007.909318

[81] T. Blumensath, "Sampling and Reconstructing Signals From a Union of Linear Subspaces," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4660–4671, Jul. 2011. doi:10.1109/TIT.2011.2146550

[82] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová, "Statistical-Physics-Based Reconstruction in Compressed Sensing," *Physical Review X*, vol. 2, no. 2, pp. (021 005–1)–(021 005–18), May 2012. doi:10.1103/PhysRevX.2.021005

[83] S. Rangan, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, Jul. 31 – Aug. 5, 2011, pp. 2168–2172. doi:10.1109/ISIT.2011.6033942

[84] ——, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," Aug. 2012, arXiv:1010.5141v2.

[85] D. L. Donoho, A. Maleki, and A. Montanari, "Message Passing Algorithms for Compressed Sensing: II. Analysis and Validation," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5. doi:10.1109/ITWKSPS.2010.5503228

[86] J. T. Parker, "Approximate Message Passing Algorithms for Generalized Bilinear Inference," Ph.D. dissertation, Graduate School of The Ohio State University, 2014.

[87] D. L. Donoho, A. Maleki, and A. Montanari, "Message Passing Algorithms for Compressed Sensing: I. Motivation and Construction," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5. doi:10.1109/ITWKSPS.2010.5503193

[88] J. Vila and P. Schniter, "Expectation-Maximization Bernoulli-Gaussian Approximate Message Passing," in *Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 6 – 9 2011, pp. 799–803. doi:10.1109/ACSSC.2011.6190117

[89] J. Ziniel, "Message Passing Approaches to Compressive Inference Under Structured Signal Priors," Ph.D. dissertation, Graduate School of The Ohio State University, 2014.

[90] M. Bayati and A. Montanari, "The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011. doi:10.1109/TIT.2010.2094817

[91] M. Bayati, M. Lelarge, and A. Montanari, "Universality In Polytope Phase Transitions And Message Passing Algorithms," *The Annals of Applied Probability*, vol. 25, no. 2, pp. 753–822, Feb. 2015. doi:10.1214/14-AAP1010

[92] C. Rush and R. Venkataramanan, "Finite-Sample Analysis of Approximate Message Passing," in *IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, Jul. 10 – 15, 2016, pp. 755–759. doi:10.1109/ISIT.2016.7541400

[93] J. Ma and L. Peng, "Orthogonal AMP," *IEEE Access*, vol. 5, pp. 2020–2033, Jan. 2017. doi:10.1109/ACCESS.2017.2653119

[94] M. F. Duarte and R. G. Baraniuk, "Kronecker Compressive Sensing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, Feb. 2012. doi:10.1109/TIP.2011.2165289

[95] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, 1st ed. Cambridge University Press, 1994.

[96] E. Anguiano and M. Aguilar, "A cross-measurement procedure (CMP) for near noise-free imaging in scanning microscopes," *Ultramicroscopy*, vol. 76, no. 1-2, pp. 39–47, Feb. 1999. doi:10.1016/S0304-3991(98)00074-6

[97] A. Labuda, M. Lysy, W. Paul, Y. Miyahara, P. Grütter, R. Bennewitz, and M. Sutton, "Stochastic noise in atomic force microscopy," *Physical Review E*, vol. 86, no. 3, pp. 1–18, Sep. 2012. doi:10.1103/PhysRevE.86.031104

[98] P. Klapetek, *Quantitative Data Processing in Scanning Probe Microscopy: SPM Applications for Nanometrology*, 1st ed., ser. Micro & Nano Technologies. Elsevier, 2013.

[99] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[100] X.-L. Meng and D. B. Rubin, "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, Jun. 1993. doi:10.2307/2337198

[101] R. D. Peng, "Reproducible Research in Computational Science," *Science*, vol. 334, no. 6060, pp. 1226–1227, Dec. 2011. doi:10.1126/science.1213847

[102] Editorial, "Devil in the details," *Nature*, vol. 470, no. 7334, pp. 305–306, Feb. 2011. doi:10.1038/470305b

[103] R. J. LeVeque, I. M. Mitchell, and V. Stodden, "Reproducible Research for Scientific Computing: Tools and Strategies for Changing the Culture," *Computing in Science & Engineering*, vol. 14, no. 4, pp. 13–17, Jul. 2012. doi:10.1109/MCSE.2012.38

[104] S. Fomel and J. F. Claerbout, "Guest Editors' Introduction: Reproducible Research," *Computing in Science & Engineering*, vol. 11, no. 1, pp. 5–7, Jan. 2009. doi:10.1109/MCSE.2009.14

[105] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, and M. Taufer, "Enhancing reproducibility for computational methods," *Science*, vol. 354, no. 6317, pp. 1240–1241, Dec. 2016. doi:10.1126/science.aah6168

[106] B. A. Nosek, G. Alter, G. C. Banks, D. Borsboom, S. D. Bowman, S. J. Breckler, S. Buck, C. D. Chambers, G. Chin, G. Christensen, M. Contestabile, A. Dafoe, E. Eich, J. Freese, R. Glennerster, D. Goroff, D. P. Green, B.Hesse, M. Humphreys, J. Ishiyama, D. Karlan, A. Kraut, A. Lupia, P. Mabry, T. Madon, N. Malhotra, E. Mayo-Wilson, M. McNutt, E. Miguel, E. L. Paluck, U. Simonsohn, C. Soderberg, B. A. Spellman, J. Turitto, G. VandenBos, S. Vazire, E. J. Wagenmakers, R. Wilson, and T. Yarkoni, "Promoting an open research culture," *Science*, vol. 348, no. 6242, pp. 1422–1425, Jun. 2015. doi:10.1126/science.aab2374

[107] Editorial, "Code share," *Nature*, vol. 514, p. 536, Oct. 2014. doi:10.1038/514536a

[108] V. Stodden, P. Guo, and Z. Ma, "Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals," *PLoS ONE*, vol. 8, no. 6, p. e67111, Jun. 2013. doi:10.1371/journal.pone.0067111

[109] Z. Merali, "Computational science: ...Error ...why scientific programming does not compute." *Nature*, vol. 467, pp. 775–777, Oct. 2010. doi:10.1038/467775a

[110] M. Rupp, F. Gini, A. Pérez-Neira, B. Pesquet-Popescu, A. Pikrakis, B. Sankur, P. Vandewalle, and A. Zoubir, "Reproducible research in signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 93, no. 1, pp. 1–2, Oct. 2011. doi:10.1186/1687-6180-2011-93

[111] C. Collberg and T. A. Proebsting, "Repeatability in Computer Systems Research," *Communications of the ACM*, vol. 59, no. 3, pp. 62–69, Mar. 2016. doi:10.1145/2812803

[112] G. Miller, "A Scientist's Nightmare: Software Problem Leads to Five Retractions," *Science*, vol. 314, no. 5807, pp. 1856–1857, Dec. 2006. doi:10.1126/science.314.5807.1856

[113] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. C. Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson, "Best Practices for Scientific Computing," *PloS Biology*, vol. 12, no. 1, p. e1001745, Jan. 2014. doi:10.1371/journal.pbio.1001745

[114] D. Kelly, D. Hook, and R. Sanders, "Five Recommended Practices for Computational Scientists Who Write Software," *Computing in Science & Engineering*, vol. 11, no. 5, pp. 48–53, Sep. 2009. doi:10.1109/MCSE.2009.139

[115] A. Prlić and J. B. Procter, "Ten Simple Rules for the Open Development of Scientific Software," *PLoS Computational Biology*, vol. 8, no. 12, p. e1002802, Dec. 2012. doi:10.1371/journal.pcbi.1002802

[116] J. Freire, D. Koop, E. Santos, and C. T. Silva, "Provenance for Computational Tasks: A Survey," *Computing in Science & Engineering*, vol. 10, no. 3, pp. 11–21, May 2008. doi:10.1109/MCSE.2008.79

[117] A. Shade and T. K. Teal, "Computing Workflows for Biologists: A Roadmap," *PLoS Biology*, vol. 13, no. 11, p. e1002303, Nov. 2015. doi:10.1371/journal.pbio.1002303

[118] K. Hinsen, "Computational science: shifting the focus from tools to models," *F1000Research*, vol. 3, no. 101, pp. 1–16, Jun. 2014. doi:10.12688/f1000research.3978.2

[119] P. Vandewalle, J. Kovačević, and M. Vetterli, "Reproducible Research in Signal Processing [What, why, and how]," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 37–47, May 2009. doi:10.1109/MSP.2009.932122

[120] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten Simple Rules for Reproducible Computational Research," *PLoS Computational Biology*, vol. 9, no. 10, p. e1003285, Oct. 2013. doi:10.1371/journal.pcbi.1003285

[121] A. P. Davison, "Automated Capture of Experiment Context for Easier Reproducibility in Computational Research," *Computing in Science & Engineering*, vol. 14, no. 4, pp. 48–56, Jul. 2012. doi:10.1109/MCSE.2012.41

[122] V. Stodden and S. Miguez, "Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research," *Journal of Open Research Software*, vol. 2, no. 1, pp. 1–6, Jul. 2014. doi:10.5334/jors.ay

[123] V. Stodden, F. Leisch, and R. D. Peng, Eds., *Implementing Reproducible Research*, ser. Chapman & Hall/CRC The R Series. CRC Press, 2014.

[124] S. Fomel, P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, "Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments," *Journal of Open Research Software*, vol. 1, no. 1, p. e8, Nov. 2013. doi:http://doi.org/10.5334/jors.ag

[125] K. Hinsen, "ActivePapers: a platform for publishing and archiving computer-aided research," *F1000Research*, vol. 3, no. 289, p. 14, Jul. 2015. doi:10.12688/f1000research.5773.3

[126] S. R. Piccolo and M. B. Frampton, "Tools and techniques for computational reproducibility," *GigaScience*, vol. 5, no. 1, pp. 1–13, Jul. 2016. doi:10.1186/s13742-016-0135-4

[127] J. D. Blischak, E. R. Davenport, and G. Wilson, "A Quick Introduction to Version Control with Git and GitHub," *PLoS Computational Biology*, vol. 12, no. 1, p. e1004668, Jan. 2016. doi:10.1371/journal.pcbi.1004668

[128] T. J. McCabe, "A Complexity Measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, Dec. 1976. doi:10.1109/TSE.1976.233837

[129] A. H. Watson and T. J. McCabe, "Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric," National Institute of Standards and Technology (NIST), Special Publication 500-235, Sep. 1996.

[130] C. S. Oxvig, P. S. Pedersen, T. Arildsen, J. Østergaard, and T. Larsen, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images," *Journal of Open Research Software*, vol. 2, no. 1, p. e29, Oct. 2014. doi:10.5334/jors.bk

[131] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011. doi:10.1109/MCSE.2011.37

[132] P. S. Pedersen, C. S. Oxvig, J. Østergaard, and T. Larsen, "Validating Function Arguments in Python Signal Processing Applications," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 106–113.

[133] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, Jan. 2015. doi:10.1145/2723872.2723882

[134] C. S. Oxvig, T. Arildsen, and T. Larsen, "Storing Reproducible Results from Computational Experiments using Scientific Python Packages," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 45–50.

[135] *The JSON Data Interchange Format*, ECMA International Std., Rev. 1, Oct. 2013.

[136] T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, May 2007. doi:10.1109/MCSE.2007.58

[137] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, Jan. 2009. doi:10.1109/MSP.2008.930649

[138] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004. doi:10.1109/TIP.2003.819861

[139] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*, ser. Synthesis Lectures on Image, Video, and Multimedia Processing. Morgan & Claypool Publishers, 2006. doi:10.2200/S00010ED1V01Y200508IVM003

[140] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469–483, Dec. 1996. doi:10.1145/235815.235821

[141] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. 32 Avenue of the Americas, New York, NY 10013-2473, USA: Cambridge University Press, 2007.

[142] P. Alfeld, "A trivariate Clough-Tocher scheme for tetrahedral data," *Computer Aided Geometric Design*, vol. 1, no. 2, pp. 169–181, Nov. 1984. doi:10.1016/0167-8396(84)90029-3

[143] S. Maneewongvatana and D. M. Mount, "Analysis of Approximate Nearest Neighbor Searching with Clustered Point Sets," in *Workshop on Algorithm Engineering and Experimentation (ALENEX)*, Baltimore, Maryland, USA, Jan. 15 – 16, 1999.

[144] A. Beck and M. Teboulle, "Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009. doi:10.1109/TIP.2009.2028250

[145] P. L. Combettes and J.-C. Pesquet, "A Douglas-Rachford Splitting Approach to Nonsmooth Convex Variational Signal Recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, Dec. 2007. doi:10.1109/JSTSP.2007.910264

[146] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Blackwell, 2000.

[147] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing Sparsity by Reweighted l1 Minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, p. 877–905, Dec. 2008. doi:10.1007/s00041-008-9045-x

Part II

Individual Manuscripts & Datasets

Paper A

Reconstruction Algorithms in Undersampled AFM Imaging

Thomas Arildsen, Christian Schou Oxvig, Patrick Steffen Pedersen,
Jan Østergaard, Torben Larsen

The paper has been published in
IEEE Journal of Selected Topics in Signal Processing,
vol. 10, no. 1, pp. 31–46, Feb. 2016.
doi:10.1109/JSTSP.2015.2500363

Reconstruction Algorithms in Undersampled AFM Imaging

Thomas Arildsen, *Member, IEEE,* Christian Schou Oxvig, *Student Member, IEEE,* Patrick Steffen Pedersen, *Member, IEEE,* Jan Østergaard, *Senior Member, IEEE,* and Torben Larsen, *Senior Member, IEEE,*

***Abstract*—This paper provides a study of spatial undersampling in atomic force microscopy (AFM) imaging followed by different image reconstruction techniques based on sparse approximation as well as interpolation. The main reasons for using undersampling is that it reduces the path length and thereby the scanning time as well as the amount of interaction between the AFM probe and the specimen. It can easily be applied on conventional AFM hardware. Due to undersampling, it is necessary to subsequently process the acquired image in order to reconstruct an approximation of the image. Based on real AFM cell images, our simulations reveal that using a simple raster scanning pattern in combination with conventional image interpolation performs very well. Moreover, this combination enables a reduction by a factor 10 of the scanning time while retaining an average reconstruction quality around $36\,\mathrm{dB}$ PSNR on the tested cell images.**

***Index Terms*—atomic force microscopy, undersampling, image reconstruction, sparse approximation, interpolation, compressed sensing**

I. Introduction

ATOMIC force microscopy (AFM) is a scanning probe microscopy technique that offers several interesting possibilities in the imaging of biological materials such as cells. Atomic force microscopy complements other microscopy techniques such as optical microscopy or scanning electron microscopy (SEM) by enabling three-dimensional imaging of cell surfaces and imaging cells and bio-molecules in more natural environments than other techniques. This also enables imaging of live cells [1]. Imaging biological material such as live cells does, however, entail some challenges such as the risk of damaging the cells due to interaction with the microscope probe tip [2], [3].

Imaging with AFM equipment is a relatively time-consuming process, taking on the order of seconds to minutes or even higher to image a region of interest using commercial AFM equipment [4], [5]. While this may be inconvenient to the operator of AFM equipment, it can become an impediment when imaging temporally evolving material and organisms, i.e. the AFM equipment may simply not be able to scan the specimen sufficiently fast to be able to follow the process [6], [7]. Several approaches to achieving higher-speed scanning in AFM have been explored. These include approaches dealing with the mechanical characteristics of the equipment, control of the probe, or design of special sampling patterns that allow faster movement of the probe [6], [8]–[10]. However, since it is often necessary to probe the specimen with great caution, particularly in the case of live cell imaging, efforts to scan faster and yet interact as little/carefully with the specimen as possible may well run counter to each other.

One way to combat this dilemma could be to use sparser sampling patterns than the patterns typically used in AFM. The typical way to sample the topography of a specimen in AFM is to scan the probe across the surface in a dense raster pattern [11]. This process can be sped up by using a sparser sampling pattern, i.e. effectively letting the scan path cover the surface less densely and thereby enabling a shorter and thus faster scan path. This approach simultaneously causes the probe tip to interact less with the specimen. Such an approach can potentially solve the dilemma of careful interaction with a fragile specimen vs. fast scanning. In exchange, this necessitates reconstruction of the full surface topography (image) from considerably fewer samples instead.

In this paper we survey a range of methods that can be applied in order to achieve faster and/or less destructive cell imaging using AFM.[1] In particular, we compare two different sampling patterns (raster and spiral) in combination with a selection of image reconstruction techniques based on sparse approximation and an interpolation technique used as reference. For the comparison, we use seven AFM cell image specimens. We identify useful combinations of scanning patterns and reconstruction algorithms that provide good reconstruction quality and are sufficiently fast w.r.t. scan time as well as reconstruction time. A perhaps somewhat surprising finding is that the naive approach of simply scanning the cell specimen with a less dense raster pattern, effectively skipping a fraction of the lines, and then combined with standard image interpolation, leads to the best overall objective reconstruction quality for a fixed undersampling ratio. It is worth noticing that if obtaining a less dense raster scanning pattern is supported by the AFM equipment, then this technique does not require

The authors are with the Faculty of Engineering and Science, Department of Electronic Systems, Aalborg University, Aalborg, Denmark e-mail: {tha,cso,psp,jo,tl}@es.aau.dk.

This project has been supported by 1) The Danish Council for Independent Research (DFF/FTP) for project number 1335-00278B/12-134971, and 2) by the Danish e-Infrastructure Cooperation (DeIC) for project number DeIC2013.12.23.

[1]The underlying code-base and images required for reproducing all results in this article is freely available at:

- **Code** http://dx.doi.org/10.5281/zenodo.32959
- **Results** http://dx.doi.org/10.5281/zenodo.32958
- **Images** http://dx.doi.org/10.5281/zenodo.17573

any hardware modification of the AFM equipment. The reconstruction algorithm is then a purely software-based approach that can be performed on a standard PC or enabled through a firmware upgrade of the equipment. We also find that for the seven AFM cell images considered in this study, interpolation and total variation techniques work better with raster scanning patterns, whereas sparse approximation techniques with DCT dictionaries work better with spiral scanning patterns.

The paper is outlined as follows. In Section II we give an overview of signal processing for atomic force microscopy, introducing the basics of AFM equipment as well as the necessary processing required to obtain useful images from the equipment, possible impairments, and notation details. In Section III we give an overview of image reconstruction based on sparse approximation techniques and introduce a reference method, interpolation, for comparison. Section IV details our experiments regarding reconstruction of images in AFM. Section V presents results from numerical experiments with the presented reconstruction methods. Finally, Section VI summarizes and concludes the paper.

II. Notation and Framework

A. Introduction to AFM

Atomic force microscopy (AFM) is one of the most advanced techniques for investigating and manipulating surfaces on the atomic scale. By working on this scale, AFM provides magnification which is orders of magnitude beyond the capabilities of optical, confocal, and even scanning electron microscopy or transmission electron microscopy techniques [12]. This is generally the case for scanning probe microscopy (SPM) which encompasses the families of AFM and scanning tunneling microscopy (STM). Whereas STM requires the surface of interest to be electrically conductive, AFM does not [11] and is therefore the technique of interest to the present paper due to the potential application to live cell imaging. However, many of the thoughts presented should be applicable to SPM in general.

Being a state-of-the-art technique, AFM is used extensively within nanoscale science and technology [13]. Partly because the technique can be used on surfaces in both vacuum, air, and liquids, there is a large variability in the applications [7]. A number of applications relate to materials science, some to the study of biological processes, and some to the study of biological materials [9]. Yet other applications use AFM for surface manipulation including lithography, nanomanipulation, and nanoassembly [14].

In the context of surface investigation, AFM is most commonly used to generate a 3D surface map of the object of interest [6]. Loosely speaking, a probe is used to measure the height of the surface while the probe and the surface are moved relative to each other. Specifically, the vertical position of the probe is controlled by a piezo which is itself controlled by a feedback loop. This feedback loop keeps a particular measured property constant, such as the deflection of the cantilever on which the probe tip is located, in order to ensure that the probe traces the surface. Independently of this control loop, the probe and the surface are moved relative to each other by the use of additional piezos [12].

B. Image acquisition

To prepare the AFM equipment for operation, the user must perform an initial system setup which consists of a number of steps. Some of these steps require the user to make decisions based on the surface under investigation. These decisions include selecting a cantilever, operating mode (contact mode, acoustic AC mode, or magnetic AC mode), servo settings or AC mode settings, and scanner settings. Although these heavily affect the quality of the measurements of the surface, an in-depth coverage of the initial system setup is beyond the scope of the present paper. It is, however, worth mentioning that the degree of interaction between the probe tip and sample depends heavily on the chosen operating mode. In contact mode, the probe tip is "dragged" across the surface and thus typically applies a near-constant force to it [12]. In AC mode on the other hand, the cantilever is oscillated and thus only applies force to the surface a fraction of the time [15]. The interaction between probe tip and sample is particularly important when dealing with soft materials such as biological cells [16].

The setup of the AFM equipment includes a number of steps related to the movement of the probe and the surface relative to each other: i.e. the scanning path. Traditionally, a raster scanning path is used [4], and this only requires the user to decide which surface region to scan, how densely to scan it, and how fast to scan it. If the raster scanning path should not be used then the user is required to decide on the actual scanning path, the movement speed of the probe, and the sampling frequency. Additionally, when deciding on a non-raster scanning path, the user is required to somehow implement this scanning if it is not already available in the AFM equipment.

The actual scanning path is subject to two major constraints: 1) The probe cannot easily or effectively "jump" from one point on the surface to another. Therefore, the path must be continuous. 2) The piezos have a band limited frequency response. Therefore, when combined with a specific probe movement speed, the path must have frequency contents which are limited to that band in order to avoid distorting the scanning path.

C. Acquisition impairments

The image acquisition process is subject to a number of impairments. Some of these impairments may severely affect the image quality when image reconstruction is introduced [17]. To ensure successful image reconstruction, the impairments must be considered. Fortunately, some of these can be mitigated by careful setup of the AFM equipment whereas others must be accounted for in the image reconstruction [11]. This section highlights some of the possible impairments but should in no way be considered a complete list.

Some of the impairments relate to the object of interest. First of all, when this object is put in place, the surface is likely to be tilted since the user cannot likely ensure that the surface is perfectly normal to the probe tip. This impairment should be accounted for by the image reconstruction. Next, when acquiring an image, the surface may be deformed [18] since

the AFM equipment applies force to the object which may consist of a soft material. This impairment can be mitigated by careful setup of the AFM equipment but it inevitably distorts the image slightly [6].

Some of the impairments relate to the physical parts of the AFM equipment. The probe tip is affected by an area of the surface rather than a single point because of the shape and size of the probe [11]. Depending on the probe, the slope of the surface, and the desired physical resolution, this may distort the image slightly. The sensors used in state-of-the-art AFM equipment are sufficiently accurate and precise to only cause negligible impairments [9]. The piezos used to move the sample and probe relative to each other are intrinsically subject to non-linearity, hysteresis, and creep [14]. However, state-of-the-art AFM equipment can operate in closed-loop mode in order to mitigate these effects [8].

Some of the impairments relate to the control loops and the applied signal processing. The probe is part of a cantilever which is deflected when the probe tip is affected by the sample surface. This deflection is measured and compared to the desired deflection resulting in an error signal which is used to control the piezo. However, due to the filters used, the type of controller, and the physics of piezos, the piezo does not instantaneously compensate for changes in cantilever deflection which may distort the image [18]. This impairment may be mitigated by reducing the probe movement speed.

Finally, there are also possible issues of stochastic measurement noise. Several factors specific to the equipment contribute to stochastic noise in AFM [19]. These are for example related to the optical system that is used to control the deflection of the cantilever.

D. Discretization

As described in Section II-A, imaging with AFM can be seen as measuring the surface height of a specimen across a continuous two-dimensional surface (topography). The end goal we consider here is conveying this measured topography visually. This typically entails displaying an image of the measured surface as points on a uniform grid, e.g. a computer screen. This can be done in various ways, some of which are described in the following. In order to do that, we first establish some notation and general principles here.

We consider a region $\Omega \subset \mathbb{R}^2$ within which we wish to image the topography of the continuous surface of the specimen, denoted X. The surface of the specimen is sampled along a scan path, on which the AFM probe, represented by the sampling operator ϕ, collects m samples $\phi(X) \in \mathbb{R}^{m\times 1}$ at discrete points on the surface X. From these samples, we wish to reconstruct an $h \times w$ (pixels) image of the surface. We refer to this image representation of the surface (with values located on a uniform pixel grid over Ω) as a matrix, $\mathbf{X} \in \mathbb{R}^{h\times w}$, or as a vector, $\mathbf{x} \in \mathbb{R}^{hw\times 1}$, containing the stacked columns of the matrix with the left-most column of $\mathbf{X}$ as the top entries of $\mathbf{x}$ etc. The reconstruction of this image is correspondingly denoted $\hat{\mathbf{X}}$ or $\hat{\mathbf{x}}$.

In the case of raster scanning as traditionally applied in AFM, the sampled points can be chosen naturally to lie close to a uniform grid that fits directly into an image interpretation. In this case the sampled points $\phi(X)$ correspond directly to the image $\mathbf{X}$ with the possible addition of noise and various scanning artifacts $\mathbf{E} \in \mathbb{R}^{h\times w}$ as described in Section II-C:

$$\phi(X) \;=\; \mathbf{X} + \mathbf{E} \tag{1}$$

If one deviates from this traditional raster scanning and sampling approach by either changing the scan pattern or using non-uniform sampling, the acquired image samples do not generally fall on a uniformly spaced grid corresponding to the pixels of $\mathbf{X}$. Having a uniformly spaced pixel grid is attractive from a mathematical point of view, since the image can be represented in a matrix form with an intuitive interpretation of the physical locations of the sampling points. In this case, we consider reconstruction $\hat{\mathbf{X}}$ of the hypothetical image $\mathbf{X}$ from which measurements are obtained via an intermediate interpolation from the measured points to the pixel grid. In this spatially discretized setting, the obtained samples can be seen as located at points on the uniform pixel grid as well such that:

$$\phi(X) \;=\; \mathbf{\Phi x} \tag{2}$$

The pattern of sampling points represented by the spatially discrete matrix $\mathbf{\Phi}$ is referred to as the sampling pattern.

The variables defined in this section form the basis of the different reconstruction approaches presented in the following sections.

III. Sparse Approximation

Sparse regularization and/or approximation is a well-known approach to solving ill-posed optimization problems, early examples of which include [20]–[22]. The principle of compressed sensing which has emerged quite recently has popularized the sparse regularization principle [23], [24]. For an introduction, see [25]–[27].

In this paper, we demonstrate a selection of reconstruction algorithms based on sparse regularization. For this purpose, we consider the following linear measurement model:

$$\mathbf{y} \;=\; \mathbf{A}\boldsymbol{\alpha} \tag{3}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^{n\times 1}$ is a sparse vector, i.e. it contains only $k \ll n$ non-zero entries, also expressed as $\|\boldsymbol{\alpha}\|_0 = k$ in the ℓ_0 pseudo-norm. The matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a sensing matrix applied to the sparse vector to sample the measurements $\mathbf{y} \in \mathbb{R}^{m\times 1}$.

Typically, as is the case for the application to AFM proposed here, a signal is not sparse in the domain we can observe it in. The following more general model is therefore used:

$$\mathbf{y} \;=\; \mathbf{\Phi x} \tag{4}$$

where:

$$\mathbf{x} \;=\; \mathbf{\Psi}\boldsymbol{\alpha} \tag{5}$$

Here $\mathbf{x} \in \mathbb{R}^{p\times 1}$ is the observable signal vector and $\mathbf{\Phi} \in \mathbb{R}^{m\times p}$ is a measurement matrix applied to sample the measurements. The matrix $\mathbf{\Psi} \in \mathbb{R}^{p\times n}$ represents the dictionary, enabling a sparse representation $\boldsymbol{\alpha}$ of the observable signal $\mathbf{x}$.

If we consider the above formulations in the setting of reconstructing an AFM image from samples scanned on a specimen, $\mathbf{y}$ corresponds to the scanned samples of the image $\mathbf{x}$, a vector version of $\mathbf{X}$. We wish to reconstruct an estimate $\hat{\mathbf{x}}$ of the image from $\mathbf{y}$. The fact that $m < p \leq n$ means that (3), or equivalently (4) together with (5), constitutes an under-determined linear equation system which we cannot directly invert to obtain $\hat{\mathbf{x}}$. Sparse regularisation as used in, e.g. compressed sensing enables solving (3) for $\boldsymbol{\alpha}$, and equivalently for $\mathbf{x}$ through (5), by solving the following (non-convex) optimization problem [23]:

$$\hat{\boldsymbol{\alpha}} \;=\; \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_0 \quad \text{s.t.} \quad \mathbf{y} \;=\; \mathbf{A}\boldsymbol{\beta} \tag{6}$$

Unfortunately (6) is an intractably difficult combinatorial problem to solve exactly. However, compressed sensing theory shows that (6) can be replaced by the following convex relaxation of the problem [23], [26]:

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{A}\boldsymbol{\beta} \tag{7}$$

The optimization problem in (7) can solve (3) exactly under certain conditions [26], [29]. The convex relaxation in (7) is one approach to approximating a solution to (6). However, there exist a number of different approaches which we survey a selection of in Section III-B.

The reconstruction method can be generalized to the case of noisy measurements and/or signals that are not exactly sparse but rather "compressible" in the sense that they are accurately approximated by a few of the largest entries in $\boldsymbol{\alpha}$:

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x} + \mathbf{e} \tag{8}$$

The vector $\mathbf{e} \in \mathbb{R}^{m\times 1}$ represents noise in the acquired measurements, e.g. the impairments described in Section II-C, and/or the error resulting from sparsely approximating a signal that is not strictly sparse. In this case, the following optimization problem reconstructs the signal [26]:

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad \text{s.t.} \quad \|\mathbf{A}\boldsymbol{\beta} \;-\mathbf{y}\|_2 \leq \epsilon \tag{9}$$

The parameter ϵ bounds the 2-norm of the error $\mathbf{e}$.

The sparse representation model (5) is known as the sparse synthesis model – for its ability to synthesize a signal $\mathbf{x}$ from a sparse vector $\boldsymbol{\alpha}$. This model also has a counterpart: the co-sparse analysis model [30]:

$$\boldsymbol{\alpha} \;=\; \boldsymbol{\Psi}^\top \mathbf{x} \tag{10}$$

This model admits a sparse representation of the signal $\mathbf{x}$ after multiplication by an analysis dictionary $\boldsymbol{\Psi}^\top$. Note here that good dictionaries for the analysis model are not necessarily simply a transpose of a corresponding synthesis dictionary, but we use this notation here in order not to complicate the notation with additional symbols. The optimization problem for reconstructing $\mathbf{x}$ from the analysis model, as a counterpart to (9) can be stated as:

$$\hat{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}}} \|\boldsymbol{\Psi}^\top \tilde{\mathbf{x}}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{\Phi}\tilde{\mathbf{x}} \;-\mathbf{y}\|_2 \leq \epsilon \tag{11}$$

A number of theoretical conditions for compressed sensing reconstruction to succeed can be found in the literature [26], [31], but most of the theory relies on the measurement matrix $\boldsymbol{\Phi}$ having i.i.d. random entries. A random measurement matrix is difficult to achieve when scanning a specimen in an efficient manner in AFM and further, the continuous trajectory typically used in AFM in this case violates the assumption of i.i.d. entries. Therefore, the imaging techniques explored in this paper are not strictly compressed sensing. Nevertheless, we investigate some of the reconstruction algorithms known especially from compressed sensing to assess the value of reconstructing images in AFM by sparse approximation.

Previous work has shown that for particular AFM images having much greater energy in the high-frequency domain than in the low-frequency domain, sparse approximation techniques generally perform better than (Delaunay) interpolation-based techniques, whereas for low-frequency AFM images, excellent performance can be obtained with Delaunay interpolation [32].

A. Measurement and Dictionaries

As mentioned in Section III, the sensing matrix $\mathbf{A}$ can be considered as the product of a separate measurement matrix $\boldsymbol{\Phi}$ and (synthesis) dictionary matrix $\boldsymbol{\Psi}$ where the purpose of the measurement matrix is to represent the process that physically measures the sample. The purpose of the dictionary matrix is to enable a sparse representation of the image $\mathbf{x}$.

When considering separate measurement matrices $\boldsymbol{\Phi}$ and dictionaries $\boldsymbol{\Psi}$, the matrices should be selected from incoherent orthogonal bases Φ and Ψ. Coherence, μ, is a measure of the similarity of the vectors Φ and Ψ [26]. A low coherence is better. The measurement matrix $\boldsymbol{\Phi}$ should consist of rows selected uniformly at random from Φ while the columns of the dictionary matrix $\boldsymbol{\Psi}$ should be the vectors from Ψ [26]. The preceding descriptions apply to the case of a dictionary $\boldsymbol{\Psi}$ corresponding to an orthonormal basis [33], i.e. $m = p$ in (5). However, the more general case of over-complete dictionaries where $m < p$ is also possible [34].

An overview of the past and present directions in the design of dictionaries is given in [35]. The possibilities range from the fixed, general purpose, orthogonal dictionaries over more adapted over-complete dictionaries [36], [37] to the highly data- and application-specific dictionaries designed using a Karhunen-Loeve transform [38] (also known as a PCA transform [39]) or a learning approach [40], [41]. Although any of these approaches may be applicable for AFM image representation, here we only discuss fixed dictionaries such as the discrete cosine transform (DCT)[2] or the discrete wavelet transform (DWT). These transforms are of particular interest due to their simplicity, their celebrated applicability in general compressive imaging [27], and the availability of efficient implementations requiring only $\mathcal{O}(n\,\log(n))$ computations as well as relaxed memory requirements due to an implicit representation of the dictionary matrix [42].

The DCT is used in the JPEG coding standard [43] and as such is known to be successful in sparsely representing

[2]Here we consider the DCT as a representative of the family of sinusoidal transforms which also includes, e.g. the discrete Fourier transform (DFT). It is our experience that the use of the DFT gives reconstructions comparable to those based on the DCT.

smooth images. However, in JPEG the DCT dictionary is used on smaller patches of the image whereas we only apply dictionaries to the full image in this study. In terms of its applicability in AFM imaging, the DCT is the dictionary used in two independent recent studies on the applicability of compressed sensing in AFM [32] and SEM [44]. The DWT is generally very successful in sparsely representing piecewise smooth images. Consequently, it is the transform chosen in the JPEG2000 coding standard [45]. The DWT has been successfully used in a number of compressive imaging studies, see e.g. [46] and [47]. Note, however, that these studies use dense sampling matrices with random entries and not the sparse point sampling used in AFM.

The excellent sparse representation capabilities of the DCT and DWT on natural images make them both good candidates for use in reconstruction of undersampled AFM images. We have, however, found that the DWT can be problematic when used in combination with point sampling. As (4) shows, measurements in compressed sensing are generally random linear combinations of the entries of the observable signal. In AFM, however, the physical properties of the probe tip only allow the microscope to sample the specimen in discrete points, i.e. each row in $\mathbf{\Phi}$ contains only *one* 1-entry while the other entries are 0, making this point sampling measurement matrix extremely sparse. The DWT dictionary matrix is relatively sparse compared to the DCT and it follows that their product is likely to be sparse, where some columns can become all-zero. Thus, the null-space of the product matrix is non-empty and there exist sparse solutions which cannot be represented by the particular pair of measurement and dictionary matrix. Intuitively, the DWT basis functions are not able to smoothly interpolate between points spaced too far apart due to being very localized in the image domain. Hence, we may experience low incoherence between the DWT and point sampling. This is not the case for the DCT dictionary matrix, which is dense and maximally incoherent with point sampling [26]. To demonstrate the difference in reconstruction capabilities between DCT and DWT dictionaries, we include results of experiments with both dictionary types in Section IV. As the results show, the performance depends strongly on the type of sampling pattern used.

B. Reconstruction Algorithms

In the following, we review a number of reconstruction algorithms that can be used to reconstruct undersampled AFM images by sparse approximation.

1) Convex Optimization: The classic approach to solving sparse approximation problems described by (6) is using the ℓ_1 norm convex optimization formulations introduced by (7) and (9). The constrained convex formulation (9) is also commonly found in a regularized form:

$$\hat{\boldsymbol{\alpha}} \;=\; \arg\min_{\boldsymbol{\beta}} \left\{ \tau\|\boldsymbol{\beta}\|_1 + \frac{1}{2}\|\mathbf{A}\boldsymbol{\beta} \;- \mathbf{y}\|_2^2 \right\} \qquad (12)$$

Although (12) appears different from (9) at first glance, they can produce identical solutions for given pairs of (ϵ, τ) [48]. These convex optimization formulations are also known as the least absolute shrinkage and selection operator (LASSO) or basis pursuit de-noising (BPDN) [49], [50].

Equations (7), (9), (12) are formulations of the problem to solve. However, various different algorithms can be employed to compute the actual solution [51], [52]. Some solvers for this type of problems are implemented in for example PyUNLocBoX[3] [53], SPGL1[4] [54], YALL1[5] [55], and TFOCS[6].

Another convex optimization method following the approach in (11) is using total variation (TV) minimization [56]. Proposed for image denoising in the context of image processing [57], TV is a measure that quantifies the variation in some function. In image reconstruction, the TV measure is used to minimize the variation in the reconstructed image. That is, this approach takes advantage of the fact that natural images tend to consist of relatively large smooth regions and exploits this fact to fill in missing regions between the known parts of the image. Anisotropic TV can be seen as analysis co-sparse approximation with a discrete difference operator as the analysis dictionary, see (10). The anisotropic TV operator can be found in, e.g. [58]. As an example of analysis-based sparse approximation, we apply a slightly different variant; isotropic TV. This has also been applied to AFM image reconstruction in [7]. A related application is found in [59] where a slightly different but similar approach known as heat equation in-painting is used. The isotropic TV convex optimization problem can be posed as [60]:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathrm{tv}(\mathbf{x}) \quad \text{s.t.} \quad \|\mathbf{\Phi}\mathbf{x} \;- \mathbf{y}\|_2 \leq \epsilon \qquad (13)$$

In (13), we have used vector notation for the image $\mathbf{x}$ to simplify the constraint. For the purposes of numerical computation used in image processing, a discrete approximation of the TV norm is used since the image is discretized to a pixel grid. One definition of this discretization can be found in [60]:

$$\begin{aligned}\mathrm{tv}(\mathbf{X}) \;=\; & \sum_{k=0}^{h-2}\sum_{l=0}^{w-2} \Big(\big|\mathbf{X}_{(k+1,l)} - \mathbf{X}_{(k,l)}\big|^2 + \\ & \big|\mathbf{X}_{(k,l+1)} - \mathbf{X}_{(k,l)}\big|^2 \Big)^{\frac{1}{2}} + \sum_{k=0}^{h-2} \big|\mathbf{X}_{(k+1,w-1)} - \mathbf{X}_{(k,w-1)}\big|^2 \\ & + \sum_{l=0}^{w-2} \big|\mathbf{X}_{(h-1,l+1)} - \mathbf{X}_{(h-1,l)}\big|^2 \end{aligned} \qquad (14)$$

In (14), we have used matrix notation for the image $\mathbf{X}$ to simplify indexing; $\mathbf{X}_{(k,l)}$ indexes the (k,l)'th entry in $\mathbf{X}$. Equation (14) is the isotropic version of the discrete TV norm.

The problem (13) can be solved using different algorithms such as split Bregman [61] or Douglas-Rachford splitting [53], [62]. Implementations of algorithms solving TV optimization can be found in PyUNLocBox, which can solve (13) as described in [63]. TFOCS[6] [64] also implements a solution.

2) Greedy Pursuits: An alternative to the convex optimization based reconstruction algorithms is using the class of so-called *greedy* reconstruction algorithms. The term *greedy* is

[3]Available at https://github.com/epfl-lts2/pyunlocbox.
[4]Available at http://www.cs.ubc.ca/labs/scl/spgl1.
[5]Available at http://yall1.blogs.rice.edu/.
[6]Available at http://cvxr.com/tfocs/.

used because these algorithms iteratively take decisions that are locally optimal in each iteration [65].

Generally, an iteration of a greedy algorithm involves a greedy selection of support elements (columns of $\mathbf{A}$) followed by a coefficient update (an update of $\hat{\boldsymbol{\alpha}}$), see e.g. [65, Algorithm 8.1]. The simplest examples are so-called greedy pursuits like Matching Pursuit (MP) [36] or Orthogonal Matching Pursuit (OMP) [66] that only allow for a continued increase of the support. Algorithms like Iterative Hard Thresholding (IHT) [67] or Iterative Soft Thresholding (IST)[7] [68] have an ability to also prune elements from the support. For common parameter choices, the IHT interchanges the optimization criterion and the constraint in (9) and uses the ℓ_0 pseudo-norm from (6) instead of the ℓ_1 relaxation:

$$\hat{\boldsymbol{\alpha}} \;=\; \arg\min_{\boldsymbol{\beta}} \|\mathbf{A}\boldsymbol{\beta} \;-\mathbf{y}\|_2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \;\leq\; k \tag{15}$$

In particular the IHT and IST algorithms have a simple iterative form which can be written (with iteration index k) as:

$$\boldsymbol{\beta}^{(k+1)} \;=\; \mathrm{T}_{\mu}\left(\boldsymbol{\beta}^{(k)} + \kappa\mathbf{A}^{\top}\left(\mathbf{y} - \mathbf{A}\boldsymbol{\beta}^{(k)}\right)\right) \tag{16}$$

The parameter κ is a step size. The function T_{μ} is a thresholding operator applied entry-wise to each of the entries v of a vector $\mathbf{v}$. For IHT the hard thresholding operator [67] is:

$$\mathrm{T}_{\mu}\,(v) \;=\; \begin{cases} 0 & \text{for } |v| \leq \mu \\ v & \text{otherwise} \end{cases} \tag{17}$$

For IST the soft thresholding operator [68] is:

$$\mathrm{T}_{\mu}\,(v) \;=\; \begin{cases} 0 & \text{for } |v| \leq \mu \\ \operatorname{sgn}(v)\,(|v| - \mu) & \text{otherwise} \end{cases} \tag{18}$$

Even more advanced algorithms exist such as Subspace Pursuit [69] or CoSaMP [70] that introduce a two-stage thresholding scheme with an intermediate support element selection and coefficient update.

Although some of the greedy algorithms can be shown to have theoretical recovery guarantees that match those of the ℓ_1 based convex relaxation methods [65], empirical evidence suggests that they do not perform as well as ℓ_1 optimization [71], [72]. The greedy algorithms are, however, worth considering due to their low computational complexity. The computational cost in an iteration of e.g., MP, IHT, or IST is dominated by the computation of matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^{\top}$, thus, having complexity $\mathcal{O}(n^2)$. If fast transforms are available, as is the case when using e.g. the DCT as described in Section III-A, the computational cost is $\mathcal{O}(n\,\log(n))$. This has a significant impact on the time it takes to do the reconstruction for large problem sizes such as a $256 \times 256 = 65536$ pixels AFM image.

3) Approximate Message Passing: Probabilistic Message Passing (MP) algorithms based on graphical belief models are known from Bayesian inference used in machine learning [73]. This is an advanced method of reconstruction, which takes into account prior information the user may have on signal characteristics [74]. It unfortunately suffers from severe computational load and may also show poor convergence properties if the algorithmic assumptions are not fulfilled [75]. The Approximate Message Passing (AMP) algorithm is derived as a first order approximation, which reduces the computational burden significantly [73], [76]–[78]. AMP exists in several variants allowing different signal priors [79], inclusion of parameters as variables [80] etc. The following is based on a reasonably simple AMP method using a Bayesian framework for probabilities. Maleki and Baraniuk [81] showed links between AMP and Iterative Soft Thresholding (IST) in terms of identical convergence properties, and it has also been shown that the AMP algorithm can solve the LASSO (Least Absolute Shrinkage and Selection Operator) problem formulated in (12) [80].

The Minimum Mean-Square Error (MMSE) signal reconstruction estimate for x_n can be found from a marginal Bayesian mean of the posterior marginal estimate as [82]:

$$\hat{x}_n^{\mathrm{MMSE}} \;=\; \int_{x^*} x_n\, \wp_{X|Y}(x_n|\mathbf{y})\,\mathrm{d}x_n \tag{19}$$

where $\wp_{X|Y}(x_n|\mathbf{y})$ is the conditioned posterior pdf (probability density function), and x^* is the space of x_n. To compute the MMSE estimate in (19) we need to determine the conditioned posterior probability $\wp_{X|Y}(x_n|\mathbf{y})$, which can be done via Bayes' rule [82]:

$$\wp_{X|Y}(x|y) \;=\; \frac{\wp_{Y|X}(y|x)\,\wp_X(x)}{\wp_Y(y)} \tag{20}$$

$$= \frac{\wp_{Y|X}(y|x)\,\wp_X(x)}{\int_{x^*}\wp_{Y|X}(y|x)\,\wp_X(x)\,\mathrm{d}x} \tag{21}$$

For the sparse input signal $\mathbf{x}$ we assume all components to be i.i.d. Bernoulli-Gaussian with marginal pdf:

$$\wp_X(x_n) \;=\; \rho\mathcal{N}(x_n;\,\mu_{\mathrm{x}},\sigma_{\mathrm{x}}^2) \;+\; (1-\rho)\,\delta_{\mathrm{dirac}}(x_n) \tag{22}$$

where $\rho \in [0;1]$, $\delta_{\mathrm{dirac}}(\cdot)$ is the Dirac δ-function [83], and the general Gaussian function is:

$$\mathcal{N}(x_n;\,\mu_{\mathrm{x}},\sigma_{\mathrm{x}}^2) \;=\; \frac{1}{\sqrt{2\,\pi}\,\sigma_{\mathrm{x}}}\exp\left(\frac{-(x_n-\mu_{\mathrm{x}})^2}{2\,\sigma_{\mathrm{x}}^2}\right) \tag{23}$$

The noise in (8) is modeled as additive white Gaussian noise with a pdf given by:

$$\wp_E(e_n) \;=\; \mathcal{N}(e_n;\,0,\sigma_{\mathrm{e}}^2) \tag{24}$$

The Message Passing (MP) is then included to describe the steps:

$$\boldsymbol{\alpha} \;\rightarrow\; \mathbf{z} = \mathbf{A}\,\boldsymbol{\alpha} \;\rightarrow\; \mathbf{y} = \mathbf{z} + \mathbf{e} = \mathbf{A}\,\boldsymbol{\alpha} + \mathbf{e} \tag{25}$$

remembering that $\mathbf{x} = \boldsymbol{\Phi}\boldsymbol{\alpha}$. When $\wp_X(x)$ is unknown, the expression above represents an assumption of the pdf. Other pdfs may be used such as Laplace and Bernoulli-Gaussian Mixture models [80]. The output can be based on any separable distribution. In the special case of a Laplace pdf, the algorithm can be reduced to a simple thresholding algorithm similar to (16) with an additional correction term in the argument to the threshold operator (18). The algorithmic complexity of the AMP algorithm based on Bernoulli-Gaussian input prior and Gaussian output prior is $\mathcal{O}(m\,n)$.

[7] IST can also solve a variant of the ℓ_1 minimization problem and as such is not completely distinct from them.

4) Reference Method: Interpolation: Interpolation using irregularly spaced samples is a widely studied topic and used in diverse disciplines such as signal processing [84], [85], computational geometry [86], and geoscience [87]. Here The computationally simplest approaches to interpolation are nearest neighbor interpolation, where the nearest known pixels are simply copied to the unknown pixel locations, and linear interpolation, where nearby pixels are linearly combined to provide the values for the missing pixels. The weights used in the linear combination are often empirically chosen such that an average of the neighboring pixels is obtained or they depend upon the distance between the pixels as is the case with Kriging linear interpolation [87]. The weights can also be analytically chosen to satisfy e.g., well-established sampling theorems in shift-invariant spaces such as non-uniform interpolation with b-splines [88] and sinc kernels [89]. Another common approach is to use Delaunay triangularization, where the surface area is subdivided into non-overlapping triangles. The vertices of the triangles are assigned the measured points, and any point within a triangle can be obtained by (non)-linear interpolation methods such as linear, cubic, and nearest-neighbor interpolation between its three vertices [86]. In this study we use the latter interpolation method as a reference to compare the sparse approximation methods against.

IV. Experiments

In order to give an overview of the possibilities of image reconstruction from sparsely sampled images we have conducted an extensive set of experiments to showcase the capabilities of different reconstruction approaches presented in Section III. The experiments cover basic variants of the involved reconstruction algorithms, i.e., no attempts were made to exploit special structure in the images or apply dictionary learning etc. It is therefore also likely that specialization of the algorithms may offer further reconstruction quality improvements or mitigation of some of the impairments described in Section II-C.

A. Quality Indicators

In order to assess the reconstruction quality in the experiments, we apply two standardized image quality indicators. The first is peak signal-to-noise ratio (PSNR):

$$\mathrm{PSNR} \;=\; 10\,\log_{10}\left(\frac{P^2}{\sum_{k=0}^{h-1}\sum_{l=0}^{w-1}\left(\mathbf{X}_{(k,l)}-\hat{\mathbf{X}}_{(k,l)}\right)^2}\right) \tag{26}$$

The value P is the maximum possible value of a pixel in $\mathbf{X}$, i.e. $P = 1$ according to the numeric representation of the images described in Section IV-C.

The second metric is the structural similarity (SSIM) index which we use according to the definition in [90]. In particular, we use: window size 7, $K_1 = 0.01$, $K_2 = 0.03$, $C_3 = \frac{C_2}{2}$, and $\alpha = \beta = \gamma = 1$, cf. [90, Eq. (13)].

All reconstructed images are scaled to have pixel values in the range $[0, 1]$ prior to application of the PSNR and SSIM indicators.

The color map referred to as "cool-warm" in [91] (exemplified in Figure 1a) is used for visualizing the ground truth images in Figure 2 as well as the reconstructed images. We have found through perceptual evaluation of the ground truth images that this color map is better for discerning image details otherwise lost in the color map traditionally applied in AFM imaging which is exemplified in Figure 1b.

(a) "Cool-warm" color map.

(b) Traditional AFM color map.

Fig. 1. Color maps for visualization of images.

B. Sampling Pattern

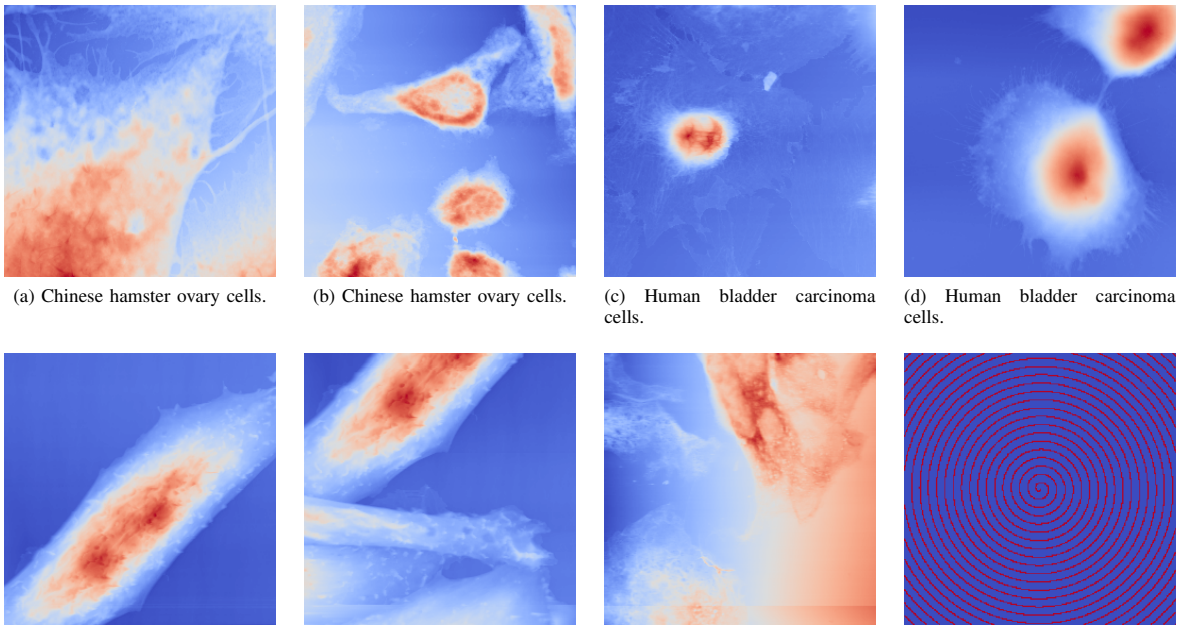
We investigate reconstruction performance under varying density of the applied sampling pattern. The density of the sampling pattern is expressed in terms of an undersampling ratio defined as follows: the undersampling ratio is measured with respect to the length of the scan path as this can reasonably be assumed proportional to the amount of time required to scan the image. We take as reference scan path length the length of the dense raster pattern used to scan the original images in Figure 2. This reference length is approximated as:

$$L_{\mathrm{ref}} \;=\; 2\,w\,h \tag{27}$$

That is, the length of each horizontal line w times the number of lines h, expressed in pixels. The multiplication by 2 stems from the fact that the probe is scanned both back and forth once in each direction for each line counted. This in principle results in two images; one composed of the left-to-right-scanned samples and one composed of the right-to-left-scanned samples. Only one of these images is used as they are usually equivalent (but not completely identical) for practical purposes. The undersampling ratio δ is finally calculated as

$$\delta \;=\; \frac{L}{L_{\mathrm{ref}}} \tag{28}$$

The length L is the length of the applied sampling pattern in units of pixels. In reconstruction experiments involving the spiral sampling pattern, we have simulated a scan path that scans beyond the square region of the original image until the spiral pattern fills the corners of the square as can be seen in Figure 2h. In calculating the resulting undersampling ratio, we also include the parts of the spiral scan path outside the square image region for fairness of comparison. This is done because the AFM equipment would have to traverse these unused regions outside the image region in order not to introduce scanning artifacts by deviating from the smooth curve of the spiral path. Note that (28) is not equal to the undersampling ratio measured in image pixels and (28) reflects the fact that we wish to focus on the potential time savings in applying sparse sampling patterns in AFM.



(a) Chinese hamster ovary cells. (b) Chinese hamster ovary cells. (c) Human bladder carcinoma cells. (d) Human bladder carcinoma cells.

(e) Chinese hamster ovary cells. (f) Chinese hamster ovary cells. (g) Chinese hamster ovary cells. (h) Sampling pattern example.

Fig. 2. (a)–(g) The seven ground truth images used in reconstruction experiments here shown before de-tilting; (h) shows an example of the spiral sampling pattern with $\delta = 0.1$.

C. Image Material

As examples of cell images we have selected the seven images shown in Figure 2. The images have originally been scanned for a size of 512×512 pixels, but have been subsequently decimated to 256×256 pixels to reduce the computational complexity of the reconstruction[8]. The images are stored and processed as double precision (64-bit) floating point values in the interval $[0, 1]$. Images (a), (c), (e), and (f) have been acquired in acoustic AC mode; images (b), (d), and (g) have been acquired in contact mode. The images have been acquired on Keysight Technologies ILM6000 and 7500 AFM equipment. The original image files are available along with this paper[9].

We demonstrate the performance of the reconstruction algorithms on images sampled using raster-, respectively, spiral-shaped scanning paths. In the experiments, we did not have access to images scanned along a spiral scan path. For this reason, the measurements used in the reconstruction experiments were constructed as follows: the original images were acquired using a dense raster scan path with one line per line of pixels in the resulting image; spiral-scanned measurements were simulated by picking pixels from the original images in a spiral-shaped pattern as illustrated in Figure 2h; for fairness of comparison, the raster-scanned measurements used in the experiments were similarly picked as horizontal lines–joined at the ends by vertical segments–of pixels from the original images. The undersampling ratio defined in Section IV-A is varied among the following values:

$$\delta \in \{0.1 + n \cdot 0.025 \mid n = 0, 1, \ldots, 8\} \tag{29}$$

In the reconstruction experiments, the images have been de-tilted prior to reconstruction. This is done by least-squares-fitting a plane through the available measurements according to the applied sampling pattern. The fitted plane is then subtracted from the measurements. When evaluating PSNR or SSIM of the reconstructed images, the reconstructed images are compared to the de-tilted original.

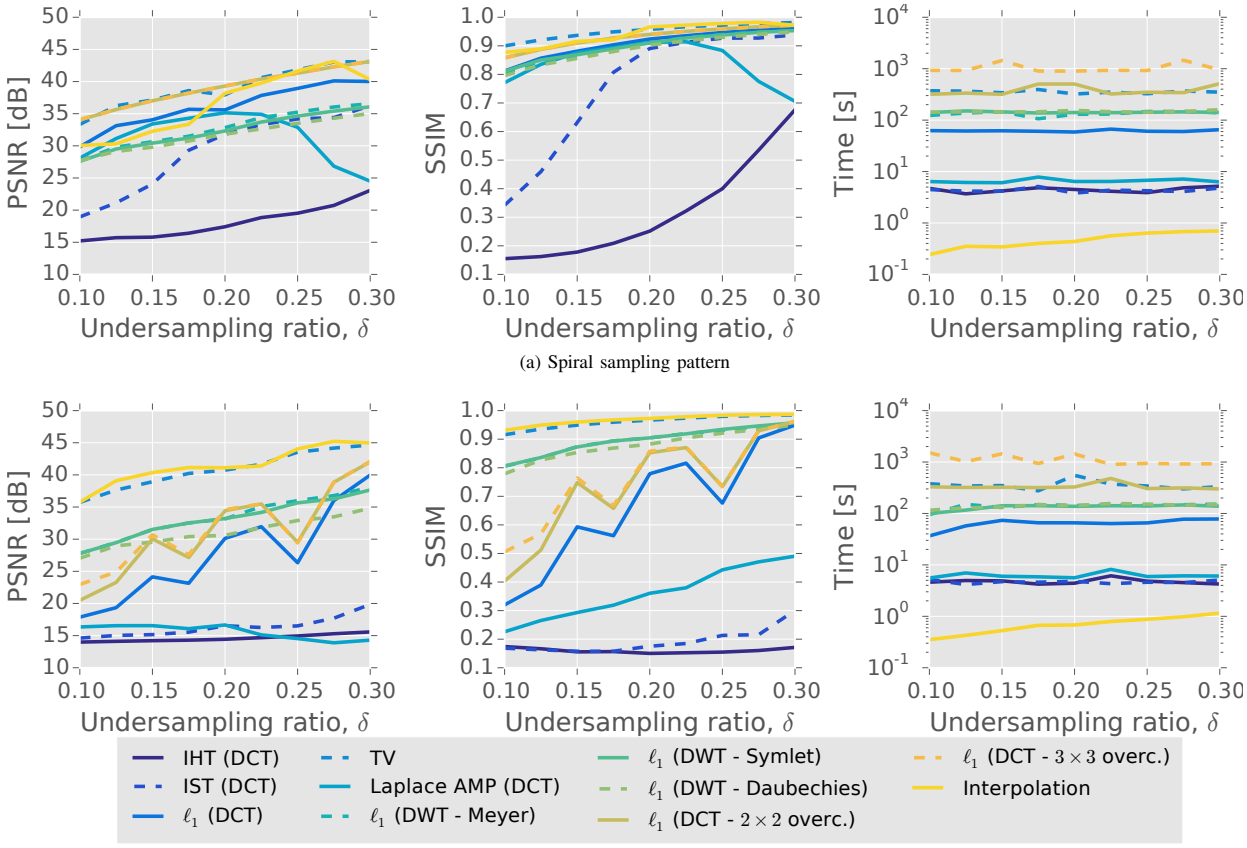
There was not sufficient information available regarding the physical experimental set-up used in producing the images in Figure 2 to analyze and estimate the amount of measurement noise in the images as described in, e.g. [19]. When available, such estimates of measurement noise should be included appropriately in the reconstruction algorithms. For example, in the cases of (13) and (9) the estimated noise variance can be used to determine ϵ.

D. Algorithm Implementations

For each of the sampling patterns (raster or spiral) and each of the undersampling ratios, we reconstruct each of the seven images using the following reconstruction algorithms: ℓ_1-minimization (Section III-B1), AMP – with Laplace prior and with Bernoulli-Gaussian prior (Section III-B3), IST and IHT (Section III-B2), TV minimization (Section III-B1), cubic interpolation via Delaunay triangulation (Section III-B4). The simulation code has been implemented in Python, which is a popular, open and suitable ecosystem for scientific computing

[8]Most of the tested algorithms can actually handle images of size 512×512, but the Bernoulli-Gaussian AMP algorithm described in Section III-B3 was unable to handle larger images on the available hardware due to memory requirements.

[9]http://dx.doi.org/10.5281/zenodo.17573



(a) Spiral sampling pattern

(b) Raster sampling pattern

Fig. 3. Comparison of reconstruction PSNR, SSIM, and run-time across the tested reconstruction algorithms, at 256×256 pixels.

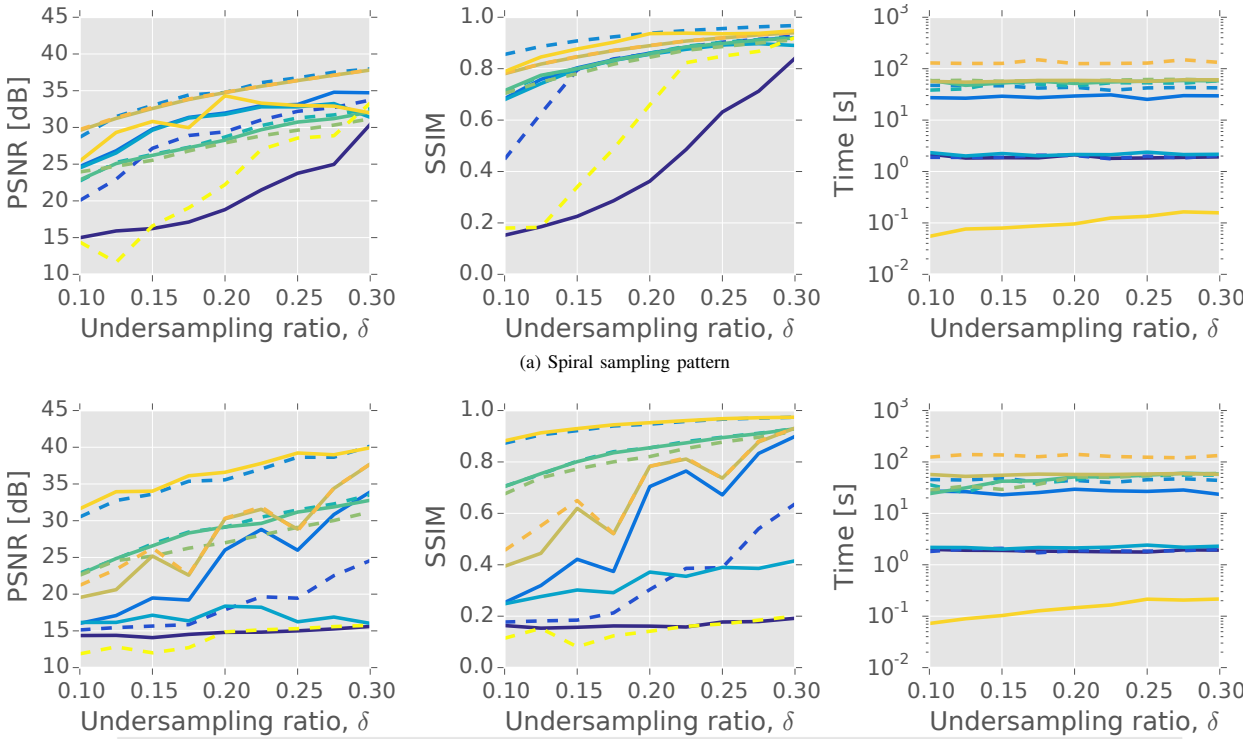
[92], [93]. Python also supports the ideas of reproducible research which are considered important in the present simulation rich context [94], [95]. The complete Python code used to conduct the reconstruction experiments is available along with its results[10]. Interaction with the data files from the AFM equipment, generation of sampling patterns, measurement and dictionary matrices (for the algorithms utilizing the latter) as well as evaluation of quality indicators and visualization of reconstruction results are handled through the Magni software package[11] described in [96]. Some of the applied reconstruction algorithms are provided as part of the Magni package while others are available in other packages; see details in the following:

1) ℓ_1*-minimization* For reconstruction via ℓ_1 minimization, we solved (9). Reconstructions using ℓ_1-minimization were performed using an orthogonal DCT dictionary as well as over-complete DCT dictionaries. The over-complete dictionaries applied 2 and 3 times oversampling along each dimension of the frequency domain. This amounts to a total of 4 and 9 times oversampling, respectively. The over-complete DCT dictionaries were implemented by applying zero-padding in the image domain. Additionally, reconstructions using ℓ_1-minimization were performed using orthogonal DWT dictionaries with three different types of wavelets: Meyer, Daubechies, and symlets. All three wavelet types were used in their longest available form in the PyWavelets toolbox for Python[12]. Wavelets with the longest available filters were chosen to mitigate possible problems with measurement-dictionary coherence and non-empty null-space discussed in Section III-A.
The solver iterates until the constraint in (9) is met or a limit of 2000 iterations has been reached. Other settings in the solver may influence the stopping conditions; these have been left at their standard values.
2) *Approximate Message Passing* For reconstruction via AMP, we have implemented the algorithm in Python for a Laplace as well as a Bernoulli-Gaussian (BG) prior. The code is included in the software accompanying this paper. The algorithms are iterated until they reach an upper limit of 300 iterations or if the residual:

$$\|\mathbf{y} - \mathbf{A}\,\hat{\boldsymbol{\alpha}}\|_2 \;<\; \epsilon \|\mathbf{y}\|_2, \quad \epsilon \;=\; 10^{-3} \tag{30}$$

Our current implementation of Bernoulli-Gaussian AMP (BG-AMP) cannot handle images in 256×256 pix-

[10]http://dx.doi.org/10.5281/zenodo.32959 and .../zenodo.32958.
[11]http://dx.doi.org/10.5278/VBN/MISC/Magni
[12]https://github.com/PyWavelets/pywt



(a) Spiral sampling pattern

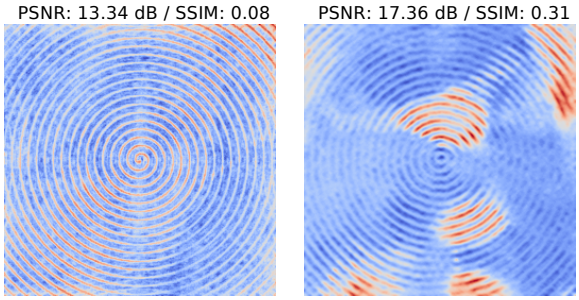
(b) Raster sampling pattern

Fig. 4. Comparison of reconstruction PSNR, SSIM, and run-time across the tested reconstruction algorithms, at 128×128. This figure serves to compare BG-AMP to the other algorithms.

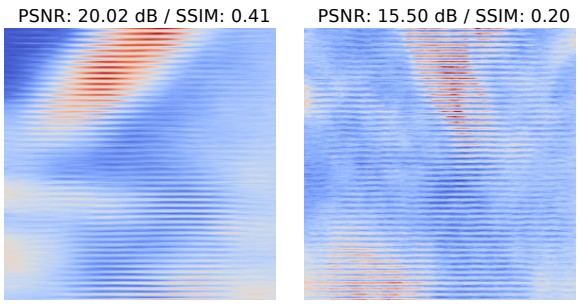
els due to severe memory requirements. Therefore our experiments with this particular algorithm have been conducted with the images in Figure 2 decimated to 128×128 pixels. All experiments with the other algorithms have additionally been repeated at this image size for the purpose of comparison with this algorithm. Reconstructions using AMP were only performed using an orthogonal DCT dictionary.

3) *Iterative Soft and Hard Thresholding* For reconstruction via IST as well as IHT, we have implemented these algorithms in Python. The code is included in the software accompanying this paper. The algorithms are iterated until they reach an upper limit of 300 iterations or meet the condition in (30).
   Reconstructions using IHT and IST were only performed using an orthogonal DCT dictionary.
4) *Total Variation* For reconstruction via TV minimization, we solved (13) using Douglas-Rachford splitting. We used the solver implemented in the PyUNLocBox package for Python, referenced in Section III-B1. The solver iterates until the constraint in (13) is met or a limit of 2000 iterations has been reached. Other settings in the solver may influence the stopping conditions; these have been left at their standard values.
5) *Interpolation* For reconstruction via interpolation, we used cubic Bezier polynomial interpolation over triangles formed by triangulating the available measurements $\mathbf{\Phi x}$ as implemented in the `scipy.interpolate` Python module [97].

For the convex optimization-based reconstruction approaches (items 1 and 4 above) we have repeated the reconstructions over a wide range of the regularization parameter ϵ and selected the reconstructions with highest PSNR/SSIM, averaged over all images for each algorithm and undersampling ratio δ. Similarly for the IHT and IST algorithms, we repeated the reconstructions over a wide range of the sparsity parameter k and selected the reconstructions with highest PSNR/SSIM, averaged over all images for each algorithm and undersampling ratio δ. This was done in order to provide a fair basis for comparison between the different algorithms since the measurement noise is unknown and thus unavailable to estimate ϵ, as explained in Section IV-C. Also, the images are not truly sparse – merely well approximated as such. There is thus no true parameter k available and this parameter is very problem-dependent. This choice of regularization parameters is not feasible in practice since the original image is unavailable for



(a) Image 2a: IHT, spiral. (b) Image 2b: IST, spiral.



(c) Image 2f: ℓ_1-min., raster. (d) Image 2g: Laplace AMP, raster.

Fig. 5. Examples of low-quality reconstructions at $\delta = 0.1$. Images reconstructed from measurements of the ground truth images 2 (a), (b), (f), and (g). 256×256 pixels.

evaluating the reconstruction quality. It was however chosen in order to compare the mentioned algorithms on an equal footing.

For each reconstruction experiment we have measured the reconstruction time as a practical indicator of the run-time complexity. Reconstruction time can of course vary depending on the specific algorithm and the implementation of it used to reconstruct the image. Our reconstruction results could thus possibly be reproduced with different measured run-times. However, the measured reconstruction times provide a useful indicator of what is practically achievable.

V. Results

For each of the reconstruction approaches, the performance in terms of both PSNR and SSIM at 256×256 pixels is plotted in Figure 3 along with reconstruction time against the tested undersampling ratios, δ. All three panels display results from the reconstructions resulting in the best PSNR among the tested regularization parameters. Figure 3a shows results for the spiral sampling pattern and 3b shows results for the raster sampling pattern, cf. Section IV-B.

As expected, reconstructed image quality in terms of PSNR as well as SSIM decreases as δ decreases. At the low undersampling end, $\delta = 0.1$, this results in reconstructions of very low PSNR/SSIM, a few examples of which are shown in Figure 5.

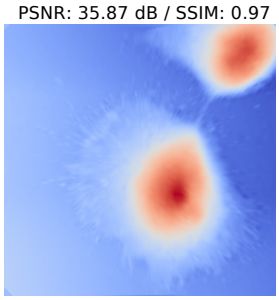
Figure 3 shows that interpolation and TV minimization reconstruct the images best among the tested algorithms, both in terms of PSNR and SSIM. As shown in Figure 3b, interpolation results in the highest PSNR as well as SSIM averaged over the seven ground truth images, for the raster sampling pattern. TV reconstruction with the raster pattern results in slightly lower PSNR than interpolation, $0.6\,\mathrm{dB}$ on average. The spiral sampling pattern results in lower PSNR for both interpolation and TV minimization, $4.9\,\mathrm{dB}$ worse on average for interpolation while only $1.7\,\mathrm{dB}$ worse for TV. This also means that reconstruction by TV minimization results in $2.5\,\mathrm{dB}$ higher PSNR than interpolation for the spiral sampling pattern.

Reconstruction by ℓ_1-minimization with DCT dictionaries and the spiral sampling pattern results in the highest PSNR performance after TV optimization, where 2×2 and 3×3 overcomplete DCT result in PSNR performance close to that of TV optimization. For the spiral sampling pattern, interpolation only performs comparable to ℓ_1-minimization with orthogonal DCT dictionary. The ℓ_1-minimization with DWT dictionary performs substantially worse than for DCT dictionary with the spiral sampling pattern. Here the Meyer wavelet is slightly better than the symlet, which is again slightly better than the Daubechies wavelet. Laplace AMP exhibits a trend in PSNR performance that deviates from that of the other algorithms, deteriorating severely for $\delta > 0.2$. This is not intuitive as the algorithms have more information available for higher δ. This is likely due to unfavorable configuration of this algorithm's parameters. IST performs substantially worse than the above algorithms for $\delta < 0.2$ but comparable to ℓ_1-minimization with DWT dictionary for $\delta \geq 0.2$. Finally, IHT reconstructs at the lowest PSNR among all of the algorithms at more than $10\,\mathrm{dB}$ below ℓ_1-minimization with DWT dictionary.

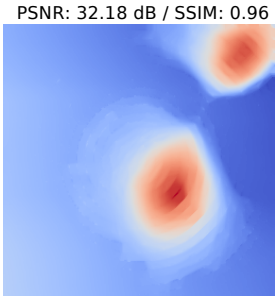
The raster sampling pattern performs much worse than the spiral sampling pattern for all of the DCT dictionary-based methods (ℓ_1-minimization with DCT dictionaries, Laplace AMP, IHT and IST). IHT, IST, and Laplace AMP with the raster sampling pattern benefit very little from increased δ. The described tendencies in PSNR figures are reflected similarly in the SSIM figures. On the other hand, ℓ_1-minimization with DWT dictionaries results in PSNR as well as SSIM figures that are very close the corresponding figures for the spiral sampling pattern. Finally, both TV minimization and interpolation perform similar and best among all of the studied algorithms for the raster sampling pattern. PSNR of the latter two ranges from approximately $35\,\mathrm{dB}$ at $\delta = 0.1$ to approximately $45\,\mathrm{dB}$ at $\delta = 0.3$.

As mentioned in Section IV-D, reconstruction experiments using Bernoulli-Gaussian AMP have only been performed on images at 128×128 pixels. Figure 4 displays results of the same experiments as in Figure 3 at 128×128 pixels for comparison to BG-AMP. Data plotted in Figure 4 stems from the reconstructions resulting in the highest PSNR among the tested regularization parameters[13]. The performance of BG-AMP in terms of PSNR and SSIM lies between that of IHT and IST for the spiral sampling pattern and even slightly below IHT for the raster sampling pattern. Laplace AMP was observed to perform substantially better than BG-AMP

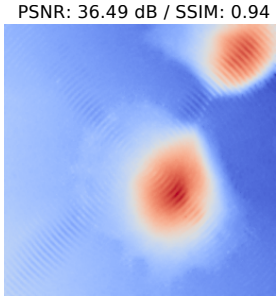
[13]The corresponding regularization parameter values are not necessarily the same as the values resulting in Figure 3.



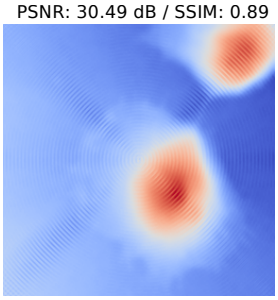
(a) Interp., spiral, $\delta = 0.1$.



(b) TV, spiral, $\delta = 0.1$.

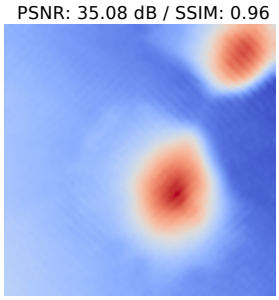


(c) Laplace AMP, spiral, $\delta = 0.15$.

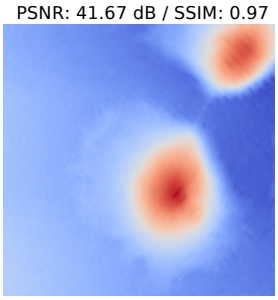


(d) IST, spiral, $\delta = 0.225$.

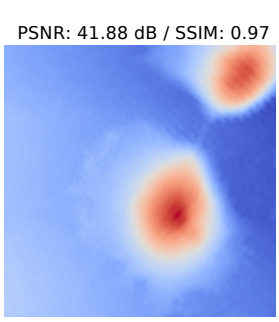
Fig. 6. Examples of reconstructions for the lowest δ yielding SSIM > 0.9. All images reconstructed based on measurements of the image in Figure 2d. 256×256 pixels.



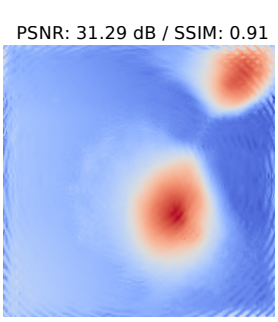
(a) ℓ_1-min. DCT, spiral, $\delta = 0.15$.



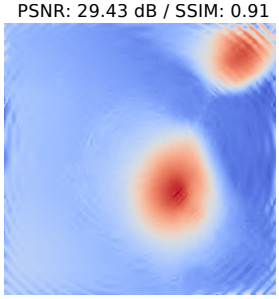
(b) ℓ_1-min. DCT (2×2), spiral, $\delta = 0.1$.



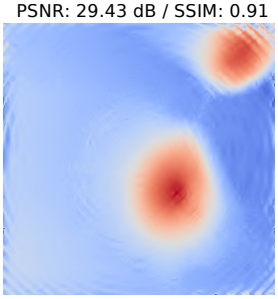
(c) ℓ_1-min. DCT (3×3), spiral, $\delta = 0.1$.



(d) ℓ_1-min. DWT (Daub.), spiral, $\delta = 0.125$.



(e) ℓ_1-min. DWT (Meyer), spiral, $\delta = 0.1$.



(f) ℓ_1-min. DWT (symlet), spiral, $\delta = 0.1$.

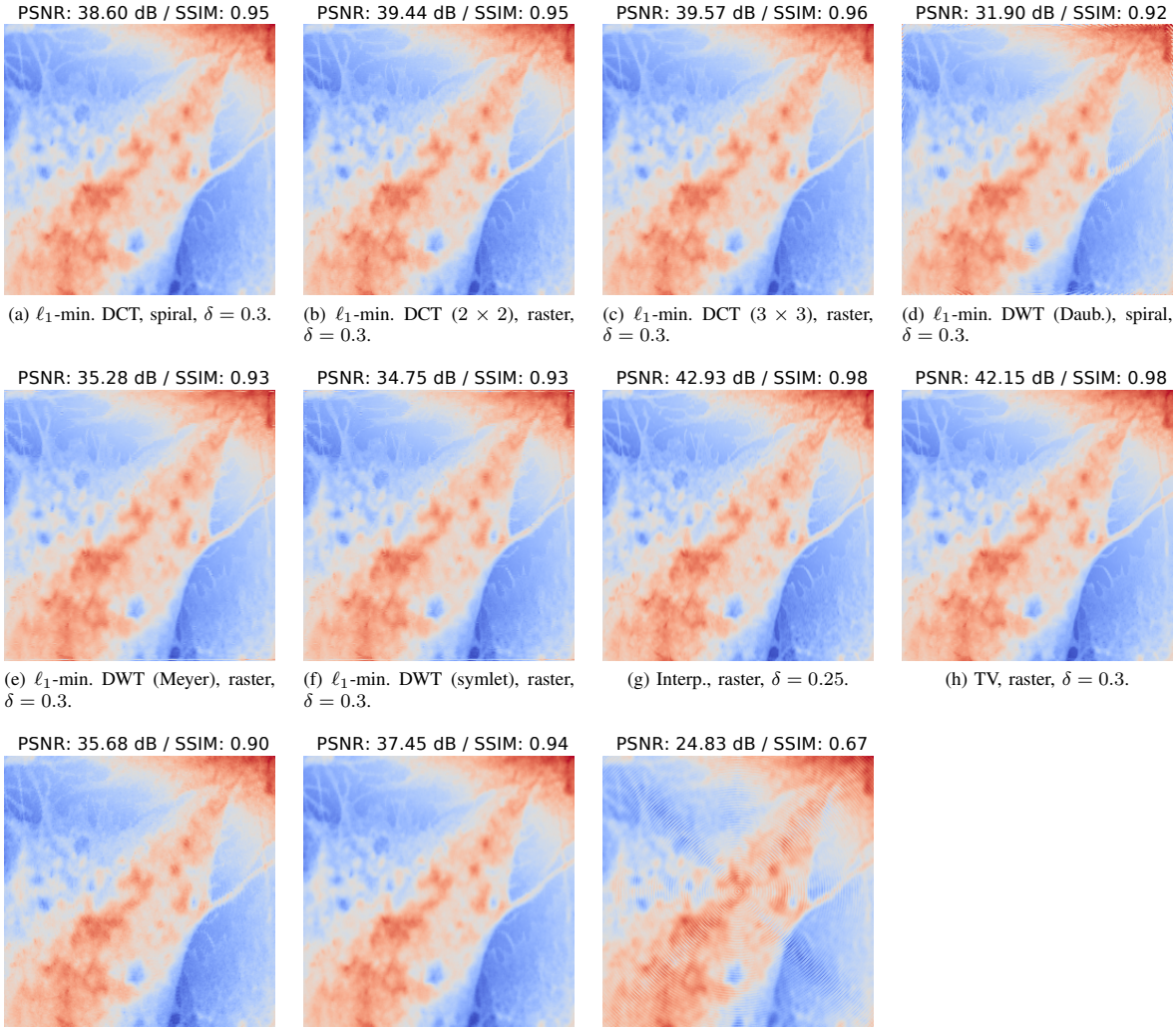
Fig. 7. Examples of reconstructions for the lowest δ yielding SSIM > 0.9. All images reconstructed based on measurements of the image in Figure 2d. 256×256 pixels.

in terms of both PSNR and SSIM. A likely reason for this is that the DCT coefficients of the images have been observed to follow a probability distribution resembling the Laplace distribution rather than the Bernoulli-Gaussian. The remaining algorithms exhibit the same trends as for the 256×256 pixels images in Figure 3, except for Laplace AMP which has a much less outspoken tendency to degrade for larger values of δ than for the 256×256 pixels images.

The different reconstruction algorithms require different amounts of samples to reconstruct images satisfactorily. As examples of reconstructions of reasonable quality, we display an image reconstruction for each of the algorithms for the lowest δ that achieves a SSIM > 0.9 in Figures 6 and 7. These images represent reconstructions of somewhat degraded quality compared to the original where reconstruction artifacts typical of the tested reconstruction algorithms are evident. IHT is left out in Figures 6 and 7 since it reconstructs the image at SSIM < 0.9. BG-AMP is likewise left out since it has only been run for images at 128×128 pixels. Interpolation (Figure 6a) tends to produce artifacts that appear as if small regions of the image are smeared radially outwards from the center. Figure 6b demonstrates how TV reconstruction tends to produce reconstructions of piece-wise constant value – here particularly concentrated around the lines of the raster sampling pattern. The sparse approximation methods (Figures 6c-6d and 7a-7f) tend to leave traces of the sampling pattern in the reconstructed image, which is particularly visible in the reconstructions with DWT dictionaries: Figures 7d-7f.

To exemplify the best performance of the tested algorithms, the best reconstruction in terms of PSNR of the image in Figure 2a is shown for each algorithm in Figure 8. All of the algorithms reconstruct the image at a legible quality but preserve finer details with varying success; interpolation and TV result in the best reconstruction quality both in terms of PSNR and SSIM (Figures 8g and 8h) while the ℓ_1 minimization algorithms with DCT dictionaries perform slightly worse (Figures 8a-8c). ℓ_1 minimization algorithms with DWT dictionaries (Figures 8d-8f) perform somewhat worse than with DCT dictionaries, producing visible edge artifacts in the reconstructed images. IHT (Figure 8k) is the only algorithm among these specific examples which leaves clearly visible sampling pattern artifacts in the reconstructed image and results in low quality.

The results indicate that reconstruction methods favoring



(a) ℓ_1-min. DCT, spiral, $\delta = 0.3$.
(b) ℓ_1-min. DCT (2×2), raster, $\delta = 0.3$.
(c) ℓ_1-min. DCT (3×3), raster, $\delta = 0.3$.
(d) ℓ_1-min. DWT (Daub.), spiral, $\delta = 0.3$.
(e) ℓ_1-min. DWT (Meyer), raster, $\delta = 0.3$.
(f) ℓ_1-min. DWT (symlet), raster, $\delta = 0.3$.
(g) Interp., raster, $\delta = 0.25$.
(h) TV, raster, $\delta = 0.3$.
(i) Laplace AMP, spiral, $\delta = 0.225$.
(j) IST, spiral, $\delta = 0.3$.
(k) IHT, spiral, $\delta = 0.3$.

Fig. 8. Examples of reconstructions with the highest PSNR for each algorithm. All images reconstructed based on measurements of the image in Figure 2a. 256×256 pixels.

image smoothness (interpolation and TV) work slightly better than methods based on sparse approximation with DCT or DWT dictionaries. It is particularly favorable for interpolation that this method was also the fastest to compute among the tested algorithms: approximately 0.3 s-1 s depending on δ (Figure 3, right panel).

Although BG-AMP was demonstrated to work particularly poorly in the examples studied here, this type of algorithm has potential. As the Laplace variant demonstrated, selecting a more appropriate prior (Laplace) distribution of the image transform coefficients can result in better reconstruction. Furthermore, this family of algorithms can be adapted more specifically to different measurement noise distributions than for example the ℓ_1 minimization approaches and may be able to address the impairments described in Section II-C.

We stress here that the sparse approximation reconstruction algorithms were selected to show an overview of the basic form of some popular algorithms. These sparse approximation algorithms can be specialized further to for example take advantage of image structure [98], [99], dictionary learning [41], or sparsity (ℓ_1) in an ensemble of several different dictionaries can be combined [100]. In summary, there is potential for further advances in AFM image reconstruction using sparse approximation methods.

VI. Conclusion

We have proposed to reduce the critical scanning time and probe-specimen interaction by AFM measurement via undersampling achieved through the use of a sparse sampling pattern. In the present study we investigated the raster sampling pattern as well as an undersampling spiral pattern; both

of varying densities. We studied the performance of a number of image reconstruction algorithms applied to measured AFM images of cell material via numerical experiments and evaluated their reconstruction performance in terms of PSNR and SSIM. We compared the central algorithms on a best-case basis over a range of different regularization parameters in order to reduce the effect of the choice of regularization parameters on the reconstruction results.

The studied algorithms include sparse approximation methods with discrete cosine transform and discrete wavelet transform dictionaries as well as total variation. These algorithms were compared to a reference method – cubic interpolation. The experimental results showed that most of the basic forms of sparse approximation algorithms studied could not quite match the reference interpolation method in terms of PSNR and SSIM. Only total variation minimization resulted in comparable PSNR and SSIM. Furthermore, interpolation was the fastest method at $0.3\,\mathrm{s}$-$1\,\mathrm{s}$ depending on undersampling ratio. Based on the tested algorithms and images, it was found that the scan time or probe-specimen interaction can be reduced by a factor of 10 compared to dense raster scanning while retaining a reconstruction $\mathrm{PSNR} \simeq 36\,\mathrm{dB}$, or by a factor of 4 for a reconstruction $\mathrm{PSNR} \simeq 44\,\mathrm{dB}$. These reductions in scan time / probe-specimen interaction are attainable on any existing AFM hardware capable of varying the line density of a horizontal-line raster pattern.

ACKNOWLEDGMENT

The images used in this paper were kindly provided by Keysight Technologies.

REFERENCES

[1] D. P. Allison, N. P. Mortensen, C. J. Sullivan, and M. J. Doktycz, "Atomic force microscopy of biological samples," *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology*, vol. 2, no. 6, pp. 618–634, 2010.

[2] P. Chen, H. Dong, L. Chen, Q. Sun, and D. Han, "Application of atomic force microscopy to living samples from cells to fresh tissues," *Chinese Science Bulletin*, vol. 54, no. 14, pp. 2410–2415, Jul. 2009.

[3] D. J. Müller and Y. F. Dufrêne, "Atomic force microscopy: a nanoscopic window on the cell surface," *Trends in Cell Biology*, vol. 21, no. 8, pp. 461–469, 2011.

[4] Y. K. Yong, A. Bazaei, S. O. R. Moheimani, and F. Allgöwer, "Design and Control of a Novel Non-Raster Scan Pattern for Fast Scanning Probe Microscopy," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Kachsiung, Taiwan, Jul., 11 – 14, 2012, pp. 456–461.

[5] G. Agarwal and T. M. Nocera, *The Nanobiotechnology Handbook*. CRC Press, 2012, ch. Atomic Force Microscopy, pp. 369–392.

[6] G. Schitter and M. J. Rost, "Scanning probe microscopy at video-rate," *Materials Today*, vol. 11, no. 0, pp. 40–48, 2008.

[7] B. Song, N. Xi, R. Yang, K. W. C. Lai, and C. Qu, "Video Rate Atomic Force Microscopy (AFM) Imaging using Compressive Sensing," in *11th IEEE International Conference on Nanotechnology*, Portland, Oregon, USA, Aug. 15-18, 2011, pp. 1056–1059.

[8] Y. K. Yong, S. O. R. Moheimani, and I. R. Petersen, "High-speed cycloid-scan atomic force microscopy," *Nanotechnology*, vol. 21, no. 36, p. 4, Aug. 2010.

[9] T. Tuma, J. Lygeros, V. Kartik, A. Sebastian, and A. Pantazi, "High-speed multiresolution scanning probe microscopy based on Lissajous scan trajectories," *Nanotechnology*, vol. 23, no. 18, p. 9, Apr. 2012.

[10] T. Tuma, A. Sebastian, J. Lygeros, and A. Pantazi, "The four pillars of nanopositioning for scanning probe microscopy: The position sensor, the scanning device, the feedback controller, and the reference trajectory," *IEEE Control Systems*, vol. 33, no. 6, pp. 68–85, Dec. 2013.

[11] B. Bhushan and O. Marti, "Scanning Probe Microscopy – Principle of Operation, Instrumentation, and Probes," in *Springer Handbook of Nanotechnology*, B. Bhushan, Ed. Springer Berlin Heidelberg, 2010, ch. 21, pp. 573–617.

[12] D. Y. Abramovitch, S. B. Andersson, L. Y. Pao, and G. Schitter, "A Tutorial on the Mechanisms, Dynamics, and Control of Atomic Force Microscopes," in *American Control Conference*, New York City, USA, Jul. 11-13, 2007, pp. 3488–3502.

[13] T. Tuma, J. Lygeros, A. Sebastian, and A. Pantazi, "Optimal scan trajectories for high-speed scanning probe microscopy," in *American Control Conference (ACC)*, Montréal, Canada, Jun. 27 – 29, 2012, pp. 3791–3796.

[14] I. A. Mahmood, S. O. R. Moheimani, and B. Bhikkaji, "A New Scanning Method for Fast Atomic Force Microscopy," *IEEE Transactions on Nanotechnology*, vol. 10, no. 2, pp. 203–216, Mar. 2011.

[15] T. Ando, N. Kodera, E. T. A. Maruyama, K. Saito, and A. Toda, "A high-speed atomic force microscope for studying biological macromolecules," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 22, p. 12468–12472, Oct. 2001.

[16] S. B. Andersson and L. Y. Pao, "Non-Raster Sampling in Atomic Force Microscopy: A Compressed Sensing Approach," in *American Control Conference (ACC)*, Montréal, Canada, Jun. 27-29, 2012, pp. 2485–2490.

[17] T. R. Meyer, D. Ziegler, C. Brune, A. Chen, R. Farnham, N. Huynh, J.-M. Chang, A. L. Bertozzi, and P. D. Ashby, "Height drift correction in non-raster atomic force microscopy," *Ultramicroscopy*, vol. 137, pp. 48–54, Feb 2014.

[18] A. Chen, A. L. Bertozzi, P. D. Ashby, P. Getreuer, and Y. Lou, "Enhancement and Recovery in Atomic Force Microscopy Images," in *Excursions in Harmonic Analysis*, ser. Applied and Numerical Harmonic Analysis,, T. D. Andrews, R. Balan, J. J. Benedetto, W. Czaja, and K. A. Okoudjou, Eds. Birkhäuser Boston, 2013, vol. 2, pp. 311–332.

[19] A. Labuda, M. Lysy, W. Paul, Y. Miyahara, P. Grütter, R. Bennewitz, and M. Sutton, "Stochastic noise in atomic force microscopy," *Phys. Rev. E*, vol. 86, p. 031104, Sep. 2012.

[20] J. F. Claerbout and F. Muir, "Robust modeling with erratic data," *Geophysics*, vol. 38, no. 5, pp. 826–844, 1973.

[21] S. Levy and P. K. Fullagar, "Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution," *Geophysics*, vol. 46, no. 9, pp. 1235–1243, 1981.

[22] F. Santosa and W. W. Symes, "Linear inversion of band-limited reflection seismograms," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, 1986.

[23] E. J. Candès, J. Romberg, and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction From Highly Incomplete Frequency Information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[24] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[25] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[26] E. J. Candès and M. B. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[27] J. Romberg, "Imaging via Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, Mar. 2008.

[28] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1 1949.

[29] E. J. Candès, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9–10, pp. 589–592, 2008.

[30] S. Nam, M. Davies, M. Elad, and R. Gribonval, "The cosparse analysis model and algorithms," *Applied and Computational Harmonic Analysis*, vol. 34, no. 1, pp. 30–56, 2013.

[31] D. L. Donoho and J. Tanner, "Precise Undersampling Theorems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 913–924, Jun. 2010.

[32] T. L. Jensen, T. Arildsen, J. Østergaard, and T. Larsen, "Reconstruction of Undersampled Atomic Force Microscopy Images : Interpolation versus Basis Pursuit," in *International Conference on Signal-Image Technology&Internet-Based Systems (SITIS)*, Kyoto,Japan, Dec. 2 – 5, 2013, p. 6.

[33] G. Strang, *Introduction to Linear Algebra*, 2nd ed. Wellesley-Cambridge Press, 1998.

[34] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[35] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, Jun. 2010.

[36] S. G. Mallat and Z. Zhang, "Matching Pursuits With Time-Frequency Dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[37] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit," *Siam Review*, vol. 43, no. 1, pp. 129–159, Mar. 2001.

[38] W. D. Ray and R. M. Driver, "Further Decomposition of the Karhunen-Loève Representation of a Stationary Random Process," *IEEE Transactions on Information Theory*, vol. IT-16, no. 6, pp. 663–668, Nov. 1970.

[39] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Blackwell, 2000.

[40] B. A. Olshausen and D. J. Field, "Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.

[41] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[42] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, USA: Wellesley-Cambridge Press, 1996.

[43] G. K. Wallace, "The JPEG Picture Compression Standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[44] H. S. Anderson, J. Ilic-Helms, B. Rohrer, J. Wheeler, and K. Larson, "Sparse imaging for fast electron microscopy," in *Computational Imaging XI, Proceedings of the SPIE*, vol. 8657, Burlingame, California, USA, Feb. 3 2013, pp. (86 570C–1)–(86 570C–12).

[45] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 Still Image Compression Standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[46] L. He and L. Carin, "Exploiting Structure in Wavelet-Based Bayesian Compressive Sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 9, pp. 3488–3497, Sep. 2009.

[47] S. Som and P. Schniter, "Compressive Imaging Using Approximate Message Passing and a Markov-Tree Prior," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3439–3448, Jul. 2012.

[48] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2345–2356, Sep. 2010.

[49] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[50] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[51] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[52] D. Bertsekas, *Convex Optimization Algorithms*. Athena Scientific, 2015.

[53] P. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, ser. Springer Optimization and Its Applications, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. Springer New York, 2011, pp. 185–212.

[54] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[55] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 250–278, 2011.

[56] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods," *Numerical Algorithms*, vol. 53, no. 1, pp. 67–92, 2010.

[57] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[58] D. Needell and R. Ward, "Stable Image Reconstruction Using Total Variation Minimization," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 1035–1058, 2013.

[59] D. Ziegler, T. R. Meyer, R. Farnham, C. Brune, A. L. Bertozzi, and P. D. Ashby, "Improved accuracy and speed in scanning probe microscopy by image reconstruction from non-gridded position sensor data," *Nanotechnology*, vol. 24, no. 33, p. 335703, 2013.

[60] P. Combettes and J. Pesquet, "Image restoration subject to a total variation constraint," *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1213–1222, Sep. 2004.

[61] T. Goldstein and S. Osher, "The Split Bregman Method for L1-Regularized Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009.

[62] P. Combettes and J. Pesquet, "A douglas-rachford splitting approach to nonsmooth convex variational signal recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, Dec. 2007.

[63] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.

[64] S. Becker, E. J. Candès, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 165–218, 2011.

[65] T. Blumensath, M. E. Davies, and G. Rilling, "Greedy algorithms for compressed sensing," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, ch. 8, pp. 348–393.

[66] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Twenty-Seventh Asilomar Conference on Signals,Systems and Computers*, vol. 1, Pacific Grove,California,USA, Nov. 1 – 3, 1993, pp. 40–44.

[67] T. Blumensath and M. E. Davies, "Iterative Thresholding for Sparse Approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, Sep. 2008.

[68] I. Daubechies, M. Defrise, and C. D. Mol, "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

[69] W. Dai and O. Milenkovic, "Subspace Pursuit for Compressive Sensing Signal Reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.

[70] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, p. 301–321, May 2009.

[71] A. Maleki and D. L. Donoho, "Optimally Tuned Iterative Reconstruction Algorithms for Compressed Sensing," *IEEE Journal Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 330–341, Apr. 2010.

[72] J. D. Blanchard and J. Tanner, "Performance comparisons of greedy algorithms in compressed sensing," *Numerical Linear Algebra with Applications*, 2014.

[73] A. Maleki and A. Montanari, "Analysis of Approximate Message Passing Algorithm," in *44th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New York, USA, Mar. 17–19, 2010.

[74] J. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," in *46th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New Jersey, USA, Mar. 21 – 23, 2012, p. 6.

[75] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of approximate message passing with arbitrary matrices," in *Information Theory (ISIT), 2014 IEEE International Symposium on*, June 2014, pp. 236–240.

[76] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, p. 18914–18919, Nov. 2009.

[77] ——, "Message Passing Algorithms for Compressed Sensing: I. Motivation and Construction," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5.

[78] ——, "Message Passing Algorithms for Compressed Sensing: II. Analysis and Validation," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5.

[79] S. Rangan, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, Jul. 31 – Aug. 5, 2011, pp. 2168–2172.

[80] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.

[81] A. Maleki and R. Baraniuk, "Least Favorable Compressed Sensing Problems for First-Order Methods," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, Jul. 31 – Aug. 5, 2011, pp. 134–138.

[82] V. Smidl and A. Quinn, *The Variational Bayes Method in Signal Processing*. Springer, 2006.
[83] M. Lighthill, *An Introduction to Fourier Analysis and Generalised Functions*. Cambridge University Press, 1958.
[84] S. S. Goha and I. G. H. Ong, "Reconstruction of bandlimited signals from irregular samples," *Signal Processing*, vol. 46, no. 3, pp. 315–329, 1995.
[85] L. P. Yaroslavsky, G. Shabat, B. G. Salomon, I. A. Ideses, and B. Fishbain, "Nonuniform sampling, image recovery from sparse data and the discrete sampling theorem," *J. Opt. Soc. Am. A*, vol. 26, no. 3, pp. 566–575, Mar. 2009.
[86] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer Berlin Heidelberg, 1997.
[87] I. Clark, *Practical Geostatistics*. Alloa, Scotland: Geostokos Limited, 1979.
[88] S. Lee, G. Wolberg, and S. Y. Shin, "Scattered data interpolation with multilevel B-splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 228–244, Jul. 1997.
[89] A. Aldroubi, "Non-uniform weighted average sampling and reconstruction in shift-invariant and wavelet spaces," *Applied and Computational Harmonic Analysis*, vol. 13, no. 2, pp. 151–161, 2002.
[90] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[91] K. Moreland, "Diverging Color Maps for Scientific Visualization," in *Advances in Visual Computing: 5th International Symposium, ISVC*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, Eds. Las Vegas, NV, USA,: Springer Berlin Heidelberg, Nov. 30 – Dec. 2 2009, vol. 5876, pp. 92–103, proceedings Part II.
[92] F. Pérez, B. E. Granger, and J. D. Hunter, "Python: An ecosystem for scientific computing," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 13–21, Mar.–Apr. 2011.
[93] T. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.
[94] P. Vandewalle, J. Kovačević, and M.Vetterli, "Reproducible research in signal processing [what, why, and how]," *IEEE Signal Processing Magazine*, pp. 37–47, May 2009.
[95] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000.
[96] C. S. Oxvig, P. S. Pedersen, T. Arildsen, J. Østergaard, and T. Larsen, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images," *Journal of Open Research Software*, vol. 2, no. 1, p. e29, Oct. 2014.
[97] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2015-02-25].
[98] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-Based Compressive Sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, Apr. 2010.
[99] M. F. Duarte and Y. C. Eldar, "Structured Compressed Sensing: From Theory to Applications," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053–4085, Sep. 2011.
[100] J. Yang, Y. Zhang, and W. Yin, "A fast alternating direction method for tvl1-l2 signal reconstruction from partial fourier data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 288–297, Apr. 2010.

Thomas Arildsen (S'03–M'04–S'06–M'10) received the M.Sc.E.E. and Ph.D. degrees from Aalborg University, Aalborg, Denmark, in 2004 and 2010, respectively. From 2004 to 2006, he worked as project engineer at RTX Telecom A/S in Aalborg, Denmark. He has been a visiting Ph.D. student at University of Miami, United States in 2007. He has worked at Aalborg University since 2010, as associate professor since 2014. His research interests include compressed sensing, image processing, analog-to-information conversion, scientific computing, statistical signal processing, estimation, communication signal processing.

Christian Schou Oxvig (S'14) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Aalborg University in 2011 and 2013, respectively. He is currently a Ph.D. student in the Signal and Information Processing Section, Department of Electronic Systems, Aalborg University. His research interests include scientific computing, statistical signal processing, and reproducibility in computer and simulation experiments.

Patrick Steffen Pedersen (M'15) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Aalborg University in 2011 and 2013, respectively. He is currently pursuing the Ph.D. degree at the Signal and Information Processing Section of the Department of Electronic Systems at Aalborg University. His current research interests lie in the area of signal processing and compressed sensing.

Jan Østergaard (S'98–M'99–SM'11) received the M.Sc.E.E. from Aalborg University, Aalborg, Denmark, in 1999 and the PhD degree (cum laude) from Delft University of Technology, Delft, The Netherlands, in 2007. From 1999 to 2002, he worked as an R&D Engineer at ETI A/S, Aalborg, Denmark, and from 2002 to 2003, he worked as an R&D Engineer at ETI Inc., Virginia, United States. Between September 2007 and June 2008, he worked as a post-doctoral researcher at The University of Newcastle, NSW, Australia. From June 2008 to March 2011, he worked as a post-doctoral researcher/Assistant Professor at Aalborg University. Since 2011 he has been an Associate Professor at Aalborg University. He has been a visiting researcher at Tel Aviv University, Tel Aviv, Israel, and at Universidad Técnica Federico Santa María, Valparaíso, Chile. He has received a Danish Independent Research Council's Young Researcher's Award, a best PhD thesis award by the European Association for Signal Processing (EURASIP), and fellowships from the Danish Independent Research Council and the Villum Foundation's Young Investigator Programme. He is an Associate Editor of EURASIP Journal on Advances in Signal Processing.

Torben Larsen (S'88–M'99–SM'04) received the M.Sc.E.E. and Dr.Techn. degrees from Aalborg University, Aalborg, Denmark, in 1988 and 1998, respectively. Since 2001, he has been a Full Professor at Aalborg University in electronic circuits, signals and systems theory. He has industrial experience working as senior engineer at Bosch Telecom and Siemens Mobile Phones. He was member in 2005-2010 and vice-chairman in 2009-2010 of the Danish Research Council for Technology and Production Sciences. In 2011 he was appointed director of the doctoral school at The Faculty of Engineering and Science, Aalborg University, with more than 650 enrolled PhD students. He has authored or co-authored over 130 peer-reviewed journal and conference papers and contributed to four internationally published books. He received the Spar Nord Research Prize in 1999 and "Aalborg University Teacher of the year 2013". Since 2007 he has been member of the Academy of Technical Sciences, Denmark. He has supervised approx. 25 PhD students. He has been Vice Dean at the Faculty of Engineering and Science, Aalborg University since 2015. His recent research interests mainly include scientific computing, compressive sensing, numerical algorithms etc. in the areas of signals and systems theory.

Paper B

Structure Assisted Compressed Sensing Reconstruction of Undersampled AFM Images

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen

The paper has been published in
Ultramicroscopy, vol. 172, pp. 1–9, Jan. 2017.
doi:10.1016/j.ultramic.2016.09.011

The succeeding 9 non-blank pages constitute the published manuscript which is subject to the following copyright notice:

Ultramicroscopy 172 (2017) 1–9

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

Structure assisted compressed sensing reconstruction of undersampled AFM images

Christian Schou Oxvig*, Thomas Arildsen, Torben Larsen

Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University, Fredrik Bajers Vej 7, DK-9220 Aalborg, Denmark

ARTICLE INFO

Keywords:
Atomic force microscopy
Compressed sensing
Compressive sampling
Undersampling
Image reconstruction
Sparsity modelling

ABSTRACT

The use of compressed sensing in atomic force microscopy (AFM) can potentially speed-up image acquisition, lower probe-specimen interaction, or enable super resolution imaging. The idea in compressed sensing for AFM is to spatially undersample the specimen, i.e. only acquire a small fraction of the full image of it, and then use advanced computational techniques to reconstruct the remaining part of the image whenever this is possible. Our initial experiments have shown that it is possible to leverage inherent structure in acquired AFM images to improve image reconstruction. Thus, we have studied structure in the discrete cosine transform coefficients of typical AFM images. Based on this study, we propose a generic support structure model that may be used to improve the quality of the reconstructed AFM images. Furthermore, we propose a modification to the established iterative thresholding reconstruction algorithms that enables the use of our proposed structure model in the reconstruction process. Through a large set of reconstructions, the general reconstruction capability improvement achievable using our structured model is shown both quantitatively and qualitatively. Specifically, our experiments show that our proposed algorithm improves over established iterative thresholding algorithms by being able to reconstruct AFM images to a comparable quality using fewer measurements or equivalently obtaining a more detailed reconstruction for a fixed number of measurements.

1. Introduction

In atomic force microscopy (AFM) there are several strong arguments for aiming at reducing the image acquisition time. A general wish for minimising operator time spent at the equipment is one argument [1]. The wish for imaging dynamic processes that evolve over time is another argument [2]. Yet another argument is that probe interaction with certain specimens should be kept at a minimum to reduce damage caused by the probe as e.g. experienced in cell imaging [3].

Image scan speed may be increased by improving the mechanics of the AFM equipment [1,2] which may in turn reduce the image scan time. On top of that, advanced sampling methods may be used to reduce the scan time as well as the necessary specimen interaction during the scan. The compressed sensing (CS) paradigm is a recent and still developing concept in the field of sampling digital signals [4–7]. It offers the possibility to undersample signals, i.e., acquire fewer samples than would normally be needed, while still allowing for near perfect reconstruction of the original signal under certain conditions. In AFM, CS may be used to only acquire part of an image and then advanced reconstruction algorithms may be used to reconstruct the full image. This spatial undersampling approach to general microscopy imaging has already been explored in several studies, e.g., [8–13].

The studies just mentioned have all used reconstruction methods based on so-called convex relaxation in CS. However, there are a variety of other options for reconstruction algorithms; see e.g. [7,14,15]. Additionally, there are numerous ways to spatially undersample the specimen. Besides doing a more coarse raster scan, some examples from the literature include patterns such as a spiral [16], a double Archimedian spiral [17], small horizontal linear micro-paths [18], or so-called Lissajous patterns [19]. Independently of the method of choice, CS does require computing the reconstruction which is going to take time. The time spent on computing the reconstruction may differ depending on the choice of reconstruction algorithm. If a speed-up of the total image acquisition process is sought, it is clear that this reconstruction computation time must not exceed the achieved reduction in image scan time.

The present study demonstrates that a model based CS approach [6,20] utilising extended signal structure may be advantageous in reconstructing spatially undersampled AFM images. Specifically, we develop a model of the structure in the discrete cosine transform (DCT) [21] coefficients of typical AFM images and apply this model in

* Corresponding author.
E-mail addresses: cso@es.aau.dk (C.S. Oxvig), tha@es.aau.dk (T. Arildsen), tl@es.aau.dk (T. Larsen).

http://dx.doi.org/10.1016/j.ultramic.2016.09.011
Received 10 June 2015; Received in revised form 23 August 2016; Accepted 15 September 2016
Available online 21 September 2016

iterative thresholding reconstruction algorithms [22] known for their low computational complexity and noise reducing capabilities [23]. Based on a large set of experiments we illustrate the image reconstruction capabilities of the proposed reconstruction method and discuss various trade-offs associated with the undersampling approach to AFM imaging.

The paper is organised as follows. In Section 2, we briefly describe the background on CS as applicable to AFM. Section 3 outlines the development of a model of typical structure in AFM image. Section 4 details how the developed model may be used in the reconstruction. Section 5 provides results from a large experimental evaluation of the proposed structure-utilising reconstruction method. A discussion of the results is given in Section 6. Finally, Section 7 states the conclusions of our work.

2. Compressed sensing in AFM

Undersampling and thus faster scan times or less probe-specimen interaction is the motivation for using CS in AFM in a number of studies as briefly covered in the Section 1. However, CS may also be used to possibly achieve super resolution imaging as shown in e.g. [24,25]. All of this motivates the following introduction to the general CS framework as applicable to AFM.

An AFM (or scanning probe microscopy (SPM)) topography image may be thought of as a 2D array (a matrix) of pixels. By stacking the columns of the matrix, one gets a vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$ where n is the total number of image pixels, i.e., the height times the width of the image measured in pixels. CS [26,27] deals with a general linear measurement process that collects a sample vector $\mathbf{y} \in \mathbb{R}^{m\times 1}$ of the image vector $\mathbf{x}$. That is, we consider measurements of the form

$$\mathbf{y} = \mathbf{\Phi x} \tag{1}$$

where $\mathbf{\Phi} \in \mathbb{R}^{m\times n}$ is a measurement matrix with $m \ll n$, i.e. we only acquire what corresponds to a fraction of the full image vector. In the general CS case, $\mathbf{\Phi}$ can be dense. However, in the AFM application, $\mathbf{\Phi}$ is an identity matrix with some of its rows removed. That is, it is a fat matrix with only a single one in each row and zeros elsewhere. Such a matrix models the measurement process in which the AFM probe tip only measures a single pixel of the image at a time as it moves around.[1] Key to the CS theory is the idea of finding a dictionary or basis in which $\mathbf{x}$ is sparse or approximately sparse (compressible). That is, we seek an approximately sparse vector of coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{n\times 1}$ such that

$$\mathbf{x} = \mathbf{\Psi}\boldsymbol{\alpha} \tag{2}$$

where $\mathbf{\Psi} \in \mathbb{R}^{n\times n}$ represents the dictionary.[2] By combining $\mathbf{\Phi}$ and $\mathbf{\Psi}$ into a single CS matrix $\mathbf{A} = \mathbf{\Phi\Psi}$ and adding measurement noise $\mathbf{e} \in \mathbb{R}^{m\times 1}$, we get the linear model:

$$\mathbf{y} = \mathbf{A}\boldsymbol{\alpha} + \mathbf{e} \tag{3}$$

For sufficiently sparse $\boldsymbol{\alpha}$ and under certain conditions, CS does provide some guarantees under which the undersampled system in (3) may be solved for $\boldsymbol{\alpha}$ [4]. In the noiseless case, recovering $\boldsymbol{\alpha}$ may be cast as the NP-complete combinatorial optimisation problem [5]

$$\text{minimise} \quad \| \boldsymbol{\alpha} \|_0 \text{ subject to } \quad \mathbf{y} = \mathbf{A}\boldsymbol{\alpha} \tag{4}$$

where $\| \boldsymbol{\alpha} \|_0$ is the ℓ_0 "pseudo"-norm from [26], i.e., the number of non-zero entries in $\boldsymbol{\alpha}$. For noisy measurement, one typically bounds the noise in e.g. the two norm and searches for a sparse solution subject to $\| \mathbf{y} - \mathbf{Ax} \|_2 \leq \epsilon$ for some ϵ describing the bound on the noise. Fortunately, there are several ways to find solutions to (4) despite it being NP-complete. The probably most used method is based on convex relaxation, i.e., replacing the ℓ_0 norm with the ℓ_1 norm as e.g. in the Lasso [28]

$$\text{minimise} \quad \frac{1}{2} \| \mathbf{y} - \mathbf{A}\boldsymbol{\alpha} \|_2^2 + \lambda \| \boldsymbol{\alpha} \|_1 \tag{5}$$

Having obtained an estimate of the sparse vector, $\hat{\boldsymbol{\alpha}}$, an estimate of the reconstructed image may be found using (2). See e.g. [4] for a further discussion of CS or [29] for an introduction to the CS imaging application.

3. Structure in AFM images

To be able to apply CS in AFM, there must exist some dictionary in which AFM images are sparse or compressible as described in Section 2. The DCT is one such dictionary that is applicable to general imaging [29] due to its ability to successfully approximate smooth images using only few coefficients which has made it the basis for the widely used JPEG image compression standard [30]. Furthermore, it has been used in a previous scanning electron microscopy (SEM) study utilising CS methods [9]. Since AFM images generally contain smooth regions or repetitive patterns, it is to be expected that the DCT performs well for compressing such images. The DCT also has a few other advantages. Firstly, it is maximally incoherent with the sparse point sampling in (1) [4]. This CS property, intuitively, states that it is likely that the dense DCT is able to combine the very sparse measurements to get the full image. Secondly, it has a fast implementation, i.e. a FFT based implementation [31]. Even for the most advanced CS reconstruction algorithms, a fast implementation may be essential in reducing the time it takes to perform the signal recovery as shown in [32]. Another typical choice of dictionary in CS imaging is the a discrete wavelet transform (DWT) [29]. However, as reported in [13] and further elaborated on in [33], the DCT provides better reconstructions than the DWT in the CS for AFM application. Thus, in the rest of this paper, we focus on the DCT as the dictionary of choice. We note, though, that our methods are applicable to any dictionary, e.g. the DWT.

Following the model based CS framework of [6,20], we now turn our attention toward not only sparsity but structured sparsity of AFM images in the DCT domain. That is, we seek to exploit additional information about the structure of the sparsity to enhance the reconstruction of an undersampled image. To motivate this, consider the AFM topography image and its reconstruction from the 10% largest DCT coefficients in Fig. 1. This reconstruction exemplifies the image quality that is achievable using only 10% DCT coefficients. The right image in Fig. 1 illustrates the positions of these coefficients in the DCT domain. From this illustration, we make the following observations:

1. The largest DCT coefficients are inherently low-frequent. That is, they are found in the upper left corner of the DCT domain.
2. There is a dispersion in the coefficients from the low-frequency area towards the high-frequency area such that:
   (a) The magnitudes of the coefficients generally decrease as the frequency increases.
   (b) The coefficients are spaced further apart as the frequency increases.

A somewhat similar low-frequency model is used in JPEG image compression by virtue of ordering the coefficients in a "zig-zag" sequence [30]. Additionally, the set of experiments detailed in Section 5 provides further evidence that the above observations are general to AFM images..

Note that if the image matrix in Fig. 1 is transposed, the DCT domain is also transposed. This suggests that there should be symmetry around the diagonal in a model of the DCT coefficients. Based on all these observations, we consider a model of the DCT coefficients that

[1] When using this in practice, the AFM would directly acquire $\mathbf{y}$ and $\mathbf{\Phi}$ would be a model of the acquisition system. Thus, $\mathbf{x}$ would not be available and one would consequently have to reconstruct an estimate, $\hat{\mathbf{x}}$, of it. In this study we do, however, simulate the sampling of $\mathbf{y}$ based on an $\mathbf{x}$ obtained using a raster scan. This allows us to compare the two imaging methods.

[2] Both $\boldsymbol{\alpha}$ and $\mathbf{\Psi}$ may be complex in the AFM application. However, in this study we only consider real matrices and vectors.

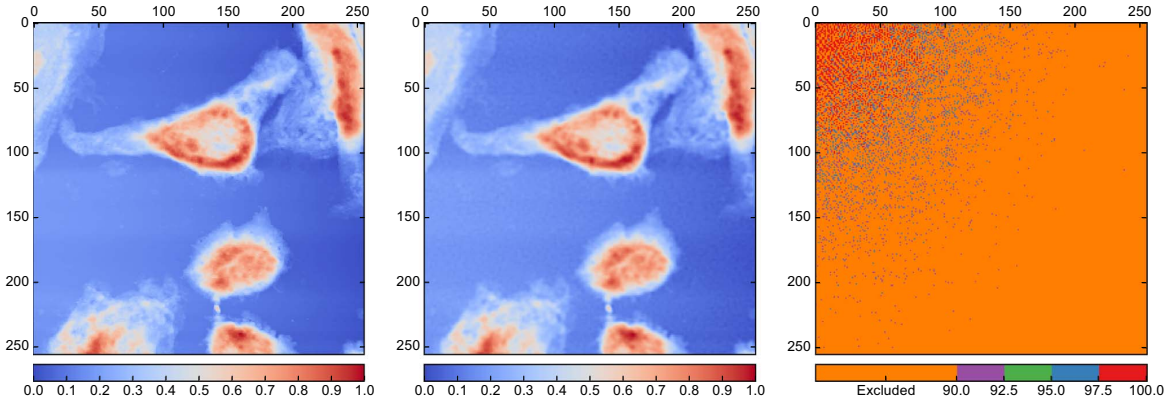


Fig. 1. Example of an AFM cell image (left), approximation from 10% DCT coefficients (middle), and locations of the 10% coefficients in the DCT domain (right). The image approximation in the middle is based on the 10% largest DCT coefficients of the image to the left. Reconstruction metrics for this image are: PSNR: 36.59 dB, SSIM: 0.96. The colours in the right figure illustrate the locations of the 90% smallest, and thus excluded, DCT coefficients (orange) and 10% largest DCT coefficients divided into four intervals. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is characterised by:

1. A smooth transition from a peak value in the low-frequency area to a small constant value in the high-frequency area. Examining the full DCT domain suggests an exponentially decaying transition.
2. A sufficient number of degrees of freedom to model the dispersion in the coefficients from low-frequencies to high-frequencies while still imposing symmetry around the diagonal.

Specifically, we propose to model the dispersion in the DCT coefficient values using a Gaussian function f of the form:

$$
\begin{aligned}
f(\mathbf{x}) &= a \cdot \exp(-(\mathbf{z}-\mathbf{b})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{z}-\mathbf{b})) \\
\mathbf{z} &= \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b \\ b \end{bmatrix} \\
\mathbf{R} &= \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix} \\
\mathbf{C}^{-1} &= \mathbf{R}\begin{bmatrix} 1/c_1 & 0 \\ 0 & 1/c_2 \end{bmatrix}\mathbf{R}^{\mathrm{T}}
\end{aligned} \tag{6}
$$

where:

- z_1, z_2 are the two pixel coordinates.
- a is a user specified parameter that determines the flatness of the model relative to the data.
- b is the "mean" value positioned on the $z_1 = z_2$ diagonal.
- $\mathbf{C}$ is the "covariance" matrix rotated to ensure symmetry around the diagonal.

We use the quadratic form $(\mathbf{z}-\mathbf{b})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{z}-\mathbf{b})$, to model the shape of the spread. Choosing both entries in $\mathbf{b}$ to be equal, forces the peak value to be on the diagonal. Combined with the fixed rotation matrix $\mathbf{R}$ that aligns the principal axes with the diagonals, this forces the desired symmetry. The exponential function taken on the quadratic form is used to model the smooth transition from the peak value at $\mathbf{b}$ towards zero away from $\mathbf{b}$. The value of a determines the scaling of the peak value in the model, thus, allowing for forcing a model with a smaller (or larger) peak value. A further discussion of the impact of the choice of a is given in Section 6. We suggest fitting this Gaussian function f to the DCT coefficients of an image using a least squares fit:

$$
\underset{b,c_1,c_2}{\text{minimise}} \sum_{\mathbf{z}} \left(|\tilde{\boldsymbol{\alpha}}(\mathbf{z})| - f(b, c_1, c_2; a, \mathbf{z})\right)^2 \tag{7}
$$

Here we take $\tilde{\boldsymbol{\alpha}}(\mathbf{z})$ be the 2D representation the vector $\boldsymbol{\alpha}$ in (2) indexed by $\mathbf{z}$. Furthermore, f is regarded a function of b, c_1, c_2 with a fixed. An example of the full DCT domain and a Gaussian model fitted to this DCT domain is given in Fig. 2..

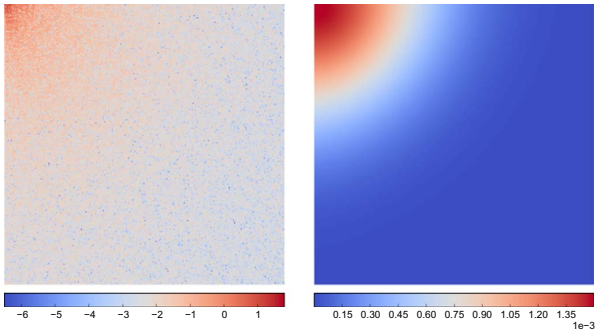


Fig. 2. Illustration of the absolute value of the full DCT domain of the AFM cell image in Fig. 1 (left) and Gaussian model fitted to this DCT domain (right). The colour map used in the left figure is based on a log-scale.

Note that image height, image width, and topography height in this fit may be arbitrarily scaled and offset during the image processing. Due to possible numerical issues in solving (7), it is convenient to not work with quantities in the order of e.g. 10^{-9} but to re-scale and offset to some larger quantity. We have successfully used a re-scaling and offset of both image height, image width, and topography height to the interval [0, 1]. That is, for each of the three lengths, we have found scaling and offset values in an affine transform that transforms the minimum value to 0, the maximum value to 1 and all other values to the interval (0, 1). Using this re-scaling and offset, the parameters of the Gaussian model in Fig. 2 are $a = 1.5{\cdot}10^{-3}$, $b = 0.18$, $c_1 = 0.12$, $c_2 = 0.19$. Also note that this model only depends on the nature of the specimen being imaged. It is independent of imaging technique. Thus, since other SPM modalities produce images with smooth regions or repetitive patterns similar to those found in AFM images, our model of the DCT coefficients is likely to be applicable to any SPM technique.

4. Reconstructions using structure

Having established a model of the DCT coefficients of AFM images, we now turn to the CS reconstruction algorithm. The choice of reconstruction algorithm entails a trade-off between reconstruction quality (e.g. peak-signal-to-noise-ratio (PSNR)), reconstruction capability, and the time required to do the reconstruction [34]. For some fixed reasonable reconstruction quality and required reconstruction

time, we are interested in assessing the reconstruction capabilities, i.e, the number of measurements in $\mathbf{y}$ needed to reconstruct $\mathbf{x}$ to the specified quality.

As exemplified in Fig. 1 when using the DCT as a dictionary, the reconstruction quality depends both on the number of coefficients used in the reconstruction as well as the values of these coefficients. We are thus interested in a CS reconstruction algorithm that allows us to fix the number of coefficients and thereby roughly fix the reconstruction quality assuming that the algorithm is capable of identifying the coefficient values that optimises the reconstruction quality as e.g. measured in PSNR. Specifically, we consider iterative thresholding algorithms [22] of the general form in Algorithm 1.

Algorithm 1. Iterative thresholding (based on [22,23]).

initialise: $\hat{\boldsymbol{\alpha}} = \mathbf{0}$, $\mathbf{r} = \mathbf{y}$
for i=1 **to** $I_{\max}$ **do**
  $\mathbf{c} \leftarrow \mathbf{A}^{\mathrm{T}}\mathbf{r}$
  $\hat{\boldsymbol{\alpha}} \leftarrow \eta_t(\hat{\boldsymbol{\alpha}} + \kappa \cdot \mathbf{c}_i)$
  $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{A}\hat{\boldsymbol{\alpha}}$
  if $\| \mathbf{r} \|_2 < \epsilon_r \cdot \| \mathbf{y} \|_2$ **then**
  break
  end if
end for

In Algorithm 1, η_t is a scalar and non-linear threshold function that operates on each entry of the vector independently. The main reason for our interest in this type of algorithm is its simplicity and its applicability in a model based CS framework [20]. The algorithm is simple in the sense that it is computationally dominated by the matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^{\mathrm{T}}$. Thus, it has a complexity of $O(m\,n)$. If fast transforms are available such as a FFT based DCT as described in Section 3, then the computational complexity is only $O(n\log(n))$ which is a significant reduction for problem sizes of e.g. 256×256 pixels.

In Algorithm 1, one has to decide on the stopping criterion tolerance ϵ_r, the maximum number of iterations $I_{\max}$, the step-size κ, and the non-linear threshold operator η_t. The step-size κ is an extension of the algorithm in [22] and is critical in terms of the convergence of the algorithm [23]. Note that other stopping criteria may be used instead of the one stated in Algorithm 1, e.g. a criterion based on the change in $\hat{\boldsymbol{\alpha}}$ in each iteration. The outset for iterative thresholding algorithms is the assumption of a signal model as in (3). There are several options for the choice of threshold function η_t. Two well described threshold operators in the literature are:

- Hard [35]: $\eta_t^{\mathrm{H}}(x) = x \cdot \mathbb{1}_{|x|>t}$
- Soft [36]: $\eta_t^{\mathrm{S}}(x) = \mathrm{sgn}(x)(|x|-t)_+$

with $(x)_+ = x \cdot \mathbb{1}_{x>0}$ and $\mathbb{1}_x$ the indicator function. Here, η_t^{H} and η_t^{S} are operators that work on each entry x of some vector $\mathbf{x}$. Specifically, η_t^{H} sets all entries in the vector smaller than some threshold t (in absolute value) to zero whereas η_t^{S} sets all entries in the vector smaller than some threshold t (in absolute value) to zero and pulls the remaining entries towards zero by the amount t, i.e. subtracts t from the positive entries and adds t to the negative entries.

When using the hard threshold, Algorithm 1 attempts to solve the following variant of the CS reconstruction problem [35]:

$$\text{minimise} \quad \|\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 \quad \text{subject to} \quad \| \boldsymbol{\alpha} \|_0 \leq k \tag{8}$$

where k is the assumed sparsity of $\boldsymbol{\alpha}$, e.g. a fixed number of DCT coefficients. The minimisation is done in a *greedy*, that is, locally optimal, way. When using the soft threshold there is no straight counterpart to (8) since we fix the number of coefficients k and let the threshold t vary accordingly in each iteration. Had the threshold t been fixed to, say, λ, it can be shown that the soft threshold strategy attempts to solve the Lasso problem in (5) [37]. In addition to the soft thresholding case with p=1 in

$$\text{minimise} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 + \lambda \sum_j |\alpha_j|^p \tag{9}$$

it can be shown that the hard thresholding case can be seen as the solution to (9) in the limit when p goes to zero [37].

Based on the idea of adapting the thresholding operators in model based CS [20], we propose the following two weighted thresholding operators:

- Weighted hard: $\eta_t^{\mathrm{wH}}(x) = x \cdot \mathbb{1}_{|wx|>t}$
- Weighted soft: $\eta_t^{\mathrm{wS}}(x) = \frac{1}{w}\,\mathrm{sgn}(x)(|wx| - t)_+$

where w is a weight applied to the coefficient before thresholding. These weights allow us to attach additional importance to certain entries in the vector when deciding which entries to set to zero. Thus, in the AFM application, we have a vector of weights $\mathbf{w} \in \mathbb{R}^{n\times 1}$ corresponding to the model f described in (6). That is, the weights, $\mathbf{w}$, are a possibly re-scaled and offset version of the vector obtained by stacking the columns of a model based on f in (6). Note that the weights in the weighted thresholding operators merely determine which coefficients are zeroed in the thresholding, i.e., they favour zeroing coefficients that have corresponding low weights. The weights have no impact on the values of the coefficients that are not zeroed when choosing the threshold t based on fixing the number of coefficients that are zerored. Thus, in principle, scaling and offsetting $\mathbf{w}$ has no impact on the performance of the thresholding operator. In practice, one must consider the following when choosing weights:

1. Coefficient values may be very small or very large. Thus, to avoid numerical over- or underflow, the weights should neither be too small nor too large.
2. When using the weighted soft threshold operator, it must be asserted that $w \neq 0$ to avoid a division by zero.

When using iterative thresholding with the hard threshold operator, η_t^{H}, the algorithm is known as Iterative Hard Thresholding (IHT) [35]. Similarly using the soft threshold operator, η_t^{S}, the algorithm is known as Iterative Soft Thresholding (IST) [38]. Following this scheme, we denote our corresponding weighted algorithms as w-IHT when using η_t^{wH} and w-IST when using η_t^{wS}. We note that all four algorithms, IHT, IST, w-IHT, and w-IST, may be used with any $\mathbf{A}$, i.e. one may use other sampling patterns and dictionaries than the ones used in this study.

5. Experiments

To assess the reconstruction capabilities of our proposed iterative thresholding algorithms, we have conducted a large set of simulation experiments.

5.1. Experimental setup

Based on an image gallery containing the 17 AFM images in Fig. 3, all with a resolution of 256-by-256 pixels, we have simulated a reconstruction of each image for each combination of:

- An undersampling ratio δ (according to (12)) from a set of 20 evenly spaced values in the range [0.1, 0.4].
- A fixed sparsity level $\tau = k/n$ from a set of 20 evenly spaced values in the range [0.05, 0.20]. The threshold level t in Algorithm 1 is then chosen based on this fixed sparsity level.
- A reconstruction algorithm from the set {IHT, IST, w-IST, w-IHT}.

.

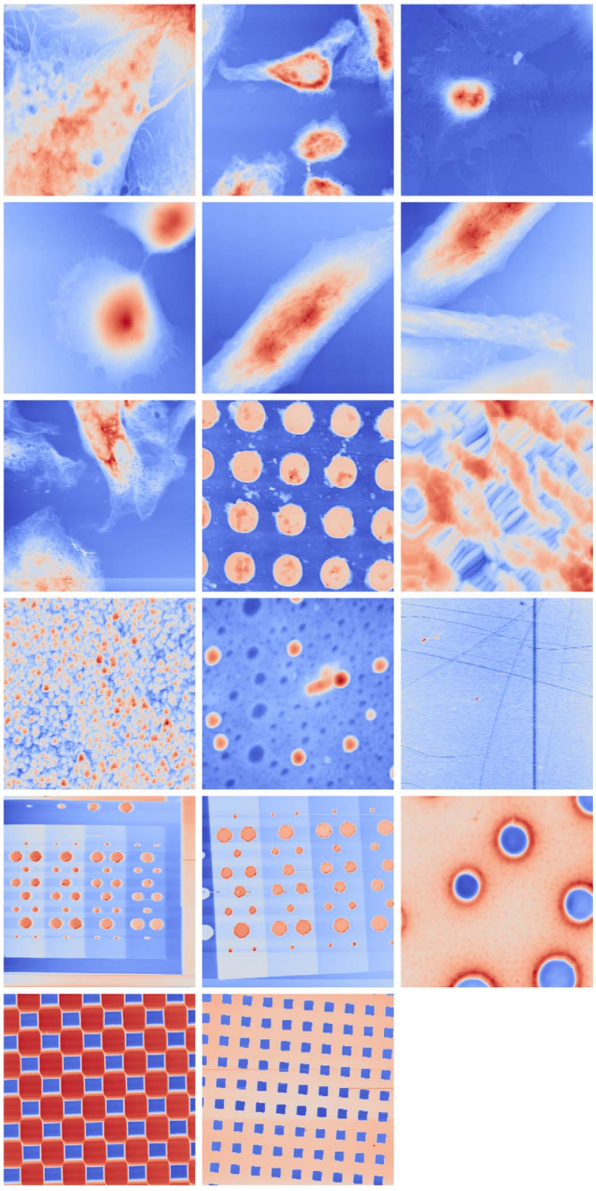


Fig. 3. The 17 de-tilted AFM images that constitute the set of test images used in the experiments.

Furthermore, as a CS reference, we have included reconstructions using ℓ_1 minimisation which is an instance of the well established convex relaxation approach to CS (see e.g. [7] for an introduction). In particular, we reconstruct the undersampled images by solving the optimisation problem:

$$\text{minimise} \quad \| \boldsymbol{\alpha} \|_1 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}\boldsymbol{\alpha}\|_2^2 \leq \epsilon_{\ell_1} \tag{10}$$

where we take $\epsilon_{\ell_1} = \epsilon_r \cdot \| \mathbf{y} \|_2$ in order to have the same fidelity constraint as used in the stopping criterion in Algorithm 1. In the sequel, we refer to the optimisation problem in (10) as Basis Pursuit Denoising (BPDN).

To be able to compare the reconstructed images to a raster scan (our reference image) of the same specimen, we have simulated the undersampled scanning from the raster scanned images. For down-sampling, we have used the spiral sampling pattern proposed in [16]

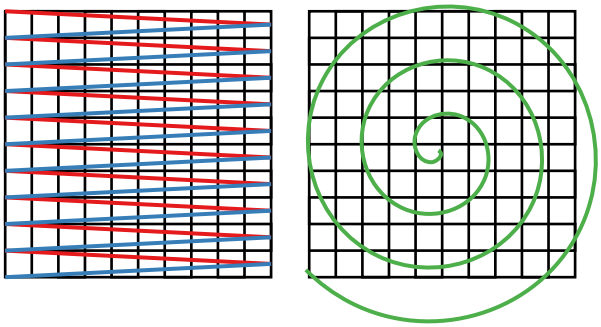


Fig. 4. Illustration of the path lengths involved in calculating the undersampling ratio, δ, in (12). The black pixel grid in the left sub-figure is scanned using a raster scan resulting in two buffers, one for the left-to-right scan (blue), and one for the right-to-left scan (red). The green spiral scan pattern in the right sub-figure only covers some of the pixels in the black grid. In (12), L_{ref} is approximately the sum of the lengths of the red and blue paths whereas L is the length of the green path. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and suggested for the CS for AFM application in [13]. We define the undersampling ratio in terms of the distance travelled by the probe during the scan, i.e. the scan path length. The distance travelled by the probe in pixels during a raster scan is approximated as:

$$L_{\text{ref}} = 2 \cdot n = 2 \cdot \text{height} \cdot \text{width} \tag{11}$$

The factor of two is a result of the usual approach of acquiring two topography buffers, one based on the left-to-right scan and one based on the right-to-left scan. Although almost identical, usually, only one of the buffers is used and displayed as is the case in Fig. 3. Now let L denote the length of a continuous scan path (in pixels) resulting in a spiral scan as exemplified in Fig. 8. We then define the undersampling ratio as

$$\delta = \frac{L}{L_{\text{ref}}} \tag{12}$$

See Fig. 4 for an illustration of this undersampling ratio definition.

Before reconstructing the undersampled images, we remove any tilt in the measurements by subtracting a least squares fit of a plane to the measurement as outlined in [39]. Note that all the images in Fig. 3 have been de-tilted using the same least squares plane fit method, albeit based on the entire image.

In the evaluation of the performance of the Gaussian model in (6), we have used a jack knife approach [40]. That is, when reconstructing a given image with either w-IHT or w-IST, we have used a Gaussian model based on the average DCT domains of all other images. We have fitted the model using the optimisation problem in (7) which we have solved using the implementation of Powell's method found in SciPy[3] using an initial guess of b=0.005, $c_1 = c_2 = 0.1$. Powell's method is a conjugate directions minimisation method that does not need information about the gradient of the cost function [41]. Before using the fitted model as the weights, $\mathbf{w}$, we have re-scaled and offset the weights to be in the interval $[10^{-3}, 1]$ using the same re-scale and offset approach as explained in Section 3.

In terms of the free parameters in Algorithm 1 and (6), we have used the following values: $\epsilon_r = 10^{-3}$, $\kappa = 0.6$, $I_{\max} = 300$, $a = 2.5 \cdot 10^{-3}$.

Finally, to quantitatively evaluate the image reconstruction quality, we have used the two image quality indicators, PSNR, and mean structural similarity index (SSIM). For a reconstructed image, $\hat{\mathbf{x}}$, the PSNR is defined as follows:

[3] See http://www.scipy.org/

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{P^2}{\frac{1}{n}\|\hat{\mathbf{x}}-\mathbf{x}\|_2^2}\right) \tag{13}$$

where we take P=1 since we re-scale and offset both $\mathbf{x}$ and $\hat{\mathbf{x}}$ such that all values are in the interval $[0, 1]$ before computing the quality indicators. We compute the mean SSIM according to the definition in [42] with the parameter values: window size 7, $K_1 = 0.01$, $K_2 = 0.03$, $C_3 = C_2/2$, $\alpha = \beta = \gamma = 1$.

5.2. Implementation

We have used the implementations of the iterative thresholding algorithms available in the open source Python package `magni` described in [43]. Release 1.5.0[4] of `magni` was used in all experiments. The `magni` package is a collection of fully documented functionality for compressive sampling and reconstruction of AFM images. It focuses on producing correct results and as such comes with an extensive input validation system [44] as well as several tests to exercise the entire public API. Furthermore, magni provides several routines that may help in improving the reproducibility [45] of the results obtained using the package. For the BPDN reconstructions, i.e. solving (10), we used the Douglas-Rachford solver from PyUNLocBox[5] in its default configuration with the exception of allowing only 300 iterations as we do in the iterative thresholding methods.

The Anaconda[6] Python distribution version 4.1.1 based on Python 3.5 was used for running the simulations.[7] The Python script used to run the simulations and the full set of simulation results is available at Zenodo.[8] A Jupyter Notebook that reproduces the figures in this paper is available from the authors' institutional repository.[9] The simulations were conducted on a compute server based on two Intel Xeon E5-2697V2 CPUs and 384 GiB RAM. This 24 core setup with 48 hyper threads allowed us to run 48 simulations concurrently.

5.3. Results

The reconstruction capability results from our experiments are presented in Figs. 5, 6, and 7. The average SSIM of the reconstructions for all combinations of sparsity level τ and undersamling ratio δ are show in Fig. 5 whereas Fig. 6 details the improvement in PSNR and SSIM in using w-IHT or w-IST over IHT or IST, respectively. Finally, Fig. 7 shows the highest obtainable average PSNR and SSIM across all sparsity levels for the iterative thresholding algorithms and includes the BPDN (ℓ_1) results for comparison.

Fig. 8 visually illustrates the quality gain in using a weighted iterative thresholding scheme. Note that the three reconstruction examples shown in Fig. 8 have been selected to visually illustrate typical gains in using w-IST, i.e., they are examples from the red area in the IST SSIM Improvement subfigure in Fig. 6. As illustrated in the left part of Fig. 6 there are various combinations of sparsity level and undersampling ratio that may enable IST or w-IHT (but not IHT) to produce comparable reconstruction to those illustrated in Fig. 8. However, as stated in Section 4, our goal has been to lower the required number of measurements needed to provide a reasonable reconstruction quality, e.g., a SSIM $\geq$0.90. Towards that end, in our experiments, w-IST proved to be the superior algorithm for undersampling ratios $0.15 \lesssim \delta \lesssim 0.30$ as witnessed in Fig. 7. For undersampling ratios $\delta \gtrsim 0.30$, w-IST and BPDN (ℓ_1) perform equally well whereas for $\delta \lesssim 0.15$, only BPDN gives somewhat reasonable reconstructions.

The average reconstruction times measured in our simulations were IHT: 2.2 s, IST: 2.3 s, w-IHT: 2.2 s, w-IST: 2.4 s, BPDN (ℓ_1): 0.8 s. Here it should be noted that `magni` is a Python package that does not provide algorithms optimised for execution speed but rather correctness of the results. Execution speeds may vary a lot with parameter choices and input images. Thus, these numbers should only be considered indicators of the execution time that one may roughly expect.

6. Discussion

The PSNR and SSIM image quality indicators used in Figs. 5, 6, and 7 provide a quantitative measure of the image reconstruction quality. Another aspect of the reconstruction is the visual quality of the reconstruction, i.e, the suppression of noise and highlighting of important details in the images. Unfortunately, there is not a one-to-one correspondence between higher PSNR (or SSIM) values and better visual quality. PSNR and SSIM weight various errors in the reconstruction differently and, thus, may favour certain details in the reconstruction more than other. See e.g. [46] for a further discussion of these aspects of measuring image reconstruction quality.

That being said, we find that acceptable reconstructions are typically found for PSNRs $\gtrsim$35 dB or SSIM $\gtrsim$0.90. As such, the SSIM sub-figures in Fig. 5 give a good overview of the undersampling ratio δ and sparsity level τ needed to obtain a successful reconstruction, i.e., a reconstruction with SSIM $\geq$0.90. Generally, each of the SSIM sub-figures in Fig. 5 may be divided into three regions. A region where reconstruction succeeds (red area), a region where reconstruction fails (blue area), and a region in-between the two other regions where there is a transition from reconstruction success to reconstruction failure. As such the figures somewhat resemble the phase transition framework in [38], albeit using $\tau = k/n$ instead of $\rho = k/m$. A general characteristic of this transition region is that fewer measurements are needed for lower sparsity levels to obtain acceptable reconstructions. However, one must keep in mind that using fewer DCT coefficients in the reconstruction leads to fewer details in the reconstruction and thereby a more smooth (blurred) image.

Comparing the sub-figures in Fig. 5 as well as the results in Fig. 7, we see that IHT does not seem to produce acceptable reconstructions for any combination of τ and δ. Switching to w-IHT, it is possible to obtain reasonable reconstructions. That is, at least in our experiments, the use of weights in w-IHT enables the use of an IHT based algorithm for reconstruction of undersampled AFM images. Using IST on the other hand still provides more combinations of τ and δ for which acceptable reconstructions are obtained than do w-IHT. This is even further increased when switching to w-IST, i.e., the transition region is pushed to the left which is also clear from studying the improvement sub-figures in Fig. 6. Also interesting to note from these improvement sub-figures is that they divide into similar regions as do the SSIM sub-figures. Specifically, in the region where reconstruction is successful, no or a minor improvement is seen. In the region where reconstruction is unsuccessful, no or a minor degradation is seen. The most interesting region, though, is the transition region where a major improvement is seen. In other words our experiments show that using weights in iterative thresholding yield similar or improved image reconstruction quality. A degradation in image quality is only seen when the reconstruction is already useless, e.g. with a SSIM $\leq$0.5.

Returning to the quality-capability-time trade-off inherent in any reconstruction method, we make the following notes. Our proposed Gaussian model in (6) is a low-frequency model which favours reconstructions that are more smooth and with less high-frequency details. To include more details in the reconstruction more DCT coefficients are needed which in turn means that more measurements are needed. However, this may still be acceptable so long as the overall

[4] All magni releases are available at http://dx.doi.org/10.5278/VBN/MISC/Magni.
[5] Specifically, we used the code from https://github.com/epfl-lts2/pyunlocbox, master branch, tag: v0.2.1-211-g585027a.
[6] See http://www.continuum.io/anaconda
[7] Refer to the annotations in the results database for a full specification of the versions of all the scientific Python packages used.
[8] See http://dx.doi.org/10.5281/zenodo.60512
[9] See http://dx.doi.org/10.5278/vbn/projects/wISTwIHT

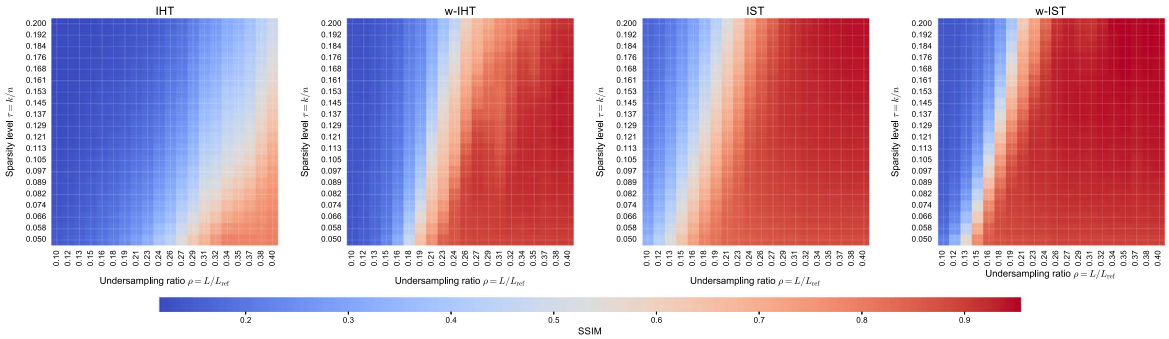


Fig. 5. Average SSIM values for each iterative thresholding reconstruction method. Each sub-figure shows the average SSIM across all 17 test images (colour coded) versus sparsity level and undersampling ratio. All sub-figures share the same colour scale.

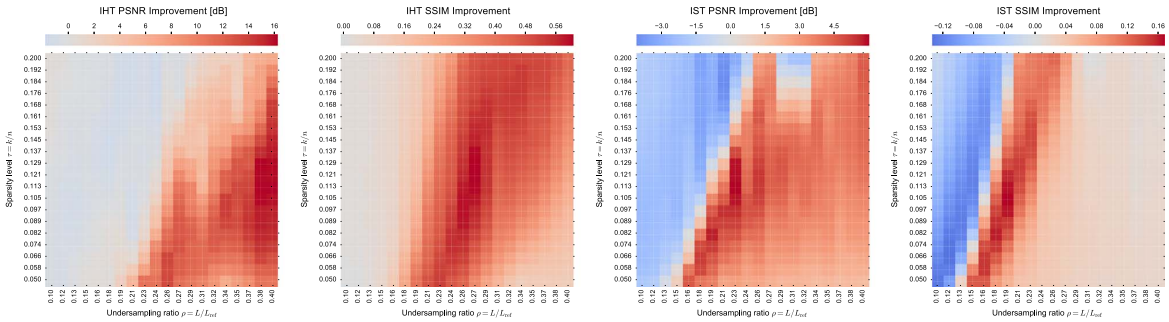


Fig. 6. PSNR and SSIM improvements in using a weighting scheme in the reconstruction algorithm, i.e., the difference between w-IST and IST, w-IHT and IHT, respectively, in both average SSIM and average PSNR. Note that the colour scales are different in the four sub-figures since the colour scales are "centered" around zero (white colour) with blue colours for negative values and red colours for positive values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

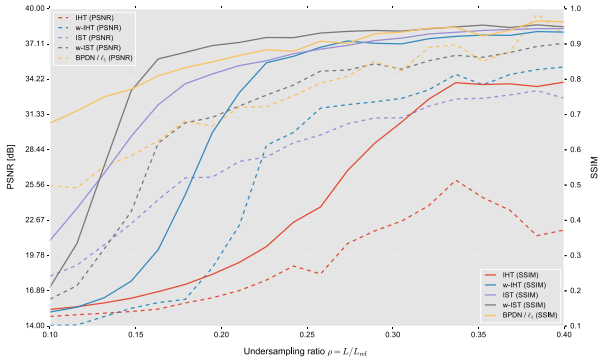


Fig. 7. Best case PSNR and SSIM performance vs undersampling ratio for all algorithms. The best case PSNR and SSIM values for the iterative thresholding algorithms have been found by choosing the highest PSNR (averaged across all test images), respectively SSIM, value across all sparsity levels for each undersampling ratio.

imaging time, i.e. image acquisition time plus image reconstruction time, is still lower than the time it takes to do a traditional raster scan. Additionally, it may be beneficial to fit different models to different images types, e.g. fitting a specific model to the very low-frequency cell images. Possibly, several reconstructions based on different models may be done simultaneously on a multi core processor if the user is not able to pick a reasonable model a-priori.

Also key in the quality-capability-time trade-off is the choice of free parameters in the algorithms. We note that our choice of ϵ_τ, κ, $I_{\max}$ are based on empirical studies (our own as well as [38]) showing that these are good choices on average. For the choice of sparsity level τ, we suggest using our results presented in Fig. 5 as guidance, i.e. choosing the sparsity level that yields the highest SSIM for a given undersampling rate as we did when constructing Fig. 7. The choice of a in (6) is related to the high-frequency versus low-frequency trade-off. A lower value of a implies less weight on low-frequency content compared to high-frequency content. As such the user may also tweak this trade-off by adjusting a. Additionally, when imaging in other resolutions than 256-by-256 pixels, it may be necessary to adjust a compared to the value, we have used.

All the results presented in this paper are based on experiments with the 17 AFM images in Fig. 3. When imaging other types of specimens the applicability of our proposed iterative thresholding methods may vary. However, since our test images are qualitatively very different and due to the general applicability of the DCT in image compression, we do believe that our results generalise to allow for reconstruction of a broad range of undersampling images in various areas of AFM, SPM, and SEM.

7. Conclusions

We have detailed a generic model of the structure in DCT coefficients of a set of typical AFM images. The model emphasises low-frequency content and assumes a smooth and diagonal-symmetric transition to high-frequency content. Together with the model we have proposed a weighted hard threshold function and a weighted soft threshold function that allow us to exploit the model structure in a general iterative threshold algorithm. A large set of experiments indicate that this setup is capable of reconstructing undersampled AFM cell images. Furthermore, the results indicate that our proposed weighted thresholding schemes outperform the corresponding state-of-the-art non-weighted schemes as well as ℓ_1 minimisation based reconstruction for some undersampling ratios. Concurrently with the presentation of our proposed reconstruction method, we have discussed several practical issues in implementing CS for AFM imaging. This discussion includes pointers on the choice of dictionary as well as various elements that may help in retaining numerical stability in the

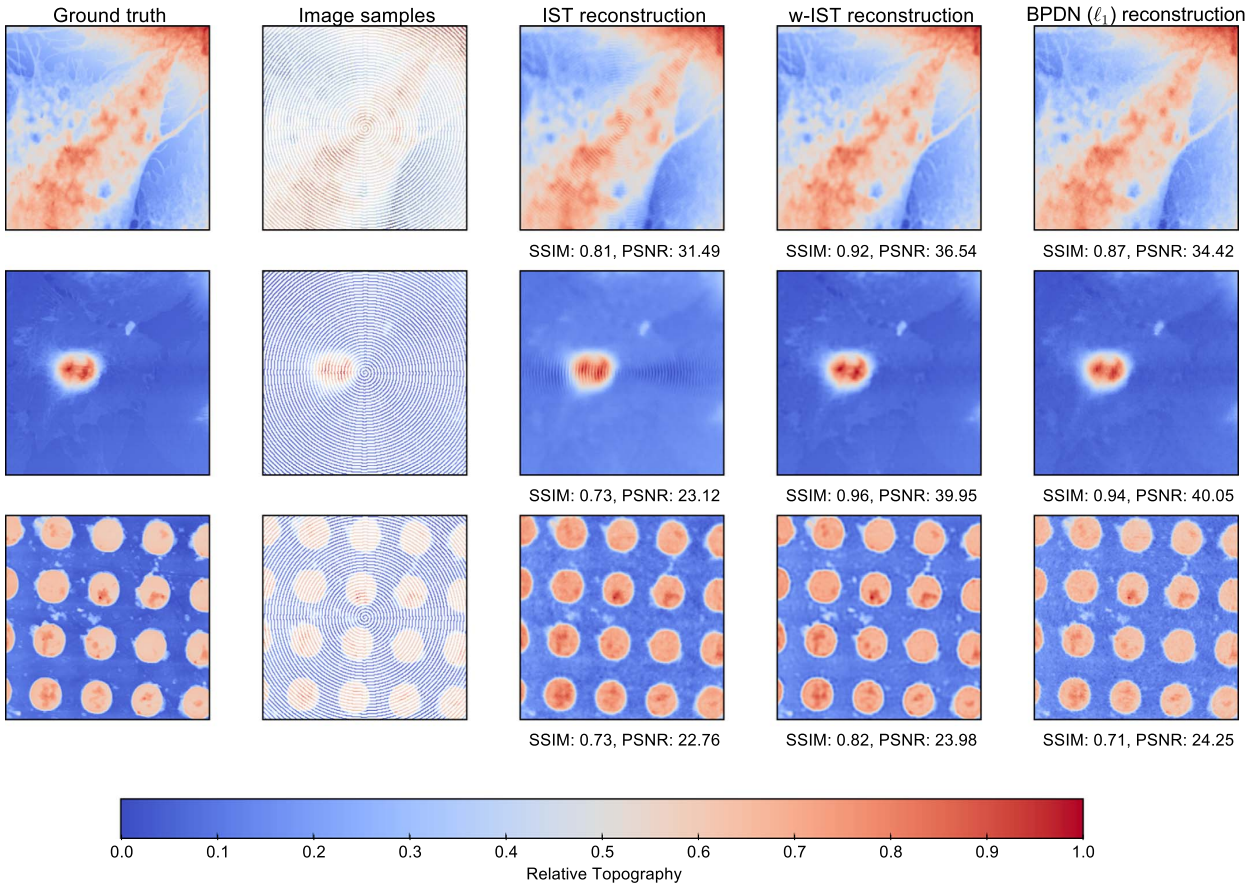


Fig. 8. Examples of typical improvements in reconstruction quality when using a weighting scheme in IST. From left to right, the images are: The original "ground truth" image, the samples used in the reconstruction, an IST reconstruction, a w-IST reconstruction, and the BPDN (ℓ_1) reconstruction for comparison. All three examples are based on an undersampling rate of $\delta = 0.19$ and a sparsity level of $\tau = 0.082$. The PSNR and SSIM metrics refer to a comparison to the corresponding "ground truth" test image.

reconstructions. We believe that our results are extensible to various applications in AFM, SPM, and SEM imaging.

Acknowledgements

This project has been supported by (1) The Danish Council for Independent Research (DFF/FTP) for project number 1335-00278B/12-134971, and (2) by the Danish e-Infrastructure Cooperation (DeIC) for project number DeIC2013.12.23. The experiments have been based on the *Atomic Force Microscopy Images of Cell Specimens*[10] and *Atomic Force Microscopy Images of Various Specimens*.[11] (shown in Fig. 3) by Christian Rankl licensed under CC BY 4.0.[12] Adapted images are part of the experimental results.

References

[1] P.K. Hansma, G. Schitter, G.E. Fantner, C. Prater, High-speed atomic force microscopy, Science 314 (5799) (2006) 601–602. http://dx.doi.org/10.1126/science.1133497.
[2] G. Schitter, M.J. Rost, Scanning probe microscopy at video-rate, Mater. Today 11 (0) (2008) 40–48. http://dx.doi.org/10.1016/S1369-7021(09)70006-9.
[3] D.J. Müller, Y.F. Dufrêne, Atomic force microscopy: a nanoscopic window on the cell surface, Trends Cell Biol. 21 (8) (2011) 461–469. http://dx.doi.org/10.1016/j.tcb.2011.04.008.
[4] E.J. Candés, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30. http://dx.doi.org/10.1109/MSP.2007.914731.
[5] R.G. Baraniuk, Compressive sensing [lecture notes], IEEE Signal Process. Mag. 24 (4) (2007) 118–121. http://dx.doi.org/10.1109/MSP.2007.4286571.
[6] M.F. Duarte, Y.C. Eldar, Structured compressed sensing: from theory to applications, IEEE Trans. Signal Process. 59 (9) (2011) 4053–4085. http://dx.doi.org/10.1109/TSP.2011.2161982.
[7] Y.C. Eldar, G. Kutyniok (Eds.), Compressed Sensing: Theory and Applications, Cambridge University Press, 2012.
[8] S.B. Andersson, L.Y. Pao, Non-Raster Sampling in Atomic Force Microscopy: a compressed Sensing Approach, in: American Control Conference (ACC), Montral, Canada, 2012, pp. 2485–2490. http://dx.doi.org/10.1109/ACC.2012.6315406.
[9] H.S. Anderson, J. Ilic-Helms, B. Rohrer, J. Wheeler, K. Larson, Sparse imaging for fast electron microscopy, in: Computational Imaging XI, Proceedings of the SPIE, Vol. 8657, Burlingame, California, USA, 2013, pp. (86570C–1)–(86570C–12) http://dx.doi.org/10.1117/12.2008313.
[10] B. Song, N. Xi, R. Yang, K.W.C. Lai, C. Qu, Video Rate Atomic Force Microscopy (AFM) Imaging using Compressive Sensing, in: 11th IEEE International Conference on Nanotechnology, Portland, Oregon, USA, 2011, pp. 1056–1059 http://dx.doi.org/10.1109/NANO.2011.6144587.
[11] P. Binev, W. Dahmen, R. DeVore, P. Lamby, D. Savu, R. Sharpley, Compressed Sensing and Electron Microscopy, in: T. Vogt, W. Dahmen, P. Binev (Eds.), Modeling Nanoscale Imaging in Electron Microscopy, Springer, 2012, pp. 73–126 http://dx.doi.org/10.1007/978-1-4614-2191-7_4.
[12] R. Leary, Z. Saghi, P.A. Midgley, D.J. Holland, Compressed sensing electron tomography, Ultramicroscopy 131 (2013) 70–91. http://dx.doi.org/10.1016/j.ultramic.2013.03.019.
[13] T.L. Jensen, T. Arildsen, J. Østergaard, T. Larsen, Reconstruction of Undersampled Atomic Force Microscopy Images: Interpolation versus Basis Pursuit, in: International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Kyoto, Japan, 2013, p. 6 http://dx.doi.org/10.1109/SITIS.2013.32.
[14] A. Chen, A.L. Bertozzi, P.D. Ashby, P. Getreuer, Y. Lou, Enhancement and Recovery in Atomic Force Microscopy Images, in: T.D. Andrews, R. Balan, J.J. Benedetto, W.

[10] The images and the license conditions are available at http://dx.doi.org/10.5281/zenodo.17573
[11] The images and the license conditions are available at http://dx.doi.org/10.5281/zenodo.60434
[12] See http://creativecommons.org/licenses/by/4.0/.

Czaja, K.A. Okoudjou (Eds.), Excursions in Harmonic Analysis, Vol. 2 of Applied and Numerical Harmonic Analysis, Birkhuser Boston, 2013, pp. 311–332 http://dx.doi.org/10.1007/978-0-8176-8379-5_16.
[15] Y. Luo, S.B. Andersson, A fast image reconstruction algorithm for compressed sensing-based atomic force microscopy, in: American Control Conference (ACC), Chicago, Illinois, USA, 2015, pp. 3503–3508 http://dx.doi.org/10.1109/ACC.2015.7171873.
[16] I.A. Mahmood, S.O.R. Moheimani, B. Bhikkaji, ANew scanning method for fast atomic force microscopy, IEEE Trans. Nanotechnol. 10 (2) (2011) 203–216. http://dx.doi.org/10.1109/TNANO.2009.2036844.
[17] T.R. Meyer, D. Ziegler, C. Brune, A. Chen, R. Farnham, N. Huynh, J.-M. Chang, A.L. Bertozzi, P.D. Ashby, Height drift correction in non-raster atomic force microscopy, Ultramicroscopy 137 (2014) 48–54. http://dx.doi.org/10.1016/j.ultramic.2013.10.014.
[18] B.D. Maxwell, S.B. Andersson, A compressed sensing measurement matrix for atomic force microscopy, in: American Control Conference (ACC), Portland, Oregon, USA, 2014, pp. 1631–1636.http://dx.doi.org/10.1109/ACC.2014.6858710
[19] T. Tuma, J. Lygeros, V. Kartik, A. Sebastian, A. Pantazi, High-speed multiresolution scanning probe microscopy based on Lissajous scan trajectories, Nanotechnology 23 (18) (2012) 9. http://dx.doi.org/10.1088/0957-4484/23/18/185501.
[20] R.G. Baraniuk, V. Cevher, M.F. Duarte, C. Hegde, Model-based compressive scensing, IEEE Trans. Inf. Theory 56 (4) (2010) 1982–2001. http://dx.doi.org/10.1109/TIT.2010.2040894.
[21] N. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, IEEE Trans. Comput. C. 23 (1) (1974) 90–93. http://dx.doi.org/10.1109/T-C.1974.223784.
[22] K.K. Herrity, A.C. Gilbert, J.A. Tropp, Sparse Approximation via Iterative Thresholding, in: IEEE International Conference on Acoustics, Speech and Signal Proccesing (ICASSP), Toulouse, France, 2006, pp. III–624 – III–627.http://dx.doi.org/10.1109/ICASSP.2006.1660731
[23] T. Blumensath, M.E. Davies, G. Rilling, Greedy algorithms for compressed sensing, in: Y.C. Eldar, G. Kutyniok (Eds.), Compressed Sensing: Theory and Applications, Cambridge University Press, 2012, Ch. 8, pp. 348–393.
[24] L. Zhu, W. Zhang, D. Elnatan, B. Huang, Faster STORM using compressed sensing, Nat. Methods 9 (7) (2012) 721–723. http://dx.doi.org/10.1038/nmeth.1978.
[25] P. Sen, S. Darabi, Compressive Image Super-resolution, in: Forty Third Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, California, USA, 2009, pp. 1235–1242.http://dx.doi.org/10.1109/ACSSC.2009.5469968
[26] D.L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (4) (2006) 1289–1306. http://dx.doi.org/10.1109/TIT.2006.871582.
[27] E.J. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2) (2006) 489–509. http://dx.doi.org/10.1109/TIT.2005.862083.
[28] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B (Methodol.) 58 (1) (1996) 267–288.
[29] J. Romberg, Imaging via compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 14–20. http://dx.doi.org/10.1109/MSP.2007.914729.
[30] G.K. Wallace, The JPEG picture compression standard, IEEE Trans. Consum. Electron. 38 (1) (1992) 18–34. http://dx.doi.org/10.1109/30.125072.
[31] G. Strang, T. Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press, Wellesley, USA, 1996.
[32] J.P. Vila, P. Schniter, Expectation-maximization gaussian-mixture approximate message passing, IEEE Trans. Signal Process. 61 (19) (2013) 4658–4672. http://dx.doi.org/10.1109/TSP.2013.2272287.
[33] T. Arildsen, C.S. Oxvig, P.S. Pedersen, J. Østergaard, T. Larsen, Reconstruction algorithms in undersampled AFM imaging, IEEE J. Sel. Top. Signal Process. 10 (1) (2016) 31–46. http://dx.doi.org/10.1109/JSTSP.2015.2500363.
[34] C.S. Oxvig, P.S. Pedersen, T. Arildsen, T. Larsen, Surpassing the Theoretical 1-norm Phase Transition in Compressive Sensing by Tuning the Smoothed l0 algorithm, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, 2013, pp. 6019–6023.http://dx.doi.org/10.1109/ICASSP.2013.6638820
[35] T. Blumensath, M.E. Davies, Iterative thresholding for sparse approximations, J. Fourier Anal. Appl. 14 (5–6) (2008) 629–654. http://dx.doi.org/10.1007/s00041-008-9035-z.
[36] D.L. Donoho, De-noising by soft-thresholding, IEEE Trans. Inf. Theory 41 (3) (1995) 613–627. http://dx.doi.org/10.1109/18.382009.
[37] M.A.T. Figueiredo, J.M. Bioucas-Dias, R.D. Nowak, Majorization-minimization algorithms for wavelet-based image restoration, IEEE Trans. Image Process. 16 (12) (2007) 2980–2991. http://dx.doi.org/10.1109/TIP.2007.909318.
[38] A. Maleki, D.L. Donoho, Optimally tuned iterative reconstruction algorithms for compressed sensing, IEEE J. Sel. Top. Signal Process. 4 (2) (2010) 330–341. http://dx.doi.org/10.1109/JSTSP.2009.2039176.
[39] P. Klapetek, Quantitative Data Processing in Scanning Probe Microscopy: SPM Applications for Nanometrology, 1st Edition, Micro & Nano Technologies, Elsevier, 2013.
[40] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd Edition, Wiley-Blackwell, 2000.
[41] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd Edition, Cambridge University Press, 32 Avenue of the Americas, New York, NY 10013-2473, USA, 2007.
[42] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612. http://dx.doi.org/10.1109/TIP.2003.819861.
[43] C.S. Oxvig, P.S. Pedersen, T. Arildsen, J. Østergaard, T. Larsen, Magni: a python package for compressive sampling and reconstruction of atomic force microscopy images, J. Open Res. Softw. 2 (1) (2014) e29. http://dx.doi.org/10.5334/jors.bk.
[44] P.S. Pedersen, C.S. Oxvig, J. stergaard, T. Larsen, Validating Function Arguments in Python Signal Processing Applications, in: Proceedings of the 15th Python in Science Conference, Austin, Texas, USA, 2016, pp. 106-113.
[45] C.S. Oxvig, T. Arildsen, T. Larsen, Storing Reproducible Results from Computational Experiments using Scientific Python Packages, in: Proceedings of the 15th Python in Science Conference, Austin, Texas, USA, 2016, pp. 45-50.
[46] Z. Wang, A.C. Bovik, Modern Image Quality Assessment, Synthesis Lectures on Image, Video, and Multimedia Processing, Morgan & Claypool Publishers, 2006. http://dx.doi.org/10.2200/S00010ED1V01Y200508IVM003.

Paper C

Entrywise Squared Transforms for High Dimensional Signal Reconstruction via Generalized Approximate Message Passing

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen

The paper has been submitted to
IEEE Transactions on Computational Imaging

The succeeding 12 non-blank pages constitute the submitted manuscript which is subject to the following copyright notice:

Entrywise Squared Transforms for High Dimensional Signal Reconstruction via Generalized Approximate Message Passing

Christian Schou Oxvig, *Student Member, IEEE,* Thomas Arildsen, and Torben Larsen, *Senior Member, IEEE*

***Abstract*—A variety of image reconstruction applications such as undersampled scanning probe microscopy or magnetic resonance imaging are characterised by measurements that may be regarded as linear combinations of a (row) sub-sampled structured transform, e.g. a Fourier, Cosine or Hadamard transform. This characterisation may either stem from the modelling of the physical image acquisition system e.g., Fourier modes acquisition in magnetic resonance imaging or may simply be due to the availability of a fast and memory efficient implementation of the structured transform - a property that is essential in high dimensional signal processing applications. The recently proposed Approximate Message Passing (AMP) methods for signal reconstruction rely on not only the first order characterisation of the acquisition system but also the second order characterisation, i.e. the entrywise absolute value squared system transform, as additional information to enhance the reconstruction. We derive efficient (in terms of memory requirement and computational cost) ways to implement such entrywise absolute value squared transforms for high dimensional signal reconstruction using AMP when the system is described by a sub-sampled separable structured transform. Through a large set of simulation experiments, we detail the performance of our efficient entrywise absolute value squared transforms compared to the alternative sum approximations. In particular, we point out the cases where it is a necessity to use our proposed method in order to reconstruct undersampled signals.**

***Index Terms*—Image Reconstruction, Scientific Computing, Computational Complexity, Computational Efficiency, Estimation, Message Passing, Compressed Sensing.**

I. Introduction

METHODS for reconstructing undersampled high dimensional signals have been heavily investigated in various imaging applications. For instance, in Magnetic Resonance Imaging (MRI) in an attempt to increase imaging speed [1] or in hyperspectral imaging applications [2]. Or in Scanning Probe Microscopies such as Atomic Force Microscopy [3] and Scanning Electron Microscopy [4] in an attempt to reduce imaging time [5] or reduce the risk of damaging the specimen by reducing the interaction with it, as is critical in cell imaging [6]. One way to reconstruct undersampled high dimensional signals is by use of the Approximate Message Passing (AMP) based algorithms [7], [8] which have recently been shown to yield excellent reconstruction performance in a broad range of applications. When using the AMP algorithm, information about an assumed statistical prior on the signal to be reconstructed may be used to enhance the reconstruction. The Generalized Approximate Message Passing (GAMP) [9] algorithm additionally allows for including information about a broad range of measurement channels in the reconstruction. Thus, due to the generality of GAMP in terms of modelling prior information on the signal and measurement channel as well as its state-of-the-art reconstruction capabilities, it seems compelling to use GAMP for reconstruction in imaging and other high dimensional undersampling applications.

Several studies have already explored the use of (G)AMP algorithms in imaging applications, e.g. in MRI [10], hyperspectral imaging [2], or in more general imaging and high dimensional applications [11], [12], [13], [14], [15]. In the present work, we are considering several practical implementation issues in using GAMP for high dimensional signal reconstruction. The rest of this introductory section of the paper is devoted to giving an informal overview of the problem, we are considering. Having motivated the problem and sketched our contributions, the remainder of the paper is devoted to providing a formal disclosure of the problem, our results and contributions, and their relation to existing literature on the topic.

A. Motivation

A subtle, but critical, detail in using the GAMP algorithm for reconstruction of undersampled images (and other high dimensional signals) is the handling of the second order characterisation of the imaging system. That is, if the imaging system is modelled by the matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ (with $m \ll n$ in the undersampling regime), then the GAMP algorithm additionally relies on the characterisation of the entrywise absolute sqaured system matrix $|\mathbf{A}|^{\circ 2}$, i.e. the matrix with entries $|a_{ji}|^2$ for all entries a_{ij} in $\mathbf{A}$. Specifically, the GAMP algorithm requires computing matrix-vector products involving $\mathbf{A}$ and $|\mathbf{A}|^{\circ 2}$. For large problem sizes, i.e. for large n, it may easily become infeasible to store $\mathbf{A}$, and $|\mathbf{A}|^{\circ 2}$ in memory on a computer. Additionally, it may also become infeasible to compute the matrix-vector products involving $\mathbf{A}$, and $|\mathbf{A}|^{\circ 2}$. This is a particularly distinct problem in high dimensional signal reconstruction (2D images or higher dimensional signals) where the problem size n becomes the product of the sizes of all the signal dimensions and, thus, easily gets very large. In

The authors are with the Faculty of Engineering and Science, Department of Electronic Systems, Aalborg University, Aalborg, Denmark e-mail: {cso,tha,tl}@es.aau.dk.

This project has been supported by 1) The Danish Council for Independent Research (DFF/FTP) for project number 1335-00278B/12-134971, and 2) by the Danish e-Infrastructure Cooperation (DeIC) for project number DeIC2013.12.23.

order to handle such large problem sizes many signal processing techniques leverage fast transforms for implementing the required system transform matrix-vector products involving $\mathbf{A}$, e.g. a Fast Fourier Transform (FFT) based method for implementing Fourier-like system transforms. Ways to handle the second order system transforms involving $|\mathbf{A}|^{\circ 2}$ are less clear, though.

B. The Problem

In this work we consider ways to implement the system transform matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ for the case of sub-sampled separable transforms which are applicable in many practical high dimensional settings. That is, in its simplest form, we consider a system matrix with a structure such that $\mathbf{A} = \mathbf{D}_\Omega(\mathbf{F}_1 \otimes \mathbf{F}_2)$ where $\mathbf{D}_\Omega \in \mathbb{R}^{m \times n}$ is a sub-sampling operator (an identity matrix with some of its rows removed) that selects rows from a matrix $\mathbf{F} = \mathbf{F}_1 \otimes \mathbf{F}_2 \in \mathbb{R}^{n \times n}$ defined by the Kronecker product of the matrices $\mathbf{F}_1$ and $\mathbf{F}_2$. Additionally, we study the performance of *sum approximations* to matrix-vector products involving $|\mathbf{A}|^{\circ 2}$. That is, under certain conditions, the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ in GAMP may be replaced by scaled sums of the vectors.

C. Our Contributions

We present theoretical results on ways to efficiently implement $|\mathbf{A}|^{\circ 2}$ and provide comprehensive empirical evaluations of the performance of our proposed implementations of $|\mathbf{A}|^{\circ 2}$. Specifically, our contributions are

- We present several theorems on Hadamard Powers of Kronecker products. These theorems define ways to efficiently implement $|\mathbf{A}|^{\circ 2}$ in the case of sub-sampled separable transforms. In particular, in its simplest form, we have the result $|\mathbf{A}|^{\circ 2} = |\mathbf{D}_\Omega|^{\circ 2}(|\mathbf{F}_1|^{\circ 2} \otimes |\mathbf{F}|_2^{\circ 2})$ for $\mathbf{A} = \mathbf{D}_\Omega(\mathbf{F}_1 \otimes \mathbf{F}_2)$. Our results are much more general, though, and defines ways to implement $|\mathbf{A}|^{\circ 2}$ for many practically applicable $\mathbf{A}$ used in high dimensional signal reconstruction applications.
- We discuss the differences between the various sum approximations to matrix-products involving $|\mathbf{A}|^{\circ 2}$ in GAMP. These sum approximations are well known in the literature. However, to the best of our knowledge, no other work has discussed the practical implications of using such sum approximations in high dimensional signal reconstruction. In particular, we give pointers on the critical parameters that must be chosen (or estimated) correctly in order for the sum approximations to yield usable results. This is of great importance in practical implementations of GAMP based on sum approximations since the algorithm tends to be less robust to the choice of these parameters.
- We present results from a large simulation study that evaluates the performance of GAMP for practical image reconstruction when using our proposed methods for implementing $|\mathbf{A}|^{\circ 2}$ as well as when using the various sum approximations. Our simulation results support our claim that the parameter choices for the sum approximations are critical since performance may significantly degrade if the parameters are chosen incorrectly. Furthermore, our simulation results suggest that it may not always be possible to use GAMP with sum approximations in practical high dimensional signal reconstruction. It seems that there are cases which require implementing GAMP using the full $|\mathbf{A}|^{\circ 2}$ transform. Thus, our theorems on Hadamard Powers for Kronecker products may potentially enable the use of GAMP in some high dimensional signal reconstruction applications where no other GAMP solution is feasible.

The remainder of the paper is organised as follows. In Section II, we introduce our notation and briefly review the GAMP algorithm and its use in reconstruction of undersampled signals. In Section III, we formally introduce the high dimensional signal reconstruction setting and the system transforms that we are considering. Section IV presents our main theoretical results for such transforms based on Hadamard powers of sub-sampled Kronecker products. In Section V we detail the sum approximations that may be used as an alternative to the entrywise absolute value squared transform in GAMP. Section VI provides an overview of, and results from, a set of numerical experiments used to evaluate the performance of our proposed method compared to the sum approximations. A discussion of our results is given in Section VII whereas Section VIII states our conclusions. Finally, proofs of our theoretical results are given in Appendix A.

II. Generalized Approximate Message Passing

We consider the reconstruction of a vector $\boldsymbol{\alpha} \in \mathbb{C}^{n \times 1}$ from noisy measurements $\mathbf{y} \in \mathbb{C}^{m \times 1}$ with $m \ll n$, i.e. in an undersampling regime. We assume that $\boldsymbol{\alpha}$ has been measured through a linear transform $\mathbf{A} \in \mathbb{C}^{m \times n}$ such that

$$\mathbf{z} = \mathbf{A}\boldsymbol{\alpha} \tag{1}$$
$$= \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} \tag{2}$$

for the non-noisy (and generally unknown) measurements $\mathbf{z} \in \mathbb{C}^{m \times 1}$ and a decomposition of $\mathbf{A}$ into a measurement matrix $\boldsymbol{\Phi} \in \mathbb{C}^{m \times p}$ and a dictionary matrix $\boldsymbol{\Psi} \in \mathbb{C}^{p \times n}$. The known noisy measurements $\mathbf{y}$ may be given by

$$\mathbf{y} = \mathbf{z} + \mathbf{e} \tag{3}$$
$$= \mathbf{A}\boldsymbol{\alpha} + \mathbf{e} \tag{4}$$

where in this example we are considering an additive measurement noise $\mathbf{e} \in \mathbb{C}^{m \times 1}$. Note, however, that other measurement channels may be assumed when considering the relation between $\mathbf{y}$ and $\mathbf{z}$.

The above is the well-known setting used in the Compressed Sensing (CS) theory (see e.g. [16], [17], [18]). In CS one assumes that $\boldsymbol{\alpha}$ has some structure, e.g. is sparse in the dictionary $\boldsymbol{\Psi}$. Oftentimes, the dictionary $\boldsymbol{\Psi}$ is introduced since the signal of interest (e.g. an image) $\mathbf{x} \in \mathbb{C}^{p \times 1}$ is not itself sparse but is assumed sparse in e.g. the wavelet domain. Once (an estimate of) the sparse $\boldsymbol{\alpha}$ has been found, one may then find the signal of interest

$$\mathbf{x} = \boldsymbol{\Psi}\boldsymbol{\alpha} \tag{5}$$

with

$$\mathbf{z} = \mathbf{\Phi x} \tag{6}$$

The Generalized Approximate Message Passing (GAMP) algorithm [9] allows for reconstructing signals from the undersampled noisy measurements $\mathbf{y}$. In a Bayesian interpretation of GAMP, one may define priors on both $\boldsymbol{\alpha}$ and the measurement channel (e.g. a prior on the additive noise $\mathbf{e}$). The GAMP algorithm is then capable of finding either minimum mean squared error (MMSE) or maximum a posteriori (MAP) estimates of $\boldsymbol{\alpha}$. Here we focus on the MMSE GAMP as shown in Table I.

In the MMSE GAMP, we have the initialisation

$$\mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}] = \int_{\boldsymbol{\alpha}} \boldsymbol{\alpha} p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I) d\boldsymbol{\alpha} \tag{7}$$

$$\mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha}) = \int_{\boldsymbol{\alpha}} |\boldsymbol{\alpha}|^2 p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I) d\boldsymbol{\alpha} - |\mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]|^2 \tag{8}$$

for some separable prior distribution $p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I) = \prod_j p(\alpha_j|[\boldsymbol{\theta}_I]_j)$ on $\boldsymbol{\alpha}$ parameterised by the vector of parameters $\boldsymbol{\theta}_I$. Furthermore, we have the scalar output channel functions $f_{\bar{z}}$, $f_{\tilde{z}}$ and scalar input channel functions $f_{\bar{\alpha}}$, $f_{\tilde{\alpha}}$ that operate independently on each entry of the vector arguments according to

$$f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{1}{\mathcal{Z}_o} \int_z z p(y|z; \boldsymbol{\theta}_o) \mathcal{N}(z; o, v) dz \tag{9}$$

$$f_{\tilde{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{1}{\mathcal{Z}_o} \int_z |z|^2 p(y|z; \boldsymbol{\theta}_o) \mathcal{N}(z; o, v) dz - |f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o)|^2 \tag{10}$$

$$f_{\bar{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{1}{\mathcal{Z}_I} \int_\alpha \alpha p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha \tag{11}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{1}{\mathcal{Z}_I} \int_\alpha |\alpha|^2 p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha - |f_{\bar{\alpha}}(s, r; \boldsymbol{\theta}_I)|^2 \tag{12}$$

$$\mathcal{Z}_I = \int_\alpha p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha \tag{13}$$

$$\mathcal{Z}_o = \int_z p(y|z; \boldsymbol{\theta}_o) \mathcal{N}(z; o, v) dz \tag{14}$$

$$\mathcal{N}(\alpha; r, s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{1}{2}\frac{(\alpha - r)^2}{s}\right) \tag{15}$$

$$\mathcal{N}(z; o, v) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{1}{2}\frac{(z - o)^2}{v}\right) \tag{16}$$

for some separable distribution $p(\mathbf{y}|\mathbf{z}; \boldsymbol{\theta}_o) = \prod_i p(y_i|z_i; [\boldsymbol{\theta}_o]_i)$ on $\mathbf{y}$ given $\mathbf{z}$ parameterised by the vector of parameters $\boldsymbol{\theta}_o$. Finally, $\mathbf{0}_m$ is the zero-vector in $\mathbb{R}^{m\times 1}$, $\mathbf{1}_n$ is the one-vector in $\mathbb{R}^{n\times 1}$, and $|\mathbf{A}|^{\circ 2}$ is the entrywise absolute value squared version of the system matrix $\mathbf{A}$, i.e. we are considering the matrix with entries $|a_{ij}|^2$ for all a_{ij} in $\mathbf{A}$. The output of MMSE GAMP is the MMSE estimate of $\boldsymbol{\alpha}$, i.e. $\bar{\boldsymbol{\alpha}}$, and the corresponding estimate of the variances of the entries in $\bar{\boldsymbol{\alpha}}$, i.e. $\tilde{\boldsymbol{\alpha}}$.

As pointed out in [19], the GAMP iteration is generally computationally dominated by the matrix-vector products involving $\mathbf{A}$, $\mathbf{A}^H$, $|\mathbf{A}|^{\circ 2}$, and $(|\mathbf{A}|^{\circ 2})^T$. However, this is only true if the computational cost of evaluating the integrals in (9)-(16) is negligible, i.e. if the integrals have closed form solutions. In the general case one may have to resort to numerical integration which may easily become more computationally expensive than evaluating the matrix-vector products. To further analyse the computational bottlenecks of the GAMP algorithm, we profiled[1] the algorithm in Table I with channel functions that have simple closed form solutions. The results showed that even when using optimised Basic Linear Algebra Subroutines (BLAS) libraries to accelerate the computation of the matrix-vector products, around 99% of the time was spent on computing these matrix-vector products (lines 3, 4, 9, and 10 in Table I). For large problem sizes this $\mathcal{O}(mn)$ cost of computing matrix-vector products may become infeasible. Also memory requirements for storing $\mathbf{A}$ and $|\mathbf{A}|^{\circ 2}$ may likewise become infeasible for large problem sizes.

TABLE I
MINIMUM MEAN SQUARED ERROR (MMSE) GENERALIZED APPROXIMATE MESSAGE PASSING (GAMP) [9]. NOTE THAT WE USE $\circ$ TO DENOTE ENTRYWISE MULTIPLICATION OF VECTORS AND $\oslash$ TO DENOTE ENTRYWISE DIVISION OF VECTORS.

1 **initialise:** $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\tilde{\boldsymbol{\alpha}}_0 = \mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})$, $\mathbf{q}_0 = \mathbf{0}_m$
2 **for** $t = 1$ **to** $T_{\max}$ **do**
3 $\quad \mathbf{v}_t = |\mathbf{A}|^{\circ 2}\tilde{\boldsymbol{\alpha}}_{t\text{-}1}$ Factor side / output updates
4 $\quad \mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \mathbf{v}_t \circ \mathbf{q}_{t\text{-}1}$
5 $\quad \bar{\mathbf{z}}_t = f_{\bar{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
6 $\quad \tilde{\mathbf{z}}_t = f_{\tilde{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
7 $\quad \mathbf{q}_t = (\bar{\mathbf{z}}_t - \mathbf{o}_t) \oslash \mathbf{v}_t$
8 $\quad \mathbf{u}_t = (\mathbf{v}_t - \tilde{\mathbf{z}}_t) \oslash (\mathbf{v}_t \circ \mathbf{v}_t)$
9 $\quad \mathbf{s}_t = \mathbf{1}_n \oslash ((|\mathbf{A}|^{\circ 2})^T \mathbf{u}_t)$ Variable side / input updates
10 $\quad \mathbf{r}_t = \bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{s}_t \circ \mathbf{A}^H \mathbf{q}_t$
11 $\quad \bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
12 $\quad \tilde{\boldsymbol{\alpha}}_t = f_{\tilde{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
13 $\quad$ **if** stop criterion is met **then**
14 $\quad\quad$ **break**
15 $\quad$ **end if**
16 **end for**

A. Some Additional Notation Details

In the remainder of the paper, we make use of the Kronecker and Hadamard products of matrices as well as the Hadamard power of a matrix (see e.g. [20]). The Kronecker product of two matrices $\mathbf{A} \in \mathbb{C}^{m\times n}$ and $\mathbf{B} \in \mathbb{C}^{k\times l}$, denoted by $\mathbf{A}\otimes\mathbf{B} \in \mathbb{C}^{mk\times nl}$, is

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \tag{17}$$

where a_{ij} denotes the ij-th entry of $\mathbf{A}$.

[1]The details of the profiling may be found in the profiling script which is available at https://dx.doi.org/10.5278/240710282

The Hadamard product (the entrywise product) of two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{m\times n}$, denoted by $\mathbf{A} \circ \mathbf{B} \in \mathbb{C}^{m\times n}$, is

$$\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} a_{11} \cdot b_{11} & \cdots & a_{1n} \cdot b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot b_{m1} & \cdots & a_{mn} \cdot b_{mn} \end{bmatrix} \tag{18}$$

The p-th Hadamard power of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ with $p \in \mathbb{Z}$, denoted by $\mathbf{A}^{\circ p} \in \mathbb{C}^{m\times n}$, is

$$\mathbf{A}^{\circ p} = \begin{bmatrix} a_{11}^p & \cdots & a_{1n}^p \\ \vdots & \ddots & \vdots \\ a_{m1}^p & \cdots & a_{mn}^p \end{bmatrix} \tag{19}$$

That is, it is the Hadamard product with itself p times.

III. High Dimensional Signal Reconstruction

In this work, we are interested in the large problem setting, i.e. $n \gtrsim 10000$. Such large problems often occur in applications involving high dimensional signals (two or more dimensions), e.g. a reconstruction of a 256-by-256 pixels image. The signal of interest $\mathbf{x}$ is then a vector representation of the high dimensional signal, e.g. in the 256-by-256 image case, one may stack the columns of the image matrix to obtain an $\mathbf{x} \in \mathbb{R}^{65536\times 1}$. Such large problems pose several challenges in implementing a reconstruction method like GAMP. Consider for instance a 256-by-256 pixels image undersampled such that $m = 0.5n$. In this case, storing $\mathbf{A}$ in a computer in double precision (8 bytes per entry) requires $8 \cdot 0.5 \cdot (256^2)^2/1024^3 = 16$ GiB RAM. Thus, for larger problem sizes or higher dimensional signals, it quickly becomes infeasible to store $\mathbf{A}$ due to the $\mathcal{O}(n^4)$ memory scaling. Also, reconstruction algorithms such as GAMP generally depend on computing matrix-vector products involving $\mathbf{A}$ which may become computationally infeasible when considering these large problem sizes. For that reason (or simply due to the physical nature of the imaging system), one typically assumes a structure on the problem that allows for more efficient implementations of the linear transforms involving $\mathbf{A}$ and $\mathbf{A}^H$. One such general structure framework is the so-called Structually Random Matrices (SRM) [21] which are closely related to structured random matrices [22] and spread-spectrum form matrices [23]. The SRM framework includes matrices described[2] by

$$\mathbf{A} = \mathbf{D}_\Omega \mathbf{F} \tag{20}$$

where $\mathbf{D}_\Omega \in \mathbb{C}^{m\times n}$ is created from the diagonal matrix $\mathbf{D} = \operatorname{diag}(d_1, \ldots, d_n) \in \mathbb{C}^{n\times n}$ (usually an identity matrix) by only keeping the rows that are indexed by the set Ω and $\mathbf{F} \in \mathbb{C}^{n\times n}$ is an orthogonal matrix. We give concrete examples of this type of transform in Section IV-B. We note that the sub-sampling matrix $\mathbf{D}_\Omega$ may be efficiently implemented using a lookup-table to keep track of the rows that are kept. As pointed out in [21], [22], $\mathbf{F}$ is oftentimes for practical reasons chosen to have a fast transform, e.g. a Fast Fourier Transform (FFT) if $\mathbf{F}$ is a Fourier matrix. Such a fast transform essentially solves the memory and computation issues in implementing reconstruction algorithms that make use of matrix-vector products of $\mathbf{A}$.

One particularly interesting case, that is the focus of the present work, is when $\mathbf{F}$ may be expressed as a Kronecker product of l lower dimensional matrices as in the theory of Kronecker Compressive Sensing [24]

$$\mathbf{F} = \mathbf{F}_1 \otimes \cdots \otimes \mathbf{F}_l \tag{21}$$

where l is the signal dimension, e.g. $l = 2$ for an image. In such cases, matrix-vector products involving $\mathbf{F}$ may be computed efficiently by applying the lower dimensional matrices independently to each signal dimension. Consider for instance the case of a 1D transform $\mathbf{T} \in \mathbb{C}^{\sqrt{n}\times\sqrt{n}}$ (assuming $\sqrt{n}$ an integer) from which as 2D transform may be defined by the Kronecker product $(\mathbf{T} \otimes \mathbf{T}) \in \mathbb{C}^{n\times n}$, e.g. a 2D Discrete Fourier Transform (DFT). And consider an image $\mathbf{X} \in \mathbb{R}^{\sqrt{n}\times\sqrt{n}}$ with $\mathbf{x} \in \mathbb{R}^{n\times 1}$ a vector representation of $\mathbf{X}$ created by stacking its columns. Then we may apply the 2D transform by applying the 1D transform to each dimension (the rows of and columns of $\mathbf{X}$). Specifically we have (see e.g. [25])

$$\mathbf{Z} = (\mathbf{T}(\mathbf{T}\mathbf{X})^T)^T \tag{22}$$
$$= (\mathbf{T}\mathbf{X}^T\mathbf{T}^T)^T \tag{23}$$
$$= \mathbf{T}\mathbf{X}\mathbf{T}^T \tag{24}$$
$$\mathbf{z} = (\mathbf{T} \otimes \mathbf{T})\mathbf{x} \tag{25}$$

for $\mathbf{z}$ a vector representation of $\mathbf{Z}$ created by stacking its columns. Thus, in using this separability property of the transform, the computational complexity is reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\sqrt{n})$. Possibly even more importantly, the memory requirement is reduced from n^2 entries for storing $\mathbf{T} \otimes \mathbf{T}$ to n entries for storing $\mathbf{T}$.

In Table II, this difference is summarised for the 2D case with the $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$ transforms used in MMSE GAMP (Table I). For the 2D MMSE GAMP case, we note that going from a computational cost of $4mn$ to $4\sqrt{n}n$ is a significant result since normally $\sqrt{n} \ll m$, even in the undersampling setting. Likewise, going from a memory cost of mn to $2n$ is a significant result since for large problem sizes, we normally have $m \gg 2$. To further analyse the gain in using such fast transforms, we profiled[3] the GAMP algorithm similarly to the profiling reported on in Section II. The only difference is that this time we consider a 2D case and use a FFT based transform in computing the matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^H$ and a 2D separable transform in computing the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$. The in- and output channels are the same. Using

[2] A SRM is described by $\mathbf{A} = \mathbf{D}_\Omega \mathbf{F}\mathbf{R}$ with $\mathbf{R}$ a pre-randomisation matrix. Specifically, $\mathbf{R}$ is either a permutation matrix or a diagonal matrix with uniformly random sign changes on the diagonal entries. We note that our framework for finding $|\mathbf{A}|^{\circ 2}$ extends to the full SRM by arguments similar to those used for $\mathbf{D}_\Omega$ since both $\mathbf{D}_\Omega$ and $\mathbf{R}$ are essentially scaled and permuted identity matrices. However, in order to keep our discussion concise and clear, we restrict our attention to SRMs without the pre-randomization matrix as given in (20).

[3] The details of the profiling may be found in the profiling script which is available at https://dx.doi.org/10.5278/240710282

TABLE II
COMPARISON OF COMPUTATIONAL AND STORAGE COSTS IN IMPLEMENTING $|\mathbf{A}|^{\circ 2}$ AND $(|\mathbf{A}|^{\circ 2})^T$ FOR THE 2D CASE.

	Matrix-vector product	Separable Transform	Sum Approximation
Computational cost	$m(2n-1)+n(2m-1) \approx 4mn$	$m+2\sqrt{n}\sqrt{n}2(\sqrt{n}-1)+n \approx 4\sqrt{n}n$	$m+n$
Memory cost	mn	$2\sqrt{n}\sqrt{n}=2n$	1

this setup we find that around 20% of the time is spent on computing the transforms involving $\mathbf{A}$ and $\mathbf{A}^H$ (lines 4 and 10 in Table I), around 50% of the time is spent on computing the transforms involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$ (lines 3 and 9 in Table I), whereas around 30% of the time is spent on evaluating the in- and output channels (lines 5, 6, 11, and 12 in Table I). Thus, the savings in computing the linear transforms have now caused the channel evaluations to become computationally significant, even for channels that have closed form solutions.

At this point we note that the AMP algorithms have only been proven to converge for certain classes of $\mathbf{A}$ matrices and only in the large system limit $n \to \infty$. In particular, convergence has been proven when $\mathbf{A}$ has i.i.d. entries drawn from a Gaussian distribution [26] or a sub-Gaussian distribution [27]. For certain other matrices (including sub-sampled unitary matrices), a damping strategy may be used to ensure convergence [28]. Also extensive empirical evidence suggests that GAMP converges for various other practically applicable matrices and for finite problem sizes [27], [29]. These empirical results are in line with our numerical experiments described in Section VI.

IV. ENTRYWISE SQUARED ABSOLUTE VALUE TRANSFORMS IN GAMP

As is evident from the MMSE GAMP algorithm in Table I and our profiling results reported on in Sections II and III, it is only valuable to have efficient transforms for $\mathbf{A}$ and $\mathbf{A}^H$ if there also exists efficient transforms for the entrywise absolute value squared transforms $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$ since all of these transforms scale similarly with the problem size in terms of both memory and computational cost. Thus, we now turn the attention towards efficient implementations of the transforms $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$.

It may be that the system matrix $\mathbf{A}$ has a structure that allows for particularly simple implementations of matrix-vector products involving $|\mathbf{A}|^{\circ 2}$, e.g. when $\mathbf{A}$ is a sub-sampled Fourier or Hadamard transform. In such cases the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$ reduce to simple sums of the vector [30]. Alternatively, one may potentially mitigate the large problem size issues by processing smaller blocks of the full scale problem at a time if the problem permits such block processing. Another potential solution is to use the sum approximations detailed in Section V if these approximations are applicable to the problem. Here, though, we focus on the case of the sub-sampled separable $\mathbf{A}$ described by (20) and (21). It turns out that in this case it is possible to find similar sub-sampled separable transforms for $|\mathbf{A}|^{\circ 2}$.

A. Efficient Entrywise Squared Absolute Value Transforms for the Sub-sampled Separable Case

We now present our main theoretical contributions, i.e. our results on Hadamard powers of sub-sampled Kronecker products. We then show that $|\mathbf{A}|^{\circ 2}$ is a special case of such Hadamard powers of Kronecker products which allows us to use our results for finding efficient transforms for matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ in GAMP. Our first result is on the computation of Hadamard powers of sub-sampled system matrices as the SRM in (20).

Theorem 1: Let $\mathbf{D} = \text{diag}(d_1, \dots, d_n) \in \mathbb{C}^{n \times n}$ be any diagonal matrix with diagonal entries $d_1, \dots, d_n$. Furthermore, let $\mathbf{D}_\Omega \in \mathbb{C}^{m \times n}$ be a matrix created from $\mathbf{D}$ by taking only the rows indexed by the set of indices Ω with $|\Omega| = m \leq n$. That is, the rows not indexed by Ω are removed from $\mathbf{D}$. Now, take any matrix $\mathbf{A} \in \mathbb{C}^{n \times l}$ and $p \in \mathbb{Z}$. Then,

$$(\mathbf{D}_\Omega \mathbf{A})^{\circ p} = \mathbf{D}_\Omega^{\circ p} \mathbf{A}^{\circ p} \tag{26}$$

where $\mathbf{D}_\Omega \mathbf{A} \in \mathbb{C}^{m \times l}$ denotes the matrix product of $\mathbf{D}_\Omega$ and $\mathbf{A}$. A proof of the theorem is given in Appendix A.

Next, we give three results on Hadamard powers of system matrices that are described by Kronecker products as in (21). The first result is on a system matrix described by a series of Kronecker products.

Theorem 2: For any $\mathbf{A}_1 \in \mathbb{C}^{m_1 \times n_1}, \dots, \mathbf{A}_s \in \mathbb{C}^{m_s \times n_s}$, and any $p \in \mathbb{Z}$,

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p} = \mathbf{A}_1^{\circ p} \otimes \cdots \otimes \mathbf{A}_s^{\circ p} \tag{27}$$

A proof of the theorem is given in Appendix A.

The last two results relate to system matrices that are described by matrix products of matrices that each are described by Kronecker products. The two results are closely connected and may be combined if needed.

Theorem 3: Let $\mathbf{A}_1 \in \mathbb{C}^{m_1 \times n_1}, \dots, \mathbf{A}_s \in \mathbb{C}^{m_s \times n_s}$, $\mathbf{B}_1 \in \mathbb{C}^{n_1 \times k_1}, \dots, \mathbf{B}_s \in \mathbb{C}^{n_s \times k_s}$, and $\mathbf{E}_s = \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s, \mathbf{F}_s = \mathbf{B}_1 \otimes \cdots \otimes \mathbf{B}_s$. Then for any $p \in \mathbb{Z}$,

$$(\mathbf{E}_s \mathbf{F}_s)^{\circ p} = (\mathbf{A}_1 \mathbf{B}_1)^{\circ p} \otimes \cdots \otimes (\mathbf{A}_s \mathbf{B}_s)^{\circ p} \tag{28}$$

A proof of the theorem is given in Appendix A.

Theorem 4: Let $\mathbf{A}_1 \in \mathbb{C}^{m_1 \times n_1}, \mathbf{A}_2 \in \mathbb{C}^{n_1 \times n_2}, \dots, \mathbf{A}_s \in \mathbb{C}^{n_{s-1} \times n_s}$, $\mathbf{B}_1 \in \mathbb{C}^{l_1 \times q_1}, \mathbf{B}_2 \in \mathbb{C}^{q_1 \times q_2}, \dots, \mathbf{B}_s \in \mathbb{C}^{q_{s-1} \times q_s}$, and $\mathbf{E}_1 = \mathbf{A}_1 \otimes \mathbf{B}_1, \dots, \mathbf{E}_s = \mathbf{A}_s \otimes \mathbf{B}_s$. Then for any $p \in \mathbb{Z}$,

$$(\mathbf{E}_1 \dots \mathbf{E}_s)^{\circ p} = (\mathbf{A}_1 \dots \mathbf{A}_s)^{\circ p} \otimes (\mathbf{B}_1 \dots \mathbf{B}_s)^{\circ p} \tag{29}$$

A proof of the theorem is given in Appendix A.

We note that for any two complex numbers $a_1, a_2 \in \mathbb{C}$ given in polar coordinates $(\bar{r}, \varphi)$, i.e., $a_1 = \bar{r}_1[\cos(\varphi_1) +$

$j\sin(\varphi_1)], a_2 = \bar{r}_2[\cos(\varphi_2) + j\sin(\varphi_2)]$ and any $p \in \mathbb{Z}$, we have (see e.g. [31])

$$|a_1 a_2| = |a_1||a_2| \tag{30}$$
$$a_1 a_2 = \bar{r}_1\bar{r}_2[\cos(\varphi_1+\varphi_2) + j\sin(\varphi_1+\varphi_2)] \tag{31}$$
$$a_1^p = \bar{r}_1^p[\cos(p\varphi_1) + j\sin(p\varphi_1)] \tag{32}$$

Combining (31) and (32), we get

$$(a_1a_2)^p$$
$$= (\bar{r}_1\bar{r}_2)^p[\cos(p(\varphi_1+\varphi_2)) + j\sin(p(\varphi_1+\varphi_2))] \tag{33}$$
$$= \bar{r}_1^p\bar{r}_2^p[\cos(p\varphi_1+p\varphi_2) + j\sin(p\varphi_1+p\varphi_2)] \tag{34}$$

Applying (30) gives

$$|(a_1a_2)^p| = \bar{r}_1^p\bar{r}_2^p \tag{35}$$
$$= |a_1|^p|a_2|^p \tag{36}$$
$$= |a_1a_2|^p \tag{37}$$

Since the Hadamard and Kronecker products only involve products of the (possibly complex) entries of matrices, it is clear from (35) - (37) that any result regarding Hadamard powers of Kronecker products of complex matrices also holds for the moduli of the matrices, e.g. for Theorem 2, we have

$$|(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p}| = |(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)|^{\circ p} \tag{38}$$
$$= |\mathbf{A}_1|^{\circ p} \otimes \cdots \otimes |\mathbf{A}_s|^{\circ p} \tag{39}$$

Additionally, since $(\mathbf{D}_\Omega\mathbf{A})^{\circ p}$ from Theorem 1 also only consists of products of entries of the matrices (see the proof of the theorem in Appendix A), we have the similar result

$$|(\mathbf{D}_\Omega\mathbf{A})^{\circ p}| = |\mathbf{D}_\Omega|^{\circ p}|\mathbf{A}|^{\circ p} \tag{40}$$

B. Examples for the 2D Case

Having stated our main theoretical contributions, we now present two examples of their applicability in terms of finding efficient transforms for the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ in the GAMP algorithm in Table I. For simplicity, we consider two idealised 2D imaging cases. Our first example is based on the Scanning Electron Microscopy (SEM) Compressed Sensing setup described in [4]. The Scanning Electron Microscope measures individual pixels in the image domain. In the CS setup, this image domain is undersampled uniformly at random and the image is attempted reconstructed using the Discrete Cosine Transform (DCT) as a sparsifying transform. Thus, this SEM example is described by the system matrix

$$\mathbf{A} = \mathbf{D}_\Omega\mathbf{\Psi}_{\text{i2DDCT}} \tag{41}$$

with $\mathbf{D}_\Omega$ being a sub-sampled identity matrix modelling the random selection of pixels and $\mathbf{\Psi}_{\text{i2DDCT}}$ being a 2D inverse DCT matrix, i.e. $\mathbf{\Psi}_{\text{i2DDCT}} = \mathbf{\Psi}_{\text{iDCT}} \otimes \mathbf{\Psi}_{\text{iDCT}}$ for $\mathbf{\Psi}_{\text{iDCT}}$ a 1D inverse DCT matrix. Using Theorems 1 and 2, we may express $|\mathbf{A}|^{\circ 2}$ as

$$|\mathbf{A}|^{\circ 2} = |\mathbf{D}_\Omega|^{\circ 2}|\mathbf{\Psi}_{\text{i2DDCT}}|^{\circ 2} \tag{42}$$
$$= |\mathbf{D}_\Omega|^{\circ 2}(|\mathbf{\Psi}_{\text{iDCT}}|^{\circ 2} \otimes |\mathbf{\Psi}_{\text{iDCT}}|^{\circ 2}) \tag{43}$$
$$= \mathbf{D}_\Omega(\mathbf{\Psi}_{\text{iDCT}}^{\circ 2} \otimes \mathbf{\Psi}_{\text{iDCT}}^{\circ 2}) \tag{44}$$

where in (44), we have used that $\mathbf{D}_\Omega$ in this example is a sub-sampled identity matrix and that the DCT transform is a real transform.

Our second example is inspired by the Magnetic Resonance Imaging (MRI) CS setups described in [1] and [10]. That is, we consider a setup in which an image is undersampled in the Fourier domain and then attempted reconstructed in the wavelet domain. Thus, this MRI example is described by the system matrix

$$\mathbf{A} = \mathbf{D}_\Omega\mathbf{\Phi}_{\text{2DDFT}}\mathbf{\Psi}_{\text{i2DDWT}} \tag{45}$$
$$= \mathbf{D}_\Omega((\mathbf{\Phi}_{\text{DFT}} \otimes \mathbf{\Phi}_{\text{DFT}})(\mathbf{\Psi}_{\text{iDWT}} \otimes \mathbf{\Psi}_{\text{iDWT}})) \tag{46}$$
$$= \mathbf{D}_\Omega((\mathbf{\Phi}_{\text{DFT}}\mathbf{\Psi}_{\text{iDWT}}) \otimes (\mathbf{\Phi}_{\text{DFT}}\mathbf{\Psi}_{\text{iDWT}})) \tag{47}$$

with $\mathbf{D}_\Omega$ modelling the sub-sampling pattern, $\mathbf{\Psi}_{\text{2DDFT}}$ being a 2D forward Discrete Fourier Transform (DFT) matrix, and $\mathbf{\Psi}_{\text{i2DDWT}}$ being a 2D separable inverse Discrete Wavelet Transform (DWT). Again, $\mathbf{\Phi}_{\text{DFT}}$ and $\mathbf{\Psi}_{\text{iDWT}}$ are the corresponding 1D transforms of $\mathbf{\Phi}_{\text{2DDFT}}$ and $\mathbf{\Psi}_{\text{i2DDWT}}$, respectively. Furthermore, we have used the *mixed product property* of the Kronecker product (see e.g. [20]) in going from (46) to (47). Using Theorems 1 and 4, we may express $|\mathbf{A}|^{\circ 2}$ as

$$|\mathbf{A}|^{\circ 2} = |\mathbf{D}_\Omega|^{\circ 2}|\mathbf{\Phi}_{\text{2DDFT}}\mathbf{\Psi}_{\text{i2DDWT}}|^{\circ 2} \tag{48}$$
$$= |\mathbf{D}_\Omega|^{\circ 2}(|\mathbf{\Phi}_{\text{DFT}}\mathbf{\Psi}_{\text{iDWT}}|^{\circ 2} \otimes |\mathbf{\Phi}_{\text{DFT}}\mathbf{\Psi}_{\text{iDWT}}|^{\circ 2}) \tag{49}$$

In both examples, we see that the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ may be computed using the fast and memory efficient methods for implementing a sub-sampled separable transform as outlined in Section III.

V. Sum Approximations

As an alternative to the entrywise absolute value squared transform $|\mathbf{A}|^{\circ 2}$, one may use various sum approximations in the GAMP algorithm in Table I if certain assumptions on $\mathbf{A}$ are met. We now describe these alternative sum approximations.

Krzakala et al. [7] considered the case of a homogeneous $\mathbf{A}$ with i.i.d. zero mean random entries with variance $\text{Var}(\mathbf{A}) = 1/n$ and proposed to replace the matrix-vector products involving $(|\mathbf{A}|^{\circ 2})^T$ and $|\mathbf{A}|^{\circ 2}$ in Table I with scaled sums of the vectors, i.e. use the following scalar state updates instead of lines 3 and 9, respectively

$$\bar{v}_t = \frac{1}{\text{Var}(\mathbf{A})}\sum_j [\tilde{\boldsymbol{\alpha}}_{t\text{-}1}]_j \tag{50}$$
$$\bar{s}_t = \left(\frac{1}{\text{Var}(\mathbf{A})}\sum_i [\mathbf{u}_t]_i\right)^{-1} \tag{51}$$

The scalar updates $\bar{v}_t$ and $\bar{s}_t$ are then used in place of $\mathbf{v}_t$ and $\mathbf{s}_t$, respectively, in Table I, effectively turning the entrywise vector multiplications/divisions into multiplications/divisions

TABLE III
MINIMUM MEAN SQUARED ERROR (MMSE) GENERALIZED APPROXIMATE MESSAGE PASSING (GAMP) WITH SCALAR VARIANCES [19].

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\breve{\alpha}_0 = \frac{1}{n}\sum_j \left[\text{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})\right]_j$, $\mathbf{q}_0 = \mathbf{0}_m$
for $t = 1$ **to** $T_{\max}$ **do**
  $\breve{v}_t = \frac{1}{m}||\mathbf{A}||_F^2 \breve{\alpha}_{t\text{-}1}$ — Factor side / output updates
  $\mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \breve{v}_t \mathbf{q}_{t\text{-}1}$
  $\bar{\mathbf{z}}_t = f_{\bar{z}}(\breve{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\tilde{\mathbf{z}}_t = f_{\tilde{z}}(\breve{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\mathbf{q}_t = (\bar{\mathbf{z}}_t - \mathbf{o}_t)\breve{v}_t^{-1}$
  $\breve{u}_t = \frac{1}{m}\sum_i (\breve{v}_t - [\tilde{\mathbf{z}}_t]_i)(\breve{v}_t^2)^{-1}$
  $\breve{s}_t = (\frac{1}{n}||\mathbf{A}||_F^2 \breve{u}_t)^{-1}$ — Variable side / input updates
  $\mathbf{r}_t = \bar{\boldsymbol{\alpha}}_{t\text{-}1} + \breve{s}_t \mathbf{A}^H \mathbf{q}_t$
  $\bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\breve{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  $\breve{\alpha}_t = \frac{1}{n}\sum_j [f_{\tilde{\alpha}}(\breve{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)]_j$
  if stop criterion is met **then**
    break
  end if
end for

of a scalar and a vector. We note that this sum approximation is based on i.i.d. random matrices with $\text{Var}(\mathbf{A}) = 1/n$. It may or may not work with other types of matrices or other variances. In order to use this sum approximation, one must be able to determine the variance $\text{Var}(\mathbf{A})$ or estimate it.

Rangan [19] considered the case of an $\mathbf{A}$ with approximately equal entries such that $|a_{ij}|^2 \approx \frac{||\mathbf{A}||_F^2}{mn}$ for all i, j ($||\mathbf{A}||_F$ being the Frobenius norm of $\mathbf{A}$) and then proposed to replace the vector variance states $\mathbf{v}_t$, $\mathbf{u}_t$, $\mathbf{s}_t$, and $\tilde{\boldsymbol{\alpha}}_t$ with the corresponding scalar variance states $\breve{v}_t$, $\breve{u}_t$, $\breve{s}_t$, $\breve{\alpha}_t$ to yield the MMSE GAMP with scalar variances algorithm in Table III. We note that in order to use this sum approximation one must be able to compute (or approximate) $||\mathbf{A}||_F^2$ which may be difficult in cases where it is infeasible to store $\mathbf{A}$ in memory on a computer.

Common to both Krzakala's and Rangan's sum approximations are that they significantly reduce the memory and computation cost of the GAMP algorithm. The matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ and $(|\mathbf{A}|^{\circ 2})^T$ become simple sums for which one must only store the scaling constant in memory. We have summarised these savings for the 2D case in Table II.

A. Relation to the Donoho/Maleki/Montanari AMP

The sum approximations are closely related to Donoho/Maleki/Montanari Approximate Message Passing (DMM AMP) algorithm [8], [32] in Table IV where η_t is a scalar thresholding function that operates independently on each entry of its vector argument and η_t' is the entrywise derivative of η_t with respect to its argument. Finally, $\langle\cdot\rangle$ denotes the average. As detailed in [33], when using an additive white gaussian noise (AWGN) GAMP output

TABLE IV
DONOHO/MALEKI/MONTANARI APPROXIMATE MESSAGE PASSING (DMM AMP) [8], [32].

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}_n$, $\boldsymbol{\chi}_0 = \mathbf{0}_m$
for $t = 1$ **to** $T_{\max}$ **do**
  $\bar{\boldsymbol{\alpha}}_t = \eta_t(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^H \boldsymbol{\chi}_{t\text{-}1})$
  $\boldsymbol{\chi}_t = \mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_t + \frac{n}{m}\langle \eta_t'(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^H \boldsymbol{\chi}_{t\text{-}1})\rangle \boldsymbol{\chi}_{t\text{-}1}$
  if stop criterion is met **then**
    break
  end if
end for

channel and a choice of threshold functions according to (52), (53),

$$\eta_t(\cdot) = f_{\bar{\alpha}}(s, \cdot; \boldsymbol{\theta}_I) \tag{52}$$

$$\eta_t'(\cdot) = \frac{f_{\tilde{\alpha}}(s, \cdot; \boldsymbol{\theta}_I)}{s} \tag{53}$$

the DMM AMP is equivalent to Krzakala's sum approximation MMSE GAMP under the assumption of $\text{Var}(\mathbf{A}) = 1/m$. The Bayesian DMM AMP from [34] uses this choice of threshold function. Usually, though, in the DMM AMP, the threshold function is chosen to be the soft threshold [8], [32], [34]. We note that it is the DMM AMP with soft thresholding that has been used in the previous studies of AMP for image reconstruction based on Kronecker products [12], [35].

VI. NUMERICAL EXPERIMENTS

We evaluate the reconstruction capabilities of the various (G)AMP implementations by estimating phase transitions as defined in [36]. Specifically, we empirically estimate the phase transition boundaries of the algorithms using the phase transition simulation framework from [37]. This is a well known method for assessing the performance of (G)AMP algorithms (see e.g. [7], [8], [29], [38]).

A. Phase Transitions

Consider the so-called phase space [36] defined by all $(\delta, \rho) \in [0, 1]^2$ for an undersampling ratio $\delta = m/n$ and a signal sparsity (density) $\rho = k/m$ with k being the true number of non-zero entries in a sparse signal $\boldsymbol{\alpha}$. The probability of successfully reconstructing $\boldsymbol{\alpha}$ from the noiseless undersampled measurements $\mathbf{z}$ is then evaluated on the phase space. It is less likely to have successful reconstruction of $\boldsymbol{\alpha}$ as δ decreases (fewer measurements are used) and as ρ increases (the signal is less sparse). The phase space is generally divided into two regions separated by a phase transition zone that becomes narrower as the problem size n increases. Below the phase transition zone, successful reconstruction is very likely to happen whereas the opposite is true above the phase transition zone where successful reconstruction is very unlikely to happen.

In our practical simulations based on [37], we attempt reconstructions over a 4000 point uniformly spaced grid in the phase space using

$$\delta \in \{0.025, 0.05, \ldots, 1.00\} \quad (54)$$

$$\rho \in \{0.01, 0.02, \ldots, 1.00\} \quad (55)$$

with 10 different draws of $\mathbf{A}$ and $\boldsymbol{\alpha}$ in each grid point. We then consider a reconstruction $\bar{\boldsymbol{\alpha}}$ successful if

$$\frac{||\bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}||_2^2}{||\boldsymbol{\alpha}||_2^2} < 10^{-4} \quad (56)$$

In order to estimate the 50 % successful reconstruction phase transition boundary we use a logistic regression for modelling the shape of the phase transition zone as suggested in [39]. That is, for a fixed δ, as ρ increases, we assume a change in the probability of successful reconstruction from 100 % to 0 % with a transition modelled by a logistic regression.

In all of our simulations we draw the non-zero entries of $\boldsymbol{\alpha}$ i.i.d. from a zero-mean, unit variance Gaussian distribution. We consider three different types of $\mathbf{A}$ matrix ensembles:

1) An $\mathbf{A}$ matrix drawn from the Uniform Spherical Ensemble (USE) [39], i.e. a matrix with entries drawn i.i.d. zero-mean, unit variance Gaussian followed by a normalisation to obtain unit column norms.
2) A 2D SEM inspired $\mathbf{A}$ matrix as specified in (41) with uniformly random sub-sampling.
3) A 2D MRI inspired $\mathbf{A}$ matrix as specified in (45). For this setup we used random sub-sampling under the constraint of forcing hermitian symmetry such that the image formed by applying the inverse DFT remained real[4]. Furthermore, we used single level 2D separable Haar wavelets as the sparsifying basis.

Finally, in order to allow for size 32-by-32 2D cases, we used a problem size of $n = 32^2 = 1024$ which is along the lines of the experiments in [29] where they used $n = 1000$.

B. Experimental Setup

In our empirical phase transition experiments, we evaluated the performance of five different (G)AMP algorithms:

1) DMM AMP (Table IV) using soft thresholding for the threshold operator η_t.
2) MMSE GAMP (Table I) using the full $|\mathbf{A}|^{\circ 2}$ transform.
3) MMSE GAMP (Table I) using Krzakala's sum approximation (SA) updates in (50) and (51) with $\text{Var}(\mathbf{A}) = 1/m$.
4) MMSE GAMP (Table I) using Krzakala's sum approximation updates in (50) and (51) with $\text{Var}(\mathbf{A}) = 1/n$.
5) MMSE GAMP with scalar variances (Table III) based on Rangan's sum approximation.

For the DMM AMP algorithm, we tested both the median (denoted M in Figures 1 - 3) and residual (denoted R in Figures 1 - 3) based threshold levels from [32]. Specifically, we used an iteration dependent soft threshold level of $\theta\hat{\tau}_t$ with

$$\hat{\tau}_t = \frac{1}{\Phi_{\mathcal{N}}^{-1}(0.75)} \cdot \text{mdn}(\boldsymbol{\chi}_t) \quad \text{(median based)} \quad (57)$$

$$\hat{\tau}_t = \sqrt{\frac{1}{m}||\boldsymbol{\chi}_t||_2^2} \quad \text{(residual based)} \quad (58)$$

where $\Phi_{\mathcal{N}}^{-1}$ is the inverse cumulative distribution function of a zero-mean, unit variance Gaussian random variable, $\text{mdn}(\cdot)$ denotes the median, and θ is a tuning parameter that we chose to the minimax optimal value as detailed in [8]. We initialised $\hat{\tau} = 1$.

For the GAMP algorithms we used an AWGN output channel and an i.i.d. sparse Bernoulli-Gaussian input channel, i.e. we assumed channels described by [29]

$$p(y|z; \boldsymbol{\theta}_o) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-z)^2}{2\sigma^2}\right) \quad (59)$$

$$p(\alpha; \boldsymbol{\theta}_I) = (1-\tau)\delta_{\text{Dirac}}(\alpha) + \tau \frac{1}{\sqrt{2\pi\tilde{\theta}}} \exp\left(-\frac{(\alpha-\bar{\theta})^2}{2\tilde{\theta}}\right) \quad (60)$$

where σ^2 is the output channel noise variance parameter such that $\boldsymbol{\theta}_o = [\sigma^2]$, τ is the signal density, $\bar{\theta}$ is the Gaussian mean, and $\tilde{\theta}$ is the Gaussian variance such that $\boldsymbol{\theta}_I = [\tau, \bar{\theta}, \tilde{\theta}]^T$. Closed form expressions for the channels functions $f_{\bar{z}}$, $f_{\tilde{z}}$ based on (59) and $f_{\bar{\alpha}}$, $f_{\tilde{\alpha}}$ based on (60) may be found in [33] and [29], respectively. We tested both a genie version of the GAMP algorithms that knew the true input channel parameters $\boldsymbol{\theta}_I = [\frac{k}{n}, 0, 1]^T$ as well as a version that used Expectation Maximization (EM) for learning the input parameters as detailed in [29]. For the EM initialisation of the input channel parameters we used the suggestion given in [29] with $\bar{\theta} = 0$ and an "assumed" SNR of 100. In all GAMP algorithms, we used EM for learning the output noise variance σ^2 according to the description in [7]. For the input genie versions we initialised $\sigma^2 = 1$ whereas in input EM versions, we used the suggested initialisation of σ^2 from [29] with an "assumed" SNR 100. In all EM cases we did a single EM update as part of the channel function evaluation following the actual channel function evaluation.

In all (G)AMP algorithms we used the normalised mean squared error (NMSE) stop criterion suggested in [29], i.e.

$$\frac{||\boldsymbol{\alpha}_{t\text{-}1} - \boldsymbol{\alpha}_t||_2^2}{||\boldsymbol{\alpha}_{t\text{-}1}||_2^2} < \epsilon \quad (61)$$

with a tolerance $\epsilon = 10^{-6}$. Also common to all tested algorithms was a maximum number of iterations of $T_{\max} = 500$. Finally, all computations where done using double precision floats.

In terms of implementing the $|\mathbf{A}|^{\circ 2}$ transform in GAMP, we used the straight up computation of $|\mathbf{A}|^{\circ 2}$ for the USE $\mathbf{A}$ matrix. For the 2D SEM $\mathbf{A}$ matrix, we used (44) together with the separability property in (22) - (24) in implementing the transforms involving $|\mathbf{A}|^{\circ 2}$. Similarly, for the 2D MRI $\mathbf{A}$ matrix, we used (49) in implementing the transforms involving $|\mathbf{A}|^{\circ 2}$. For both the 2D SEM and 2D MRI cases, we used fast transforms in implementing $\mathbf{A}$, e.g. an FFT for the Fourier transform in the 2D MRI case.

We used the implementations of the (G)AMP algorithms available in release 1.6.0[5] of the open source `magni` Python Package described in [40]. The `magni` package focuses on

[4]See the simulation script available at https://dx.doi.org/10.5281/zenodo.165051 for the details.

[5]All releases of `magni` are available at https://dx.doi.org/10.5278/VBN/MISC/Magni

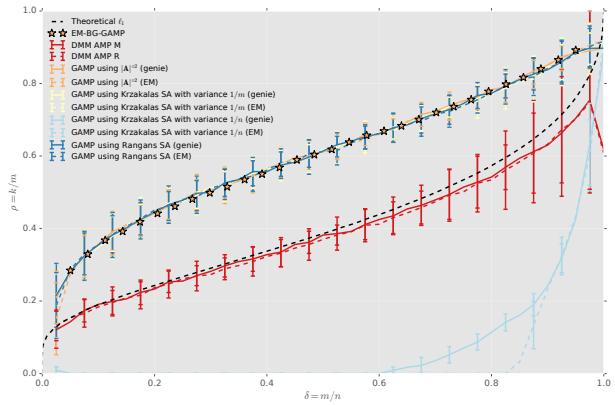


Fig. 1. Empirical phase transition boundaries for the USE system matrix. The theoretical ℓ_1 phase transition from [36] is included for reference. The curves show the 50 % transition boundary of the fitted logistic regression. The error bars mark the 10 % and 90 % transition levels of the fitted logistic regression as an indicator of the phase transition zone width. The EM-BG-GAMP markers are the corresponding results from [29], [44][12].

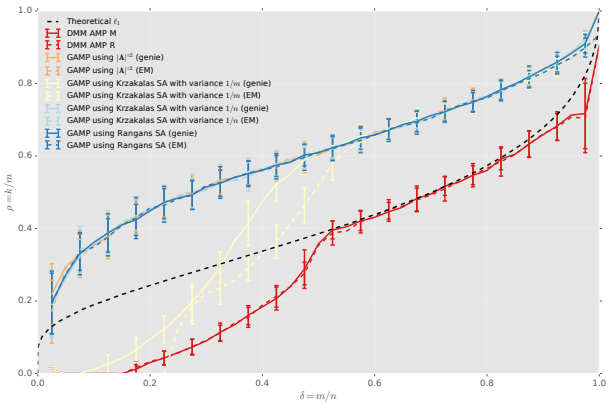


Fig. 2. Empirical phase transition boundaries for the SEM inspired system matrix. The theoretical ℓ_1 phase transition from [36] is included for reference. The curves show the 50 % transition boundary of the fitted logistic regression. The error bars mark the 10 % and 90 % transition levels of the fitted logistic regression as an indicator of the phase transition zone width.

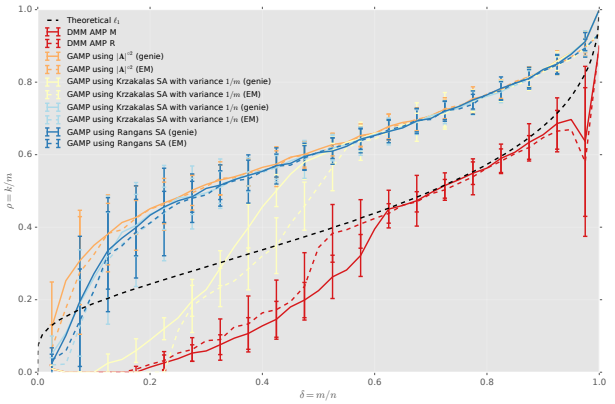


Fig. 3. Empirical phase transition boundaries for the MRI inspired system matrix. The theoretical ℓ_1 phase transition from [36] is included for reference. The curves show the 50 % transition boundary of the fitted logistic regression. The error bars mark the 10 % and 90 % transition levels of the fitted logistic regression as an indicator of the phase transition zone width.

correctness of results and as such is fully documented, comes with an extensive test suite, and an input argument validation framework [41]. It has been developed according to best practices in scientific software development [42] and provides tools for aiding in making results from computational experiments reproducible [43]. All of these features of `magni` were used to ensure the correctness as well as the reproducibility of our numerical experiments. We used the Anaconda[6] Python distribution version 4.1.1 based on Python 3.5 for running the simulations[7] in combination with the Haar wavelet implementation in the PyWavelets[8] library version 0.4.0. The Python scripts used to run the simulations and the full set of simulation results may be obtained from Zenodo[9]. A Jupyter Notebook that reproduces the figures in this paper may be obtained from the authors' institutional repository[10]. The simulations were conducted on a compute cluster consisting of five nodes each featuring two Intel Xeon E5-2697V2 CPUs and 384 GiB RAM and running Ubuntu 14.04.3 LTS.

C. Results

Our empirically determined 50 % phase transition boundaries are depicted in Figures 1 (USE), 2 (SEM), and 3 (MRI). Each of the figures display the full phase space and the placement of the phase transitions for all tested algorithms along with the theoretical ℓ_1 phase transition from [36]. Error bars are used in the figures to display the 10 % and 90 % transition levels which give an indication of the phase transition zone width.

[6]See http://www.continuum.io/anaconda

[7]Refer to the annotations of the results databases for a detailed list of scientific Python packages used in the simualtions

[8]See http://pywavelets.readthedocs.io/en/latest/

[9]See https://dx.doi.org/10.5281/zenodo.165051

[10]See https://dx.doi.org/10.5278/240710282

[12]The EM-BG-GAMP results were replicated from [29], [44] using the material from https://doi.org/10.5281/zenodo.160700

VII. Discussion

The USE matrix ensemble results depicted in Figure 1 are mainly included as a reference. The USE matrix ensemble is based on the i.i.d. Gaussian matrices (with variance $1/m$) that are the basis for the derivation of the (G)AMP algorithms. As such, our results for the USE matrix ensemble obtained using the `magni` Python (G)AMP implementations are as expected. The DMM AMP results closely follow the theoretical ℓ_1 curve as is expected for this minimax tuned algorithm. The GAMP results closely follow the corresponding simulation results reported in [29]. We do note, though, that the results for GAMP with Krzakala's sum approximation using a (mistuned) variance of $1/n$ performs significantly worse than its $1/m$ counterpart. This result clearly indicates the importance of tuning the variance parameter in Krzakala's sum approximation. Apart from this lack of robustness of Krzakala's sum approximation towards changes in the variance parameter, all

sum approximation methods seem to perform as well as the GAMP algorithm using the full $|\mathbf{A}|^{\circ 2}$ transform for the USE matrix ensemble.

For the SEM results in Figure 2, we again see the lack of robustness towards changes in the variance parameter for not only the GAMP with Krzakala's sum approximation but also for the DMM AMP. In particular, we see that for $\delta \lesssim 0.5$, the performance of Krzakala's sum approximation based on a variance of $1/m$ and DMM AMP (which also assumes a variance of $1/m$ as detailed in Section V-A) degrades. The SEM ensemble is not based on a matrix with i.i.d. random entries, though we still use random sampling such that the variance of the entries is on the order of $1/n$. For large values of δ, we have $m \approx n$ and still reasonable performance of all methods. However, for $\delta \lesssim 0.5$ (which is the interesting case in undersampling applications), the choice of algorithm becomes critical.

The MRI matrix ensemble is also based on random sampling that results in entries with a variance on the order of $1/n$. Thus, the same remarks regarding the robustness towards the choice of variance parameter as in the SEM matrix ensemble holds for the MRI ensemble as seen in Figure 3. More interestingly, though, is that for this matrix ensemble, we also see a degradation in performance for both GAMP with Krzakala's sum approximation with variance $1/n$ and the scalar variance GAMP based on Rangan's sum approximation when $\delta \lesssim 0.2$. In this particular setting, the GAMP algorithms based on the application of the full transforms involving $|\mathbf{A}|^{\circ 2}$ perform significantly better than their sum approximation counterparts. Additional simulations using problem sizes of 64^2 and 128^2 and reported on in the supplementary material[13] confirm that this difference is also present for larger problem sizes, though the large problem size generally also helps in stabilising the transition zone and moving the 50 % boundary slightly upward in the phase space.

Based on all of our simulations, we note that the various sum approximations generally work well if one is able to correctly determine or approximate the scaling factors, i.e. the variance scaling factor for Krzakala's sum approximation and the Frobenius norm of $\mathbf{A}$ for Rangan's sum approximation. In such cases significant memory and computation cost reductions may be obtained as exemplified for the 2D case in Table II. However, our simulation results also show that one may have to use the GAMP algorithm based on the application of the full transforms involving $|\mathbf{A}|^{\circ 2}$ to get the best performance in terms of reconstruction capabilities. In this case, our proposed methods for implementing $|\mathbf{A}|^{\circ 2}$ for high dimensional sub-sampled separable transforms yield significant memory and computation cost reductions over explicitly forming $|\mathbf{A}|^{\circ 2}$ and computing the matrix-vector products directly.

VIII. Conclusions

We have derived expressions to efficiently implement Hadamard Powers of sub-sampled Kronecker products. A particularly interesting application of these expressions is in implementing fast and memory efficient transforms involving the entrywise absolute value squared system matrix used in the GAMP algorithm. Such efficient transforms may be needed in high dimensional signal undersampling applications where the problem size makes it infeasible to store the full transform matrices in memory on a computer. We have detailed the practical implications of using our proposed transforms compared to the alternative sum approximations. In particular we have discussed the quantities that one must be able to correctly determine or estimate in order to use the sum approximations. Furthermore, we have detailed a large set of phase transition simulations of (G)AMP algorithms. Our simulations allow for a comparison of our proposed efficient full transform implementations to the alternative sum approximations. Though the sum approximations generally perform very well, we find that there are cases such as our MRI inspired example where the best performance is only achievable using the full transforms.

[13]The supplementary material is available at the authors' institutional repository at https://dx.doi.org/10.5278/240710282

Appendix A
Proofs of Hadamard Powers Relations

Here we give proofs of the theorems presented in Section IV-A.

Proof of Theorem 1: Let $(\mathbf{D}_\Omega\mathbf{A})_i^{\circ p}$ and $\mathbf{A}_i$ denote the i-th rows of the matrices $(\mathbf{D}_\Omega\mathbf{A})^{\circ p}$ and $\mathbf{A}$, respectively. Furthermore, let a_{ij} denote the ij-th entry of $\mathbf{A}$. Then,

$$(\mathbf{D}_\Omega\mathbf{A})_i^{\circ p} = (d_i\mathbf{A}_i)^{\circ p} \tag{A.62}$$
$$= \left[(d_i a_{i1})^p \dots (d_i a_{il})^p\right] \tag{A.63}$$
$$= \left[d_i^p a_{i1}^p \dots d_i^p a_{il}^p\right] \tag{A.64}$$
$$= d_i^p \mathbf{A}_i^{\circ p} \tag{A.65}$$

for any $i \in 1 \dots m$. Therefore, $(\mathbf{D}_\Omega\mathbf{A})^{\circ p} = \mathbf{D}_\Omega^{\circ p}\mathbf{A}^{\circ p}$. ∎

Proof of Theorem 2: Consider matrices $\mathbf{A} \in \mathbb{C}^{m_1\times n_1}$, $\mathbf{B} \in \mathbb{C}^{m_2\times n_2}$. Let b_{ij} denote the ij-th entry of $\mathbf{B}$ and note that for any $\beta \in \mathbb{C}$ and any $p \in \mathbb{Z}$,

$$(\beta\mathbf{B})^{\circ p} = \begin{bmatrix} (\beta b_{11})^p & \cdots & (\beta b_{1n_2})^p \\ \vdots & \ddots & \vdots \\ (\beta b_{m_2 1})^p & \cdots & (\beta b_{m_2 n_2})^p \end{bmatrix} \tag{A.66}$$
$$= \begin{bmatrix} \beta^p b_{11}^p & \cdots & \beta^p b_{1n_2}^p \\ \vdots & \ddots & \vdots \\ \beta^p b_{m_2 1}^p & \cdots & \beta^p b_{m_2 n_2}^p \end{bmatrix} \tag{A.67}$$

It then follows that

$$(\mathbf{A}\otimes\mathbf{B})^{\circ p} = \begin{bmatrix} (a_{11}\mathbf{B})^{\circ p} & \cdots & (a_{1n_1}\mathbf{B})^{\circ p} \\ \vdots & \ddots & \vdots \\ (a_{m_1 1}\mathbf{B})^{\circ p} & \cdots & (a_{m_1 n_1}\mathbf{B})^{\circ p} \end{bmatrix} \tag{A.68}$$
$$= \begin{bmatrix} a_{11}^p\mathbf{B}^{\circ p} & \cdots & a_{1n_1}^p\mathbf{B}^{\circ p} \\ \vdots & \ddots & \vdots \\ a_{m_1 1}^p\mathbf{B}^{\circ p} & \cdots & a_{m_1 n_1}^p\mathbf{B}^{\circ p} \end{bmatrix} \tag{A.69}$$
$$= \mathbf{A}^{\circ p}\otimes\mathbf{B}^{\circ p} \tag{A.70}$$

Now consider the recursive definition of $\mathbf{A}_1^{\circ p}\otimes\cdots\otimes\mathbf{A}_s^{\circ p}$,

$$(\mathbf{A}_s^{'})^{\circ p} = (\mathbf{A}_{s-1}^{'})^{\circ p} \otimes \mathbf{A}_s^{\circ p}, \quad s \in \{2,3,4,\dots\} \tag{A.71}$$
$$(\mathbf{A}_1^{'})^{\circ p} = \mathbf{A}_1^{\circ p} \tag{A.72}$$

Then by induction, we have

$$(\mathbf{A}_1^{'})^{\circ p} = \mathbf{A}_1^{\circ p} = (\mathbf{A}_1)^{\circ p} \tag{A.73}$$
$$(\mathbf{A}_{s+1}^{'})^{\circ p} = (\mathbf{A}_s^{'})^{\circ p} \otimes \mathbf{A}_{s+1}^{\circ p} \tag{A.74}$$
$$= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p} \otimes \mathbf{A}_{s+1}^{\circ p} \tag{A.75}$$
$$= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s \otimes \mathbf{A}_{s+1})^{\circ p} \tag{A.76}$$

Therefore, $(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_s)^{\circ p} = \mathbf{A}_1^{\circ p} \otimes \cdots \otimes \mathbf{A}_s^{\circ p}$. ■

Proof of Theorem 3: Consider the recursive definition of $(\mathbf{A}_1\mathbf{B}_1)^{\circ p} \otimes \cdots \otimes (\mathbf{A}_s\mathbf{B}_s)^{\circ p}$,

$$(\mathbf{A}_s\mathbf{B}_s)^{'\circ p} = (\mathbf{A}_{s-1}\mathbf{B}_{s-1})^{'\circ p} \otimes (\mathbf{A}_s\mathbf{B}_s)^{\circ p}$$
$$s \in \{2,3,4,\dots\} \tag{A.77}$$
$$(\mathbf{A}_1\mathbf{B}_1)^{'\circ p} = (\mathbf{A}_1\mathbf{B}_1)^{\circ p} \tag{A.78}$$

Then by induction, we have

$$(\mathbf{A}_1\mathbf{B}_1)^{'\circ p} = (\mathbf{A}_1\mathbf{B}_1)^{\circ p} = (\mathbf{E}_1\mathbf{F}_1)^{\circ p} \tag{A.79}$$
$$(\mathbf{A}_{s+1}\mathbf{B}_{s+1})^{'\circ p} = (\mathbf{A}_s\mathbf{B}_s)^{'\circ p} \otimes (\mathbf{A}_{s+1}\mathbf{B}_{s+1})^{\circ p} \tag{A.80}$$
$$= (\mathbf{E}_s\mathbf{F}_s)^{\circ p} \otimes (\mathbf{A}_{s+1}\mathbf{B}_{s+1})^{\circ p} \tag{A.81}$$
$$= ((\mathbf{E}_s\mathbf{F}_s) \otimes (\mathbf{A}_{s+1}\mathbf{B}_{s+1}))^{\circ p} \tag{A.82}$$
$$= ((\mathbf{E}_s \otimes \mathbf{A}_{s+1})(\mathbf{F}_s \otimes \mathbf{A}_{s+1}))^{\circ p} \tag{A.83}$$
$$= (\mathbf{E}_{s+1}\mathbf{F}_{s+1})^{\circ p} \tag{A.84}$$

where we have used Theorem 2 in (A.82) and the *mixed-product property* of the Kronecker product [20] in (A.83). Therefore, $(\mathbf{E}_s\mathbf{F}_s)^{\circ p} = (\mathbf{A}_1\mathbf{B}_1)^{\circ p} \otimes \cdots \otimes (\mathbf{A}_s\mathbf{B}_s)^{\circ p}$. ■

Proof of Theorem 4: Using the *mixed-product property* of the Kronecker product [20], we get

$$(\mathbf{E}_1 \dots \mathbf{E}_s)^{\circ p}$$
$$= ((\mathbf{A}_1 \otimes \mathbf{B}_1) \dots (\mathbf{A}_s \otimes \mathbf{B}_s))^{\circ p} \tag{A.85}$$
$$= (((\mathbf{A}_1\mathbf{A}_2) \otimes (\mathbf{B}_1\mathbf{B}_2)) \dots (\mathbf{A}_s \otimes \mathbf{B}_s))^{\circ p} \tag{A.86}$$
$$= ((\mathbf{A}_1 \dots \mathbf{A}_s) \otimes (\mathbf{B}_1 \dots \mathbf{B}_s))^{\circ p} \tag{A.87}$$
$$= (\mathbf{A}_1 \dots \mathbf{A}_s)^{\circ p} \otimes (\mathbf{B}_1 \dots \mathbf{B}_s)^{\circ p} \tag{A.88}$$

where we have used Theorem 2 in (A.88). Therefore, $(\mathbf{E}_1 \dots \mathbf{E}_s)^{\circ p} = (\mathbf{A}_1 \dots \mathbf{A}_s)^{\circ p} \otimes (\mathbf{B}_1 \dots \mathbf{B}_s)^{\circ p}$. ■

REFERENCES

[1] Michael Lustig, David Donoho, and John M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, Dec. 2007.

[2] Jin Tan, Yanting Ma, Hoover Rueda, Dror Baron, and Gonzalo R. Arce, "Compressive hyperspectral imaging via approximate message passing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 389–401, Mar. 2016.

[3] Thomas Arildsen, Christian Schou Oxvig, Patrick Steffen Pedersen, Jan Østergaard, and Torben Larsen, "Reconstruction Algorithms in Undersampled AFM Imaging," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 1, pp. 31–46, Feb. 2016.

[4] Hyrum S. Anderson, Jovana Ilic-Helms, Brandon Rohrer, Jason Wheeler, and Kurt Larson, "Sparse imaging for fast electron microscopy," in *Computational Imaging XI, Proceedings of the SPIE*, Burlingame, California, USA, Feb. 3 2013, vol. 8657, pp. (86570C–1)–(86570C–12).

[5] Sean B. Andersson and Lucy Y. Pao, "Non-Raster Sampling in Atomic Force Microscopy: A Compressed Sensing Approach," in *American Control Conference (ACC)*, Montréal, Canada, June 27-29, 2012, pp. 2485–2490.

[6] Daniel J. Müller and Yves F. Dufrêne, "Atomic force microscopy: a nanoscopic window on the cell surface," *Trends in Cell Biology*, vol. 21, no. 8, pp. 461–469, Aug. 2011.

[7] Florent Krzakala, Marc Mézard, Francois Sausset, Yifan Sun, and Lenka Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices," *Journal of Statistical Mechanics: Theory and Experiment*, vol. P08009, pp. 1–57, Aug. 2012.

[8] David L. Donoho, Arian Maleki, and Andrea Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, pp. 18914–18919, Nov. 2009.

[9] Sundeep Rangan, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, July 31 – Aug. 5, 2011, pp. 2168–2172.

[10] Kyunghyun Sung, Bruce L. Daniel, and Brian A. Hargreaves, "Location Constrained Approximate Message Passing for Compressed Sensing MRI," *Magnetic Resonance in Medicine*, vol. 70, no. 2, pp. 370–381, Aug. 2013.

[11] Subhojit Som and Philip Schniter, "Compressive Imaging Using Approximate Message Passing and a Markov-Tree Prior," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3439–3448, July 2012.

[12] Akira Hirabayashi, Jumpei Sugimoto, and Kazushi Mimura, "Approximate Message Passing Algorithm for Complex Separable Compressed Imaging," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Kaohsiung, Taiwan, Oct. 29 – Nov. 1, 2013, pp. 1–5.

[13] Jun Fang, Lizao Zhang, and Hongbin Li, "Two-Dimensional Pattern-Coupled Sparse Bayesian Learning via Generalized Approximate Message Passing," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2920–2930, June 2016.

[14] Jin Tan, Yanting Ma, and Dror Baron, "Compressive Imaging via Approximate Message Passing With Image Denoising," *IEEE Transactions on Signal Processing*, vol. 63, no. 8, pp. 2085–2092, Apr. 2015.

[15] Philip Schniter and Sundeep Rangan, "Compressive Phase Retrieval via Generalized Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1043–1055, Feb. 2015.

[16] Emmanuel J. Candès and Michael B. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[17] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, July 2007.

[18] David L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[19] Sundeep Rangan, "Generalized approximate message passing for estimation with random linear mixing," arXiv:1010.5141v2, Aug. 2012.

[20] Roger A. Horn and Charles R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1st paperback edition, 1994.

[21] Thong T. Do, Lu Gan, Nam H. Nguyen, and Trac D. Tran, "Fast and efficient compressive sensing using structurally random matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 139–154, Jan. 2012.

[22] Holger Rauhut, "Compressive Sensing and Structured Random Matrices," in *Theoretical Foundations and Numerical Methods for Sparse Recovery*, Massimo Fornasier, Ed., vol. 9 of *Radon Series on Computational and Applied Mathematics*, pp. 1–92. De Gruyter, 2010.

[23] Mark Borgerding, Philip Schniter, Jeremy Vila, and Sundeep Rangan, "Generalized Approximate Message Passing for Cosparse Analysis Compressive Sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, Apr. 19 – 24, 2015, pp. 3756–3760.

[24] Marco F. Duarte and Richard G. Baraniuk, "Kronecker Compressive Sensing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, Feb. 2012.

[25] Ali N. Akansu, Richard A. Haddad, and Paul R. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, Academic Press, 2nd edition, 2000.

[26] Mohsen Bayati, Marc Lelarge, and Andrea Montanari, "Universality in Polytope Phase Transitions and Iterative Algorithms," in *IEEE International Symposium on Information Theory (ISIT)*, Cambridge, Massachusetts, USA, July 1 – 6, 2012, pp. 1643–1647.

[27] Mohsen Bayati and Andrea Montanari, "The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.
[28] Sundeep Rangan, Philip Schniter, and Alyson Fletcher, "On the Convergence of Approximate Message Passing with Arbitrary Matrices," in *IEEE International Symposium on Information Theory (ISIT)*, Honolulu, Hawaii, USA, June 29 – July 4, 2014, pp. 236–240.
[29] Jeremy P. Vila and Philip Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.
[30] Jean Barbier, Eric W. Tramel, and Florent Krzakala, "Scampi: a robust approximate message-passing framework for compressive imaging," in *International Meeting on High-Dimensional Data-Driven Science*, Kyoto, Japan, Dec. 14 – 17, 2015, vol. 699 of *Journal of Physics: Conference Series*, pp. 1–13.
[31] Erwin Kreyszig, *Advanced Engineering Mathematics*, Wiley, 9th edition, 2006.
[32] Andrea Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing: Theory and Applications*, Yonina C. Eldar and Gitta Kutyniok, Eds., chapter 9, pp. 394–438. Cambridge University Press, 2012, ISBN: 9781107005587.
[33] Jason T. Parker, *Approximate Message Passing Algorithms for Generalized Bilinear Inference*, Ph.D. thesis, Graduate School of The Ohio State University, 2014.
[34] David L. Donoho, Arian Maleki, and Andrea Montanari, "Message Passing Algorithms for Compressed Sensing: I. Motivation and Construction," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5.
[35] Akia Hirabayashi, Jumpei Sugimoto, and Kazushi Mimura, "Complex Approximate Message Passing Algorithm for Two-Dimensional Compressed Sensing," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E96-A, no. 12, pp. 2391–2397, Dec. 2013.
[36] David L. Donoho and Jared Tanner, "Precise Undersampling Theorems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 913–924, June 2010.
[37] Christian Schou Oxvig, Patrick Steffen Pedersen, Thomas Arildsen, and Torben Larsen, "Surpassing the Theoretical 1-norm Phase Transition in Compressive Sensing by Tuning the Smoothed l0 algorithm," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 26 – 31, 2013, pp. 6019–6023.
[38] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová, "Statistical-Physics-Based Reconstruction in Compressed Sensing," *Physical Review X*, vol. 2, no. 2, pp. (021005–1)–(021005–18), May 2012.
[39] Arian Maleki and David L. Donoho, "Optimally Tuned Iterative Reconstruction Algorithms for Compressed Sensing," *IEEE Journal Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 330–341, Apr. 2010.
[40] Christian Schou Oxvig, Patrick Steffen Pedersen, Thomas Arildsen, Jan Østergaard, and Torben Larsen, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images," *Journal of Open Research Software*, vol. 2, no. 1, pp. e29, Oct. 2014.
[41] Patrick Steffen Pedersen, Christian Schou Oxvig, Jan Østergaard, and Torben Larsen, "Validating Function Arguments in Python Signal Processing Applications," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, July 11 – 17, 2016, pp. 106–113.
[42] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson, "Best Practices for Scientific Computing," *PloS Biology*, vol. 12, no. 1, pp. e1001745, Jan. 2014.
[43] Christian Schou Oxvig, Thomas Arildsen, and Torben Larsen, "Storing Reproducible Results from Computational Experiments using Scientific Python Packages," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, July 11 – 17, 2016, pp. 45–50.
[44] Jeremy Vila and Philip Schniter, "Expectation-Maximization Bernoulli-Gaussian Approximate Message Passing," in *Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 6 – 9 2011, pp. 799–803.

Christian Schou Oxvig (S'14) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Aalborg University in 2011 and 2013, respectively. He is currently a Ph.D. student in the Signal and Information Processing Section, Department of Electronic Systems, Aalborg University. His research interests include scientific computing, statistical signal processing, and reproducibility in computer and simulation experiments.

Thomas Arildsen (S'03–M'04–S'06–M'10) received the M.Sc.E.E. and Ph.D. degrees from Aalborg University, Aalborg, Denmark, in 2004 and 2010, respectively. From 2004 to 2006, he worked as project engineer at RTX Telecom A/S in Aalborg, Denmark. He has been a visiting Ph.D. student at University of Miami, United States in 2007. He has worked at Aalborg University since 2010, as associate professor since 2014. His research interests include compressed sensing, image processing, analog-to-information conversion, scientific computing, statistical signal processing, estimation, communication signal processing.

Torben Larsen (S'88–M'99–SM'04) received the M.Sc.E.E. and Dr.Techn. degrees from Aalborg University, Aalborg, Denmark, in 1988 and 1998, respectively. Since 2001, he has been a Full Professor at Aalborg University in electronic circuits, signals and systems theory. He has industrial experience working as senior engineer at Bosch Telecom and Siemens Mobile Phones. He was member in 2005-2010 and vice-chairman in 2009-2010 of the Danish Research Council for Technology and Production Sciences. In 2011 he was appointed director of the doctoral school at The Faculty of Engineering and Science, Aalborg University, with more than 650 enrolled PhD students. He has authored or co-authored over 130 peer-reviewed journal and conference papers and contributed to four internationally published books. He received the Spar Nord Research Prize in 1999 and "Aalborg University Teacher of the year 2013". Since 2007 he has been member of the Academy of Technical Sciences, Denmark. He has supervised approx. 25 PhD students. He has been Vice Dean at the Faculty of Engineering and Science, Aalborg University since 2015. His recent research interests mainly include scientific computing, compressive sensing, numerical algorithms etc. in the areas of signals and systems theory.

Paper D

Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images

Christian Schou Oxvig, Patrick Steffen Pedersen, Thomas Arildsen,
Jan Østergaard, Torben Larsen

The paper has been published in
Journal of Open Research Software, vol. 2, no. 1, pp. e29, Oct. 2014.
doi:10.5334/jors.bk

The succeeding 6 non-blank pages constitute the published manuscript which is subject to the following copyright notice:

Journal of
open research software

Oxvig, C S, Pedersen , P S, Arildsen, T, Østergaard, J and Larsen, T 2014 Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images. *Journal of Open Research Software,* 2: e29, DOI: http://dx.doi.org/10.5334/jors.bk

SOFTWARE METAPAPER

Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images

Christian Schou Oxvig[1], Patrick Steffen Pedersen[1], Thomas Arildsen[1], Jan Østergaard[1] and Torben Larsen[1]

[1] Signal and Information Processing Section, Department of Electronic Systems, Faculty of Engineering of Science, Aalborg University, Aalborg, Denmark

Magni is an open source Python package that embraces compressed sensing and Atomic Force Microscopy (AFM) imaging techniques. It provides AFM-specific functionality for undersampling and reconstructing images from AFM equipment and thereby accelerating the acquisition of AFM images. Magni also provides researchers in compressed sensing with a selection of algorithms for reconstructing undersampled general images, and offers a consistent and rigorous way to efficiently evaluate the researchers own developed reconstruction algorithms in terms of phase transitions. The package also serves as a convenient platform for researchers in compressed sensing aiming at obtaining a high degree of reproducibility of their research.

Keywords: Atomic Force Microscopy; Compressive Sensing; Python; Image Reconstruction; Reproducible Research

Funding Statement: TThis project has been supported by 1) The Danish Council for Independent Research (DFF) via funding from DFF/Technology and Production (FTP), grant DFF–1335–00278, for the project "Enabling Fast Image Acquisition for Atomic Force Microscopy using Compressed Sensing", and 2) by the Danish e-Infrastructure Cooperation (DeIC) via a grant for a high performance computing system for the project "High Performance Computing SMP Server for Signal Processing".

(1) Overview

Introduction

In our research group at Aalborg University (AAU) we have recently launched a new research project, FastAFM[1], seeking to utilise compressed sensing in accelerating the acquisition of atomic force microscopy images. This is a relatively unexplored application area where results have only just started to appear [1], [2]. With the present paper, we present the general software package `magni`, which we have developed to combine compressed sensing and AFM imaging techniques.

Compressed sensing is a theory which has attracted a great deal of attention recently. In brief, the theory states that a wide range of possible signal types can be accurately represented from a greatly reduced number of acquired samples [3], [4]. That is, these signal types can be accurately reconstructed from samples taken significantly below the Shannon-Nyquist rate which is normally seen as the ultimate limit.

Atomic Force Microscopy (AFM) is one of the most advanced tools for high-resolution imaging and manipulation of nanoscale matter [5]. When used for imaging, it is able to generate a 3D surface map with sub nanometer resolution of an object [6]. To generate this map, a sharp probe is brought close to the surface of the object, and the probe tip and the object are then moved relative to each other. The mechanical probe tip is affected by the force on the surface and, loosely speaking, "feels" the surface [6], [7]. Unfortunately, standard AFM imaging requires a timescale on the order of minutes to hours to acquire an image [7].

In the course of our work with compressed sensing and AFM, we have identified three shortcomings. We find that these are not adequately met by available free and open source research software in this area:

1. Software for reconstruction of compressed sensing signals.
2. Software for consistent and rigorous testing of reconstruction algorithms, particularly of their reconstruction capabilities in terms of phase transition.
3. Software for acquisition and processing of AFM images in relation to compressed sensing.

While free and open source software for compressed sensing signal reconstruction is available as such [8], [9], most of this software relies on Matlab from MathWorks

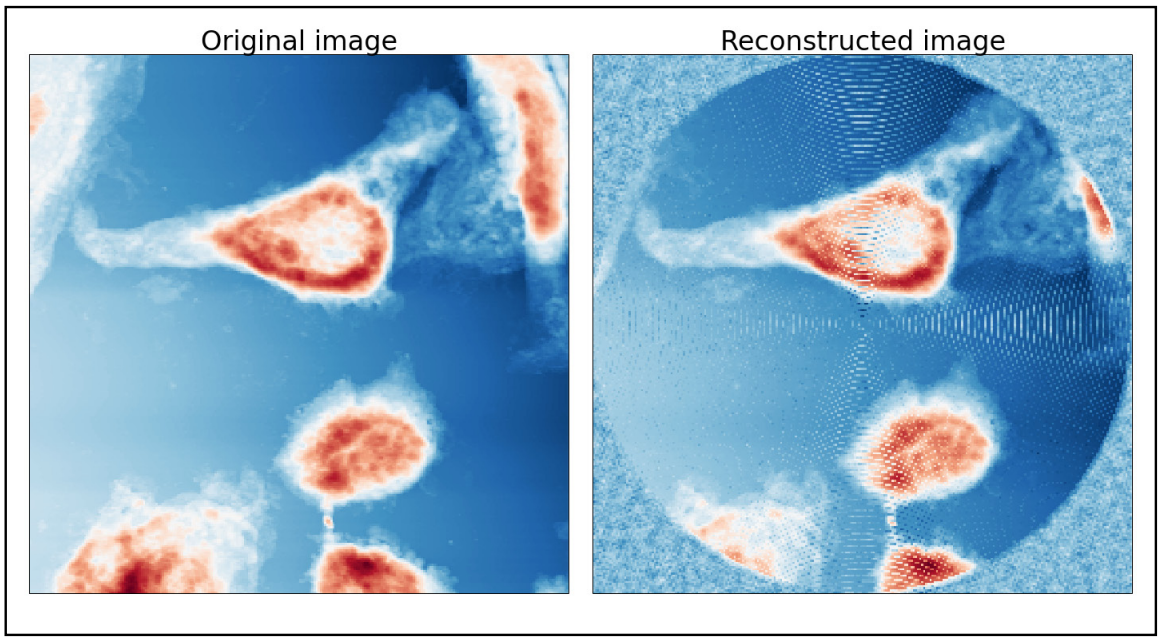


Figure 1: An example compressive sensing reconstruction of an AFM image.

which limits the reproducibility. Also, the available free and open source software for AFM image post-processing and visualisation [10] does not include compressed sensing related functionality. Instead, to mitigate these shortcomings, we have built the `magni` software package to ensure the highest degree of reproducibility defined for signal processing [11]. This has been done by relying on the free and open source programming language Python[2] and by making all examples, figures, etc. easily reproducible. An example of the reconstruction of an AFM image is shown in **Figure 1**. Using `magni`, the original image was loaded, preprocessed, sampled, reconstructed and displayed in less than 25 lines with intuitive calls to `magni` such as:

```
f>>> magni.imaging.measurements.spiral_sample_
image(h, w, scan_length, num_points)
>>> magni.imaging.measurements.construct_meas
urement_matrix(img_coords, h, w)
>>> magni.imaging.dictionaries.get_DCT((h, w))
>>> magni.afm.reconstruction.reconstruct(domain.
measurements, Phi, Psi)
```

We have designed the `magni` software package to address the above three needs: it contains a selection of compressed sensing reconstruction algorithms, a framework for evaluating reconstruction algorithms through Monte Carlo Simulations, and more AFM-specific functionality for sampling and reconstructing images from AFM equipment. Further development of the package is planned through our ongoing FastAFM research project as this progresses over the coming years. This further development aims to extend the functionality of the package both in terms of directly interfacing the AFM equipment and in terms of adding more post-processing and reconstruction algorithms.

Implementation and architecture

The `magni` package is written in the Python programming language[2]. Python combined with a set of third-party libraries is an excellent tool for scientific and engineering applications [12]. The `magni` package uses the following third-party libraries to exploit code reuse, to ease the quality control process, and to enhance the end user experience:

- The `numpy` and `scipy` libraries are used for handling data (using the efficient `ndarray` data container class [13]) and for performing numerical computations. These are two of the core libraries for scientific computing using Python [12].
- The `pytables` library [14] is used for storing data through a high-abstraction HDF5 database interface.
- The `matplotlib` library [15] is used for visualising data.
- The easy-to-use `IPython` [16] Notebook is used for presenting a number of examples showing the capabilities of the `magni` package.

The `magni` package is itself a library, i.e. it is a collection of Python sub-packages and modules and as such does not provide any (graphical) user interface. The functionality provided by `magni` may be grouped into five categories with a sub-package assigned to each category, as illustrated in **Figure 2**. Furthermore, each sub-package has a number of modules or nested sub-packages to group related functionality.

As for coding style, procedural programming is preferred over object-oriented programming, to avoid unnecessary overhead [17]. Also, the developers found procedural programming more transparent for implementing the

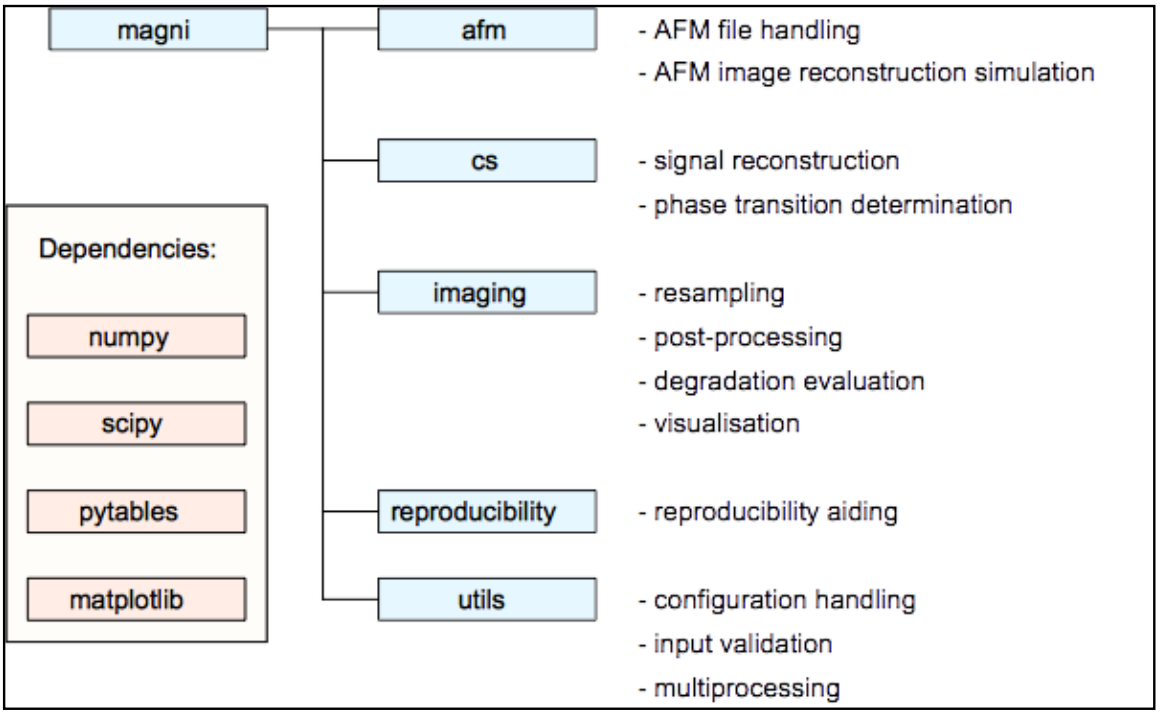


Figure 2: The functionality of the 5 sub-packages of magni along with the dependencies of `magni`.

desired functionality. Only in a few cases where the use of classes leads to significantly cleaner code, object-oriented programming is applied. Thus each module has its functionality encapsulated in a number of functions and classes, for which a distinction is made between public, internal, and private accessibility [18]. These accessibility levels are reflected in the code by use of the weak "internal use" indicator underscore convention as suggested by PEP8[3]:

- Private functionality is used only by the module itself. An underscore precedes the name of such functions or classes.
- Internal functionality is used by modules in the same or a nested sub-package. No underscore precedes the name of such functions or classes, but an underscore precedes the name of the module.
- Public functionality is available to the end-users and used by the package itself. No underscore precedes the name of such functions or classes, and no underscore precedes the name of the module.

Both functions and methods are implemented as to ensure readability in addition to efficiency by limiting the number of logical tasks per routine, the cyclomatic complexity [19], [20], and the number of physical code lines[4]. The cyclomatic complexity, i.e. the number of independent paths through the function, is kept below 10 for core functionality, consistent with observations on the level which programmers can usually handle flawlessly. This has been validated via the static code analyser `radon`[5]. The number of physical code lines is kept below 50 which is consistent with recommendations used at IBM and TRW [19] and general experiences in this field [21], [22].

The `magni` package complies with the PEP8 recommendation for Python coding conventions. This ensures that all Python code conforms to a number of recommendations with the aim of making the code user-friendly and thus easier and more robust from a maintenance point-of-view. The recommendations cover e.g. line width, variable naming conventions, package importing, indentations, and source encoding. Furthermore, `magni` is extensively documented using numpydoc[6] formatted doc-strings which describe the objective of the code, specify inputs and outputs of functions, elaborate on the functionality of the code, mention relevant references, and present examples of the use of the package. Finally, the input of every public (i.e. user-accessible) function and class is validated according to the known requirements with appropriate Python exceptions raised for invalid input. This is done to avoid runtime errors with hard-to-debug messages and stack traces.

Quality control

The code development procedure was built on what was found to be the best choice of methods from: 1) Well defined stage-based methods such as the structured waterfall approach [23] and the spiral approach [24] allowing backward interaction between different development phases; and 2) The test and adaptive centred Agile procedure [25] including e.g. Scrum [26], [27] and extreme programming [28], [29] with parts such as code reviews, code iteration, simplicity of design, frequent refactoring and collective ownership. All code modules were first

developed with tight links to the algorithm and refactoring was then performed to ensure maintainability, readability, robustness and sufficient performance. Multiple smaller and one large code review were held by 2-6 researchers including the main developers. Throughout the development process, Git was used for version control and issue tracking [30], and multiple branches were used to ensure that only tested code entered the master branch.

Testing and code validation has been handled by different instruments:

- 15 carefully designed end-to-end examples have been implemented in `IPython` Notebook (the .ipynb files). United, these examples exercise all critical code segments and serve the purpose of integration and regression testing.
- Doc-strings for all public functions include examples that are used in automated doctests[7]. This helps with the regression testing and ensures that the docstrings are kept up-to-date.
- `pyflakes` and `pylint` static source code analysers for Python have been used in the code development process to catch bugs and bad coding quality.

As always, no software package is better than its documentation and examples provided along with the package. The examples and part of the documentation have already been mentioned. Some of the examples use an AFM image, which is provided with the package. Furthermore, a full documentation in html is automatically generated from the doc-strings. A pdf version of this is shipped with the code.

(2) Availability

Operating system

Tested on Ubuntu 12.04 LTS Linux, Apple Mac OS X 10.9, and Microsoft Windows 7. Since `magni` is written in pure Python, it should run on any system on which Python and the `magni` dependencies run.

Programming language

The `magni` package is written in pure Python. Python 2 (>=2.7) or Python 3 (>=3.3) is required to use the package. The package has been tested with the Anaconda[8] Python distribution by Continuum Analytics.

Additional system requirements

`magni` is designed to process data sets of all sizes. Hardware requirements in terms of processor power, memory capacity, etc. depend primarily on the size of the data sets that are processed.

Dependencies

`magni` depends on numpy, `scipy`, `pytables`, and `matplotlib`. The package has been tested with:

- `numpy` version 1.8
- `scipy` version 0.13
- `pytables` version 3.1
- `matplotlib` version 1.3

The following libraries are optional requirements for `magni`:

- `IPython` Notebook >= 1.1 (for running examples)
- Math Kernel Library (`mkl`) >= 11.1 (for accelerated vector operations)
- `sphinx` >= 1.2 (for building the documentation from source)
- `napoleon` >= 0.2.6 (for building the documentation from source)

List of contributors

- Christian Schou Oxvig (Aalborg University) - Development
- Patrick Steffen Pedersen (Aalborg University) - Development
- Jan Østergaard (Aalborg University) - Testing and code review
- Thomas Arildsen (Aalborg University) - Testing and code review
- Tobias L. Jensen (Aalborg University) - Testing and code review
- Torben Larsen (Aalborg University) - Testing and code review

Archive

Name

Videnbasen (VBN), Aalborg University

Persistent identifier

DOI: http://doi.org/10.5278/VBN/MISC/Magni

License

BSD 2-Clause

Publisher

Christian Schou Oxvig

Date published

23/05/14

Code Repository

Name

GitHub

Identifier

https://github.com/SIP-AAU/Magni/

License

BSD 2-Clause

Date published

23/05/14

Language

English

(3) Reuse potential

The `magni` package has been designed to facilitate reuse through extensive documentation of functionality and interfaces. The code has been implemented with focus on readability. And the package is accompanied by a number of examples to demonstrate its use in various use-cases.

We expect the `magni` package to have significant reuse potential for researchers in the area of AFM, particularly in relation to compressed sensing acquisition and reconstruction of AFM images. This applies to both users interested in developing and testing new sampling patterns for use in conjunction with compressed sensing techniques and users developing new algorithms for compressed sensing in the context of AFM.

Furthermore, the `magni` package is applicable to compressed sensing in general and can be particularly useful to those looking for compressed sensing reconstruction algorithms for use in Python, which have so far been scarce. In addition to reconstruction algorithms, the package provides a consistent framework which can be used to empirically estimate the reconstruction capabilities of the users' own reconstruction algorithms in terms of reconstruction phase transitions.

Due to the `magni` package being based on well established Python libraries, it fits naturally into the Python ecosystem [31] of high-quality tools for scientific computing. The software complies with the reproducible research paradigm as used in the field of signal processing [11]. The intent of reproducible research is to create an open and transparent approach to the software related to some specific conducted research – see e.g. [32], [33], [34], [35]. We thus provide full open access to all source code and full reuse rights via the generous BSD 2-Clause license, making it easy for others to use the code base. While it is the plan of the developers to continuously expand the functionality of the software, others are free to use it in separate branches. The `reproducibility` subpackage goes one step further by providing functionality for reading and writing the version and complete configuration of `magni`. Furthermore, information about conda, git revision, and the system platform is included if available. This information can be automatically shipped alongside the results, by letting `magni` use the same HDF5 database for storing the two. With these features, the developers hope to inspire others to make their results reproducible.

Notes

[1] See http://dx.doi.org/10.5278/vbn/projects/FastAFM/.
[2] See https://www.python.org/.
[3] Python Enhancement Proposal, see http://legacy.python.org/dev/peps/pep-0008/.
[4] See https://docs.python.org/2/reference/lexical_analysis.html.
[5] See https://github.com/rubik/radon.
[6] See https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt/.
[7] See https://docs.python.org/2/library/doctest.html/.
[8] See http://www.continuum.io/anacondace.html/.

References

1. **Song, B, Xi, N, Yang, R, Lai, K W C** and **Qu, C** 2011 Video Rate Atomic Force Microscopy (AFM) Imaging using Compressive Sensing. *11th IEEE International Conference on Nanotechnology*. 15-18 August 2011, Portland, Oregon, USA: 1056–1059, DOI: http://dx.doi.org/10.1109/NANO.2011.6144587
2. **Andersson, S B** and **Pao, L Y** 2012 Non-Raster Sampling in Atomic Force Microscopy: A Compressed Sensing Approach. *American Control Conference (ACC)*. 27-29 June 2012, Montréal, Canada: 2485–2490.
3. **Baraniuk, R G** 2007 Compressive sensing [lecture notes]. *IEEE Signal Processing Magazine* 24(4): 118–121, DOI: http://dx.doi.org/10.1109/MSP.2007.4286571
4. **Candès, E J** and **Wakin, M B** 2007 An Introduction To Compressive Sampling. *IEEE Signal Processing Magazine* 25(2): 21–30, DOI: http://dx.doi.org/10.1109/MSP.2007.914731
5. **Abramovitch, D Y, Andersson, S B, Pao, L Y** and **Schitter, G** 2007 A Tutorial on the Mechanisms, Dynamics, and Control of Atomic Force Microscopes. *American Control Conference*. 11-13 July 2007, New York City, USA: 3488–3502, DOI: http://dx.doi.org/10.1109/ACC.2007.4282300
6. **Bhushan, B** and **Marti, O** 2010 Scanning Probe Microscopy – Principle of Operation, Instrumentation, and Probes In: *Springer Handbook of Nanotechnology*. Berlin Heidelberg: Springer, DOI: http://dx.doi.org/10.1007/978-3-642-02525-9_21
7. **Hansma, P K, Schitter, G, Fantner, G E** and **Prater, C** 2006 High-Speed Atomic Force Microscopy. *Science* 314(5799): 601–602, DOI: http://dx.doi.org/10.1126/science.1133497
8. **van den Berg, E** and **Friedlander, M P** 2008 Probing the Pareto Frontier for Basis Pursuit Solutions. *SIAM Journal on Scientific Computing* 31(2): 890–912, DOI: http://dx.doi.org/10.1137/080714488
9. **Yang, J** and **Zhang, Y** 2008 Alternating Direction Algorithms for l1-Problems in Compressive Sensing. *SIAM Journal on Scientific Computing* 33(1): 250–278, DOI: http://dx.doi.org/10.1137/090777761
10. **Klapetek, P** 2013 *Quantitative Data Processing in Scanning Probe Microscopy: SPM Applications for Nanometrology*. 1st ed. Elsevier.
11. **Vandewalle, P, Kovačević,** J and **Vetterli, M** 2009 Reproducible Research in Signal Processing [What, why, and how]. *IEEE Signal Processing Magazine* 26(3): 37–47, DOI: http://dx.doi.org/10.1109/MSP.2009.932122
12. **Oliphant, T E** 2007 Python for Scientific Computing. *Computing in Science & Engineering* 9(3): 10–20, DOI: http://dx.doi.org/10.1109/MCSE.2007.58
13. **van der Walt, S, Colbert, S C** and **Varoquaux, G** 2011 The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering* 13(2): 22–30, DOI: http://dx.doi.org/10.1109/MCSE.2011.37
14. **Alted, F** and **Fernández-Alonso, M** 2003 PyTables : Processing And Analyzing Extremely Large Amounts Of Data In Python. *PyCon2003*. April 2003, Washington, D.C., USA: 1–9.

15. **Hunter, J D** 2007 Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9(3): 90–95, DOI: http://dx.doi.org/10.1109/MCSE.2007.55
16. **Pérez, F** and **Granger, B E** 2007 IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering* 9(3): 21–29, DOI: http://dx.doi.org/10.1109/MCSE.2007.53
17. **Jürgens, D** 2009 *Survey on Software Engineering for Scientific Applications: Reuseable Software, Grid Computing and Applications*. Germany: Institute of Scientific Computing, Carl-Friedrich-Gauss-Fakultät, Technische Universität Braunschweig.
18. **Moock, C** 2007 Essential ActionScript 3.0. 1st ed. O'Reilly Media / Adobe Dev Library.
19. **McCabe, T J** 1976 A Complexity Measure. *IEEE Transactions on Software Engineering* SE-2(4): 308–320, DOI: http://dx.doi.org/10.1109/TSE.1976.233837
20. **Watson, A H** and **McCabe, T J** 1996 Structured Testing: *A Testing Methodology Using the Cyclomatic Complexity Metric*. National Institute of Standards and Technology (NIST): 500-235.
21. **Shen, V Y, Yu, T J, Thebaut, S M** and **Paulsen, L R** 1985 Identifying Error-Prone Software -- An Empirical Study. *IEEE Transactions on Software Engineering* 11(4): 317–324, DOI: http://dx.doi.org/10.1109/TSE.1985.232222
22. **Kelly, D, Hook, D** and **Sanders, R** 2009 Five Recommended Practices for Computational Scientists Who Write Software. *Computing in Science & Engineering* 11(5): 48–53, DOI: http://dx.doi.org/10.1109/MCSE.2009.139
23. **Royce, W W** 1970 Managing the Development of Large Software Systems. *IEEE WESCON.* August 1970, : 328–338.
24. **Boehm, B W** 1988 A Spiral Model of Software Development and Enhancement. *Computer* 21(5): 61–72, DOI: http://dx.doi.org/10.1109/2.59
25. **Sletholt, M T, Hannay, J E, Pfahl, D** and **Langtangen, H P** 2012 What Do We Know about Scientific Software Development's Agile Practices?. *Computing in Science & Engineering* 14(2): 24–37, DOI: http://dx.doi.org/10.1109/MCSE.2011.113
26. **Mahnic, V** 2012 A Capstone Course on Agile Software Development Using Scrum. *IEEE Transactions on Education* 55(1): 99–106, DOI: http://dx.doi.org/10.1109/TE.2011.2142311
27. **Rising, L** and **Janoff, N S** 2000 The Scrum Software Development Process for Small Teams. *IEEE Software* 17(4): 26–32, DOI: http://dx.doi.org/10.1109/52.854065
28. **Beck, K** 1999 Embracing Change with Extreme Programming. *Computer* 32(10): 70–77, DOI: http://dx.doi.org/10.1109/2.796139
29. **Maurer, F** and **Martel, S** 2002 Extreme Programming: Rapid Development for Web-Based Applications. *IEEE Internet Computing* 6(1): 86–90, DOI: http://dx.doi.org/10.1109/4236.989006
30. **Loeliger, J** and **McCullough, M** 2012 *Version Control with Git: Powerful tools and techniques for collaborative software development*. 2nd ed. O'Reilly Media.
31. **Pérez, F, Granger, B E** and **Hunter, J D** 2011 Python: An Ecosystem for Scientific Computing. *Computing in Science & Engineering* 13(2): 13–21, DOI: http://dx.doi.org/10.1109/MCSE.2010.119
32. **Schwab, M, Karrenbach, M** and **Claerbout, J** 2011 Making Scientific Computations Reproducible. *Computing in Science & Engineering* 2(6): 61–67, DOI: http://dx.doi.org/10.1109/5992.881708
33. **LeVeque, R J, Mitchell, I M** and **Stodden, V** 2012 Reproducible Research for Scientific Computing: Tools and Strategies for Changing the Culture. *Computing in Science & Engineering* 14(4): 13–17, DOI: http://dx.doi.org/10.1109/MCSE.2012.38
34. **Barni, M** and **Perez-Gonzalez, F** 2005 Pushing Science into Signal Processing. *IEEE Signal Processing Magazine* 22(4): 120–119, DOI: http://dx.doi.org/10.1109/MSP.2005.1458324
35. **Fomel, S** and **Claerbout, J F** 2009 Guest Editors' Introduction: Reproducible Research. *Computing in Science & Engineering* 11(1): 5–7, DOI: http://dx.doi.org/10.1109/MCSE.2009.14

How to cite this article: Oxvig, C S, Pedersen , P S, Arildsen, T, Østergaard, J and Larsen, T 2014 Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images. *Journal of Open Research Software,* 2: e29, DOI: http://dx.doi.org/10.5334/jors.bk

Published: 07 October 2014

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS

Paper E

Validating Function Arguments in Python Signal Processing Applications

Patrick Steffen Pedersen, Christian Schou Oxvig,
Jan Østergaard, Torben Larsen

The paper has been published in
Proceedings of the 15th Python in Science Conference (SciPy), pp. 106–113,
Austin, Texas, USA, Jul. 11 – 17, 2016.
http://conference.scipy.org/proceedings/scipy2016/patrick_pedersen.html

The succeeding 8 non-blank pages constitute the published manuscript which is subject to the following copyright notice:

Validating Function Arguments in Python Signal Processing Applications

Patrick Steffen Pedersen‡*, Christian Schou Oxvig‡, Jan Østergaard‡, Torben Larsen‡

Abstract—Python does not have a built-in mechanism to validate the value of function arguments. This can lead to nonsensical exceptions, unexpected behaviour, erroneous results and the like. In the present paper, we define the concept of so-called application-driven data types which place a layer of abstraction on top of Python data types. With this concept in mind, we discuss the current argument validation solutions of PyDBC, Traitlets and Numtraits, MyPy, PyValid, and PyContracts. We find that they share the issue of expressing the validation scheme in terms of Python objects rather than in terms of the data they hold. Consequently, we lay out a suggestion for a validation strategy including what qualifies as a validation scheme, how to create an interface which promotes both usability and readability, and which Python constructs to encourage using for validation encapsulation. A reference implementation of the suggested validation strategy is part of the open-source Python package, Magni which is thus presented along with a number of examples of the usages of this package.

Index Terms—Function Argument Validation, Application-driven Data Types, Signal Processing, Computational Science

Introduction

Python is a dynamically typed language that does not have a built-in mechanism to ensure that the value of an argument passed to a function conforms to the intentions of that particular argument. This can lead to nonsensical exceptions, unexpected behaviour, erroneous results and the like. In signal processing applications and scientific computing in general, large amounts of numerical data are passed to any number of functions that inherently impose limitations upon that data. If such functions do not validate their arguments, these limitations may be violated without raising exceptions leading to potentially erroneous results. Thus, although impairing the performance, explicit validation may not only spare the user a lot of frustration by providing useful exceptions but may also prevent erroneous results and thereby ensure the credibility of works in scientific computing.

The usage of explicit function argument validation could be considered "unpythonic"[1] as it goes against dynamic typing [CVS13] and duck typing [CVS13] by not relying on documentation, clear code and testing to ensure correct usage. Even so, there exist a number of solutions for validating function arguments in Python relying on a wide range of language constructs and interfaces. The validation capabilities of these solutions vary greatly from type, attribute, and value checks to fully customisable checks. Among these solutions are PyDBC, Traitlets and Numtraits, MyPy, PyValid, and PyContracts which are all discussed later. Most of the solutions do, however, seem to have an interface which relates to the data model used by Python and therefore translates to Python check in a straightforward way.

Unfortunately, there are a number of shortcomings with the existing validation strategies as implemented in the existing Python packages with validation capabilities; in particular in signal processing applications. Some of the existing solutions lack generality, some do not promote readability, and some are inconvenient to use. However, the primary issue is that the validation scheme of function arguments is expressed in terms of Python objects rather than in terms of the data they hold, and this poses a number of problems. Even with these shortcomings, the existing solutions represent a large variety of validation strategies which are an obvious source of inspiration.

In the present effort, we suggest the concept of so-called application-driven data types as a signal processing data model for programming. These data types are intended for expressing the validation scheme of function arguments. Furthermore, based on existing solutions, we lay out a new strategy for validating function arguments in Python signal processing applications. Finally, we present the open-source Python package, Magni which includes a reference implementation of the suggested validation strategy, and we show a number of examples of the usage of this package.

The remainder of the present paper is organised as follows. We first take a look at validation in Python at a glance before presenting the concept of application-driven data types. Next, we discuss some of the existing solutions with an emphasis on the validation strategies they represent. Drawing on the observations made, we then present the suggested Python validation strategy. Following this specification, we detail a reference implementation of it and give examples of its usage. Finally, we conclude on what is achieved by the presented validation strategy and reference implementation as well as when to use them. All code examples have been run with Python 2.7 unless otherwise noted, all tracebacks have been removed to save space, and exception messages and the like have been broken across multiple lines using trailing backslashes where necessary.

* *Corresponding author: psp@es.aau.dk*
‡ *Faculty of Engineering and Science, Department of Electronic Systems, Section of Signal and Information Processing, Aalborg University, 9220 Aalborg, Denmark*

1. For an informal yet fitting definition, see http://stackoverflow.com/questions/25011078/what-does-pythonic-mean

Validation in Python at a glance

For the purpose of exemplifying the concepts discussed in this section, we define a simple Python function for returning the square root of the first item of a sequence. Obviously, only a sequence with a non-negative, numerical first item, $a_0 \in \mathbb{R}_{\geq 0}$, should be a valid argument of this function.

```
def do_something(a):
    print(a[0]**0.5)
```

To quote the Zen of Python[2], "there should be one-- and preferably only one --obvious way to do it" when faced with solving a task in Python, and the obvious ways to solve common tasks are oftentimes referred to as pythonic idioms. When it comes to function argument validation in Python, the most pythonic idiom is to clearly document what a function expects and then just try to use whatever gets passed to the function and either let exceptions propagate or catch attribute errors and raise other exceptions instead. This approach is well-suited for Python because it is a dynamically typed language. Basically, this means that variables, such as the function argument in the example, are not limited to hold values of a certain type. Instead, we can pass a number, a sequence, a mapping, or any other type to the example function. Regardless of the type, Python tries to use whatever value gets passed to the function which is a consequence of duck typing. The basic principle is that if a bird looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck. That is, if a value exhibits the desired behaviour, then that value probably is valid. Translated to our example, if the value of the function argument, `a`, has the `__getitem__` attribute which Python uses internally for retrieving the first item, then `a` probably is valid. Thus, the most pythonic idiom would rely on documentation, clear code, and testing to ensure correct usage rather than explicitly testing function arguments to ensure conformity to the intentions of the function.

What happens, then, if the value of a function argument is invalid by the reckoning of duck typing? This is the case with the following call as the built-in `int` type does not define `__getitem__`:

```
>>> integer = 42
>>> do_something(integer)
TypeError: 'int' object has no attribute \
'__getitem__'
```

With the following call, a `TypeError` exception is raised with a message that "`'int'` object has no attribute `'__getitem__'`". Even with this simple example, such an exception message is less sensible than desired. Furthermore, such an exception is as likely to occur in some obscure function call and, thus, be accompanied by a traceback with more levels than anyone would want. However, at least the presence of an exception indicates that something did not go as expected. What happens, however, if the value of a function argument is valid by the reckoning of duck typing but does not conform to the intentions of the function? This is the case with the following call as the built-in `dict` type defines `__getitem__` but with a different purpose than the `__getitem__` of sequences:

```
>>> dictionary = {-1: 0, 0: 1}
>>> do_something(dictionary)
1.0
```

The intention of the function is to operate on the first item of the function argument, but `dictionary` is unordered meaning that there is no such thing as a first item. However, the call does not raise an exception because of duck typing. This is an example of unexpected or erroneous behaviour.

The two examples of calls presented showcase how the lack of function argument validation can lead to hard-to-debug exceptions or even worse to unexpected or erroneous behaviour. The benefit of explicit function argument validation is that the mentioned problems should be avoided. Furthermore, by having such validation for functions that are part of a public API of released packages, the package is made more trustworthy and user-friendly.

How to Test for Validity

One way to test for validity would be to check if the value of a variable has a certain type. That is, to determine the validity based on what a value *is*. For example, we could rewrite the `do_something` example in the following way:

```
def do_something(a):
    if not isinstance(a, list):
        raise TypeError('Descriptive message.')

    if not isinstance(a[0], int):
        raise TypeError('Descriptive message.')

    print(a[0]**0.5)
```

Obviously, this approach to validation goes against dynamical typing as it restricts variables to only hold values of certain types. In the example, `a` may hold values of the type `list` or of a derived type, and the first item of `a` may hold values of the type `int` or of a derived type. Clearly, the validation in the above example is too restrictive: as the intention of the function is to allow a sequence with a non-negative, numerical first item, the following call should pass but instead fails the validation checks:

```
>>> sequence = (0., 1.)
>>> do_something(sequence)
TypeError: Descriptive message.
```

The issue is that a number of Python types represent sequences, and a number of Python types represent numbers. This could be accounted for in the example, but the point to stress is that the programmer should not have to know about every single Python type, nor should he or she have to explicitly list a large number of Python types for each validation check.

Another way to test for validity would be to check if the value of a variable displays a certain behaviour. That is, to determine the validity based on what a value *can do*. For example, we could rewrite the `do_something` example in the following way:

```
def do_something(a):
    if not hasattr(a, '__getitem__'):
        raise TypeError('Descriptive message.')

    if not hasattr(a[0], '__pow__'):
        raise TypeError('Descriptive message.')

    print(a[0]**0.5)
```

Clearly, this approach to validation is along the lines of duck typing as it explicitly checks for the presence of the required attribute. In the example, `a` may hold values of any type that defines the `__getitem__` attribute, and `a[0]` may hold values of any type that defines the `__pow__` attribute. Unlike with the first way to test for validity, the validation in the above example is not restrictive enough as already explained using the example with the dictionary. The same check could be achieved in a cleaner and

2. See https://www.python.org/dev/peps/pep-0020/

more thorough way using abstract base classes[3], but this solution would essentially suffer from the same type of problem.

Neither of the two ways to test for validity mentioned, consider the fact that the square root operation is only defined for non-negative `a[0]` values if complex numbers are ignored. Thus, a third way to partially test for validity would be to check if the value of a variable is in a set of valid values. That is, to determine validity based on what a value *contains*. For example, we could rewrite the `do_something` example in the following way:

```
def do_something(a):
    if len(a) < 1:
        raise ValueError('Descriptive message.')

    if a[0] < 0:
        raise ValueError('Descriptive message.')

    print(a[0]**0.5)
```

Obviously, this approach would have to be combined with something else to ensure that `a` is indeed a sequence and `a[0]` is indeed a number as covered by the first two ways to test for validity.

The Concept of Application-Driven Data Types

The approaches presented in the previous section do not even consider less common although valid cases such as non-derived types that only implicitly define the required attributes. Even more so, it is apparent that there is no straightforward way to test for validity based solely on what a value *is*, *can do*, or *contains*. A possible explanation for this is that all three approaches express the validation scheme in terms of Python objects rather than in terms of the data they hold. Indeed, it was easy to identify and in plain writing express that the function argument of the `do_something` example must be a sequence with a non-negative, numerical first item. Expressing the validation scheme in this way does provide a layer of abstraction.

Instead of checking if the value of `a` is a certain Python type, it would be convenient to be able to check if the value of `a` is a sequence. Likewise, instead of checking if the value of `a[0]` is a certain Python type containing a non-negative value, it would be convenient to be able to check if the value of `a[0]` is a non-negative, numerical type. Both "sequence" and "non-negative, numerical type" are examples of data types at a higher abstraction level than actual Python types, and we will name these abstractions application-driven data types.

In the context of scientific computing and signal processing in particular, the most relevant and interesting application-driven data types are numerical types. Here, an application-driven data type is some "mental" intersection between math and computer science in scientific computing and signal processing in particular. For example, the set of real-valued matrices with dimensions m times n, $\mathbb{R}^{m \times n}$, is an example of an application-driven data type. If the user is able to test the validity of a function argument against this application-driven data type, there is no need for the user to consider the distinction between Python floats, numpy generics, numpy ndarrays, and so on.

Existing Solutions

As mentioned in the introduction, there exist a number of solutions to validating function arguments in Python relying on a wide range of language constructs and interfaces and thereby representing a large variety of validation strategies. As these strategies are a source of inspiration for any new validation strategy, this section is used to briefly discuss some existing solutions with a focus on the three aspects which make up the suggested validation strategy: 1) The validation schemes that can be expressed and through that the abstraction level of the application-driven data types. 2) The way the interface of the implementation allows the validation scheme to be specified. 3) The Python constructs used to allow Python to validate the function arguments against the validation specification. Additionally, the relevant versions of Python are mentioned as 4) under each solution. Thus, the emphasis of this section is not to give a complete review of all existing solutions.

PyDBC

Although the original PyDBC[4] is long outdated, it represents an approach worth mentioning. The package allows so-called contracts to be specified using method preconditions, method postconditions, and class invariants. Thus, function argument validation can be performed using method preconditions. In the following example, the function argument, `a`, of the function, `exemplify` is validated to be a real scalar in the range $[0; 1]$:

```
import dbc
__metaclass__ = dbc.DBC

class Example:
    def exemplify(self, a):
        pass  # do something

    def exemplify__pre(self, a):
        assert isinstance(a, float)
        assert 0 <= a <= 1
```

When an invalid value is passed, the following assertion error occurs:

```
>>> example = Example()
>>> example.exemplify(-0.5)
AssertionError
```

As for validation strategy, the following observations are made:

1. As shown in the example above, the validation function, `exemplify__pre` contains custom validity checks, as PyDBC does not include any functionality for specifying a validation scheme.
2. Without any functionality for specifying a validation scheme, there is no fixed interface, and the user instead writes a number of `assert` statements to validate the function arguments.
3. The Python constructs used rely on object oriented Python by using metaclasses. When the metaclass creates the class, it rewrites the function `exemplify` to first invoke the function named `exemplify__pre` when `exemplify` is called following a fixed naming scheme.
4. PyDBC was intended for Python 2.2 and has not been changed since 2005, but the package does work with Python 2.7. It does, however, not work with Python 3, but the same functionality could indeed be implemented in Python 3.

3. See https://docs.python.org/2/glossary.html#term-abstract-base-class

4. See http://www.nongnu.org/pydbc/

Traits, Traitlets, and Numtraits

Traits[5] is an extensive package by Enthought which provides class attributes with the additional characteristics of customisable initialisation, validation, delegation, notification, and even visualisation. Traitlets[6] is a lightweight Traits-like module which provides customisable validation, default values, and notification. Finally, Numtraits[7] adds to Traitlets with a numerical trait with more versatility in validation than that of the numerical traits of Traitlets. Thus, although hardly as intended by the developers, function argument validation can be performed using an attribute for each function argument. In the following example, the function argument, `a`, of the function, `exemplify` is validated to be a real scalar in the range $[0; 1]$:

```
from numtraits import NumericalTrait
from traitlets import HasTraits

class Example(HasTraits):
    _a = NumericalTrait(ndim=0, domain=(0, 1))

    def exemplify(self, a):
        self._a = a

        pass  # do something
```

When an invalid value is passed, the following assertion error occurs:

```
>>> example = Example()
>>> example.exemplify(-0.5)
traitlets.traitlets.TraitError: _a should be in \
the range [0:1]
```

As for validation strategy, the following observations are made:

1. The validation scheme of Traitlets requires specifying a static Python type, allows specifying a valid range of values for numerical types, and allows specifying relevant properties for other specific types. Furthermore, the validation scheme of the numerical trait of Numtraits does not require specifying a static Python type but allows specifying the number of dimensions and the shape of a value.
2. As shown in the example above, the interface of the implementation lets the user specify the validation scheme using a single call for each function argument with named arguments, named keyword arguments and in some cases unspecified keyword arguments using `**kwargs`.
3. The Python constructs used rely on object oriented Python by using descriptors which modify the retrieving and modification of attribute values of objects. Thus, when assigning a new value to an attribute, the relevant descriptor validates the new value.
4. Traitlets and Numtraits work with Python 2.7 and with Python 3.3 or above.

Annotations, Type Hints, and MyPy

PEP 3107[8] is a Python enhancement proposal on function annotations which is a feature which has recently been added to Python. This PEP allows arbitrary annotations without assigning any meaning to the particular annotations. PEP 484[9] is a PEP on type hints which attach a certain meaning to particular annotations to hint the type of argument values and return values of functions. The most important goal of this is static analysis, but runtime type checking is mentioned as a potential goal also. For more information, see PEP 483[10] on the theory of type hints and PEP 482[11] for a literature overview for type hints. MyPy[12] is a static type checker which, thus, does not enforce data type conformance at runtime. In the following example, the function argument, `a`, of the function, `exemplify` is validated to be a real scalar:

```
def exemplify(a: float):
    pass  # do something

exemplify('0')
```

When the script above is passed to MyPy using Python 3.5, the following message is produced:

```
$ mypy example.py
example.py:4: error: Argument 1 to "exemplify" has \
incompatible type "str"; expected "float"
```

As for validation strategy, the following observations are made:

1. The validation scheme of MyPy requires specifying a static Python type or a union of static Python types. This is hardly surprising for a static type checker.
2. As mentioned, the syntax of annotations is given by PEP 3107, and the format of the type hints is given by PEP 484 making the type hints explicit and readable although a less well-known feature of Python.
3. The Python constructs used rely only on annotations and runs offline and separately of normal execution of Python code.
4. PEP 484 was accepted for Python 3.5, but the syntax is compatible with that of PEP 3107 which was accepted for Python 3.0, and thus MyPy works with Python 3.2 or above. Furthermore, PEP 484 suggests a syntax for Python 2.7 using comments instead of annotations, and MyPy supports this and thus also works with Python 2.7.

PyValid

As the name suggests, PyValid[13] is a Python validation package, and it allows validation of function arguments and function return values. In the following example, the function argument, `a`, of the function, `exemplify` is validated to be a real scalar:

```
from pyvalid import accepts

@accepts(float)
def exemplify(a):
    pass  # do something
```

When an invalid value is passed, the following assertion error occurs:

```
>>> exemplify(0)
pyvalid.__exceptions.ArgumentValidationError: The \
1st argument of exemplify() is not in a \
[<type 'float'>]
```

As for validation strategy, the following observations are made:

1. The validation scheme for PyValid requires specifying one or more static Python types and acts as a runtime

5. See http://docs.enthought.com/traits/
6. See http://traitlets.readthedocs.org/
7. See http://github.com/astrofrog/numtraits/
8. See https://www.python.org/dev/peps/pep-3107/
9. See https://www.python.org/dev/peps/pep-0484/
10. See https://www.python.org/dev/peps/pep-0483/
11. See https://www.python.org/dev/peps/pep-0482/
12. See http://mypy.readthedocs.org/
13. See http://uzumaxy.github.com/pyvalid/

type checker. Thus, in terms of validation scheme capabilities, this is equivalent to MyPy.
2. As shown in the example above, the interface of the implementation lets the user specify the validation scheme using a single call for an entire function with a single argument or keyword argument for each validated function argument.
3. The Python constructs used rely on decorators by including an `accept` decorator in order to precede function execution by function argument validation.
4. PyValid works with Python 2.6 or above and with Python 3.

PyContracts

PyContracts[14] is a Python package that allows declaring constraints on function arguments and return values. In the following example, the function argument, `a`, of the function, `exemplify` is validated to be a real scalar in the range $[0; 1]$:

```
from contracts import contract

@contract(a='float,>=0,<=1')
def exemplify(a):
    pass  # do something
```

When an invalid value is passed, the following assertion error occurs:

```
>>> exemplify(-0.5)
contracts.interface.ContractNotRespected: Breach \
for argument 'a' to exemplify().
Condition -0.5 >= 0 not respected
checking: >=0              for value: Instance of \
<type 'float'>: -0.5
checking: float,>=0,<=1    for value: Instance of \
<type 'float'>: -0.5
Variables bound in inner context:
```

As for validation strategy, the following observations are made:

1. The capabilities of PyContracts allows specifying any conceivable validation scheme. This is achieved in part through built-in capabilities including specifying one or more static types in a flexible way, specifying value ranges, and specifying flexible length/shape constraints. And in part through custom specifications by using so-called custom contracts.
2. As shown in the example above, the interface of the implementation lets the user specify the validation scheme using a single call for an entire function with a single keyword argument for each validated function argument. The validation schemes for the individual arguments are specified using a custom string format. As the validation scheme becomes more advanced, the specification becomes less Python-like and less readable. For example, the following was taken from an official presentation and allows an argument to be a list containing a maximum of two types of objects: `list(type(t)|type(u))`.
3. The Python constructs used rely on decorators by including a `contract` decorator in order to precede function execution by function argument validation. Depending on the preference of the user, the validation scheme is either specified through arguments of the decorator, through annotations in the form of type hints or custom annotations, or through docstrings following a specific format.
4. PyContracts works with Python 2 and with Python 3.

14. See http://andreacensi.github.com/contracts/

The Suggested Python Validation Strategy

This section lays out a suggestion for a Python validation strategy for validating function arguments in signal processing applications. This strategy uses the introduced concept of application-driven data types and the observations made on the strategies of existing solutions. As mentioned in the previous section, the suggested validation strategy is made up of three aspects which are discussed separately in the following.

The Suggested Validation Schemes

As described in a previous section, we want to specify validation schemes in terms of application-driven data types rather than in terms of what a valid Python object *is*, *can do*, or *contains*. Needless to say, a translation must still be made from application-driven data types to Python data types, but this task is left for the validation package according to the suggested validation strategy. For an early implementation, any application-driven data type will allow only a limited set of Python data types. This does, however, not mean that the application-driven data type is limited to a few Python data types. Rather, more Python data types may be added along the way as long as they provide the necessary attributes with the desired interpretation. Thus, effectively, the suggested validation strategy can be considered less strict than static type checking but more strict than duck type checking.

The numerical trait of the Numtraits package has an interesting approach which is not too different from the concept of application-driven data types. The numerical trait does not distinguish between Python data types as long as they are numerical, and this corresponds to the most general numerical application-driven data type able to assume any numerical value of any shape. Furthermore, the numerical trait allows restricting the data type to more restrictive data types by specifying a number of dimensions, a specific shape, and/or a range of valid values. Indeed, signal processing applications could benefit from having such an application-driven data type. However, in some applications it may be necessary to work with boolean values, integral values, real values, or complex values only. Therefore, it should be possible to restrict the data type to suit these cases in addition to the other possible restrictions allowed by numerical traits.

To summarise, in Python signal processing applications, there should be an application-driven data type representing the most general numerical value being able to assume any numerical value of any shape. This data type should be able to be restricted to less general data types by specifying the mathematical set, the range or domain of valid values, the number of dimensions, and/or the specific shape of the data type. The suggested validation schemes should be expressed in terms of the desired application-driven data type.

The Suggested Interface Type

Most of the existing solutions which were mentioned in the previous section specify the validation scheme of all function arguments of a function in a single call to the validation package in question. This is not the case with the traits of the Trailets and Numtraits packages which only specify the validation scheme of a single function argument in each call to the validation package.

From the perspective of the authors, the latter approach yields the better readability. Therefore, the suggested interface type should only let the user specify the validation scheme of a single function argument in each call.

As for the specifics of the interface, the validation scheme must be easy both for the programmer to state and for users to read. The PyContracts details its own format where the validation scheme is given by a string. However, it would be desirable to use a more standard Python interface to ease the usages even if it means having to be more verbose. On the other hand, the numerical trait of the Numtraits package uses named named arguments and keyword arguments which relate to the possible restrictions of the application-driven data types. From the perspective of the authors, the latter approach works well with application-driven data types and result in logical, easy to use interfaces. Therefore, the suggested interface should use named arguments and keyword arguments related to the possible restrictions of the general numerical application-driven data type to specify the validation scheme of function arguments.

The Suggested Python Constructs to Use

There are a lot of Python constructs which could potentially be used as showcased by the existing solutions. PyContracts allows the user to specify the validation scheme through the docstring of a function. However, most users would not expect docstrings to be parsed to yield the validation scheme, and furthermore the format used to specify the validation scheme would not be obvious because of the lack of restrictions put on docstrings. Therefore, docstrings are not suggested as a Python construct to use here. Annotations, as used by MyPy, are relatively new to Python, but that should not disqualify them from being used. However, the format used would not be obvious because there are few restrictions put on annotations so with the exception of type hints which are insufficient for this purpose. Therefore, annotations are not suggested as a Python construct to use here.

Next, there are the object oriented Python constructs. Metaclasses, as used by, PyDBC, have existed for a long time. However, these have changed over time, and so the metaclass attribute feature of Python 2 no longer works in Python 3, and only one metaclass is allowed per class in the more recent Python versions. Furthermore, the behaviour of metaclasses makes them impair the readability, especially to users that are unfamiliar with the construct. Therefore, metaclasses are not suggested as a Python construct to use here. Descriptors, as used by Traits, Traitlets, and Numtraits, are another feature applicable to object oriented Python, and these can provide flexibility and readability. However, they are limited to object oriented Python, and furthermore it seems unpythonic to validate function arguments by invoking descriptors through class instance attribute assignment. Therefore, descriptors are not suggested as a Python construct to use here.

Decorators, as used by PyValid and PyContracts, are a well-known and general Python construct. However, it is not immediately apparent if something goes on "under the hood", and the pythonic approach is to specify the validation scheme of all function arguments in a single decorator call, both of which affect readability. Therefore, decorators are not suggested as a Python construct to use here.

The suggested Python construct values explicit over implicit and promotes readability. The suggestion is to define and explicitly call a nested validation function with no arguments. There are a number of obvious alternatives which are not suggested for different reasons:

- It is not suggested to precede the function code by calls directly to a validation package because this does not clearly separate validation from the rest of the code.
- It is not suggested to use arguments for the validation function because this could potentially lead to error-prone validation if the validation function arguments are wrongly named or ordered, or the function arguments are renamed or reordered.
- It is not suggested to use a global rather than nested validation function because this could potentially separate the validation from the function and thus reduce readability.

Magni Reference Implementation

A reference implementation of the **suggested validation strategy** is made available by the open source Magni Python package [OPA+14] through the subpackage `magni.utils.validation`. The subpackage contains the following functions:

```
decorate_validation(func)
disable_validation()
validate_generic(
    name, type_, value_in=None, len_=None,
    keys_in=None, has_keys=None, ignore_none=False,
    var=None)
validate_levels(name, levels)
validate_numeric(
    name, type_, range_='[-inf;inf]', shape=(),
    precision=None, ignore_none=False, var=None)
```

Of these, `validate_generic` and `validate_levels` are concerned with validating objects outside the scope of the present paper. The function, `disable_validation` can be used to disable validation globally. Although discouraged, this can be done to remove the overhead of validating function arguments. As the name suggests, `decorate_validation` is a decorator, and this should be used to decorate every validation function with the sole purpose of being able to disable validation. Using the suggested validation strategy with Magni, the following structure is used for all validation adhering to **the suggested Python constructs to use**:

```
from magni.utils.validation import decorate_validation

def func(*args, **kwargs):
    @decorate_validation
    def validate_input():
        pass  # validation calls

    validate_input()

    pass  # the body of func
```

The remaining function, `validate_numeric`, is used to validate numeric objects based on application-driven data types as proposed by **the suggested validation scheme** of the validation strategy. This is done using the interface as proposed by **the suggested interface type** of the validation strategy: The `type_` argument is used for specifying one or more of the `boolean`, `integer`, `floating`, and `complex` subtype specifiers. The `range_` argument is used for specifying the set of valid values with a minimum value and a maximum value both of which may be included or excluded. The `shape` argument is used for specifying the shape with the entry, -1 allowing an arbitrary shape

for a given dimension and any non-negative entry giving a fixed shape for a given dimension.

The remaining arguments of `validate_numeric` are not directly related to the validation scheme but rather to the surrounding Python code. The `precision` argument is used for specifying one or more allowed precisions in terms of bits per value. The `name` argument is used for specifying which argument of the function to validate with the particular validation call. The `ignore_none` argument is a flag indicating if the validation call should ignore `None` objects and thereby accept them as valid. The `var` argument is irrelevant to the scope of the present paper and the reader is referred to the documentation for more information.

Additional resources for `magni` are:

- Official releases: doi:10.5278/VBN/MISC/Magni
- Online documentation: http://magni.readthedocs.io
- GitHub repository: https://github.com/SIP-AAU/Magni

Examples

As mentioned in relation to the suggested validation schemes, there should be an application-driven data type representing the most general numerical value being able to assume any numerical value of any shape. The following example validates a variable against exactly this application-driven data type. The validation only fails when a non-numerical object is passed as argument to `func`.

```
from magni.utils.validation import decorate_validation
from magni.utils.validation import validate_numeric
import numpy as np

def func(var):
    @decorate_validation
    def validate_input():
        all_types = ('boolean', 'integer',
                     'floating', 'complex')
        validate_numeric(
            'var', all_types, shape=None)

    validate_input()

    pass  # the body of the func
```

When valid values are passed, nothing happens:

```
>>> func(42)
>>> func(3.14)
>>> func(np.empty((5, 5), dtype=np.complex_))
```

However, when a non-numerical object is passed, the following exception occurs:

```
>>> func('string')
TypeError: The value(s) of >>var<<, 'string', must \
be numeric.
```

In the next example, the application-driven data type is any non-negative real scalar, i.e., $\mathbb{R}_{\geq 0}$.

```
from magni.utils.validation import decorate_validation
from magni.utils.validation import validate_numeric

def func(var):
    @decorate_validation
    def validate_input():
        real = ('integer', 'floating')
        validate_numeric(
            'var', real, range_='[0;inf]')

    validate_input()

    pass  # the body of the func
```

When valid values are passed, nothing happens:

```
>>> func(0)
>>> func(3.14)
```

However, when a complex object or a negative float is passed, the following exception occurs:

```
>>> func(1j)
TypeError: The value(s) of >>var.dtype<<, \
<type 'complex'>, must be in ('integer', 'floating').

>>> func(-3.14)
ValueError: The value(s) of >>min(real(var))<<, \
-3.14, must be >= 0.
```

Notice, that the `range_` argument in the validation call of the previous includes the values zero and infinity using `[...]`. One or both of these values could be excluded using `(...)` or `]...[` as is the case in the next example, i.e., $\mathbb{R}_{>0}$.

```
from magni.utils.validation import decorate_validation
from magni.utils.validation import validate_numeric

def func(var):
    @decorate_validation
    def validate_input():
        real = ('integer', 'floating')
        validate_numeric(
            'var', real, range_='(0;inf)')

    validate_input()

    pass  # the body of the func
```

When a valid value is passed, nothing happens:

```
>>> func(3.14)
```

However, when a zero-valued object is passed, the following exception occurs:

```
>>> func(0.)
ValueError: The value(s) of >>min(real(var))<<, \
0.0, must be > 0.
```

In the final example, the application-driven data type is any real matrix with its first dimension equal to 5, i.e. $\mathbb{R}^{5\times n}$ for any non-negative integer n.

```
from magni.utils.validation import decorate_validation
from magni.utils.validation import validate_numeric
import numpy as np

def func(var):
    @decorate_validation
    def validate_input():
        real = ('integer', 'floating')
        validate_numeric(
            'var', real, shape=(5, -1))

    validate_input()

    pass  # the body of the func
```

When a valid value is passed, nothing happens:

```
>>> func(np.empty((5, 5)))
>>> func(np.empty((5, 10)))
```

However, when an $\mathbb{R}^{10\times 5}$ object or an $\mathbb{R}^{5\times 5\times 5}$ object is passed, the following exception occurs:

```
>>> func(np.empty((10, 5)))
ValueError: The value(s) of>>var.shape[0]<<, 10, \
must be 5.

>>> func(np.empty((5, 5, 5)))
ValueError: The value(s) of >>len(var.shape)<<, 3, \
must be 2.
```

Requirements

The required dependencies for `magni` (as of version 1.4.0) are:

- Python >= 2.7 / 3.3
- Matplotlib [Hun07] (Tested on version >= 1.3)
- NumPy [vdWCV11] (Tested on version >= 1.8)
- PyTables[15] (Tested on version >= 3.1)
- SciPy [Oli07] (Tested on version >= 0.14)

It should be noted that the requirements other than Python and NumPy are due to `magni` rather than `magni.utils.validation`. In addition to the above requirements, `magni` has a number of optional dependencies but none of these are relevant to the usage of `magni.utils.validation`.

Quality Assurance

The Magni Python package has been developed according to best practices for developing scientific software [WAB+14], and every included piece of code has been reviewed by at least one person other than its author. Furthermore, the PEP 8[16] style guide is adhered to, no function has a cyclomatic complexity [McC76] exceeding 10, the code is fully documented, and an extensive test suite accompanies the package. More details about the quality assurance of `magni` is given in [OPA+14].

Conclusions

We have argued that function arguments should be validated according to data types at a higher abstraction level than actual Python types, and we have named these application-driven data types. Based on a discussion of existing validation solutions, we have suggested a Python validation strategy including three aspects: 1) The validation schemes that can be expressed. 2) The way the interface of the implementation allows the validation scheme to be specified. 3) The Python constructs used to allow Python to validate the function arguments. A reference implementation of this strategy is available in the open source Magni Python package which we have presented along with a number of examples. In short, `magni` and more generally the validation strategy should be used to abstract function argument validation from Python to signal processing, to make validation ease to write, and to enhance readability of validation.

Acknowledgements

This work was supported in part by the Danish Council for Independent Research (DFF/FTP) under Project 1335-00278B/12-134971 and in part by the Danish e-Infrastructure Cooperation (DeIC) under Project DeIC2013.12.23.

References

[CVS13] José Cordeiro, Maria Virvou, and Boris Shishkov. *Software and Data Technologies: 5th International Conference, ICSOFT 2010, Athens, Greece, July 22-24, 2010. Revised Selected Papers*. Springer, 2013.

[Hun07] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, May 2007. doi:10.1109/MCSE.2007.55.

[McC76] Thomas J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, December 1976. doi:10.1109/TSE.1976.233837.

[Oli07] Travis E. Oliphant. Python for Scientific Computing. *Computing in Science & Engineering*, 9(3):10–20, May 2007. doi:10.1109/MCSE.2007.58.

[OPA+14] Christian Schou Oxvig, Patrick Steffen Pedersen, Thomas Arildsen, Jan Østergaard, and Torben Larsen. Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images. *Journal of Open Research Software*, 2(1):e29, October 2014. doi:10.5334/jors.bk.

[vdWCV11] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2):22–30, March 2011. doi:10.1109/MCSE.2011.37.

[WAB+14] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best Practices for Scientific Computing. *PloS Biology*, 12(1):e1001745, January 2014. doi:10.1371/journal.pbio.1001745.

15. See http://www.pytables.org/
16. See https://www.python.org/dev/peps/pep-0008/
17. See https://travis-ci.org/

Paper F

Storing Reproducible Results from Computational Experiments using Scientific Python Packages

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen

The paper has been published in
Proceedings of the 15th Python in Science Conference (SciPy), pp. 45–50,
Austin, Texas, USA, Jul. 11 – 17, 2016.
http://conference.scipy.org/proceedings/scipy2016/christian_oxvig.html

Storing Reproducible Results from Computational Experiments using Scientific Python Packages

Christian Schou Oxvig[‡*], Thomas Arildsen[‡], Torben Larsen[‡]

Abstract—Computational methods have become a prime branch of modern science. Unfortunately, retractions of papers in high-ranked journals due to erroneous computations as well as a general lack of reproducibility of results have led to a so-called credibility crisis. The answer from the scientific community has been an increased focus on implementing reproducible research in the computational sciences. Researchers and scientists have addressed this increasingly important problem by proposing best practices as well as making available tools for aiding in implementing them. We discuss and give an example of how to implement such best practices using scientific Python packages. Our focus is on how to store the relevant metadata along with the results of a computational experiment. We propose the use of JSON and the HDF5 database and detail a reference implementation in the Magni Python package. Further, we discuss the focuses and purposes of the broad range of available tools for making scientific computations reproducible. We pinpoint the particular use cases that we believe are better solved by storing metadata along with results the same HDF5 database. Storing metadata along with results is important in implementing reproducible research and it is readily achievable using scientific Python packages.

Index Terms—Reproducibility, Computational Science, HDF5

Introduction

Exactly how did I produce the computational results stored in this file? Most data scientists and researchers have probably asked this question at some point. For one to be able to answer the question, it is of utmost importance to track the provenance of the computational results by making the computational experiment reproducible, i.e. describing the experiment in such detail that it is possible for others to independently repeat it [LMS12], [Hin14]. Unfortunately, retractions of papers in high-ranked journals due to erroneous computations [Mil06] as well as a general lack of reproducibility of computational results [Mer10], with some studies showing that only around 10% of computational results are reproducible [BE12], [RGPN+11], have led to a so-call *credibility crisis* in the computational sciences.

The answer has been a demand for requiring research to be reproducible [Pen11]. The scientific community has acknowledged that many computational experiments have become so complex that more than a textual presentation in a paper or a technical report is needed to fully detail it. Enough information to make the experiment reproducible must be included with the textual presentation [RGPN+11], [CG12], [SLP14]. Consequently, reproducibility of computational results have become a requirement for submission to many high-ranked journals [Edi11], [LMS12].

But how does one make computational experiments reproducible? Several communities have proposed best practices, rules, and tools to help in making results reproducible, see e.g. [VKV09], [SNTH13], [SM14], [Dav12], [SLP14]. Still, this is an area of active research with methods and tools constantly evolving and maturing. Thus, the adoption of the reproducible research paradigm in most scientific communities is still ongoing - and will be for some time. However, a clear description of how the reproducible research paradigm fits in with customary workflows in a scientific community may help speed up the adoption of it. Furthermore, if tools that aid in making results reproducible for such customary workflows are made available, they may act as an additional catalyst.

In the present study, we focus on giving guidelines for integrating the reproducible research paradigm in the typical scientific Python workflow. In particular, we propose an easy to use scheme for storing metadata along with results in an HDF5 database. We show that it is possible to use Python to adhere to best practices for making computational experiments reproducible by storing metadata as JSON serialized arrays along with the results in an HDF5 database. A reference implementation of our proposed solution is part of the open source Magni Python package.

The remainder of this paper is organized as follows. We first describe our focus and its relation to a more general data management problem. We then outline the desired workflow for making scientific Python experiments reproducible and briefly review the fitness of existing reproducibility aiding tools for this workflow. This is continued by a description of our proposed scheme for storing metadata along with results. Following this specification, we detail a reference implementation of it and give plenty examples of its use. The paper ends with a more general discussion of related reproducibility aiding software packages followed by our conclusions.

The Data Management Problem

Reproducibility of computational results may be considered a part of a more general problem of data management in a computational study. In particular, it is closely related to the data management tasks of documenting and describing data. A typical computational study involves testing several combinations of various elements, e.g. input data, hardware platforms, external software libraries,

* Corresponding author: cso@es.aau.dk
‡ Faculty of Engineering and Science, Department of Electronic Systems, Aalborg University, 9220 Aalborg, Denmark

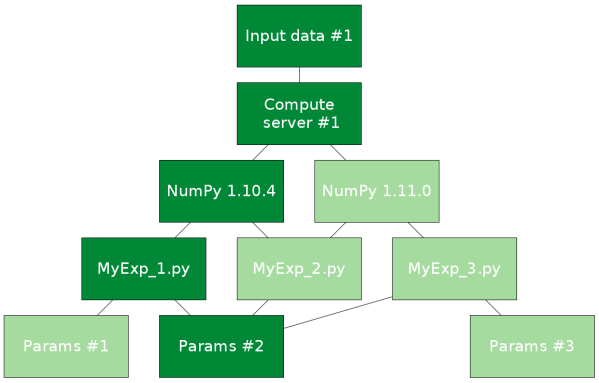


Fig. 1: *Illustration of a typical data management description problem as a layered graph. In this exemplified experiment, several combinations of input data, hardware platforms, software libraries (e.g. NumPy), algorithmic/experimental setup (described in a Python script), and parameter values are tested. The challenging task is to keep track of both the full set of combinations tested (marked by all the edges in the graph) as well as the individual simulations (e.g. the combination of highlighted vertices).*

experiment specific code, and model parameter values. Such a study may be illustrated as a layered graph like the one shown in figure 1. Each layer corresponds to one of the elements, e.g. the version of the NumPy library or the set of parameter values. The edges in the graph mark all the combinations that are tested. An example of a combination that constitutes a single simulation or experiment is the set of connected vertices that are highlighted in the graph in figure 1. In the present study, we focus on the problem of documenting and describing such a single simulation. A closely related problem is that of keeping track of all tested combinations, i.e. the set of all paths through all layers in the graph in figure 1. This is definitely also an interesting and important problem. However, once the "single simulation" problem is solved, it should be straight forward to solve the "all combinations" problem by appropriately combining the information from all the single simulations.

Storing Metadata Along With Results

For our treatment of reproducibility of computational results, we adopt the meaning of reproducibility from [LMS12], [Hin14]. That is, *reproducibility* of a study is the ability of others to repeat the study and obtain the same results using a general description of the original work. The related term *replicability* then means the ability of others to repeat the study and obtain the same results using the exact same setup (code, hardware, etc.) as in the original work[1]. As pointed out in [Hin14], reproducibility generally requires replicability.

The lack of reproducibility of computational results is oftentimes attributed to missing information about critical computational details such as library versions, parameter values, or precise descriptions of the exact code that was run [LMS12], [BPG05], [RGPN+11], [Mer10]. Several studies have given best practices for how to detail such metadata to make computational results reproducible, see e.g. [VKV09], [SNTH13], [SM14], [Dav12]. Here we detail the desired workflow for storing such metadata along with results when using a typical scientific Python workflow in the computational experiments. That is, we detail how to document a single experiment as illustrated by the highlighted vertices in figure 1.

The Scientific Python Workflow

In a typical scientific Python workflow, we define an experiment in a Python script and run that script using the Python interpreter, e.g.

```
$ python my_experiment.py
```

The content of the `my_experiment.py` script would typically have a structure like:

```
import some_library
import some_other_library

def some_func(...):
    ...

def run_my_experiment(...):
    ...

if __name__ == '__main__':
    run_my_experiment(...)
```

This is a particularly generic setup that only requires the availability of the Python interpreter and the libraries imported in the script. We argue that for the best practices for detailing a computational study to see broad adoption by the scientific Python community, three elements are of critical importance: Any method or tool for storing the necessary metadata to make the results reproducible must

1. be very easy to use and integrate well with existing scientific Python workflows.
2. be of high quality to be as trustworthy as the other tools in the scientific Python stack.
3. store the metadata in an open format that is easily inspected using standard viewers as well as programmatically from Python.

These elements are some of the essentials that have made Python so popular in the scientific community[2]. Thus, for storing the necessary metadata, we seek a high quality solution which integrates well with the above exemplified workflow. Furthermore, the metadata must be stored in such a way that is is easy to extract and inspect when needed.

Existing Tools

Several tools for keeping track of provenance and aiding in adhering to best practices for reproducible research already exist, e.g. Sumatra [Dav12], ActivePapers [Hin15], or Madagascar [Fom15]. Tools like Sumatra, ActivePapers, and Madagascar generally function as *reproducibility frameworks*. That is, when used with Python, they wrap the standard Python interpreter with a framework that in addition to running a Python script (using the standard Python interpreter) also captures and stores metadata detailing the setup used to run the experiment. E.g. when using Sumatra, one would replace `python my_experiment.py` with [Dav12]

```
$ smt run -e python -m my_experiment.py
```

1. Some authors (e.g. [SLP14]) swap the meaning of *reproducibility* and *replicability* compared to the convention, we have adopted.

2. See http://cyrille.rossant.net/why-using-python-for-scientific-computing/ for an overview of the main arguments for using Python for scientific computing.

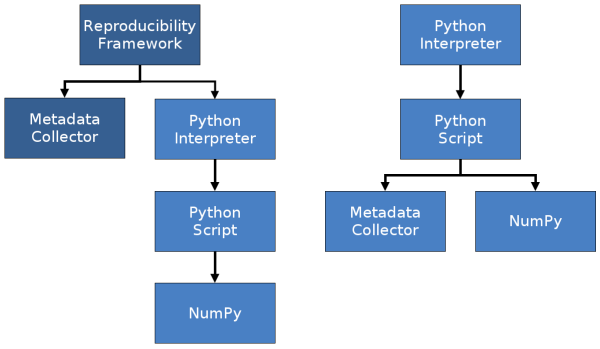


Fig. 2: *Illustration of the difference between a full reproducibility framework (on the left) and an importable Python library (on the right). The reproducibility framework calls the metadata collector as well as the Python interpreter which in turn runs the Python simulation script which e.g. imports NumPy. When using an importable library, the metadata collector is imported in the Python script alongside with e.g. NumPy.*

This idea of wrapping a computational simulation is different from the usual scientific Python workflow which consists of running a Python script that imports other packages and modules as needed, e.g. importing NumPy for numerical computations. This difference is illustrated in figure 2.

We argue that an importable Python library for aiding in making results reproducible has several advantages compared to using a full blown reproducibility framework. A major element in using any tool for computational experiments is being able to trust that the tool does what it is expected do. The scientific community trusts Python and the SciPy stack. For a reproducibility framework to be adopted by the community, it must build trust as the wrapper of the Python interpreter, it effectively is. That is, one must trust that it handles experiment details such as input parameters, library paths, etc. just as accurately as the Python interpreter would have done. Furthermore, such a framework must be able to fully replace the Python interpreter in all existing workflows which uses the Python interpreter. A traditional imported Python library does not have these potentially staggering challenges to overcome in order to see wide adoption. It must only build trust among its users in the same way as any other scientific library. Furthermore, it would be easy to incorporate into any existing workflow. Thus, ideally we seek a solution that allow us to update our `my_experiment.py` to have a structure like:

```
import some_library
import some_other_library
import reproducibility_library

def some_func(...):
    ...

def run_my_experiment(...):
    ...

if __name__ == '__main__':
    reproducibility_library.store_metadata(...)
    run_my_experiment(...)
```

Interestingly, the authors of the Sumatra package has to some degree pursued this idea by offering an API for importing the library as an alternative to using the `smt run` command line tool.

Equally important, to how to obtain the results, is how to inspect the results afterwards. Thus, one may ask: *How are the results and the metadata stored, and how may they be accessed later on?* For example, Sumatra by default stores all metadata in a SQLite database [Dav12] separate from simulation results (which may be stored in any format) whereas ActivePapers stores the metadata along with the results in an HDF5 database [Hin15]. The idea of storing (or "caching") intermediate results and metadata along with the final results has also been pursued in another study [PE09].

We argue that this idea of storing metadata along with results is an excellent solution. Having everything compiled into one standardized and open file format helps keep track of all the individual elements and makes it easy to share the full computational experiment including results and metadata. Preferably, such a file format should be easy to inspect using a standard viewer on any platform; just like the Portable Document Format (PDF) has made it easy to share and inspect textual works across platforms. The HDF5 Hierarchical Data Format [FP10] is a great candidate for such a file format due to the availability of cross-platform viewers like HDFView[3] and HDFCompass[4] as well as its capabilities in terms of storing large datasets. Furthermore, HDF5 is recognized in the scientific Python community[5] with bindings available through e.g. PyTables[6], h5py[7], or Pandas [McK10]. Also, bindings for HDF5 exists in several other major programming languages.

Suggested Library Design

Our above analysis reveals that all elements needed for implementing the reproducible research paradigm in scientific Python are in fact already available in existing reproducibility aiding tools: Sumatra may serve as a Python importable library and the ActivePapers project shows how metadata may be stored along with results in an HDF5 database. However, no single tool offers all of these elements for the scientific Python workflow. Consequently, we propose creating a scientific Python package that may be imported in existing scientific Python scripts and may be used to store all relevant metadata for a computational experiment along with the results of that experiment in an HDF5 database.

Technically, there are various ways to store metadata along with results in an HDF5 database. The probably most obvious way is to store the metadata as attributes to HDF5 tables and arrays containing the results. However, this approach is only recommended for small metadata (generally < 64KB)[8]. For larger metadata it is recommended to use a separate HDF5 array or table for storing the metadata[9]. Thus, for the highest flexibility, we propose to store the metadata as separate HDF5 arrays. This also allows for separation of specific result arrays or tables and general metadata. When using separate metadata arrays, a serialization (a representation) of the metadata must be chosen. For the metadata to be humanly readable using common HDF viewers, it must be stored in an easily readable string representation. We suggest using JSON [ECM13] for serializing the metadata. This makes for a humanly readable representation. Furthermore, JSON is a standard format with bindings for most major programming languages[10].

3. See https://www.hdfgroup.org/products/java/hdfview/
4. See https://github.com/HDFGroup/hdf-compass
5. See https://www.youtube.com/watch?v=nddj5OA8LJo
6. See http://www.pytables.org/
7. See http://www.h5py.org/

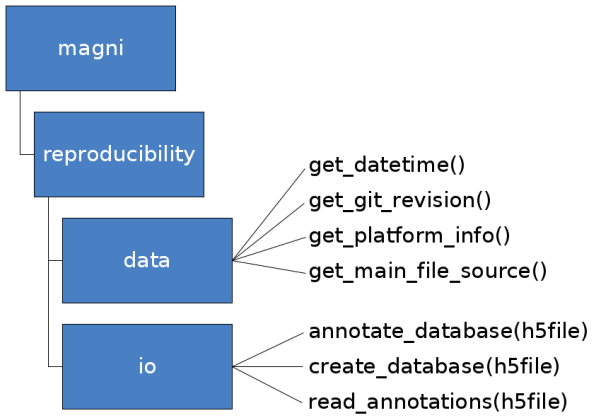


Fig. 3: *Illustration of the structure of the* `magni.reproducibility` *subpackage of Magni. The main modules are the* `data` *module for acquiring metadata and the* `io` *module for interfacing with an HDF5 database when storing as well as reading the metadata. A subset of available functions are listed next to the modules.*

In particular, Python bindings are part of the standard library (introduced in Python 2.6)[11]. This would effectively make Python >=2.6 and an HDF5 Python interface the only dependencies of our proposed reproducibility aiding library. We note, though, that the choice of JSON is not crucial. Other formats similar to JSON (e.g. XML[12] or YAML[13]) may be used as well. We do argue, though, that a humanly readable format should be used such that the metadata may be inspected using any standard HDF5 viewer.

Magni Reference Implementation

A reference implementation of the above suggested library design is available in the open source Magni Python package [OPA+14]. In particular, the subpackage `magni.reproducibility` is based on this suggested design. Figure 3 gives an overview of the `magni.reproducibility` subpackage. Additional resources for `magni` are:

- Official releases: doi:10.5278/VBN/MISC/Magni
- Online documentation: http://magni.readthedocs.io
- GitHub repository: https://github.com/SIP-AAU/Magni

In `magni.reproducibility`, a differentiation is made between *annotations* and *chases*. *Annotations* are metadata that describe the setup used for the computation, e.g. the computational environment, values of input parameters, platform (hardware/OS) details, and when the computation was done. *Chases* on the other hand are metadata describing the specific code that was used in the computation and how it was called, i.e. they *chase* the provenance of the results.

8. See http://docs.h5py.org/en/latest/high/attr.html
9. See https://www.hdfgroup.org/HDF5/doc1.6/UG/13_Attributes.html
10. See http://www.json.org/
11. See https://docs.python.org/2/library/json.html
12. See https://www.w3.org/TR/REC-xml/
13. See http://yaml.org/

Requirements

Magni uses PyTables as its interface to HDF5 databases. Thus, had `magni.reproducibility` been a package of its own, only Python and PyTables would have been requirements for its use. The full requirements for using `magni` (as of version 1.5.0) are[14]

- Python >= 2.7 / 3.3
- Matplotlib [Hun07] (Tested on version >= 1.3)
- NumPy [vdWCV11] (Tested on version >= 1.8)
- PyTables[15] (Tested on version >= 3.1)
- SciPy [Oli07] (Tested on version >= 0.14)
- Setuptools[16] (Tested on version >= 11.3)

When using the Conda[17] package management system for handling the Python environment used in the computation, `magni.reproducibility` may optionally use Conda to capture details about the Python environment. Thus, we have one optional dependency

- Conda (Tested on version >= 3.7.0)

Usage Examples

We now give several smaller examples of how to use `magni.reproducibility` to implement the best practices for reproducibility of computational result described in [VKV09], [SNTH13], [SM14]. An extensive example of the usage of `magni.reproducibility` is available at doi:10.5278/VBN/MISC/MagniRE. This extensive example is based on a Python script used to simulate the Mandelbrot set[18] using the scientific Python workflow described above. An example of a resulting HDF5 database containing both the Mandelbrot simulation result and metadata is also included. Finally, the example includes a Jupyter Notebook showing how to read the metadata using `magni.reproducibility`.

A simple example of how to acquire platform metadata using the `data` module from `magni.reproducibility` is

```
>>> from pprint import pprint
>>> from magni import reproducibility as rep
>>> pprint(rep.data.get_platform_info())
{'libc': '["glibc", "2.2.5"]',
 'linux': '["debian", "jessie/sid", ""]',
 'mac_os': '["", ["", "", ""], ""]',
 'machine': '"x86_64"',
 'node': '"eagle1"',
 'processor': '"x86_64"',
 'python': '"3.5.1"',
 'release': '"3.16.0-46-generic"',
 'status': 'All OK',
 'system': '"Linux"',
 'version': '"#62~14.04.1-Ubuntu SMP ~"',
 'win32': '["", "", "", ""]'}
```

When using the typical scientific Python workflow described above, one may use the functions in the `io` module from `magni.reproducibility` to conveniently store all relevant metadata, e.g. the `create_database(h5file)` to automatically create an HDF5 database with a set of standard annotations

14. More details about Python and the Scientific Python Stack are available at http://python.org and http://scipy.org
15. See http://www.pytables.org/
16. See http://setuptools.readthedocs.io/
17. See http://conda.pydata.org/docs/ as well as https://www.youtube.com/watch?v=UaIvrDWrIWM
18. See https://en.wikipedia.org/wiki/Mandelbrot_set

and chases. The `my_experiment.py` script would then have a structure like

```
import tables
from magni import reproducibility as rep

def run_my_experiment(...):
    ...

def store_result(h5, result):
    ...

if __name__ == '__main__':
    hdf5_db = 'database.hdf5'
    rep.io.create_database(hdf5_db)
    result = run_my_experiment(...)
    with tables.File(hdf5_db, mode='a') as h5:
        store_result(h5, result)
```

This would create an HDF5 database named `database.hdf5` which would hold both the results and all metadata. The HDF5 database may be inspected using any tool capable of reading HDF5 files. As an alternative, the `io` module from `magni.reproducibility` also includes convenience functions for reading the annotations and chases. E.g. to see the set of standard metadata stored in a database with `create_database(h5file)`, one could do

```
>>> from pprint import pprint
>>> import tables
>>> from magni import reproducibility as rep
>>> hdf5_db = 'database.hdf5'
>>> rep.io.create_database(hdf5_db)
>>> with tables.File(hdf5_db) as h5:
...     annotations = rep.io.read_annotations(h5)
...     chases = rep.io.read_chases(h5)
>>> pprint(list(annotations.keys()))
['magni_config',
 'git_revision',
 'datetime',
 'conda_info',
 'magni_info',
 'platform_info']
>>> pprint(list(chases.keys()))
['main_file_source',
 'stack_trace',
 'main_file_name',
 'main_source']
```

Quality Assurance

The Magni Python package is fully documented and comes with an extensive test suite. It has been developed using best practices for developing scientific software [WAB+14] and all code has been reviewed by at least one other person than its author prior to its inclusion in Magni. All code adheres to the PEP8[19] style guide and no function or class has a cyclomatic complexity [McC76], [WM96] exceeding 10. The source code is under version control using Git and a continuous integration system based on Travis CI[20] is in use for the git repository. More details about the quality assurance of `magni` are given in [OPA+14].

Related Software Packages

Independently of the tool or method used, making results from scientific computations reproducible is not only for the benefit of the audience. As pointed out in several studies [Fom15], [CG12], [VKV09], the author of the results gains as least as much in terms increasing one's productivity. Thus, using some method or tool to help make the results reproducible is a win for everyone. In the present work we have attempted to detail the ideal solution for how to do this for the typical scientific Python workflow.

A plethora of related alternative tools exist for aiding in making results reproducible. We have already discussed ActivePapers [Hin15], Sumatra [Dav12], and Madagascar [Fom15] which are general reproducibility frameworks that allow for wrapping most tools - not only Python based computations. Such tools are definitely excellent for some workflows. In particular, they seem fit for large fixed setups which require keeping track of several hundred runs that only differ by the selection of parameters[21] and for which the time cost of initially setting up the tool is insignificant compared to the time cost of the entire study. That is, they are useful in keeping track of the full set of combination in a large computations study as marked by all the edges in the layered graph in figure 1. However, as we have argued, they are less suitable for documenting a single experiment based on the typical scientific Python workflow. Also these tools tend to be designed for use on a single computer. Thus, they do not scale well for big data applications which run on compute clusters.

Another category of related tools are graphical user interface (GUI) based workflow managing tools like Taverna [OAF+04] or Vistrail [SFC07]. Such tools seem to be specifically designed for describing computational workflows in particular fields of research (typically bioinformatics related fields). It is hard, though, to see how they can be effectively integrated with the typical scientific Python workflow. Other much more Python oriented tools are the Jupyter Notebook[22] as well as Dexy[23]. These tools, however, seem to have more of a focus on implementing the concept of literate programming and documentation than reproducibility of results in general.

Conclusions

We have argued that metadata should be stored along with computational results in an easily readable format in order to make the results reproducible. When implementing this in a typical scientific Python workflow, all necessary tools for making the results reproducible should be available as an importable package. We suggest storing the metadata as JSON serialized arrays along with the result in an HDF5 database. A reference implementation of this design is available in the open source Magni Python package which we have detailed with several examples of its use. All of this shows that storing metadata along with results is important in implementing reproducible research and it is readily achievable using scientific Python packages.

Acknowledgements

This work was supported in part by the Danish Council for Independent Research (DFF/FTP) under Project 1335-00278B/12-134971 and in part by the Danish e-Infrastructure Cooperation (DeIC) under Project DeIC2013.12.23.

References

[BE12] C. Glenn Begley and Lee M. Ellis. Drug development: Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533, March 2012. doi:10.1038/483531a.

19. See https://www.python.org/dev/peps/pep-0008/
20. See https://travis-ci.org/
21. See e.g. https://www.youtube.com/watch?v=1YJr9c-zSng
22. See http://jupyter.org/
23. See http://www.dexy.it/ as well as https://www.youtube.com/watch?v=u6_qtDJ6ciA / https://www.youtube.com/watch?v=qFd04rA8lp0

[BPG05] Mauro Barni and Fernando Perez-Gonzalez. Pushing Science into Signal Processing. *IEEE Signal Processing Magazine*, 22(4):120–119, July 2005. doi:10.1109/MSP.2005.1458324.

[CG12] Kingshuk Roy Choudhury and Ray Gibson. Editorial: Reproducible Research in Medical Imaging. *Molecular Imaging and Biology*, 14(4):395–396, June 2012. doi:10.1007/s11307-012-0569-8.

[Dav12] Andrew P. Davison. Automated Capture of Experiment Context for Easier Reproducibility in Computational Research. *Computing in Science & Engineering*, 14(4):48–56, July 2012. doi:10.1109/MCSE.2012.41.

[ECM13] The JSON Data Interchange Format, October 2013.

[Edi11] Editorial. Devil in the details. *Nature*, 470(7334):305–306, February 2011. doi:10.1038/470305b.

[Fom15] Sergey Fomel. Reproducible Research as a Community Effort: Lessons from the Madagascar Project. *Computing in Science & Engineering*, 17(1):20–26, January 2015. doi:10.1109/MCSE.2014.94.

[FP10] Mike Folk and Elena Pourmal. Balancing Performance and Preservation Lessons learned with HDF5. In *Digital Preservation Interoperability Framework (DPIF) Workshop*, Gaithersburg, Maryland, USA, March 29 – 31, 2010. doi:10.1145/2039274.2039285.

[Hin14] Konrad Hinsen. Computational science: shifting the focus from tools to models. *F1000Research*, 3(101):1–16, June 2014. doi:10.12688/f1000research.3978.2.

[Hin15] Konrad Hinsen. ActivePapers: a platform for publishing and archiving computer-aided research. *F1000Research*, 3(289):14, July 2015. doi:10.12688/f1000research.5773.3.

[Hun07] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, May 2007. doi:10.1109/MCSE.2007.55.

[LMS12] Randall J. LeVeque, Ian M. Mitchell, and Victoria Stodden. Reproducible Research for Scientific Computing: Tools and Strategies for Changing the Culture. *Computing in Science & Engineering*, 14(4):13–17, July 2012. doi:10.1109/MCSE.2012.38.

[McC76] Thomas J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, December 1976. doi:10.1109/TSE.1976.233837.

[McK10] Wes McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 51–56, Austin, Texas, USA, June 28 – July 3, 2010.

[Mer10] Zeeya Merali. Computational science: ...Error ...why scientific programming does not compute. *Nature*, 467:775–777, October 2010. doi:10.1038/467775a.

[Mil06] Greg Miller. A Scientist's Nightmare: Software Problem Leads to Five Retractions. *Science*, 314(5807):1856–1857, December 2006. doi:10.1126/science.314.5807.1856.

[OAF+04] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, June 2004. doi:10.1093/bioinformatics/bth361.

[Oli07] Travis E. Oliphant. Python for Scientific Computing. *Computing in Science & Engineering*, 9(3):10–20, May 2007. doi:10.1109/MCSE.2007.58.

[OPA+14] Christian Schou Oxvig, Patrick Steffen Pedersen, Thomas Arildsen, Jan Østergaard, and Torben Larsen. Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images. *Journal of Open Research Software*, 2(1):e29, October 2014. doi:10.5334/jors.bk.

[PE09] Roger D. Peng and Sandrah P. Eckel. Distributed Reproducible Research Using Cached Computations. *Computing in Science & Engineering*, 11(1):28–34, January 2009. doi:10.1109/MCSE.2009.6.

[Pen11] Roger D. Peng. Reproducible Research in Computational Science. *Science*, 334(6060):1226–1227, December 2011. doi:10.1126/science.1213847.

[RGPN+11] Markus Rupp, Fulvio Gini, Ana Pérez-Neira, Beatrice Pesquet-Popescu, Aggelos Pikrakis, Bulent Sankur, Patrick Vandewalle, and Abdelhak Zoubir. Reproducible research in signal processing. *EURASIP Journal on Advances in Signal Processing*, 93(1):1–2, October 2011. doi:10.1186/1687-6180-2011-93.

[SFC07] Claudio T. Silva, Juliana Freire, and Steven P. Callahan. Provenance for Visualizations: Reproducibility and Beyond. *Computing in Science & Engineering*, 9(5):82–89, September 2007. doi:10.1109/MCSE.2007.106.

[SLP14] Victoria Stodden, Friedrich Leisch, and Roger D. Peng, editors. *Implementing Reproducible Research*. Chapman & Hall/CRC The R Series. CRC Press, 2014.

[SM14] Victoria Stodden and Sheila Miguez. Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research. *Journal of Open Research Software*, 2(1):1–6, July 2014. doi:10.5334/jors.ay.

[SNTH13] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10):e1003285, October 2013. doi:10.1371/journal.pcbi.1003285.

[vdWCV11] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2):22–30, March 2011. doi:10.1109/MCSE.2011.37.

[VKV09] P. Vandewalle, J. Kovačević, and M. Vetterli. Reproducible Research in Signal Processing [What, why, and how]. *IEEE Signal Processing Magazine*, 26(3):37–47, May 2009. doi:10.1109/MSP.2009.932122.

[WAB+14] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best Practices for Scientific Computing. *PloS Biology*, 12(1):e1001745, January 2014. doi:10.1371/journal.pbio.1001745.

[WM96] Arthur H. Watson and Thomas J. McCabe. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. Special Publication 500-235, National Institute of Standards and Technology (NIST), September 1996.

Tech Report G

Generalized Approximate Message Passing: Relations and Derivations

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen

This tech report is available at
doi:10.5278/VBN.GAMPTechReport

The succeeding 63 non-blank pages constitute the version 1.0 manuscript which is subject to the following copyright notice:

Generalized Approximate Message Passing
Relations and Derivations

Tech Report

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen

Department of Electronic Systems
Aalborg University
Denmark

April 6, 2017

Generalized Approximate Message Passing: Relations and Derivations
Tech Report
Version: 1.00, April 6, 2017
doi:10.5278/VBN.GAMPTechReport

Christian Schou Oxvig, Thomas Arildsen, Torben Larsen
Department of Electronic Systems, Aalborg University, Denmark

Author Contributions

Christian Schou Oxvig defined the research which led to the novel contributions presented in this tech report and was the main contributor to the proposed solutions. He defined and detailed the outline of the GAMP elements to include in the review parts of the tech report. Finally, he wrote the entire draft of the tech report with the exception of Section 3.3 "MMSE Channel Functions in General" and revised the draft according to comments from co-authors.

Thomas Arildsen and **Torben Larsen** supervised the work on the novel contributions presented in this tech report. They contributed several ideas that were used in the proposed solutions. They took part in several discussions of the outline of the review parts of the tech report. Finally, these authors performed several reviews of drafts of the tech report and proposed several changes that were used in the final version of the tech report. Torben Larsen wrote Section 3.3 "MMSE Channel Functions in General".

Contents

1 Introduction

This tech report details a collection of results related to the Generalised Approximate Message Passing (GAMP) [1] algorithm. It is a summary of the results that the authors have found critical in understanding the GAMP algorithm. In particular, emphasis is on the details that are crucial in implementing the GAMP algorithm on a computer but which are oftentimes left out from the literature focusing on the more theoretical aspects of GAMP. Thus, this tech report is not meant to comprehensively cover all the published works on GAMP and related algorithms.

The Generalised Approximate Message Passing (GAMP) [1] algorithm by Rangan is a generalisation of Approximate Message Passing (AMP) algorithm, independently described by Donoho et al. [2], [3], [4] and Krzakala et al. [5], [6]. The generalisation allows for arbitrary output channels to be used with AMP.

1.1 Contributions Overview

The primary focus of this tech report is on giving more elaborate derivations and discussions of GAMP key relations found in the existing literature - with an emphasis on implementation details. However, a few new results are presented. Specifically, our new contributions are:

1. The proposed General Weighted Sparse (GWS) GAMP input channel described in Section 3.5 and all of its related derivations including the EM updates described in Sections 6.2.2.2 and 6.2.2.3 and the relations described in Sections 3.7.1.2 and 3.7.2.1.

2. The Sparse Bernoulli-Laplace input channel derivations given in Section 3.7.1 and the related EM update derivations given in Sections 6.2.2.4 and 6.3.4.

3. The methods for efficiently computing the Frobenius norm of various system matrices described in Section 4.2.2.

1.2 Background

We consider the undersampling reconstruction problem of estimating $\boldsymbol{\alpha} \in \mathbb{C}^{n\times 1}$ from the measurements $\mathbf{y} \in \mathbb{C}^{m\times 1}$ when $m \leq n$ (typically $m \ll n$) and $\mathbf{y} = \mathbf{A}\boldsymbol{\alpha} + \mathbf{e}$ with:

- $\mathbf{A} \in \mathbb{C}^{m\times n}$ - the system matrix
- $\mathbf{e} \in \mathbb{R}^{m\times 1}$ - an additive measurement noise

The system matrix $\mathbf{A}$ may be further decomposed into the matrix product of the sampling operator matrix $\boldsymbol{\Phi} \in \mathbb{R}^{m\times p}$ and the dictionary matrix $\boldsymbol{\Psi} \in \mathbb{C}^{p\times n}$, i.e. $\mathbf{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$. The dictionary matrix is chosen such that the coefficient vector $\boldsymbol{\alpha}$ has some sort of *structure*, e.g. $\boldsymbol{\alpha}$ is sparse. Introducing the *signal of interest*, $\mathbf{x} \in \mathbb{C}^{p\times 1}$, as well as the noiseless measurements, $\mathbf{z} \in \mathbb{C}^{m\times 1}$, the setup is described by the following set of equations:

$$\begin{aligned}
\mathbf{y} &= \mathbf{A}\boldsymbol{\alpha} + \mathbf{e} && (1.1)\\
&= \mathbf{z} + \mathbf{e} && (1.2)\\
&= \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{\alpha} + \mathbf{e} && (1.3)\\
&= \boldsymbol{\Phi}\mathbf{x} + \mathbf{e} && (1.4)\\
\mathbf{x} &= \boldsymbol{\Psi}\boldsymbol{\alpha} && (1.5)\\
\mathbf{z} &= \mathbf{A}\boldsymbol{\alpha} && (1.6)
\end{aligned}$$

1.3 Motivation

For probabilistic inference/recovery/reconstruction problems as those described in Section 1.2, one can generally observe a *phase transition* [7] (in the large system limit $n \to \infty$ for fixed $\delta = m/n$) separating the problems for which successful inference can be done from those for which it is impossible [5], [6]. This separation is determined by the available information about the inference problem. If too little information is available, it is generally not possible to solve the inference problem. The overall goal is then to find algorithms that are able to reach the phase transition boundary, i.e., they must be able to solve the inference problems when provided with just enough information to be able to succeed. The AMP algorithm is of interest in this regard primarily due to [5], [8]:

- It finds Bayes optimal (maximum a posterior (MAP) or minimum mean squared error (MMSE)) estimates in probabilistic inference/recovery/reconstruction problems and is (under certain conditions) able to reach the phase transition boundary.
- The number of messages in the underlying message passing problem that is being approximated scales linearly with the problem size. Thus, it is a computationally feasible algorithm.

See e.g. [5] or [8] for a more detailed motivation.

1.4 Derivation

Here we give a brief overview of a more heuristic derivation of the AMP following [5] and [9]. Rigorous proofs are given in [10], [11].

1.4.1 MMSE and MAP GAMP

We consider a Bayesian probabilistic approach to reconstructing a vector $\boldsymbol{\alpha}$ from measurements $\mathbf{y}$ in a setup described by the set of Equations (1.1)-(1.6). In particular, we consider the posterior distribution:

$$p(\boldsymbol{\alpha}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\mathbf{y})} \tag{1.7}$$

$$= \frac{1}{\mathcal{Z}} p(\mathbf{y}|\boldsymbol{\alpha})p(\boldsymbol{\alpha}) \tag{1.8}$$

for a normalisation constant $\mathcal{Z}$, the likelihood $p(\mathbf{y}|\boldsymbol{\alpha})$, and $p(\boldsymbol{\alpha})$ - a prior on $\boldsymbol{\alpha}$. As is usually the case in Bayesian methods, it all boils down to finding ways to handle the otherwise intractable integretions/summations needed to determine the value of such normalisation constants as well as marginals or expectations.

Even though the posterior marginals of $\boldsymbol{\alpha}$ may be of interest as such, one usually finds point estimates from these marginals. In particular, the minimum mean squared error (MMSE) or the maximum a posteriori (MAP) estimates are often used:

$$\alpha_j^{\text{MMSE}} = \int_{\alpha_j} \alpha_j p(\alpha_j|\mathbf{y}) d\alpha_j \tag{1.9}$$

$$\alpha_j^{\text{MAP}} = \arg\max_{\alpha_j} p(\alpha_j|\mathbf{y}) \tag{1.10}$$

where the (conditional) marginals are obtained as

$$p(\alpha_j|\mathbf{y}) = \int_{\{\alpha_{j'}\}:j'\neq j} p(\boldsymbol{\alpha}|\mathbf{y}) d\boldsymbol{\alpha} \tag{1.11}$$

Thus, we are interested in finding MMSE (or MAP) estimates of $\boldsymbol{\alpha}$ which entails the need for (at least indirectly) finding the marginals. In rest of this tech report, we generally focus on GAMP for finding MMSE estimates.

1.4.2 The Message Passing Interpretation

In general, the system matrix $\mathbf{A}$ defines a dense bipartite factor graph with variables $\alpha_1, \ldots, \alpha_n$ and factors $y_1, \ldots, y_m$ [9]. Had this factor graph contained no loops, it would have been possible to obtain exact inference of the marginals in Equation (1.11) (and MMSE estimates) in a single pass over the graph by use of the sum-product message passing algorithm (sometimes also known as belief propagation) [12], [13]. However, due to the loops in the factor graph, iterative message passing over the graph only results in approximate inference. In the loopy sum-product message passing, one iteratively passes the following messages (full probability distributions) along the edges of the factor graph:

$$m_{i\to j}(\alpha_j) = \frac{1}{\mathcal{Z}^{i\to j}} \int_{\{\alpha_k\}:k\neq j} p(y_i|z_i) \prod_{k\neq j} m_{k\to i}(\alpha_k) \qquad \text{factor to variable message} \tag{1.12}$$

$$m_{j\to i}(\alpha_j) = \frac{1}{\mathcal{Z}^{j\to i}} p(\alpha_j|[\boldsymbol{\theta}_I]_j) \prod_{l\neq i} m_{l\to j}(\alpha_j) \qquad \text{variable to factor message} \tag{1.13}$$

where the $z_i = [\mathbf{A}\alpha]_i$'s are the noiseless measurements, $[\boldsymbol{\theta}_I]_j$ are some parameters of the prior distribution on α_j and $\mathcal{Z}^{i\to j}$, $\mathcal{Z}^{j\to i}$ are normalisation factors. The use of this iterative message passing scheme poses two problems:

1. One has to track full probability distributions (the messages $m_{i\to j}(\alpha_j)$, $m_{j\to i}(\alpha_j)$ are full probability distributions on α_j, i.e. functions on the real axis).

2. There is a total of $2mn$ messages that must be passed in each iteration (all m factors sends a message to all n variables and vice versa).

Thus, it is intractable to use the above message passing scheme as is. However, under certain assumptions, it is possible to use a different message passing scheme involving only $m+n$ messages. Furthermore, these messages are means and variances, i.e, they are scalars which provides for a much more tractable algorithm.

The derivation of such a message passing scheme relies on the assumption that the individual element of $\mathbf{A}$ becomes insignificant for $n \to \infty$. That is, it is assumed that all elements of $\mathbf{A}$ scale like $\frac{1}{\sqrt{n}}$ such that each element seen in isolation becomes insignificant in the large system limit $n \to \infty$. In other words, the information is spread equally throughout the graph. The use of various Taylor approximations and applications of the central limit theorem then gives the resulting new message passing scheme. All the details are given in [2], [3], [4], [5], [6], [9], [14], [15].

Most importantly from an implementation perspective, the workload reduces to the iteration of the mean and variance updates $\bar{\alpha}_j$ and $\tilde{\alpha}_j$ (along with a few other states as detailed below) for finding the MMSE estimate[1] of α_j, $j = 1, \ldots, n$

$$\bar{\alpha}_j = f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) \tag{1.14}$$

$$\tilde{\alpha}_j = f_{\tilde{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) \tag{1.15}$$

for scalar non-linear functions $f_{\bar{\alpha}_j}$, $f_{\tilde{\alpha}_j}$. These updates may be iterated to a fixed point resulting in the reconstructed signal $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \ldots, \bar{\alpha}_n]^T$. The estimated variance of the elements of $\bar{\boldsymbol{\alpha}}$ is then given by $\tilde{\boldsymbol{\alpha}}$ [16]. This variance may be used to quantify the accuracy of the reconstructed signal (it should be small). In summary, $\bar{\alpha}_j$ ends up being an approximation of α_j^{MMSE} in Equation (1.9) with $\tilde{\alpha}_j$ expressing something about the quality of this approximation.

As hinted by the scalar functions in Equations (1.14) and (1.15), one introduces additional states that represent local beliefs about means and variances at the variables and factors:

s_j: Prior side (/ AMP field / variable field) variance

r_j: Prior side mean

v_i: Factor side (/ Channel side / factor field) variance

[1] The GAMP framework of Rangan [1], [16] allows for both MMSE and MAP estimates by choosing different channel functions.

o_i: Factor side mean

Note that there is one such state for each $j = 1, \ldots, n$, $i = 1, \ldots, m$. Thus, the number of states in the AMP algorithm that must be tracked and updated in each iteration is $\mathcal{O}(m+n)$. Only the compact (and simplified) AMP iteration by Donoho/Maleki/Montanari has exactly $m+n$ messages that must be tracked in each iteration [2] - see also Section 2.1. The full GAMP iteration is given in Section 2 (Equations (2.1) - (2.12)). The generalisation in GAMP introduces further states as well as a pair of scalar output side channel functions $f_{\bar{z}_i}$, $f_{\tilde{z}_i}$ in addition to the input side channel functions $f_{\bar{\alpha}_j}$, $f_{\tilde{\alpha}_j}$. Details about these channel functions are given in Section 3.

1.4.3 State Evolution

From a theoretical perspective, the probably most appealing part of the AMP algorithm is its State Evolution (SE) formalism [2], [4], [9] which provides precise convergence guarantees for AMP. It turns out that in the large system limit $n \to \infty$ and under certain conditions, a certain state (the unthresholded $\hat{\boldsymbol{\alpha}}$) in the AMP algorithm may be interpreted as a AWGN corrupted version of the true $\boldsymbol{\alpha}$ for any iteration. Thus, the reconstruction error may be tracked by the mean squared error (MSE) of the estimate. This is, however, a purely analytical construction that may be used to theoretically characterise the AMP algorithm in the large system limit for a given problem. A rigorous proof of the SE is given in [10].

1.4.4 Theoretical Guarantees for Arbitrary System Matrices

The AMP algorithm has been rigorously proved to converge for i.i.d. Gaussian system matrices (entries are drawn i.i.d. zero-mean Gaussian) in the large system limit $n \to \infty$ [17], [18] as well as for i.i.d. sub-Gaussian system matrices [10], [11]. For i.i.d. Gaussian system matrices of finite size, it can be shown that the probability of deviation from the SE described in Section 1.4.3 decreases exponentially in n [19].

For various other types of system matrices, a damping strategy may be used to guarantee convergence of the GAMP algorithms [20] (more details are given in Section 5.2). The S-AMP algorithm described in [21] and [22] is an attempt at generalising GAMP to more general system matrix ensembles. A similar attempt at an AMP algorithm for more general system matrix ensembles is the ADMM-GAMP algorithm [23]. Yet another attempt is the Orthogonal AMP [24] algorithm which is somewhat similar to the Vector AMP algorithm[2]. Generally, these alternatives have significantly higher computationally complexity than GAMP. Thus, they all present a trade-off between convergence guarantee and computational complexity.

The fixed points of GAMP with arbitrary matrices is discussed in [25] whereas general compressed sensing phase transitions for deterministic matrices are discussed in [26]. Details about when AMP algorithms provide theoretically optimal recovery guarantees are given in [6], [27]. Finally, empirical results suggest that GAMP also converges for various other system matrices [10], [28] including matrices related to structured random matrices [29] and structurally random matrices [30].

1.4.5 Additional Practicality Notes on GAMP

The AMP algorithm is based on a zero mean assumption on $\mathbf{A}$, i.e., the entries of $\mathbf{A}$ are assumed to be zero mean. For non-zero mean $\mathbf{A}$, one may use a transformation to a new problem with a zero mean system matrix, see e.g. [5] or [31].

In the derivation of AMP, Krzakala et al. assume a $\frac{1}{\sqrt{n}}$ scaling of the entries of $\mathbf{A}$ [6] whereas Montanari et. al assume a $\frac{1}{\sqrt{m}}$ with a fixed $\delta = \frac{m}{n}$ [9]. Theoretically, the two assumptions are equivalent (in the sense of an element becoming insignificant) for $n \to \infty$ as long as m scales linearly with n, i.e., δ is fixed. However, in practice (with finite n) for low δ, the difference between m and n is large enough to impact the convergence of the AMP algorithm[3].

[2] See the pre-print available at `https://arxiv.org/abs/1610.03082`

[3] Empirical phase transition simulations reported on in a submitted but yet to be published manuscript by the authors of this tech report confirms this observation. See also Chapter 4 for a more elaborate discussion of the impact of these assumptions.

The in- and output channels are required to be separable. That is, the channel defining probabilities must be conditionally independent [1]:

$$p(\boldsymbol{\alpha}|\boldsymbol{\theta}_I) = \prod_j p(\alpha_j|[\boldsymbol{\theta}_I]_j) \tag{1.16}$$

$$p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_o) = \prod_i p(y_i|z_i;[\boldsymbol{\theta}_o]_i) \tag{1.17}$$

Thus, we consider a setup in which we imagine that a random $\boldsymbol{\alpha}$ is generated according to the input channel specification in Equation (1.16). This $\boldsymbol{\alpha}$ is measured through the linear transform in Equation (1.6). The observation $\mathbf{y}$ is then generated from $\mathbf{z}$ according to the output channel specification in Equation (1.17) which is a generalisation of the additive noise used in Equation (1.1). That being said, one may consider the $\boldsymbol{\theta}_I$'s and $\boldsymbol{\theta}_o$'s to be random variables themselves and define hyperpriors on them. This allows for arbitrary structures (subject to the above separability constraint) on the prior of $\boldsymbol{\alpha}$, e.g. a Markov chain prior [32]. For updating the beliefs across such priors, one may use the Turbo GAMP framework [33]. See also Chapter 8 for more references to works on structured priors. Independently of the choice of prior, it is important to keep in mind that the GAMP estimates, when MMSE- or MAP-optimal, are optimal under the model assumption which may only approximate the problem attempted solved.

1.5 Notation

The notation used across publications by a given author is typically consistent. Unfortunately, notation varies between different authors. Table 1.1 gives a comparison of the different notations used by some of the authors responsible for a significant part of the published works on GAMP. Also included in the table is our "unified" notation used in this note. This notation is a combination of the other notations as well as elements from the notation that is customary in the compressed sensing literature with a focus on imaging applications.

Quantity	Our notation	Krzakala	Schniter	Rangan	Donoho
System matrix	$\mathbf{A}$	$\mathbf{F}$	$\mathbf{A}$	$\mathbf{A}$	$\mathbf{A}$
Abs. entrywise squared $\mathbf{A}$	$\lvert\mathbf{A}\rvert^{\circ 2}$ *or* $\mathbf{A}_{\text{asq}}$			$\lvert\mathbf{A}\rvert^2$	
Dictionary coefficients	$\boldsymbol{\alpha}$	$\mathbf{x}$ *or* $\mathbf{s}$	$\mathbf{x}$	$\mathbf{x}$ *or* $\mathbf{s}$	$\mathbf{x}$ *or* $\mathbf{s}$
Noiseless measurements	$\mathbf{z}$		$\mathbf{z}$	$\mathbf{z}$	
Noisy measurements	$\mathbf{y}$	$\mathbf{y}$	$\mathbf{y}$	$\mathbf{y}$	$\mathbf{y}$
Additive noise	$\mathbf{e}$	$\boldsymbol{\xi}$	$\mathbf{w}$	$\mathbf{w}$	$\mathbf{w}$
Image as a vector	$\mathbf{x}$				
Image width	w				
Image height	h				
Image as a matrix	$\mathbf{M}$				
Number of measurements	m	M	m *or* M	m	n
Number of coefficients	n	N	n *or* N	n	N
Number of non-zeros	k	K	k *or* K		k
Undersampling ratio	$\delta = \frac{m}{n}$	α			δ
Sparsity level	$\rho = \frac{k}{m}$				ρ
Signal density	$\tau = \frac{k}{n}$	ρ	λ	ρ	
Dictionary	$\boldsymbol{\Phi}$				
Sampling matrix	$\boldsymbol{\Psi}$				
Input (/prior) parameters	$\boldsymbol{\theta}_I$		$\mathbf{q}$	$\mathbf{q}$	
Output parameters	$\boldsymbol{\theta}_o$				
Factor-side state #1	$\mathbf{v}$ / v	$\mathbf{V}$ / $\bar{V}$	$\boldsymbol{\mu}^p$ / μ^p	$\boldsymbol{\tau}^p$ / τ^p	γ
Factor-side state #2	$\mathbf{o}$	$\boldsymbol{\omega}$	$\hat{\mathbf{p}}$	$\hat{\mathbf{p}}$	
Channel function	f	f		g	F/G *or* $(\eta)^{\ddagger}$
Factor-side mean	$\bar{\mathbf{z}}$		$\hat{\mathbf{z}}$		
Factor-side variance	$\tilde{\mathbf{z}}$		$\boldsymbol{\mu}^z$		
Output channel state #1	$\mathbf{q}$		$\hat{\mathbf{s}}$	$\hat{\mathbf{s}}$	
Output channel state #2	$\mathbf{u}$ / u		$\boldsymbol{\mu}^s$ / μ^s	$\boldsymbol{\tau}^s$ / τ^s	
Variable-side state #1	$\mathbf{s}$ / s	$\boldsymbol{\Sigma}^2$ / Σ^2	$\boldsymbol{\mu}^r$ / μ^r	$\boldsymbol{\tau}^r$ / τ^r	
Variable-side state #2	$\mathbf{r}$	$\mathbf{R}$	$\hat{\mathbf{r}}$	$\hat{\mathbf{r}}$	
Variable-side mean	$\bar{\boldsymbol{\alpha}}$	$\mathbf{a}$	$\hat{\mathbf{x}}$	$\hat{\mathbf{x}}$	$\mathbf{x}$
Variable-side variance	$\tilde{\boldsymbol{\alpha}}$	$\mathbf{v}$	$\boldsymbol{\mu}^x$	$(\boldsymbol{\tau}^x)^{\dagger}$	
Onsager-corrected residual	$\boldsymbol{\chi}$		$\hat{\mathbf{v}}$		$\mathbf{z}$
(Marginal) prob. density	$p(x)$	$\phi(x)$	$p_X(x)$	$p_X(x)$	
Joint prob. density	$p(x,y)$	$P(x,y)$	$p_{X,Y}(x,y)$	$p_{X,Y}(x,y)$	
Conditional prob. density	$p(x\|y;\theta)$	$P(x\|y,\theta)$	$p_{X\|Y;\theta}(x\|y;\theta)$		
Normalisation factor	$\mathcal{Z}$	Z	Z	Z	Z
AWGN noise variance	σ^2	Δ	μ^w *or* ψ	τ^w	σ^2
Gaussian mean	$\bar{\theta}$	$\bar{x}$	$\hat{\theta}$ *or* θ	q	
Gaussian variance	$\tilde{\theta}$	σ^2	μ^θ *or* ϕ	τ^{x0}	
Laplacian rate parameter	λ		λ		β
Laplacian mean parameter	μ				
Convergence tolerance	ϵ		τ_{gamp}		
Iteration index	t		t	t	t
Maximum iterations	$T_{\max}$		$T_{\max}$		
Step-size parameter	κ		β		
Dirac delta	δ_{Dirac}	δ	δ		δ

‡ Only under certain conditions is the η threshold functions equivalent to the GAMP channel functions - see Section 2.1.

† Rangan uses slightly different states to allow for both MMSE and MAP estimates. See [1], [16], [20].

Table 1.1: Comparison of notations typically used by the different research groups working on (G)AMP.

2 The GAMP Iteration

The AMP iteration is given in [5]. Here we state the MMSE Generalised AMP (GAMP) iteration following Parker's presentation [15]. Rangan was the first to state the GAMP iteration [1]. However, Rangan uses special output functions which allow for both MMSE and MAP estimates. The relation between the below MMSE GAMP iteration and Rangan's more general GAMP iteration is elaborated on in Section 3. The optional use of parameter value updates is described in [5] and [28]. The MMSE GAMP iteration consists of the state updates in Equations (2.1)-(2.12).

Output (factor) side updates:

$$v_i^{t+1} = \sum_j |a_{ij}|^2 \tilde{\alpha}_j^t \tag{2.1}$$

$$o_i^{t+1} = \sum_j a_{ij} \bar{\alpha}_j^t - v_i^{t+1} q_i^t \tag{2.2}$$

$$\bar{z}_i^{t+1} = f_{\bar{z}_i}(v_i^{t+1}, o_i^{t+1}; y_i, [\boldsymbol{\theta}_o]_i^t) \tag{2.3}$$

$$\tilde{z}_i^{t+1} = f_{\tilde{z}_i}(v_i^{t+1}, o_i^{t+1}; y_i, [\boldsymbol{\theta}_o]_i^t) \tag{2.4}$$

$$q_i^{t+1} = \frac{\bar{z}_i^{t+1} - o_i^{t+1}}{v_i^{t+1}} \tag{2.5}$$

$$u_i^{t+1} = \frac{v_i^{t+1} - \tilde{z}_i^{t+1}}{(v_i^{t+1})^2} \tag{2.6}$$

Input (variable) side updates:

$$s_j^{t+1} = \left[\sum_i |a_{ij}|^2 u_i^{t+1} \right]^{-1} \tag{2.7}$$

$$r_j^{t+1} = \bar{\alpha}_j^t + s_j^{t+1} \sum_i a_{ij}^* q_i^{t+1} \tag{2.8}$$

$$\bar{\alpha}_j^{t+1} = f_{\bar{\alpha}_j}(s_j^{t+1}, r_j^{t+1}; [\boldsymbol{\theta}_I]_j^t) \tag{2.9}$$

$$\tilde{\alpha}_j^{t+1} = f_{\tilde{\alpha}_j}(s_j^{t+1}, r_j^{t+1}; [\boldsymbol{\theta}_I]_j^t) \tag{2.10}$$

Optional parameter value updates (using e.g. EM - see also Section 6):

$$[\boldsymbol{\theta}_o]_i^{t+1} = \dots \tag{2.11}$$

$$[\boldsymbol{\theta}_I]_j^{t+1} = \dots \tag{2.12}$$

where a_{ij}^* is the complex conjugate of a_{ij}.

2.1 Relation to Donoho/Maleki/Montanari AMP

Rangan states that GAMP [16] is closely related to the AMP algorithm by Donoho/Maleki/Montanari [2], [3], [4], [9], [34]. Parker [15] gives the below elaboration on this claim in relation to the MMSE GAMP (see also [35]).

We consider the MMSE GAMP with the AWGN output channel given in Equations (3.57) and (3.60) (used in computing $\bar{z}$ and $\tilde{z}$). The equivalence to the DMM AMP is based on a few simplifications of some of the GAMP states. In particular, v, u, and s become scalars (which do not depend on the index j)

$$v^{t+1} \coloneqq \frac{1}{m}\sum_j \tilde{\alpha}_j^t \approx \sum_j |a_{ij}|^2 \tilde{\alpha}_j^t \tag{2.13}$$

$$u^{t+1} \coloneqq \frac{v^{t+1} - \tilde{z}^{t+1}}{(v^{t+1})^2} = \frac{v^{t+1} - \frac{\sigma^2 v^{t+1}}{\sigma^2 + v^{t+1}}}{(v^{t+1})^2} = \frac{\frac{v^{t+1}\sigma^2 + (v^{t+1})^2 - v^{t+1}\sigma^2}{\sigma^2 + v^{t+1}}}{(v^{t+1})^2} = \frac{1}{\sigma^2 + v^{t+1}} \tag{2.14}$$

$$s^{t+1} \coloneqq \frac{1}{u^{t+1}} = \sigma^2 + v^{t+1} \approx \left[\frac{1}{m}\sum_i u_i^{t+1}\right]^{-1} \approx \left[\sum_i |a_{ij}|^2 u_i^{t+1}\right]^{-1} \tag{2.15}$$

These simplifications are closely related to the sum approximations by Krzakala et al. in Equations (4.7) and (4.8), though the scaling (i.e. the assumed variance of the the entries in $\mathbf{A}$) is slightly different: $\frac{1}{m}$ vs $\frac{1}{n}$. Based on the above simplifications, the update of the r_j state becomes

$$r_j^{t+1} = \bar{\alpha}_j^t + s^{t+1}\sum_i a_{ij}^* q_i^{t+1} \tag{2.16}$$

$$= \bar{\alpha}_j^t + s^{t+1}\sum_i a_{ij}^* \frac{\bar{z}_i^{t+1} - o_i^{t+1}}{v^{t+1}} \tag{2.17}$$

$$= \bar{\alpha}_j^t + s^{t+1}\sum_i a_{ij}^* \frac{o_i^{t+1} + \frac{v^{t+1}}{\sigma^2 + v^{t+1}}(y_i - o_i^{t+1}) - o_i^{t+1}}{v^{t+1}} \tag{2.18}$$

$$= \bar{\alpha}_j^t + s^{t+1}\sum_i a_{ij}^* \frac{y_i - o_i^{t+1}}{\sigma^2 + v^{t+1}} \tag{2.19}$$

$$= \bar{\alpha}_j^t + \sum_i a_{ij}^* (y_i - o_i^{t+1}) \tag{2.20}$$

$$= \bar{\alpha}_j^t + \sum_i a_{ij}^* \chi_i^{t+1} \tag{2.21}$$

for

$$\chi_i^{t+1} \coloneqq y_i - o_i^{t+1} \tag{2.22}$$

Now, for χ_i^{t+1}, we have

$$\chi_i^{t+1} = y_i - o_i^{t+1} \tag{2.23}$$
$$= y_i - \left(\sum_j a_{ij} \bar{\alpha}_j^t - v^{t+1} q_i^t \right) \tag{2.24}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + v^{t+1} \frac{\bar{z}_i^t - o_i^t}{v^t} \tag{2.25}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + v^{t+1} \frac{y_i - o_i^t}{\sigma^2 + v^t} \tag{2.26}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{v^{t+1}}{s^t} \chi_i^t \tag{2.27}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{\frac{1}{m} \sum_j \tilde{\alpha}_j^t}{s^t} \chi_i^t \tag{2.28}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{\frac{1}{m} \sum_j s^t g'_{in}(r_j^t, \boldsymbol{\theta}_I, s^t)}{s^t} \chi_i^t \tag{2.29}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{1}{m} \sum_j g'_{in}(r_j^t, \boldsymbol{\theta}_I, s^t) \chi_i^t \tag{2.30}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{n}{m} \langle g'_{in}(r_j^t, \boldsymbol{\theta}_I, s^t) \rangle \chi_i^t \tag{2.31}$$
$$= y_i - \sum_j a_{ij} \bar{\alpha}_j^t + \frac{1}{\delta} \langle g'_{in}(\bar{\alpha}_j^{t-1} + \sum_i a_{ij}^* \chi_i^t, \boldsymbol{\theta}_I, s^t) \rangle \chi_i^t \tag{2.32}$$

where we have used Rangan's GAMP g'_{in} channel in Equation (3.26) and $\langle \cdot \rangle$ denotes the average. Similarly for $\bar{\alpha}^{t+1}$, we have

$$\bar{\alpha}_j^{t+1} = f_{\bar{\alpha}_j}(s_j^{t+1}, r_j^{t+1}; \boldsymbol{\theta}_I) \tag{2.33}$$
$$= g_{in}(r_j^{t+1}, \boldsymbol{\theta}_I, s_j^{t+1},) \tag{2.34}$$
$$= g_{in}(\bar{\alpha}_j^t + \sum_i a_{ij}^* \chi_i^{t+1}, \boldsymbol{\theta}_I, s_j^{t+1}) \tag{2.35}$$

where we have used Rangan's GAMP g_{in} channel in Equation (3.25). Now in matrix-vector notation, Equations (2.32) and (2.35) read

$$\boldsymbol{\chi}_t = \mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \frac{1}{\delta} \langle g'_{in}(\bar{\boldsymbol{\alpha}}_{t\text{-}2} + \mathbf{A}^H \boldsymbol{\chi}_{t\text{-}1}, \boldsymbol{\theta}_I, \mathbf{s}_{t\text{-}1}) \rangle \boldsymbol{\chi}_{t\text{-}1} \tag{2.36}$$
$$\bar{\boldsymbol{\alpha}}_t = g_{in}(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^H \boldsymbol{\chi}_t, \boldsymbol{\theta}_I, \mathbf{s}_t) \tag{2.37}$$

where ^{H} denotes the Hermitian transpose (complex conjugated transpose), $\boldsymbol{\chi}$ is the so-called Onsager-corrected residual and $\mathbf{s}_t$ is a length m vector with all entries equal to the scalar in Equation (2.15). Here we note that from Equations (2.13) and(2.15), using similar steps as in deriving Equation (2.32), we have

$$v^{t+1} = \frac{1}{\delta} s^t \langle g'_{in}(\bar{\alpha}_j^{t-1} + \sum_i a_{ij}^* \chi_i^t, \boldsymbol{\theta}_I, s^t) \rangle \tag{2.38}$$
$$= \frac{1}{\delta} (\sigma^2 + v^t) \langle g'_{in}(\bar{\alpha}_j^{t-1} + \sum_i a_{ij}^* \chi_i^t, \boldsymbol{\theta}_I, \sigma^2 + v^t) \rangle \tag{2.39}$$

Thus, in order to compute the channel value based on s, one may introduce the additional recursion on the state v. If we take $\eta_t(\cdot) = g_{in}(\cdot, \boldsymbol{\theta}_I, \mathbf{s}_t)$ for the threshold function η_t in [2], then Equations

(2.36) and (2.37) constitute the Donoho/Maleki/Montanari AMP update in Equations **[2]** and **[1]**, respectively, in [2][1].

The threshold function, η_t is in general a conditional expectation [3] (as is $g_{in}(\cdot, \boldsymbol{\theta}_I, \mathbf{s}_t)$) in MMSE GAMP. However, in AMP for the Basis Pursuit and LASSO problems [2], [3] (closely linked to instances of the MAP GAMP [1], [16], the threshold function becomes the soft threshold operator for which the threshold level is chosen slightly differently - see [9] for more details. See also [15] for a further discussion of some of these subtle details in the choice of threshold function.

2.2 Relation to IST, ISTA, and ADMM

The iterative soft thresholding (IST) algorithm [36] is similar in structure to the DMM AMP updates in Equations (2.36) and (2.37). Specifically, the corresponding IST updates read

$$\boldsymbol{\chi}_t = \mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} \tag{2.40}$$

$$\bar{\boldsymbol{\alpha}}_t = \eta_t(\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^H\boldsymbol{\chi}_t) \tag{2.41}$$

for η_t being the soft threshold operator. Thus, the difference to AMP is the lack of the Onsager correction $\frac{1}{\delta}\langle g'_{in}(\bar{\boldsymbol{\alpha}}_{t\text{-}2} + \mathbf{A}^H\boldsymbol{\chi}_{t\text{-}1}, \boldsymbol{\theta}_I, \mathbf{s}_{t\text{-}1})\rangle\boldsymbol{\chi}_{t\text{-}1}$. It is this correction that gives rise to the interpretation of $\bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{A}^H\boldsymbol{\chi}_t$ being a AWGN corrupted version of the true $\boldsymbol{\alpha}$ as discussed in Section 1.4.3 [2].

GAMP may also be interpreted as certain variants of the iterative shrinkage and thresholding algorithm (ISTA) [37] and the alternating direction method of multipliers (ADMM) algorithm [38] as detailed in [25].

[1] In [2], $\bar{\boldsymbol{\alpha}}_t$ is computed prior to $\boldsymbol{\chi}_t$ which accounts for the iteration index shifts.

3 MMSE Channel Functions

The MMSE GAMP channel functions, $f_{\tilde{\alpha}}$, $f_{\tilde{\alpha}}$, $f_{\tilde{z}}$, $f_{\tilde{z}}$, used in Equations (2.1)-(2.12) are given in terms of special conditional expectations and variances since these are at the core of the MMSE GAMP as described in Section 1.4.1. The MMSE GAMP channel functions follow the AMP channel function definitions by Krzakala et. al in [5] but differs from GAMP the channel function definitions used by Rangan [1] as elaborated on in Section 3.3.1. All channel functions are scalar functions. Thus, in this chapter we drop the notational dependence on the index as well as the notational dependence on iteration. When the presented channel expressions are used with the GAMP iteration in Equations (2.1)-(2.12), the appropriate dependencies should be taken into account.

3.1 Input Side Channel Functions

The GAMP input side channel functions are:

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathbb{E}_{\alpha|s,r,\boldsymbol{\theta}_I}[\alpha] \coloneqq \frac{1}{\mathcal{Z}_I} \int_{\alpha} \alpha p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha \tag{3.1}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathrm{Var}_{\alpha|s,r,\boldsymbol{\theta}_I}(\alpha) \coloneqq \frac{1}{\mathcal{Z}_I} \int_{\alpha} |\alpha|^2 p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha - |f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I)|^2 \tag{3.2}$$

where $\mathcal{Z}_I = \int_{\alpha} p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha$ is a normalisation constant that ensures that the product $p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s)$ is a proper probability measure and

$$\mathcal{N}(\alpha; r, s) = \frac{1}{\sqrt{2\pi s}} \exp\left(-\frac{1}{2} \frac{(\alpha - r)^2}{s} \right) \tag{3.3}$$

Thus, from Equation (3.1) we find that in GAMP, the true marginal posterior $p(\alpha|\mathbf{y}; \boldsymbol{\theta}_I)$ is approximated by:

$$p(\alpha|\mathbf{y}; s, r, \boldsymbol{\theta}_I) \coloneqq \frac{p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s)}{\int_{\alpha} p(\alpha; \boldsymbol{\theta}_I) \mathcal{N}(\alpha; r, s) d\alpha} \tag{3.4}$$

which has the interpretation that if $\tilde{\mathcal{A}}$ is a random variable distributed according to $p(\alpha; \boldsymbol{\theta}_I)$ and $\tilde{\mathcal{B}} = \tilde{\mathcal{A}} + \mathcal{W}$ with $\mathcal{W}$ a zero-mean Gaussian noise with variance s, then Equation (3.1) is the conditional mean of $\tilde{\mathcal{A}}$ given $\tilde{\mathcal{B}} = r$, i.e. $\mathbb{E}[\tilde{\mathcal{A}}|\tilde{\mathcal{B}} = r]$. Similarly Equation (3.2) is the conditional variance $\mathrm{Var}(\tilde{\mathcal{A}}|\tilde{\mathcal{B}} = r)$ [3], [16].

Note that the input channel parameters $\boldsymbol{\theta}_I$ may depend on the coefficient, i.e. the $[\boldsymbol{\theta}_I]_j$'s may be different for each $j = 1, \ldots, n$. In Equation (3.4) it is to be understood that $\boldsymbol{\theta}_I$ is the vector/matrix of *all* input channel parameters independently of whether or not they depend on the index j.

All the input channel functions are scalar functions. When used with vectors as arguments, it is to be understood that a channel function is used on each element of the vector.

3.2 Output Side Channel Functions:

The GAMP output side channel functions are:

$$f_{\tilde{z}}(v, o; y, \boldsymbol{\theta}_o) = \mathbb{E}_{z|o,v,y,\boldsymbol{\theta}_o}[z] \coloneqq \frac{1}{\mathcal{Z}_o} \int_{z} z p(y|z; \boldsymbol{\theta}_o) \mathcal{N}(z; o, v) dz \tag{3.5}$$

$$f_{\tilde{z}}(v, o; y, \boldsymbol{\theta}_o) = \mathrm{Var}_{z|o,v,y,\boldsymbol{\theta}_o}(z) \coloneqq \frac{1}{\mathcal{Z}_o} \int_{z} |z|^2 p(y|z; \boldsymbol{\theta}_o) \mathcal{N}(z; o, v) dz - |f_{\tilde{z}}(v, o; y, \boldsymbol{\theta}_o)|^2 \tag{3.6}$$

where $\mathcal{Z}_o = \int_z p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)dz$ is a normalisation constant that ensures that the product $p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)$ is a proper probability measure and

$$\mathcal{N}(z;o,v) = \frac{1}{\sqrt{2\pi v}}\exp\left(-\frac{1}{2}\frac{(z-o)^2}{v}\right) \tag{3.7}$$

Thus, from Equation (3.5) we find that in GAMP, the true marginal posterior $p(z|\mathbf{y};\boldsymbol{\theta}_o)$ is approximated by:

$$p(z|\mathbf{y};o,v,\boldsymbol{\theta}_o) \coloneqq \frac{p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)}{\int_z p(y|z;\boldsymbol{\theta}_o)\mathcal{N}(z;o,v)dz} \tag{3.8}$$

which has the interpretation that if $\tilde{\mathcal{Y}}$ is a random variable distributed according to $p(y|z;\boldsymbol{\theta}_o)$ and $\tilde{\mathcal{Z}}$ is a Gaussian random variable with mean o and variance v, then Equation (3.5) is the conditional mean of $\tilde{\mathcal{Z}}$ given $\tilde{\mathcal{Y}} = y$, i.e. $\mathbb{E}[\tilde{\mathcal{Z}}|\tilde{\mathcal{Y}} = y]$. Similarly, Equation (3.6) is the conditional variance $\mathrm{Var}(\tilde{\mathcal{Z}}|\tilde{\mathcal{Y}} = y)$ [16].

Note that the output channel parameters $\boldsymbol{\theta}_o$ may depend on the coefficient, i.e. the $[\boldsymbol{\theta}_o]_i$'s may be different for each $i = 1,\ldots,m$. In Equation (3.8), it is to be understood that $\boldsymbol{\theta}_o$ is the vector/matrix of *all* output channel parameters independently of whether or not they depend on the index i.

All the output channel functions are scalar functions. When used with vectors as arguments, it is to be understood that a channel function is used on each element of the vector.

3.3 MMSE Channel Functions in General

From Equations (3.1) and (3.2) as well as Equations (3.5) and (3.6), we find that in the MMSE case, evaluating the channels functions amounts to evaluating conditional means and variances. Thus, in relation to the in- and output channel functions, we may, for a given channel distribution $p(u;\boldsymbol{\theta})$, in general, define

$$\mathcal{Z}(v,w,\boldsymbol{\theta}) \coloneqq \int_u p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \in \mathbb{R} \tag{3.9}$$

$$N_1(v,w,\boldsymbol{\theta}) \coloneqq \int_u up(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \in \mathbb{C} \tag{3.10}$$

$$N_{2a}(v,w,\boldsymbol{\theta}) \coloneqq \int_u |u|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \in \mathbb{C} \tag{3.11}$$

for which we may find mean and variance functions as

$$\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u] = \frac{N_1(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} \in \mathbb{C} \tag{3.12}$$

$$\mathrm{Var}_{u|v,w,\boldsymbol{\theta}}(u) = \frac{N_{2a}(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} - |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \in \mathbb{R} \tag{3.13}$$

The expression for the conditional variance stems from

$$\mathrm{Var}_{u|v,w,\boldsymbol{\theta}}(u) \coloneqq \frac{\int_u |u - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du}{\int_u p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du} \tag{3.14}$$

$$N_2(v,w,\boldsymbol{\theta}) \coloneqq \int_u |u - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \tag{3.15}$$

$$\begin{aligned} &= \int_u |u|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \\ &\quad - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u] \int_u u^* p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \\ &\quad + |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \int_u p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \\ &\quad - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]^* \int_u u p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \end{aligned} \tag{3.16}$$

$$\begin{aligned} &= \int_u |u|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du \\ &\quad - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]^* \mathcal{Z}(v,w,\boldsymbol{\theta}) \\ &\quad + |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \mathcal{Z}(v,w,\boldsymbol{\theta}) \\ &\quad - \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]^* \mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u] \mathcal{Z}(v,w,\boldsymbol{\theta}) \end{aligned} \tag{3.17}$$

$$= \int_u |u|^2 p(u;\boldsymbol{\theta})\mathcal{N}(u;v,w)du - |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \mathcal{Z}(v,w,\boldsymbol{\theta})^* \tag{3.18}$$

$$N_2(v,w,\boldsymbol{\theta}) = N_{2a}(v,w,\boldsymbol{\theta}) - N_{2b}(v,w,\boldsymbol{\theta}) \tag{3.19}$$

$$N_{2b}(v,w,\boldsymbol{\theta}) \coloneqq |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \mathcal{Z}(v,w,\boldsymbol{\theta})^* = |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \mathcal{Z}(v,w,\boldsymbol{\theta}) \tag{3.20}$$

$$\mathrm{Var}_{u|v,w,\boldsymbol{\theta}}(u) = \frac{N_2(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} \tag{3.21}$$

$$= \frac{N_{2a}(v,w,\boldsymbol{\theta}) - N_{2b}(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} \tag{3.22}$$

$$= \frac{N_{a2}(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} - |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \frac{\mathcal{Z}(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} \tag{3.23}$$

$$= \frac{N_{a2}(v,w,\boldsymbol{\theta})}{\mathcal{Z}(v,w,\boldsymbol{\theta})} - |\mathbb{E}_{u|v,w,\boldsymbol{\theta}}[u]|^2 \tag{3.24}$$

where * denotes complex conjugation.

3.3.1 Relation to Rangan's Channel Functions

Rangan's GAMP allows for obtaining both MAP and MMSE estimates depending on the choice of channel functions. For MAP estimates the channel functions are found from certain probability maximisation problems and are, thus, closely related to typical optimisation formulations used in e.g. sparse inference [1], [16]. For MMSE estimates the channel functions are expectations and variances as seen in the channel functions given in Sections 3.1 and 3.2. To allow for both types (MAP and MMSE) of channel functions, Rangan uses the differently defined channel functions $g_{in}, g'_{in}, g_{out}, g'_{out}$ [1], [16]. For the MMSE case, we have the following relations:

$$g_{in}(r,\boldsymbol{\theta}_I,s) = f_{\tilde{\alpha}}(s,r;\boldsymbol{\theta}_I) \tag{3.25}$$

$$g'_{in}(r,\boldsymbol{\theta}_I,s) = \frac{f_{\tilde{\alpha}}(s,r;\boldsymbol{\theta}_I)}{s} \tag{3.26}$$

$$g_{out}(o,y,v) = \frac{f_{\tilde{z}}(v,o;y,\boldsymbol{\theta}_o) - o}{v} = q \tag{3.27}$$

$$g'_{out}(o,y,v) = \frac{f_{\tilde{z}}(v,o;y,\boldsymbol{\theta}_o) - v}{v^2} = -u \tag{3.28}$$

3.4 General Sparse Input Channel

Sparsity is a typical *structure* assumed on $\boldsymbol{\alpha}$ in the reconstruction problem described in Section 1.2. Thus, when using GAMP to solve such reconstruction problems, one usually uses a sparsity promoting prior. For that reason, many of the input side channels presented in the literature are described by a probability density function which is a mixture of a Dirac delta function at zero and some other known proper density function [5], [28], i.e.

$$p(\alpha;\boldsymbol{\theta}_I) = (1-\tau)\delta_{\text{Dirac}}(\alpha) + \tau\varphi(\alpha;\boldsymbol{\theta}_I) \tag{3.29}$$

where $\tau \in [0;1]$ is the signal density and $\varphi(\alpha;\boldsymbol{\theta}_I)$ is e.g. Gaussian, Laplace, Student's t or even itself a mixture density. Such a prior is sometimes also referred to as a spike-and-slab prior because of the (sparsity promoting) spike at zero and a slab part $\varphi(\alpha;\boldsymbol{\theta}_I)$. The spike part is typically referred to as the Bernoulli part as in e.g. a sparse Bernoulli-Gaussian prior [39]. In using the expression in Equation (3.29), it is important to realise that any manipulations are to be understood as being done "inside" an integral (such as an expectation) which makes the use of the Dirac delta function well defined.

Since the Dirac delta function integrates to 1 over the real line, it is easily seen that $p(\alpha;\boldsymbol{\theta}_I)$ in Equation (3.29) integrates to 1 for any proper probability density function $\varphi(\alpha;\boldsymbol{\theta}_I)$ and is, thus, a proper probability density itself. The GAMP approximated posterior in Equation (3.4) for the general sparse prior in Equation (3.29) is

$$p(\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I) = \frac{((1-\tau)\delta_{\text{Dirac}}(\alpha)+\tau\varphi(\alpha;\boldsymbol{\theta}_I))\mathcal{N}(\alpha;r,s)}{\mathcal{Z}_I} \tag{3.30}$$

$$= \frac{((1-\tau)\delta_{\text{Dirac}}(\alpha)+\tau\varphi(\alpha;\boldsymbol{\theta}_I))\mathcal{N}(\alpha;r,s)}{(1-\tau)\int_\alpha \mathcal{N}(\alpha;r,s)\delta_{\text{Dirac}}(\alpha)d\alpha + \tau\int_\alpha \varphi(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)d\alpha} \tag{3.31}$$

$$= \frac{(1-\tau)\delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s)+\tau\varphi(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)}{(1-\tau)\mathcal{Z}_\delta+\tau\mathcal{Z}_\varphi} \tag{3.32}$$

$$= \frac{(1-\tau)\mathcal{N}(\alpha;r,s)\delta_{\text{Dirac}}(\alpha)}{(1-\tau)\mathcal{Z}_\delta+\tau\mathcal{Z}_\varphi} + \frac{\tau\mathcal{N}(\alpha;r,s)\varphi(\alpha;\boldsymbol{\theta}_I)}{(1-\tau)\mathcal{Z}_\delta+\tau\mathcal{Z}_\varphi} \tag{3.33}$$

$$= \frac{(1-\tau)\mathcal{Z}_\delta}{(1-\tau)\mathcal{Z}_\delta+\tau\mathcal{Z}_\varphi}\frac{\mathcal{N}(\alpha;r,s)\delta_{\text{Dirac}}(\alpha)}{\mathcal{Z}_\delta} + \frac{\tau\mathcal{Z}_\varphi}{(1-\tau)\mathcal{N}(0;r,s)+\tau\mathcal{Z}_\varphi}\frac{\mathcal{N}(\alpha;r,s)\varphi(\alpha;\boldsymbol{\theta}_I)}{\mathcal{Z}_\varphi} \tag{3.34}$$

$$= \left(1-\frac{\tau\mathcal{Z}_\varphi}{(1-\tau)\mathcal{Z}_\delta+\tau\mathcal{Z}_\varphi}\right)\frac{\mathcal{N}(\alpha;r,s)\delta_{\text{Dirac}}(\alpha)}{\mathcal{Z}_\delta} + \frac{\tau\mathcal{Z}_\varphi}{(1-\tau)\mathcal{N}(0;r,s)+\tau\mathcal{Z}_\varphi}\frac{\mathcal{N}(\alpha;r,s)\varphi(\alpha;\boldsymbol{\theta}_I)}{\mathcal{Z}_\varphi} \tag{3.35}$$

$$= (1-\pi(r,s,\boldsymbol{\theta}_I))\delta'_{\text{Dirac}}(\alpha) + \pi(r,s,\boldsymbol{\theta}_I)\varphi_{\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I}(\alpha;\boldsymbol{\theta}_I) \tag{3.36}$$

$$= (1-\pi(r,s,\boldsymbol{\theta}_I))\delta_{\text{Dirac}}(\alpha) + \pi(r,s,\boldsymbol{\theta}_I)\varphi_{\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I}(\alpha;\boldsymbol{\theta}_I) \tag{3.37}$$

for

$$\mathcal{Z}_I \coloneqq \int_\alpha ((1-\tau)\delta_{\text{Dirac}}(\alpha)+\tau\varphi(\alpha;\boldsymbol{\theta}_I))\mathcal{N}(\alpha;r,s)d\alpha \tag{3.38}$$

$$\mathcal{Z}_\delta \coloneqq \int_\alpha \delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s)d\alpha = \mathcal{N}(0;r,s) \tag{3.39}$$

$$\mathcal{Z}_\varphi \coloneqq \int_\alpha \varphi(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s)d\alpha \tag{3.40}$$

$$\pi(r,s,\boldsymbol{\theta}_I) \coloneqq \frac{\tau\mathcal{Z}_\varphi}{(1-\tau)\mathcal{N}(0;r,s)+\tau\mathcal{Z}_\varphi} = \frac{1}{1+\frac{(1-\tau)\mathcal{N}(0;r,s)}{\tau\mathcal{Z}_\varphi}} = \frac{1}{1+\left(\frac{\tau\mathcal{Z}_\varphi}{(1-\tau)\mathcal{N}(0;r,s)}\right)^{-1}} \tag{3.41}$$

$$\varphi_{\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I}(\alpha;\boldsymbol{\theta}_I) \coloneqq \frac{\mathcal{N}(\alpha;r,s)\varphi(\alpha;\boldsymbol{\theta}_I)}{\mathcal{Z}_\varphi} \tag{3.42}$$

$$\delta'_{\text{Dirac}}(\alpha) \coloneqq \frac{\mathcal{N}(\alpha;r,s)\delta_{\text{Dirac}}(\alpha)}{\mathcal{Z}_\delta} \tag{3.43}$$

The equality of Equations (3.36) and (3.37) is based on the manipulations being done "inside" an integral. In this case, the sampling property of the Dirac delta function $\delta_{\text{Dirac}}(\alpha)$ "sifts" the value at $\alpha = 0$ which means that the $\delta'_{\text{Dirac}}(\alpha)$ function provides a scaling of $\mathcal{N}(0;r,s)$ which is cancelled by $\mathcal{Z}_\delta = \mathcal{N}(0;r,s)$ essentially reducing $\delta'_{\text{Dirac}}(\alpha)$ to $\delta_{\text{Dirac}}(\alpha)$.

Thus, as reported in [28], for the general sparse prior in Equation (3.29), we find that the GAMP posterior in Equation (3.37) is again a sparse density consisting of a Dirac delta at zero and the *GAMP posterior* $\varphi_{\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I}(\alpha;\boldsymbol{\theta}_I)$ of $\varphi(\alpha;\boldsymbol{\theta}_I)$ with a posteriori signal density (posterior support probabilities) $\pi(r,s,\boldsymbol{\theta}_I)$. As is seen from Equation (3.41), we have $\pi(r,s,\boldsymbol{\theta}_I) \in [0;1]$ as along as $\tau \in [0;1]$ and $\varphi(\alpha;\boldsymbol{\theta}_I)$ is a proper density since all quantities are non-negative. Thus, the GAMP approximated posterior $p(\alpha|\mathbf{y};s,r,\boldsymbol{\theta}_I)$ remains a proper density.

3.5 General Weighted Sparse Input Channel

If we assume that $\boldsymbol{\alpha}$ in the reconstruction problem in Section 1.2 is not only sparse but *structured sparse* in the sense that some of the coefficient values in $\boldsymbol{\alpha}$ are more likely to be sparse than others, we may consider an independent but non-identical general sparse weighted (GWS) input channel, i.e.

$$p(\alpha_j;\boldsymbol{\theta}_I) = (1 - w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j) \tag{3.44}$$

where $\tau \in [0;1]$ models the overall signal density and the $w_j \in [0;1]$, $j = 1,\ldots,n$ are individual weights that model the belief about the sparsity of the individual coefficients. We note that the general weighted sparse input channel in Equation (3.44) reduces to the general sparse input channel in Equation (3.29) if $\forall j, w_j = 1$. Since the Dirac delta function integrates to 1 over the real line, it is easily seen that $p(\alpha_j;[\boldsymbol{\theta}_I]_j$ in Equation (3.44) integrates to 1 for any proper probability density function $\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j)$ and is, thus, a proper probability density itself.

Since the input channel acts independently on each element of $\boldsymbol{\alpha}, \mathbf{s}, \mathbf{r}$, everything still decouples in manipulations involving Equation (3.44). Thus, following the same path of derivations as was done in deriving Equation (3.29), we find that GAMP approximated posterior for the GWS prior in Equation (3.44) is

$$\begin{aligned} p(\alpha_j|\mathbf{y};s_j,r_j,[\boldsymbol{\theta}_I]_j) &= (1 - \pi_j^{\text{w}}(r_j,s_j,[\boldsymbol{\theta}_I]_j))\delta_{\text{Dirac}}(\alpha_j) \\ &\quad + \pi_j^{\text{w}}(r_j,s_j,[\boldsymbol{\theta}_I]_j)\varphi_{\alpha_j|\mathbf{y};s_j,r_j,[\boldsymbol{\theta}_I]_j}(\alpha_j;[\boldsymbol{\theta}_I]_j) \end{aligned} \tag{3.45}$$

for

$$\pi_j^{\text{w}}(r_j,s_j,[\boldsymbol{\theta}_I]_j) \coloneqq \frac{w_j\tau\mathcal{Z}_{\varphi_j}}{(1-w_j\tau)\mathcal{N}(0;r_j,s_j) + w_j\tau\mathcal{Z}_{\varphi_j}} = \frac{w_j\tau\mathcal{Z}_{\varphi_j}}{\mathcal{Z}_{I_j}} \tag{3.46}$$

$$= \frac{1}{1 + \frac{(1-w_j\tau)\mathcal{N}(0;r_j,s_j)}{w_j\tau\mathcal{Z}_{\varphi_j}}} = \frac{1}{1 + \left(\frac{w_j\tau\mathcal{Z}_{\varphi_j}}{(1-w_j\tau)\mathcal{N}(0;r_j,s_j)}\right)^{-1}} \tag{3.47}$$

$$\mathcal{Z}_{I_j} \coloneqq \int_{\alpha_j} ((1 - w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j))\mathcal{N}(\alpha_j;r_j,s_j)d\alpha_j \tag{3.48}$$

$$\mathcal{Z}_{\varphi_j} \coloneqq \int_{\alpha_j} \varphi(\alpha_j;[\boldsymbol{\theta}_I]_j)\mathcal{N}(\alpha_j;r_j,s_j)d\alpha_j \tag{3.49}$$

$$\varphi_{\alpha_j|\mathbf{y};s_j,r_j,[\boldsymbol{\theta}_I]_j}(\alpha_j;[\boldsymbol{\theta}_I]_j) \coloneqq \frac{\mathcal{N}(\alpha_j;r_j,s_j)\varphi(\alpha_j;[\boldsymbol{\theta}_I]_j)}{\mathcal{Z}_{\varphi_j}} \tag{3.50}$$

Thus, for the GWS prior in Equation (3.44), the GAMP posterior in Equation (3.45) is again a sparse density with posterior signal densities $\pi_j^{\text{w}}(r_j,s_j,[\boldsymbol{\theta}_I]_j)$, $j = 1,\ldots,n$. Again, it is clear from Equation (3.46) that $\pi_j^{\text{w}}(r_j,s_j,[\boldsymbol{\theta}_I]_j) \in [0;1]$ as long as $w_j\tau \in [0;1]$ and $\varphi_{\alpha_j|\mathbf{y};s_j,r_j,[\boldsymbol{\theta}_I]_j}(\alpha_j;[\boldsymbol{\theta}_I]_j)$

is a proper density since all quantities in $\pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j)$ are non-negative. Thus, the GAMP approximated posterior $p(\alpha_j|\mathbf{y}; s_j, r_j, [\boldsymbol{\theta}_I]_j)$ remains a proper density.

Now defining

$$N_{1\varphi_j} \coloneqq \int_{\alpha_j} \alpha_j \varphi(\alpha_j; [\boldsymbol{\theta}_I]_j) \mathcal{N}(\alpha_j; r_j, s_j) d\alpha_j \tag{3.51}$$

$$N_{2a\varphi_j} \coloneqq \int_{\alpha_j} |\alpha_j|^2 \varphi(\alpha_j; [\boldsymbol{\theta}_I]_j) \mathcal{N}(\alpha_j; r_j, s_j) d\alpha_j \tag{3.52}$$

$$\tag{3.53}$$

and using Equations (3.12), (3.13), and (3.45), we find that the MMSE GAMP input channel mean and variance functions are given by

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \frac{N_{1\varphi_j}}{\mathcal{Z}_{\varphi_j}} \tag{3.54}$$

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \frac{N_{2a\varphi_j}}{\mathcal{Z}_{\varphi_j}} - |f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j)|^2 \tag{3.55}$$

since the Dirac delta at zero does not contribute to the mean or the variance. Thus, the GWS input channel mean and variance functions may be expressed as scaled versions for the slab-part mean and variance functions with the scaling given by the posterior signal densities. This may be exploited in an implementation of the GWS input channel to separate the slab-part update from the GWS updates making it easy to re-use the GWS updates with different slab-part updates.

3.6 Analytic Expressions for Common Output Channels

In an implementation of the MMSE GAMP, one may in general have to resort to numerical integration to evaluate output the channel functions $f_{\bar{z}}$, $f_{\bar{z}}$. However, for some channels, it is possible to derive analytic solutions to the integrals involved in evaluating the channel functions. Here we present some output channels for which analytic solutions to the channel evaluation functions exist.

3.6.1 AWGN Output Channel

For an additive white Gaussian noise (AWGN) output channel with noise variance σ^2 ($\boldsymbol{\theta}_o = [\sigma^2]$), i.e.

$$p(y|z; \boldsymbol{\theta}_o) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-z)^2}{2\sigma^2}\right) \tag{3.56}$$

we have channel functions [1], [16] (Eqs. (41), (42), (43)):

$$f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{vy + \sigma^2 o}{\sigma^2 + v} \tag{3.57}$$

$$= o + \frac{v}{\sigma^2 + v}(y - o) \tag{3.58}$$

$$f_{\bar{z}}(v, o; y, \boldsymbol{\theta}_o) = \frac{\sigma^2 v}{\sigma^2 + v} \tag{3.59}$$

$$= \frac{1}{\frac{1}{\sigma^2} + \frac{1}{v}} \tag{3.60}$$

Note that Equations (3.58) and (3.60) are the expressions suggested by Parker in [15]. However, in EM-BG/GM-GAMP [28] Equations (3.58) and (3.59) are mentioned. Mathematically, there is no difference in this choice. Numerically, however, there may be.

3.6.2 AWLN Output Channel

For an additive white Laplacian noise (AWLN) output channel with rate parameter $\lambda > 0$, ($\boldsymbol{\theta}_o = [\lambda]$), i.e.

$$p(y|z;\boldsymbol{\theta}_o) = \frac{\lambda}{2}\exp(-\lambda|y-z|) \tag{3.61}$$

we have channel functions [40] (Eqs. (22), (23)):

$$f_{\bar{z}}(v,o;y,\boldsymbol{\theta}_o) = y + \frac{\underline{\mathcal{Z}}_o}{\mathcal{Z}_o}\left(\underline{o} - \sqrt{v}\frac{\phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}\right) + \frac{\bar{\mathcal{Z}}_o}{\mathcal{Z}_o}\left(\bar{o} + \sqrt{v}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}\right) \tag{3.62}$$

$$\begin{aligned} f_{\tilde{z}}(v,o;y,\boldsymbol{\theta}_o) = &-(y^2 - f_{\bar{z}}(v,o;y,\boldsymbol{\theta}_o))^2 \\ &+ \frac{\underline{\mathcal{Z}}_o}{\mathcal{Z}_o}\left[v\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)} - \frac{\underline{o}}{\sqrt{v}}\right)\right) + \left(\underline{o} - \sqrt{v}\frac{\phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right)}\right)^2\right] \\ &+ \frac{\bar{\mathcal{Z}}_o}{\mathcal{Z}_o}\left[v\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)} + \frac{\bar{o}}{\sqrt{v}}\right)\right) + \left(\bar{o} + \sqrt{v}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right)}\right)^2\right] \end{aligned} \tag{3.63}$$

for

$$\mathcal{Z}_o = \underline{\mathcal{Z}}_o + \bar{\mathcal{Z}}_o \tag{3.64}$$

$$\underline{\mathcal{Z}}_o = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 v + \check{o}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{o}}{\sqrt{v}}\right) \tag{3.65}$$

$$\bar{\mathcal{Z}}_o = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 v - \check{o}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{\bar{o}}{\sqrt{v}}\right) \tag{3.66}$$

$$\check{o} = o - y \tag{3.67}$$

$$\underline{o} = \check{o} + \lambda v \tag{3.68}$$

$$\bar{o} = \check{o} - \lambda v \tag{3.69}$$

$$\Phi_{\mathcal{N}}(\check{x}) = \int_{-\infty}^{\check{x}} \phi_{\mathcal{N}}(t)\,dt = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\check{x}} \exp\left(-\frac{t^2}{2}\right)dt \tag{3.70}$$

$$\phi_{\mathcal{N}}(\check{x}) = \mathcal{N}(\check{x},0,1) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\check{x}^2}{2}\right) \tag{3.71}$$

The derivation of these channel functions is similar to the derivation of the channel functions for the i.i.d. Sparse Bernoulli-Laplace input channel in Section 3.7.1. Note that the scaled complementary error function may be used to achieve better numerical accuracy in an implementation as detailed in Section 3.7.1.1.

3.7 Analytic Expressions for Common Input Channels

In an implementation of the MMSE GAMP, one may in general have to resort to numerical integration to evaluate the input channel functions $f_{\bar{\alpha}}$, $f_{\tilde{\alpha}}$,. However, for some channels, it is possible to derive analytic solutions to the integrals involved in evaluating the channel functions. Here we present some input channels for which analytic solutions to the channel evaluation functions exist.

3.7.1 I.i.d. Sparse Bernoulli-Laplace Input Channel

We now consider an i.i.d. BL input channel with signal density τ, Laplace mean μ, and rate parameter $\lambda > 0$, ($\boldsymbol{\theta}_I = [\tau,\mu,\lambda]^T$), i.e. an input channel described by

$$p(\alpha;\boldsymbol{\theta}_I) = (1-\tau)\delta_{\text{Dirac}}(\alpha) + \tau\frac{\lambda}{2}\exp(-\lambda|\alpha-\mu|) \tag{3.72}$$

We derive the channel functions following the general procedure described in Section 3.3. Towards this end, we make use of various tricks and techniques used in the derivation of the elastic net prior in [41] as well as in the derivation of the ALWN output channel in [40]. Starting with the product of the prior and the Gaussian GAMP field, we observe that

$$p(\alpha;\boldsymbol{\theta}_I)\mathcal{N}(\alpha;r,s) = (1-\tau)\delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s) + \tau\frac{\lambda}{2}\exp(-\lambda|\alpha-\mu|)\,\mathcal{N}(\alpha;r,s) \tag{3.73}$$

$$= (1-\tau)\delta_{\text{Dirac}}(\check{\alpha}+\mu)\mathcal{N}(\check{\alpha};\check{r},s) + \tau\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s) \tag{3.74}$$

where in Equation (3.74), we have shifted everything to align with the Laplace mean μ, i.e., $\check{\alpha} = \alpha - \mu$, $\check{r} = r - \mu$. Then the normalisation constant in Equation (3.9) is given by

$$\mathcal{Z}_I = (1-\tau)\int_{-\infty}^{\infty}\delta_{\text{Dirac}}(\check{\alpha}+\mu)\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} + \tau\int_{-\infty}^{\infty}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.75}$$

$$= (1-\tau)\int_{-\infty}^{\infty}\delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s)d\alpha + \tau\int_{-\infty}^{\infty}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.76}$$

$$= (1-\tau)\mathcal{N}(0;r,s) + \tau\int_{-\infty}^{\infty}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.77}$$

where we have used the sampling property of the generalised Dirac delta function. The absolute value in the integrand in Equation (3.77) requires considering the two cases: $\check{\alpha} < 0$, $\check{\alpha} > 0$, separately. For $\check{\alpha} < 0$, we have

$$\frac{\lambda}{2}\exp(\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s) = \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-\check{r})^2 - \lambda\check{\alpha}2s}{2s}\right) \tag{3.78}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{\check{\alpha}^2+\check{r}^2 - 2\check{\alpha}(\check{r}+\lambda s)}{2s}\right) \tag{3.79}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-(\check{r}+\lambda s))^2 - (\lambda s)^2 - 2\check{r}\lambda s}{2s}\right) \tag{3.80}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-\underline{r})^2}{2s}\right)\exp\left(\frac{1}{2}\lambda^2 s + \check{r}\lambda\right) \tag{3.81}$$

$$= \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s + \check{r}\lambda\right)\mathcal{N}(\check{\alpha};\underline{r},s) \tag{3.82}$$

for $\underline{r} = \check{r} + \lambda s$. Note that $\frac{1}{2}\lambda^2 s + \check{r}\lambda = \frac{\underline{r}^2 - \check{r}^2}{2s}$. Similarly, for $\check{\alpha} > 0$, we have

$$\frac{\lambda}{2}\exp(-\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s) = \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-\check{r})^2 + \lambda\check{\alpha}2s}{2s}\right) \tag{3.83}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{\check{\alpha}^2+\check{r}^2 - 2\check{\alpha}(\check{r}-\lambda s)}{2s}\right) \tag{3.84}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-(\check{r}-\lambda s))^2 - (\lambda s)^2 + 2\check{r}\lambda s}{2s}\right) \tag{3.85}$$

$$= \frac{1}{\sqrt{2\pi s}}\frac{\lambda}{2}\exp\left(-\frac{(\check{\alpha}-\bar{r})^2}{2s}\right)\exp\left(\frac{1}{2}\lambda^2 s - \check{r}\lambda\right) \tag{3.86}$$

$$= \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s - \check{r}\lambda\right)\mathcal{N}(\check{\alpha};\bar{r},s) \tag{3.87}$$

for $\bar{r} = \check{r} - \lambda s$. Note that $\frac{1}{2}\lambda^2 s - \check{r}\lambda = \frac{\bar{r}^2 - \check{r}^2}{2s}$.

Now returning to the Equation (3.77), we may split the integral in a lower and an upper part

$$\begin{aligned}
\mathcal{Z}_I &= (1-\tau)\mathcal{N}(0;r,s) \\
&\quad + \tau\left(\int_{-\infty}^{0}\frac{\lambda}{2}\exp(\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} + \int_{0}^{\infty}\frac{\lambda}{2}\exp(-\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha}\right) && (3.88)\\
&= (1-\tau)\mathcal{N}(0;r,s) \\
&\quad + \tau\left(\frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s+\check{r}\lambda\right)\int_{-\infty}^{0}\mathcal{N}(\check{\alpha};\underline{r},s)d\check{\alpha}\right. \\
&\quad\quad \left. + \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s-\check{r}\lambda\right)\int_{0}^{\infty}\mathcal{N}(\check{\alpha};\bar{r},s)d\check{\alpha}\right) && (3.89)\\
&= (1-\tau)\mathcal{N}(0;r,s) \\
&\quad + \tau\left(\frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s+\check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{r}}{\sqrt{s}}\right) + \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s-\check{r}\lambda\right)\left(1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)\right)\right) && (3.90)\\
&= (1-\tau)\mathcal{N}(0;r,s) \\
&\quad + \tau\left(\frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s+\check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{r}}{\sqrt{s}}\right) + \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s-\check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)\right) && (3.91)\\
&= (1-\tau)\mathcal{N}(0;r,s) + \tau(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) && (3.92)
\end{aligned}$$

for

$$\underline{\mathcal{Z}}_I = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s+\check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{r}}{\sqrt{s}}\right) \tag{3.93}$$

$$\bar{\mathcal{Z}}_I = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s-\check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right) \tag{3.94}$$

where we have introduced the cumulative distribution function (cdf) $\Phi_{\mathcal{N}}(\check{x}) = \int_{-\infty}^{\check{x}}\phi_{\mathcal{N}}(t)\,dt = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\check{x}}\exp\left(-\frac{t^2}{2}\right)dt$ of a standard normal distribution (with probability density function (pdf) $\phi_{\mathcal{N}}(\check{x}) = \mathcal{N}(\check{x},0,1) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\check{x}^2}{2}\right)$). Note that we have the following symmetry relations

$$\phi_{\mathcal{N}}(-\check{x}) = \phi_{\mathcal{N}}(\check{x}) \tag{3.95}$$

$$\Phi_{\mathcal{N}}(-\check{x}) = 1-\Phi_{\mathcal{N}}(\check{x}) \tag{3.96}$$

Using techniques similar to those used above for deriving $\mathcal{Z}_I$, we have the following expression for the N_1 quantity in Equation (3.10)

$$\begin{aligned}
N_1 &= (1-\tau)\int_{-\infty}^{\infty}(\check{\alpha}+\mu)\delta_{\text{Dirac}}(\check{\alpha}+\mu)\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \\
&\quad + \tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} && (3.97)\\
&= (1-\tau)\int_{-\infty}^{\infty}\alpha\delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s)d\alpha \\
&\quad + \tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} && (3.98)\\
&= \tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} && (3.99)\\
&= \mu\tau\int_{-\infty}^{\infty}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} + \tau\int_{-\infty}^{\infty}\check{\alpha}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} && (3.100)\\
&= \mu\tau(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\int_{-\infty}^{\infty}\check{\alpha}\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} && (3.101)
\end{aligned}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\int_{-\infty}^{0} \check{\alpha}\frac{\lambda}{2}\exp(\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} + \int_{0}^{\infty} \check{\alpha}\frac{\lambda}{2}\exp(-\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha}\right) \tag{3.102}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s + \check{r}\lambda\right)\int_{-\infty}^{0} \check{\alpha}\mathcal{N}(\check{\alpha};\underline{r},s)d\check{\alpha} + \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s - \check{r}\lambda\right)\int_{0}^{\infty} \check{\alpha}\mathcal{N}(\check{\alpha};\bar{r},s)d\check{\alpha}\right) \tag{3.103}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s + \check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{r}}{\sqrt{s}}\right)\int_{-\infty}^{0} \check{\alpha}\frac{\mathcal{N}(\check{\alpha};\underline{r},s)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}d\check{\alpha} + \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s - \check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)\int_{0}^{\infty} \check{\alpha}\frac{\mathcal{N}(\check{\alpha};\bar{r},s)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}d\check{\alpha}\right) \tag{3.104}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\int_{-\infty}^{0} \check{\alpha}\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-\underline{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}d\check{\alpha} + \bar{\mathcal{Z}}_I\int_{0}^{\infty} \check{\alpha}\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}d\check{\alpha}\right) \tag{3.105}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\int_{-\infty}^{0} \check{\alpha}\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-\underline{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}d\check{\alpha} + \bar{\mathcal{Z}}_I\int_{0}^{\infty} \check{\alpha}\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-\bar{r}}{\sqrt{s}}\right)}{1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}d\check{\alpha}\right) \tag{3.106}$$

Now, consider the *double truncated normal distribution* with probability density function [42]

$$\mathcal{TN}(\check{x},\xi,\sigma^2,a,b) = \begin{cases} 0, & \check{x} < a \\ \frac{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(\check{x}-\xi)^2}{2\sigma^2}\right)}{\int_a^b \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(t-\xi)^2}{2\sigma^2}\right)dt} = \frac{\phi_{\mathcal{N}}\left(\frac{\check{x}-\xi}{\sigma}\right)}{\sigma\left(\Phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right)-\Phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right)\right)}, & a \le \check{x} \le b \\ 0, & \check{x} > b \end{cases} \tag{3.107}$$

with mean and variance [42]

$$\mathbb{E}[\check{x}] = \int_{-\infty}^{\infty} \check{x}\mathcal{TN}(\check{x},\xi,\sigma^2,a,b)d\check{x} \tag{3.108}$$

$$= \xi + \sigma\frac{\phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right) - \phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right)}{\Phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right) - \Phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right)} \tag{3.109}$$

$$\mathrm{Var}(\check{x}) = \int_{-\infty}^{\infty} (\check{x} - \mathbb{E}[\check{x}])^2\mathcal{TN}(\check{x},\xi,\sigma^2,a,b)d\check{x} \tag{3.110}$$

$$= \sigma^2\left(1 + \frac{\frac{a-\xi}{\sigma}\phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right) - \frac{b-\xi}{\sigma}\phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right)}{\Phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right) - \Phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right)} - \left(\frac{\phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right) - \phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right)}{\Phi_{\mathcal{N}}\left(\frac{b-\xi}{\sigma}\right) - \Phi_{\mathcal{N}}\left(\frac{a-\xi}{\sigma}\right)}\right)^2\right) \tag{3.111}$$

Returning to Equation (3.106), we find that the integrals in that equation correspond to the mean values of two *singly truncated normal distributions*

$$N_1 = \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\int_{-\infty}^{0} \check{\alpha}\mathcal{TN}(\check{\alpha},\underline{r},s,-\infty,0)d\check{\alpha} + \bar{\mathcal{Z}}_I\int_{0}^{\infty} \check{\alpha}\mathcal{TN}(\check{\alpha},\bar{r},s,0,\infty)d\check{\alpha}\right) \tag{3.112}$$

Thus, using the expression in Equation (3.109) (handling $\phi_{\mathcal{N}}(\infty)$ properly as a limit - see also [43]), we have

$$N_1 = \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\left(\underline{r} + \sqrt{s}\frac{-\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right) + \bar{\mathcal{Z}}_I\left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}{1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}\right)\right) \tag{3.113}$$

$$= \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right) + \bar{\mathcal{Z}}_I\left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)\right) \tag{3.114}$$

Using techniques similar to those used above for deriving N_1, we have the following expression for the N_{2a} quantity in Equation (3.11)

$$N_{2a} = (1-\tau)\int_{-\infty}^{\infty}(\check{\alpha}+\mu)^2\delta_{\text{Dirac}}(\check{\alpha}+\mu)\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha}$$
$$+\tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)^2\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.115}$$
$$= (1-\tau)\int_{-\infty}^{\infty}\alpha^2\delta_{\text{Dirac}}(\alpha)\mathcal{N}(\alpha;r,s)d\alpha$$
$$+\tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)^2\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.116}$$
$$= \tau\int_{-\infty}^{\infty}(\check{\alpha}+\mu)^2\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.117}$$
$$= \tau\int_{-\infty}^{\infty}(\mu^2+2\mu\check{\alpha}+\check{\alpha}^2)\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.118}$$
$$= \tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + 2\mu(N_1 - \tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I))$$
$$+\tau\int_{-\infty}^{\infty}\check{\alpha}^2\frac{\lambda}{2}\exp(-\lambda|\check{\alpha}|)\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} \tag{3.119}$$
$$= -\tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + 2\mu N_1$$
$$+\tau\left(\int_{-\infty}^{0}\check{\alpha}^2\frac{\lambda}{2}\exp(\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha} + \int_{0}^{\infty}\check{\alpha}^2\frac{\lambda}{2}\exp(-\lambda\check{\alpha})\,\mathcal{N}(\check{\alpha};\check{r},s)d\check{\alpha}\right) \tag{3.120}$$
$$= -\tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + 2\mu N_1$$
$$+\tau\left(\underline{\mathcal{Z}}_I\int_{-\infty}^{0}\check{\alpha}^2\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}d\check{\alpha} + \bar{\mathcal{Z}}_I\int_{0}^{\infty}\check{\alpha}^2\frac{\frac{1}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{\check{\alpha}-\bar{r}}{\sqrt{s}}\right)}{1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}d\check{\alpha}\right) \tag{3.121}$$

The integrals in Equation (3.121) correspond to the second moments of two *singly truncated normal distributions*

$$N_{2a} = -\tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + 2\mu N_1$$
$$+\tau\left(\underline{\mathcal{Z}}_I\int_{-\infty}^{0}\check{\alpha}^2\mathcal{TN}(\check{\alpha},\underline{r},s,-\infty,0)d\check{\alpha} + \bar{\mathcal{Z}}_I\int_{0}^{\infty}\check{\alpha}^2\mathcal{TN}(\check{\alpha},\bar{r},s,0,\infty)d\check{\alpha}\right) \tag{3.122}$$

Thus, using $\mathbb{E}[\check{x}^2] = \mathrm{Var}(\check{x}) + \mathbb{E}[\check{x}]^2$ together with Equation (3.111), we get

$$\begin{aligned}
N_{2a} &= -\tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + 2\mu N_1 \\
&\quad + \tau\left(\underline{\mathcal{Z}}_I\left[s\left(1 + \frac{-\frac{-r}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)} - \left(\frac{-\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right)^2\right) + \left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right)^2\right]\right. \\
&\quad \left. + \bar{\mathcal{Z}}_I\left[s\left(1 + \frac{\frac{-\bar{r}}{\sqrt{s}}\phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}{1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)} - \left(\frac{\phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}{1-\Phi_{\mathcal{N}}\left(\frac{-\bar{r}}{\sqrt{s}}\right)}\right)^2\right) + \left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)^2\right]\right)
\end{aligned} \tag{3.123}$$

$$\begin{aligned}
&= 2\mu N_1 - \tau\mu^2(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) \\
&\quad + \tau\left(\underline{\mathcal{Z}}_I\left[s\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)} - \frac{\underline{r}}{\sqrt{s}}\right)\right) + \left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right)^2\right]\right. \\
&\quad \left. + \bar{\mathcal{Z}}_I\left[s\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)} + \frac{\bar{r}}{\sqrt{s}}\right)\right) + \left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)^2\right]\right)
\end{aligned} \tag{3.124}$$

Finally, we arrive at the following expressions for the i.i.d. BL channel functions based on Equations (3.12) and (3.13)

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{N_1}{\mathcal{Z}_I} \tag{3.125}$$

$$= \frac{\tau\mu(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) + \tau\left(\underline{\mathcal{Z}}_I\left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right) + \bar{\mathcal{Z}}_I\left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)\right)}{\mathcal{Z}_I} \tag{3.126}$$

$$= \tau\left(\mu\frac{(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)}{\mathcal{Z}_I} + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_I}\left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right) + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I}\left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)\right) \tag{3.127}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{N_{2a}}{\mathcal{Z}_I} - f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I)^2 \tag{3.128}$$

$$\begin{aligned}
&= 2\mu f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) + \tau\left\{ - \mu^2\frac{(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)}{\mathcal{Z}_I}\right. \\
&\quad + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_{\mathcal{I}}}\left[s\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)} - \frac{\underline{r}}{\sqrt{s}}\right)\right) + \left(\underline{r} - \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-r}{\sqrt{s}}\right)}\right)^2\right] \\
&\quad \left. + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I}\left[s\left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\left(\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)} + \frac{\bar{r}}{\sqrt{s}}\right)\right) + \left(\bar{r} + \sqrt{s}\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right)}\right)^2\right]\right\} \\
&\quad - f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I)^2
\end{aligned} \tag{3.129}$$

for

$$\mathcal{Z}_I = (1-\tau)\mathcal{N}(0; r, s) + \tau(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I) \tag{3.130}$$

$$\underline{\mathcal{Z}}_I = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s + \check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{-\underline{r}}{\sqrt{s}}\right) \tag{3.131}$$

$$\bar{\mathcal{Z}}_I = \frac{\lambda}{2}\exp\left(\frac{1}{2}\lambda^2 s - \check{r}\lambda\right)\Phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s}}\right) \tag{3.132}$$

$$\check{r} = r - \mu \tag{3.133}$$

$$\underline{r} = \check{r} + \lambda s \tag{3.134}$$

$$\bar{r} = \check{r} - \lambda s \tag{3.135}$$

3.7.1.1 Numerical Accuracy Considerations for the i.i.d. BL Input Channel

When implementing these channel functions, one has to pay attention to fractions of the type $\frac{\phi_{\mathcal{N}}(\check{x})}{\Phi_{\mathcal{N}}(\check{x})}$. For such fractions, improved numerical accuracy may be obtained[1] by using a reasonable implementation[2] of the scaled complementary error function [44]

$$\operatorname{erfcx}(\check{x}) \coloneqq \exp\left(\check{x}^2\right)\frac{2}{\sqrt{\pi}}\int_{\check{x}}^{\infty}\exp\left(-t^2\right)dt \tag{3.136}$$

For the scaled complementary error function, we have

$$\operatorname{erfcx}\left(\frac{-\check{x}}{\sqrt{2}}\right) = \exp\left(\left(\frac{-\check{x}}{\sqrt{2}}\right)^2\right)\frac{2}{\sqrt{\pi}}\int_{\frac{-\check{x}}{\sqrt{2}}}^{\infty}\exp\left(-t^2\right)dt \tag{3.137}$$

$$= \exp\left(\frac{\check{x}^2}{2}\right)\frac{2}{\sqrt{\pi}}\int_{-\infty}^{\frac{\check{x}}{\sqrt{2}}}\exp\left(-t^2\right)dt \tag{3.138}$$

$$= \exp\left(\frac{\check{x}^2}{2}\right)\frac{2}{\sqrt{2\pi}}\int_{-\infty}^{\check{x}}\exp\left(-\left(\frac{t}{\sqrt{2}}\right)^2\right)dt \tag{3.139}$$

$$= \exp\left(\frac{\check{x}^2}{2}\right)\frac{2}{\sqrt{2\pi}}\int_{-\infty}^{\check{x}}\exp\left(-\frac{t^2}{2}\right)dt \tag{3.140}$$

$$= \exp\left(\frac{\check{x}^2}{2}\right)2\Phi_{\mathcal{N}}(\check{x}) \tag{3.141}$$

which means that

$$\frac{\phi_{\mathcal{N}}(\check{x})}{\Phi_{\mathcal{N}}(\check{x})} = \frac{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\check{x}^2}{2}\right)}{\frac{\operatorname{erfcx}\left(\frac{-\check{x}}{\sqrt{2}}\right)}{2\exp\left(\frac{\check{x}^2}{2}\right)}} \tag{3.142}$$

$$= \frac{\frac{2}{\sqrt{2\pi}}\exp\left(-\frac{\check{x}^2}{2}\right)\exp\left(\frac{\check{x}^2}{2}\right)}{\operatorname{erfcx}\left(\frac{-\check{x}}{\sqrt{2}}\right)} \tag{3.143}$$

$$= \frac{\frac{2}{\sqrt{2\pi}}}{\operatorname{erfcx}\left(\frac{-\check{x}}{\sqrt{2}}\right)} \tag{3.144}$$

[1] This use of complementary error function was inspired by its use in the `ElasticNetEstimIn.m` file in the GAMPMatlab Toolbox version 20161005 available at `https://sourceforge.net/projects/gampmatlab/`. See also Section 7.2 for more information about the GAMPMatlab Toolbox.

[2] See: `http://scipy.github.io/devdocs/special.html#error-function-and-fresnel-integrals` and `http://ab-initio.mit.edu/wiki/index.php/Faddeeva_Package`

3.7.1.2 BL Expressions for use with the GWS Input Channel

Based on the result in Equations (3.127) and (3.129), we may identify the following i.i.d BL channel updates to be used in the GWS framework described in Section 3.5

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \Bigg\{ \mu_j + \frac{\underline{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \Bigg\} \tag{3.145}$$

$$\begin{aligned} f_{\tilde{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = 2\mu f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) + \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \Bigg\{ -\mu_j^2 \\ + \frac{\underline{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left[s_j \left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \left(\frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} - \frac{\underline{r}_j}{\sqrt{s_j}} \right) \right) + \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right)^2 \right] \\ + \frac{\bar{\mathcal{Z}}_{I_j}}{\mathcal{Z}_{\varphi_j}} \left[s_j \left(1 - \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \left(\frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} + \frac{\bar{r}_j}{\sqrt{s_j}} \right) \right) + \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right)^2 \right] \Bigg\} \\ - f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j)^2 \end{aligned} \tag{3.146}$$

for

$$\mathcal{Z}_{\varphi_j} = \underline{\mathcal{Z}}_{I_j} + \bar{\mathcal{Z}}_{I_j} \tag{3.147}$$

$$\underline{\mathcal{Z}}_{I_j} = \frac{\lambda_j}{2} \exp\left(\frac{1}{2} \lambda_j^2 s_j + \check{r}_j \lambda_j \right) \Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}} \right) \tag{3.148}$$

$$\bar{\mathcal{Z}}_{I_j} = \frac{\lambda_j}{2} \exp\left(\frac{1}{2} \lambda_j^2 s_j - \check{r}_j \lambda_j \right) \Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}} \right) \tag{3.149}$$

$$\check{r}_j = r_j - \mu_j \tag{3.150}$$

$$\underline{r}_j = \check{r}_j + \lambda_j s_j \tag{3.151}$$

$$\bar{r}_j = \check{r}_j - \lambda_j s_j \tag{3.152}$$

3.7.2 I.i.d. Sparse Bernoulli-Gauss Input Channel

For an i.i.d. BG input channel with signal density τ and Gaussian mean $\bar{\theta}$ and variance $\tilde{\theta}$ ($\boldsymbol{\theta}_I = [\tau, \bar{\theta}, \tilde{\theta}]^T$), i.e.

$$p(\alpha; \boldsymbol{\theta}_I) = (1-\tau)\delta_{\mathrm{Dirac}}(\alpha) + \tau \frac{1}{\sqrt{2\pi\tilde{\theta}}} \exp\left(-\frac{(\alpha - \bar{\theta})^2}{2\tilde{\theta}} \right) \tag{3.153}$$

we have channel functions [5][3] (Eqs. (68), (69))

$$f_{\bar{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{\tau \mathit{a} \mathit{b} \frac{1}{s+\tilde{\theta}} \mathit{c}}{(1-\tau)\mathit{d} + \tau \mathit{b} \mathit{a}} \tag{3.154}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{\tau(1-\tau)\mathit{d}\mathit{a}\mathit{b}\frac{1}{(s+\tilde{\theta})^2}(\tilde{\theta}s(s+\tilde{\theta}) + \mathit{c}^2) + \tau^2 \mathit{a}\tilde{\theta}\mathit{b}^4}{((1-\tau)\mathit{d} + \tau \mathit{b}\mathit{a})^2} \tag{3.155}$$

for

$$\mathit{a} \coloneqq \exp\left(-\frac{(r-\bar{\theta})^2}{2(s+\tilde{\theta})} \right) \tag{3.156}$$

[3] In [5] the full expressions for the channel functions are given, i.e. the intermediate variables $\mathit{a}, \mathit{b}, \mathit{c}, \mathit{d}$ are not used. These variables have been introduced by the authors of this note.

$$\mathcal{b} \coloneqq \frac{\sqrt{s}}{\sqrt{s+\tilde{\theta}}} \tag{3.157}$$

$$\mathcal{c} \coloneqq \bar{\theta}s + r\tilde{\theta} \tag{3.158}$$

$$\mathcal{d} \coloneqq \exp\left(-\frac{r^2}{2s}\right) \tag{3.159}$$

Manoel and Tramel suggested the following implementation[4] of the i.i.d. BG input channel functions

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{\frac{\mathcal{c}}{s+\tilde{\theta}}}{\mathcal{f}+1} \tag{3.160}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathcal{f}f^2_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) + \frac{\mathcal{e}}{\mathcal{f}+1} \tag{3.161}$$

for

$$\mathcal{e} \coloneqq \frac{\tilde{\theta}s}{s+\tilde{\theta}} \left[= \tilde{\theta}\mathcal{b}^2\right] \tag{3.162}$$

$$\mathcal{f} \coloneqq \frac{1-\tau}{\tau}\sqrt{\frac{\tilde{\theta}}{\mathcal{e}}}\exp\left(-\frac{1}{2}\left(\frac{r^2}{s} - \frac{(r-\bar{\theta})^2}{s+\tilde{\theta}}\right)\right)\left[= \frac{1-\tau}{\tau}\frac{1}{\mathcal{b}}\frac{\mathcal{d}}{\mathcal{a}}\right] \tag{3.163}$$

Parker suggested the following implementation of the i.i.d. BG input channel functions [15]

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \frac{\mathcal{g}}{\mathcal{k}} \tag{3.164}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathcal{g}^2\frac{\mathcal{k}-1}{\mathcal{k}^2} + \frac{\mathcal{h}}{\mathcal{k}} \tag{3.165}$$

for

$$\mathcal{g} \coloneqq \frac{\frac{\bar{\theta}}{\tilde{\theta}} + \frac{r}{s}}{\frac{1}{\tilde{\theta}} + \frac{1}{s}}\left[= \frac{\mathcal{c}}{s+\tilde{\theta}}\right] \tag{3.166}$$

$$\mathcal{h} \coloneqq \frac{1}{\frac{1}{\tilde{\theta}} + \frac{1}{s}}\left[= \mathcal{e}\right] \tag{3.167}$$

$$\mathcal{k} \coloneqq 1 + \frac{1-\tau}{\tau}\sqrt{\frac{\tilde{\theta}}{\mathcal{h}}}\exp\left(\frac{1}{2}\left[\frac{(r-\bar{\theta})^2}{\tilde{\theta}+s} - \frac{r^2}{s}\right]\right)\left[= 1 + \frac{1-\tau}{\tau}\frac{1}{\mathcal{b}}\frac{\mathcal{d}}{\mathcal{a}}\right] \tag{3.168}$$

However, in EM-BG-GAMP [39], the following slightly modified implementation of the i.i.d. BG input channel is suggested

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathcal{l}\mathcal{g}\left[= \frac{\mathcal{g}}{\mathcal{k}}\right] \tag{3.169}$$

$$f_{\tilde{\alpha}}(s, r; \boldsymbol{\theta}_I) = \mathcal{l}(\mathcal{h} + |\mathcal{g}|^2) - \mathcal{l}^2|\mathcal{g}|^2\left[= \frac{\mathcal{h}+|\mathcal{g}|^2}{\mathcal{k}} - \frac{|\mathcal{g}|^2}{\mathcal{k}^2} = \frac{\mathcal{k}\mathcal{h}+|\mathcal{g}|^2(\mathcal{k}-1)}{\mathcal{k}^2} = |\mathcal{g}|^2\frac{\mathcal{k}-1}{\mathcal{k}^2} + \frac{\mathcal{h}}{\mathcal{k}}\right] \tag{3.170}$$

for

$$\mathcal{l} \coloneqq \frac{1}{1 + \left(\frac{\tau}{1-\tau}\frac{\frac{1}{\sqrt{\tilde{\theta}+s}}\exp\left(-\frac{1}{2}\frac{(r-\bar{\theta})^2}{\tilde{\theta}+s}\right)}{\frac{1}{\sqrt{s}}\exp\left(-\frac{1}{2}\frac{r^2}{s}\right)}\right)^{-1}}\left[= \frac{1}{1+\left(\frac{\tau}{1-\tau}\mathcal{b}\frac{\mathcal{a}}{\mathcal{d}}\right)^{-1}} = \frac{1}{\mathcal{k}}\right] \tag{3.171}$$

Note that the $\mathcal{l}$ in Equation (3.171) corresponds to the $\pi(r, s, \boldsymbol{\theta}_I)$ in Equation (3.41). Also note that the absolute values are included for generality since the case of real valued Gaussians may be extended to the case of circular-complex-Gaussians [39], [28].

[4] See: `https://github.com/eric-tramel/SwAMP-Demo/blob/master/python/amp.py`

3.7.2.1 BG Expressions for use with the GWS Input Channel

Based on the result in Equations (3.170) and (3.170) as well as the multiplication rule for two Gaussian densities given in [28], we may identify the following i.i.d. BG channel updates to be used in the GWS framework described in Section 3.5

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \left(\frac{\frac{\bar{\theta}_j}{\bar{\theta}_j} + \frac{r_j}{s_j}}{\frac{1}{\bar{\theta}_j} + \frac{1}{s_j}} \right) \tag{3.172}$$

$$f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j) = \pi_j^{\mathrm{w}}(r_j, s_j, [\boldsymbol{\theta}_I]_j) \left(\frac{1}{\frac{1}{\bar{\theta}_j} + \frac{1}{s_j}} + \left(\frac{\frac{\bar{\theta}_j}{\bar{\theta}_j} + \frac{r_j}{s_j}}{\frac{1}{\bar{\theta}_j} + \frac{1}{s_j}} \right)^2 \right) - f_{\bar{\alpha}_j}(s_j, r_j; [\boldsymbol{\theta}_I]_j)^2 \tag{3.173}$$

4 Sum Approximations

From an implementation point of view, a critical element in the GAMP iteration in Equations (2.1)-(2.12) is the application of the entrywise absolute value squared system matrix

$$|\mathbf{A}|^{\circ 2} = \mathbf{A}_{\text{asq}} = \begin{bmatrix} |a_{11}|^2 & \cdots & |a_{1n}|^2 \\ \vdots & \ddots & \vdots \\ |a_{m1}|^2 & \cdots & |a_{mn}|^2 \end{bmatrix} \tag{4.1}$$

If the system matrix $\mathbf{A}$ is given explicitly, one may easily find $|\mathbf{A}|^{\circ 2}$ using Equation (4.1). However, oftentimes (especially when considering large problem sizes) it is a necessity to use a *fast transform* for implementing the matrix-vector products involving $\mathbf{A}$ and $\mathbf{A}^H$ in the GAMP algorithm in order to achieve acceptable reconstruction times and reasonable memory requirements [28], [29]. For instance, one may use a Fast Fourier Transform (FFT) based method to implement a matrix-vector product involving a Discrete Fourier Transform (DFT). However, such fast transforms are not always available for implementing the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$. As an alternative, one may use sum approximation (also known as uniform variance) GAMP. The idea, then, is to approximate the matrix-vector products involving $|\mathbf{A}|^{\circ 2}$ by certain sums.

4.1 The Sum Approximation by Krzakala et al.

In [5], Krzakala et al. consider the case where $\mathbf{A}$ is a homogeneous matrix with i.i.d. random entries having zero mean and variance $\frac{1}{n}$. In this case, the ensemble average of different realisations of $|\mathbf{A}|^{\circ 2}$ is

$$\mathbb{E}[|\mathbf{A}|^{\circ 2}] = \begin{bmatrix} \mathbb{E}[|a_{11}|^2] & \cdots & \mathbb{E}[|a_{1n}|^2] \\ \vdots & \ddots & \vdots \\ \mathbb{E}[|a_{m1}|^2] & \cdots & \mathbb{E}[|a_{mn}|^2] \end{bmatrix} \tag{4.2}$$

$$= \begin{bmatrix} \text{Var}(a_{11}) & \cdots & \text{Var}(a_{1n}) \\ \vdots & \ddots & \vdots \\ \text{Var}(a_{m1}) & \cdots & \text{Var}(a_{mn}) \end{bmatrix} \tag{4.3}$$

$$= \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix} \tag{4.4}$$

since the entries of $\mathbf{A}$ are zero mean. Now, one may consider an approximation of e.g. the GAMP factor side update in Equation (2.1)

$$\bar{v}^{t+1} = \sum_j \mathbb{E}[|\mathbf{A}|^{\circ 2}]_{ij} \tilde{\alpha}_j^t \tag{4.5}$$

$$= \frac{1}{n} \sum_j \tilde{\alpha}_j^t \tag{4.6}$$

Since the variance $\overline{(v_i^{t+1} - \bar{v}^{t+1})^2}$ is of order $\mathcal{O}\left(\frac{1}{n}\right)$ (see the details in Eq. (55) in [5]), in the large system limit as $n \to \infty$, one may consider all v_i^{t+1} equal to their average $\bar{v}^{t+1}$. Similar arguments may be used for other GAMP updates that involve $|\mathbf{A}|^{\circ 2}$. Thus, one may replace all GAMP instances of $|\mathbf{A}|^{\circ 2}$ with sums scaled by $\frac{1}{n}$, which yields the following alternatives to the GAMP

updates in Equations (2.1) and (2.7)

$$\bar{v}^{t+1} = \frac{1}{n}\sum_j \tilde{\alpha}_j^t \tag{4.7}$$

$$\bar{s}^{t+1} = \left[\frac{1}{n}\sum_i u_i^{t+1}\right]^{-1} \tag{4.8}$$

which may then be used in place of all v_i^{t+1}, s_j^{t+1} respectively, in the GAMP iteration in Equations (2.1)-(2.12). Note that the sum approximation by Krzakala et al. (with an assumed variance of $\frac{1}{m}$ instead of $\frac{1}{n}$) is closely related to the Donoho/Maleki/Montanari AMP as described in Section 2.1.

4.2 The Sum Approximation by Rangan

In [16], Rangan considers a sightly different approximation based on the assumption that $|a_{ij}|^2 \approx \frac{||\mathbf{A}||_F^2}{mn}$ for all i, j, where $||\mathbf{A}||_F^2$ is the Frobenius norm of the matrix $\mathbf{A}$. Rangan then forces all variance related components for each GAMP state to be the same. That is, $v_i^{t+1} = \breve{v}^{t+1}$, $u_i^{t+1} = \breve{u}^{t+1}$, $s_j^{t+1} = \breve{s}^{t+1}$, $\tilde{\alpha}_j^{t+1} = \breve{\alpha}^{t+1}$ for all i, j. Specifically, Rangan's MMSE GAMP iteration with scalar variances reads

Output (factor) side updates:

$$\breve{v}^{t+1} = \frac{1}{m}||\mathbf{A}||_F^2 \breve{\alpha}^t \tag{4.9}$$

$$o_i^{t+1} = \sum_j a_{ij}\bar{\alpha}_j^t - \breve{v}^{t+1} q_i^t \tag{4.10}$$

$$\bar{z}_i^{t+1} = f_{\bar{z}_i}(\breve{v}^{t+1}, o_i^{t+1}; y_i, [\boldsymbol{\theta}_o]_i^t) \tag{4.11}$$

$$\tilde{z}_i^{t+1} = f_{\tilde{z}_i}(\breve{v}^{t+1}, o_i^{t+1}; y_i, [\boldsymbol{\theta}_o]_i^t) \tag{4.12}$$

$$q_i^{t+1} = \frac{\bar{z}_i^{t+1} - o_i^{t+1}}{\breve{v}^{t+1}} \tag{4.13}$$

$$\breve{u}^{t+1} = \frac{1}{m}\sum_i \frac{\breve{v}^{t+1} - \tilde{z}_i^{t+1}}{(\breve{v}^{t+1})^2} \tag{4.14}$$

Input (variable) side updates:

$$\breve{s}^{t+1} = \left[\frac{1}{n}||\mathbf{A}||_F^2 \breve{u}^{t+1}\right]^{-1} \tag{4.15}$$

$$r_j^{t+1} = \bar{\alpha}_j^t + \breve{s}^{t+1}\sum_i a_{ij}^* q_i^{t+1} \tag{4.16}$$

$$\bar{\alpha}_j^{t+1} = f_{\bar{\alpha}_j}(\breve{s}^{t+1}, r_j^{t+1}; [\boldsymbol{\theta}_I]_j^t) \tag{4.17}$$

$$\breve{\alpha}^{t+1} = \frac{1}{n}\sum_j f_{\tilde{\alpha}_j}(\breve{s}^{t+1}, r_j^{t+1}; [\boldsymbol{\theta}_I]_j^t) \tag{4.18}$$

Optional parameter value updates (using e.g. EM - see also Section 6):

$$[\boldsymbol{\theta}_o]_i^{t+1} = \ldots \tag{4.19}$$

$$[\boldsymbol{\theta}_I]_j^{t+1} = \ldots \tag{4.20}$$

In [16], Rangan claims that simulations show that this simplified GAMP iteration works as well as the full (non-uniform variance) GAMP iteration in Equations (2.1) - (2.12). Unfortunately, neither the specific simulations nor any details about their nature are given in [16].

4.2.1 The Connection Between Sum Approximations by Krzakala et al. and Rangan

Rangan's simplifications are closely related to simplifications by Krzakala et al. To see this, note that for an $m \times n$ system matrix $\mathbf{A}$ with i.i.d. random entries having zero mean and variance σ^2, we have

$$\sigma^2 \approx \frac{1}{mn} \sum_i \sum_j |a_{ij}|^2 = \frac{1}{mn} ||\mathbf{A}||_F^2 \tag{4.21}$$

Thus, if $\sigma^2 = \frac{1}{n}$ as in the sum approximation by Krzakala et al. and $v_i^{t+1} \approx \breve{v}^{t+1}$, $s_j^{t+1} \approx \breve{s}^{t+1}$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$, we have

$$\begin{aligned}
\bar{v}^{t+1} &= \sigma^2 \sum_j \tilde{\alpha}_j^t && (4.22)\\
&\approx \frac{1}{mn} ||\mathbf{A}||_F^2 \sum_j \tilde{\alpha}_j^t && (4.23)\\
&\approx \frac{1}{m} ||\mathbf{A}||_F^2 \breve{\alpha}^t && (4.24)\\
&= \breve{v}^{t+1} && (4.25)\\
\bar{s}^{t+1} &= \left[\sigma^2 \sum_i u_i^{t+1} \right]^{-1} && (4.26)\\
&\approx \left[\frac{1}{mn} ||\mathbf{A}||_F^2 \sum_i u_i^{t+1} \right]^{-1} && (4.27)\\
&\approx \left[\frac{1}{n} ||\mathbf{A}||_F^2 \breve{u}^{t+1} \right]^{-1} && (4.28)\\
&= \breve{s}^{t+1} && (4.29)
\end{aligned}$$

An interpretation of this result is that Rangan's sum approximation adapts to the system matrix $\mathbf{A}$ (and its variance) through $||\mathbf{A}||_F^2$.

4.2.2 Efficiently Computing the Frobenius Norm of the System Matrix

The need for finding $||\mathbf{A}||_F^2$ may make it infeasible to use Rangan's sum approximation in practical applications. For instance, if one attempts to use Rangan's method in order to avoid explicitly storing $|\mathbf{A}|^{\circ 2}$ in memory on a computer because it is infeasible to do so, a method for implicitly finding $||\mathbf{A}||_F^2$ is needed. If no such method is available and one has to explicitly store $\mathbf{A}$ in memory in order to estimate $||\mathbf{A}||_F^2$, no progress is made in using Rangan's sum approximation. We now discuss a few structured system matrices that allow for efficient (in terms of memory and computation requirements) ways to compute $||\mathbf{A}||_F^2$.

If $\mathbf{A}$ is defined by a Kronecker product, i.e. $\mathbf{A} = \mathbf{B} \otimes \mathbf{C} \in \mathbb{C}^{m \times n}$ for $\mathbf{B} \in \mathbb{C}^{o \times p}$, $\mathbf{C} \in \mathbb{C}^{q \times r}$ with $m = o \cdot q$, $n = p \cdot r$, then we have

$$\begin{aligned}
||\mathbf{A}||_F^2 &= \sum_{i'=0}^{m-1} \sum_{j'=0}^{n-1} |a_{i'j'}|^2 && (4.30)\\
&= \sum_{i=0}^{o-1} \sum_{j=0}^{p-1} \sum_{k=0}^{q-1} \sum_{l=0}^{r-1} |b_{ij} c_{kl}|^2 && (4.31)\\
&= \sum_{i=0}^{o-1} \sum_{j=0}^{p-1} \sum_{k=0}^{q-1} \sum_{l=0}^{r-1} |b|_{ij}^2 |c|_{kl}^2 && (4.32)\\
&= \left(\sum_{i=0}^{o-1} \sum_{j=0}^{p-1} |b|_{ij}^2 \right) \left(\sum_{k=0}^{q-1} \sum_{l=0}^{r-1} |c|_{kl}^2 \right) && (4.33)\\
&= ||\mathbf{B}||_F^2 ||\mathbf{C}||_F^2 && (4.34)
\end{aligned}$$

Using an inductive argument, it is easy to see from the above computations that this property of the Frobenius norm generalises to Kronecker products of more than two matrices.

Now consider all matrices that are arbitrary-sign adjusted, permuted, and scaled identity matrices, i.e. any matrix $\mathbf{G} \in \mathbb{C}^{n\times n}$ formed from an identity matrix by scaling all diagonal entries by an arbitrary factor $b \in \mathbb{C}$, followed by arbitrary sign changes on the diagonal entries, followed by an arbitrary permutation of either rows and/or columns. Now since we are only moving around and scaling (with a common absolute value of the scaling factor) all entries of the identity matrix, we have the following property of the Frobenius norm of the matrix products $\mathbf{H}_1\mathbf{G}$, $\mathbf{G}\mathbf{H}_2$ for any $\mathbf{H}_1 \in \mathbb{C}^{m\times n}$, $\mathbf{H}_2 \in \mathbb{C}^{n\times m}$:

$$||\mathbf{H}_1\mathbf{G}||_F^2 = \sum_{i'=0}^{m-1}\sum_{j'=0}^{n-1} |h_{1_{i'j'}} \cdot b|^2 \tag{4.35}$$

$$= |b|^2 \sum_{i'=0}^{m-1}\sum_{j'=0}^{n-1} |h_{1_{i'j'}}|^2 \tag{4.36}$$

$$= |b|^2 \cdot ||\mathbf{H}_1||_F^2 \tag{4.37}$$

$$||\mathbf{G}\mathbf{H}_2||_F^2 = \sum_{i''=0}^{n-1}\sum_{j''=0}^{m-1} |b \cdot h_{1_{i''j''}}|^2 \tag{4.38}$$

$$= |b|^2 \cdot \sum_{i''=0}^{n-1}\sum_{j''=0}^{m-1} |h_{1_{i''j''}}|^2 \tag{4.39}$$

$$= |b|^2 \cdot ||\mathbf{H}_2||_F^2 \tag{4.40}$$

$$\tag{4.41}$$

with the i', j' indexing $\mathbf{H}_1$ according to the permutations applied by $\mathbf{G}$ and the i'', j'' indexing $\mathbf{H}_2$ according to the permutations applied by $\mathbf{G}$.

Finally, we consider the Structurally Random Matrices (SRMs) detailed in [30]. These matrices are defined by

$$\mathbf{A} = \mathbf{D}_\Omega\mathbf{F}\mathbf{R} \tag{4.42}$$

with $\mathbf{D}_\Omega \in \mathbb{R}^{m\times n}$ a sub-sampling matrix that selects a (random) subset of the rows from $\mathbf{F}\mathbf{R}$ according to the indexing set Ω, i.e. it is an identity matrix with the rows not indexed by Ω removed, $\mathbf{F} \in \mathbb{R}^{n\times n}$ an orthogonal matrix, and $\mathbf{R} \in \mathbb{R}^{n\times n}$ a prerandomization matrix, i.e. either an identity matrix with uniformly random sign changes on the diagonal entries or a permutation matrix that permutes the columns of $\mathbf{F}$ at (uniformly) random. From Equation (4.37), we have $||\mathbf{F}\mathbf{R}||_F^2 = ||\mathbf{F}||_F^2$. The sub-sampling by $\mathbf{D}_\Omega$ results in keeping only a fraction of $\frac{m}{n}$ of the (unit vector) rows in $\mathbf{F}\mathbf{R}$. Thus, for $\mathbf{A}$ a SRM, we get

$$||\mathbf{A}||_F^2 = ||\mathbf{D}_\Omega\mathbf{F}\mathbf{R}||_F^2 \tag{4.43}$$

$$= \frac{m}{n}||\mathbf{F}\mathbf{R}||_F^2 \tag{4.44}$$

$$= \frac{m}{n}||\mathbf{F}||_F^2 \tag{4.45}$$

$$= \frac{m}{n}n \tag{4.46}$$

$$= m \tag{4.47}$$

If $\mathbf{F}$ is not an orthogonal matrix, Equation (4.45) is not in general valid. However, if we assume that all entries of $|\mathbf{F}|^{\circ 2}$ are of approximately the same size, i.e. $|f_{ij}|^2 \approx \frac{||\mathbf{F}||_F^2}{n^2}$ for all i, j (essentially the same assumption that is used in Rangan's sum approximation), we have

$$||\mathbf{A}||_F^2 \approx \frac{m}{n}||\mathbf{F}||_F^2 \tag{4.48}$$

for any such $\mathbf{F}$. Furthermore, if $\mathbf{F}$ is defined by a Kronecker product, we may use Equation (4.34) to compute $||\mathbf{F}||_F^2$.

5 Implementations of the GAMP Iteration

The GAMP iteration given in Equations (2.1)-(2.12) may be implemented in a number of ways. One may elect to combine some of the states, introduce new ones, or introduce convergence supporting heuristics. In this section, we present some of the implementations of the GAMP iteration that may be found in the literature.

The algorithms presented in this section are described using matrix-vector notation with Numpy broadcasting rules[1], e.g. multiplying two (column) vectors amounts to entrywise multiplication. The matrix-vector notation takes precedence over broadcasting[2], e.g. multiplying a matrix with a vector amounts to a matrix-vector multiplication - not a broadcast along the last axis. All vectors are column vectors. Also $\boldsymbol{\theta}_I$ and $\boldsymbol{\theta}_o$ are to be understood as being the relevant channel parameters for the given iteration as detailed in Sections 3.1 and 3.2. The channel functions $f_{\bar{z}}$, $f_{\tilde{z}}$, $f_{\bar{\alpha}}$, and $f_{\tilde{\alpha}}$ are scalar functions as noted in Section 3. Thus, when used on vectors these functions operate on each element of the vectors and produce a result vector having the individual scalar results as its entries. Consequently, the result is a vector of the same size as the input vectors.

Algorithm 1 details the implementation of the MMSE GAMP from [1], [16] as described by Parker [15]. Algorithm 1 is the GAMP variant used in Schniter's and Vila's EM-BG-AMP and EM-GM-AMP algorithms [39], [45], [28].

Algorithm 1 - MMSE GAMP [1], [15]

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\tilde{\boldsymbol{\alpha}}_0 = \mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})$, $\mathbf{q}_0 = \mathbf{0}_m$ # marginal conditional expectations
for $t = 1 \dots T_{\max}$ **do**
  $\mathbf{v}_t = \mathbf{A}_{\mathrm{asq}} \tilde{\boldsymbol{\alpha}}_{t\text{-}1}$
  $\mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \mathbf{v}_t \mathbf{q}_{t\text{-}1}$
  $\bar{\mathbf{z}}_t = f_{\bar{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\tilde{\mathbf{z}}_t = f_{\tilde{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\mathbf{q}_t = \frac{\bar{\mathbf{z}}_t - \mathbf{o}_t}{\mathbf{v}_t}$
  $\mathbf{u}_t = \frac{\mathbf{v}_t - \tilde{\mathbf{z}}_t}{\mathbf{v}_t^2}$
  $\mathbf{s}_t = [\mathbf{A}_{\mathrm{asq}}^T \mathbf{u}_t]^{-1}$
  $\mathbf{r}_t = \bar{\boldsymbol{\alpha}}_{t\text{-}1} + \mathbf{s}_t \mathbf{A}^H \mathbf{q}_t$
  $\bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  $\tilde{\boldsymbol{\alpha}}_t = f_{\tilde{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  if stop criterion is met **then**
    break
  end if
end for

Parker introduced some further modifications for numerical robustness of the GAMP algorithm [15]. This modified algorithm is detailed in Algorithm 2. Compared to Algorithm 1, two modifications are made:

- Introduction of a step-size (or damping) parameter κ.
- Re-scaling of several of the states to better handle high SNR cases.

The simplified sum approximation GAMP algorithms described Section 4 have similar implementations to the algorithms presented so far. In particular, the sum approximation by Krzakala et al. in Equations (4.7) and (4.8) may be implemented in Algorithm 1 be replacing $\mathbf{A}_{\mathrm{asq}}$ with $\frac{1}{n}\mathbf{1}_n^T$ and replacing $\mathbf{A}_{\mathrm{asq}}^T$ with $\frac{1}{n}\mathbf{1}_m^T$ which means that $\mathbf{v}_t$ and $\mathbf{s}_t$ become scalar. Rangan's simplified sum

[1] See: `http://docs.scipy.org/doc/numpy-1.10.1/user/basics.broadcasting.html`

[2] The way to think of it: Whenever you encounter an undefined operation in matrix-vector notation, then use the broadcasting rules.

Algorithm 2 - Numerically Robust MMSE GAMP with damping [15]

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\tilde{\boldsymbol{\alpha}}_0 = \mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})$, $\mathbf{q}_0 = \mathbf{0}_m$, $\mu_0 = 1$, $\mathbf{v}_0 = \mathbf{0}_m$ # marginal conditional expectations
for $t = 1 \dots T_{\max}$ **do**
  $\mathbf{v}_t = \kappa \mathbf{A}_{\text{asq}} \tilde{\boldsymbol{\alpha}}_{t\text{-}1} + (1-\kappa)\mathbf{v}_{t\text{-}1}$
  $\mu_t = \frac{1}{m}\mathbf{v}_t^{\mathrm{T}}\mathbf{1}_m$
  $\mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \frac{1}{\mu_t}\mathbf{v}_t\mathbf{q}_{t\text{-}1}$
  $\bar{\mathbf{z}}_t = f_{\bar{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\tilde{\mathbf{z}}_t = f_{\tilde{z}}(\mathbf{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\mathbf{q}_t = \kappa\mu_t \frac{\bar{\mathbf{z}}_t - \mathbf{o}_t}{\mathbf{v}_t} + (1-\kappa)\mathbf{q}_{t\text{-}1}$
  $\mathbf{u}_t = \kappa\mu_t \frac{\mathbf{v}_t - \tilde{\mathbf{z}}_t}{\mathbf{v}_t^2} + (1-\kappa)\mathbf{u}_{t\text{-}1}$
  $\breve{\boldsymbol{\alpha}}_t = \kappa\bar{\boldsymbol{\alpha}}_{t\text{-}1} + (1-\kappa)\breve{\boldsymbol{\alpha}}_{t\text{-}1}$
  $\mathbf{s}_t = [\mathbf{A}_{\text{asq}}^T \mathbf{u}_t]^{-1}$
  $\mathbf{r}_t = \breve{\boldsymbol{\alpha}}_t + \mathbf{s}_t \mathbf{A}^H \mathbf{q}_t$
  $\bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  $\tilde{\boldsymbol{\alpha}}_t = \mu_t f_{\tilde{\alpha}}(\mathbf{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  if stop criterion is met **then**
    break
  end if
end for

approximation GAMP iteration in Equations (4.9)-(4.20) may be implemented in a similar way to Algorithm 1 as detailed in Algorithm 3.

Algorithm 3 - MMSE GAMP with Rangan sum approximations [16]

initialise: $\bar{\boldsymbol{\alpha}}_0 = \mathbb{E}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}[\boldsymbol{\alpha}]$, $\breve{\alpha}_0 = \frac{1}{n}\sum\left(\mathrm{Var}_{\boldsymbol{\alpha}|\boldsymbol{\theta}_I}(\boldsymbol{\alpha})\right)$, $\mathbf{q}_0 = \mathbf{0}_m$ # marginal conditional expectations
for $t = 1 \dots T_{\max}$ **do**
  $\breve{v}_t = \frac{1}{m}||\mathbf{A}||_F^2 \breve{\alpha}_{t\text{-}1}$
  $\mathbf{o}_t = \mathbf{A}\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \breve{v}_t\mathbf{q}_{t\text{-}1}$
  $\bar{\mathbf{z}}_t = f_{\bar{z}}(\breve{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\tilde{\mathbf{z}}_t = f_{\tilde{z}}(\breve{v}_t, \mathbf{o}_t; \mathbf{y}, \boldsymbol{\theta}_o)$
  $\mathbf{q}_t = \frac{\bar{\mathbf{z}}_t - \mathbf{o}_t}{\breve{v}_t}$
  $\breve{u}_t = \frac{1}{m}\sum\left(\frac{\breve{v}_t - \tilde{\mathbf{z}}_t}{\breve{v}_t^2}\right)$
  $\breve{s}_t = [\frac{1}{n}||\mathbf{A}||_F^2 \breve{u}_t]^{-1}$
  $\mathbf{r}_t = \bar{\boldsymbol{\alpha}}_{t\text{-}1} + \breve{s}_t \mathbf{A}^H \mathbf{q}_t$
  $\bar{\boldsymbol{\alpha}}_t = f_{\bar{\alpha}}(\breve{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)$
  $\breve{\alpha}_t = \frac{1}{n}\sum\left(f_{\bar{\alpha}}(\breve{s}_t, \mathbf{r}_t; \boldsymbol{\theta}_I)\right)$
  if stop criterion is met **then**
    break
  end if
end for

5.1 Stop Criteria

The GAMP iteration in Equations (2.1)-(2.12) is to be iterated *until convergence*. However, in a practical setup one may terminate the iteration once the algorithm is *sufficiently close to convergence*. The challenge is then to find some criterion that describes when the algorithm has (almost) converged. Here we discuss a few such stop criteria that may be used with Algorithms 1-3. For a more general introduction to stop criteria for iterative methods see e.g. [46].

5.1.1 Mean Squared Error Stop Criterion

Since the GAMP iteration converges to a fixed point [25], it seems natural to stop the iteration once the change in the solutions between iterations becomes sufficiently small. If the change in the solution between iterations is measured in the 2-norm, we have a mean squared error (MSE) stop criterion

$$\frac{1}{n}||\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \bar{\boldsymbol{\alpha}}_t||_2^2 < \epsilon \tag{5.1}$$

for some tolerance ϵ.

Note that if the algorithm stalls for some iterations, too early termination may occur when using this MSE stop criterion. It is the authors' experience that this may indeed happen with GAMP in some cases.

5.1.2 Normalised Mean Squared Error Stop Criterion

A stop criterion related to the MSE stop criterion in Equation (5.1) is the normalised mean squared error (NMSE) stop criterion used in e.g. [28], [31], [40]

$$\frac{||\bar{\boldsymbol{\alpha}}_{t\text{-}1} - \bar{\boldsymbol{\alpha}}_t||_2^2}{||\bar{\boldsymbol{\alpha}}_{t\text{-}1}||_2^2} < \epsilon \tag{5.2}$$

for some tolerance ϵ.

Note that this criterion is subject to a potential division by zero problem if $\bar{\boldsymbol{\alpha}}_{t\text{-}1} = \mathbf{0}$ which happens if the solution vector is initialised to the zero-vector. It is the authors' experience that this stop criterion is more robust towards stalls than the MSE criterion in Equation (5.1).

5.1.3 Residual Stop Criterion

If an additive measurement noise is assumed, as e.g. when using the AWGN GAMP output channel, one may define a stop criterion based on noise power. The GAMP iteration should then be terminated once the residual may be regarded as a reflection of the noise, i.e. once it has a signal power smaller than the noise power. Thus, we have the residual stop criterion

$$\frac{1}{m}||\mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_t||_2^2 < \epsilon \tag{5.3}$$

for some tolerance ϵ reflecting the noise power, e.g. $\epsilon = \sigma^2$ for an AWGN with variance σ^2.

5.1.4 Residual Measurements Ratio Stop Criterion

In [47] it is suggested to use the following residual measurments ratio stop criterion in iterative reconstruction methods

$$\frac{||\mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}_t||_2^2}{||\mathbf{y}||_2^2} < \epsilon \tag{5.4}$$

for some tolerance ϵ.

Note that if the initial solution vector is chosen to be the zero-vector, this stop criterion expresses the ratio of the residual at iteration t to the residual at iteration one. Thus, convergence is determined based on the reduction in the residual. Also note that such a residual ratio stop criterion reflects the error in the solution through the condition number of the system matrix (for a non-singular system matrix) as detailed in [46].

5.2 Damping and Other Methods for Improving Convergence

The MMSE GAMP in Algorithm 2 incorporates damping of the GAMP updates by virtue of κ. It has been shown that the application of sufficient damping guarantees convergence of GAMP for arbitrary system matrices, $\mathbf{A}$, and with a Gaussian distributed vector, $\boldsymbol{\alpha}$, as well as for some other

distributions on the vector $\boldsymbol{\alpha}$ under certain conditions [20]. Note that the GAMP states that are damped in [20] are slightly different from those that are damped in Algorithm 2.

An adaptive damping scheme is proposed in [31] along with a scheme for removing any non-zero mean of the system matrix. Such non-zero mean system matrices may significantly impede convergence of the GAMP algorithm [31], [48]. Another method for improving the convergence of the GAMP algorithm is to use a sequential updating scheme [48] where each of the elements in the GAMP states are updated one at a time instead of in parallel. This is the idea used in the Swept AMP from [49].

6 Parameter Learning

For the GAMP algorithm to converge, it is essential to use some learning or estimation scheme to update the AWGN output channel noise level (when using the AWGN output channel)[1]. Furthermore, several studies have shown that Expectation Maximization (EM) may be used to effectively learn input channel parameters [6], [28], [39], [45], [50] to the point where oracle-like performance is achieved. An alternative adaptive GAMP strategy for learning the channel distributions is detailed in [51] and [52].

6.1 Variance Estimates

For the AWGN output channel given in Equations (3.57) and (3.59) with noise variance σ^2, one may use a per iteration estimate of the noise variance. This may e.g. be done using the sample variance estimator

$$(\sigma^2)^{t+1} = \frac{1}{m}||\mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}^{t+1}||_2^2 \tag{6.1}$$

Another option is to use the median based estimator often preferred by Donoho and Montanari [9], [47], [53]:

$$(\sigma^2)^{t+1} = \left(\text{median}\left(\frac{|\mathbf{y} - \mathbf{A}\bar{\boldsymbol{\alpha}}^{t+1}|}{\Phi_{\mathcal{N}}^{-1}(0.75)}\right)\right)^2 \tag{6.2}$$

In Equation (6.2), the absolute value is entrywise and $\Phi_{\mathcal{N}}^{-1}$ is the inverse standard normal cumulative distribution function.

6.2 Expectation Maximization (EM)

The Expectation Maximization (EM) algorithm may be used to find maximum likelihood (ML) estimates of parameters in probabilistic models [54] (see also [55] for an introduction to EM). Here we give an introduction to the use of EM to learn GAMP channel parameters as presented in [28], [39], [45].

The complete vector of GAMP channel parameters $\boldsymbol{\theta}_C$ is the concatenation of the input channel parameter vector(/matrix) $\boldsymbol{\theta}_I$ and the output channel parameter vector(/matrix) $\boldsymbol{\theta}_o$, i.e. $\boldsymbol{\theta}_C = [\boldsymbol{\theta}_I^T, \boldsymbol{\theta}_o^T]^T$. Now, the goal in using EM is to maximise the likelihood $p(\mathbf{y}|\boldsymbol{\theta}_C)$ with respect to $\boldsymbol{\theta}_C$. This is done using an iterative scheme in which each iteration has an E-step and an M-step that is guaranteed to increase the likelihood (if not at a stationary point already). One may elect to use a partial E-step in an "incremental" EM scheme [56] for improved convergence and/or to obtain a more computationally tractable problem. A partial M-step (known as the expectation conditional maximisation (ECM) algorithm [57]) is also an option to obtain a more computationally tractable problem. Both of these partial schemes are also guaranteed to increase the likelihood (if not at a stationary point already), though not necessarily maximise it, in each iteration.

For the general GAMP channel parameter optimisation problem, the EM algorithm manifests as the recursion of the following optimisation problem[2] [28]

$$\boldsymbol{\theta}_C^{t+1} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\boldsymbol{\theta}_C^t}[\ln(p(\mathbf{y}, \boldsymbol{\alpha}; \boldsymbol{\theta}))] \tag{6.3}$$

where $\mathbf{y}$ is the vector of observed variables (the measurements) and $\boldsymbol{\alpha}$ is the vector of the latent (unobserved or hidden) variables. Specifically, $\boldsymbol{\alpha}$ is the coefficient vector in Equation (1.1). Now,

[1] At least that is what the authors of this tech report have experienced in an extensive set of simulations

[2] Strictly speaking, this is the M-step in the EM algorithm consisting of both an E-step and the M-step. However, the E-step amounts to trivially choosing the distribution $p(\boldsymbol{\alpha}|\mathbf{y}; \boldsymbol{\theta}_C^t)$ for the expectation [28].

if we consider only updating the input channel parameters (a partial M-step approach), we have

$$\begin{aligned}
\boldsymbol{\theta}_I^{t+1} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\boldsymbol{\theta}_C^t}[\ln(p(\mathbf{y},\boldsymbol{\alpha};\boldsymbol{\theta}))] && (6.4)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y},\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} && (6.5)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}_o)p(\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} && (6.6)\\
&= \arg\max_{\boldsymbol{\theta}} \left(\int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}_o))d\alpha + \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha}\right) && (6.7)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.8)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} && (6.9)\\
&= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\boldsymbol{\theta}_C^t}[\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}))] && (6.10)
\end{aligned}$$

since, by definition, $\boldsymbol{\theta}_I$ is only used in the specification of the prior $p(\boldsymbol{\alpha};\boldsymbol{\theta}_I)$. That is, given a specific value of $\boldsymbol{\alpha}$, the value of $\mathbf{y}$ no longer depends on $\boldsymbol{\theta}_I$. Thus, w.r.t. elements of $\boldsymbol{\theta}_I$, $p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}_o)$ is a constant.

In order to find a similar expression for a separate update of the output channel parameters (i.e. a partial M-step approach update of $\boldsymbol{\theta}_o$), one must realise that the distribution of $\mathbf{y}$ depends only on $\boldsymbol{\alpha}$ through $\mathbf{z}$ since $\mathbf{z} = \mathbf{A}\boldsymbol{\alpha}$ (a deterministic relation) as specified in Equation (1.6). Due to the separability assumption of GAMP (see the discussion below Equations (1.16) and (1.17)), we then have a Markov chain $\boldsymbol{\alpha} \to \mathbf{z} \to \mathbf{y}$ meaning that $p(\boldsymbol{\alpha},\mathbf{y}|\mathbf{z}) = p(\boldsymbol{\alpha}|\mathbf{z})p(\mathbf{y}|\mathbf{z})$. The deterministic relation $\mathbf{z} = \mathbf{A}\boldsymbol{\alpha}$ also means that the joint distribution $p(\boldsymbol{\alpha},\mathbf{z})$ is degenerate[3]. All of this makes for a series of mathematical subtleties in the below expression. However, in accepting these expressions, it is probably most important to realise that *the distribution of* $\mathbf{y}$ *depends only on* $\boldsymbol{\alpha}$ *through* $\mathbf{z}$.

$$\begin{aligned}
\boldsymbol{\theta}_o^{t+1} &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\boldsymbol{\theta}_C^t}[\ln(p(\mathbf{y},\boldsymbol{\alpha};\boldsymbol{\theta}))] && (6.11)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y},\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} && (6.12)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta})p(\boldsymbol{\alpha};\boldsymbol{\theta}_I))d\boldsymbol{\alpha} && (6.13)\\
&= \arg\max_{\boldsymbol{\theta}} \left(\int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\alpha + \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}_I))d\boldsymbol{\alpha}\right) && (6.14)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{y};\boldsymbol{\theta}_C^t)\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.15)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} \frac{p(\boldsymbol{\alpha},\mathbf{y};\boldsymbol{\theta}_C^t)}{p(\mathbf{y};\boldsymbol{\theta}_C^t)}\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.16)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}}\int_{\mathbf{z}} \frac{p(\boldsymbol{\alpha},\mathbf{y},\mathbf{z};\boldsymbol{\theta}_C^t)}{p(\mathbf{y};\boldsymbol{\theta}_C^t)}d\mathbf{z}\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.17)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}}\int_{\mathbf{z}} \frac{p(\boldsymbol{\alpha},\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_C^t)p(\mathbf{z};\boldsymbol{\theta}_C^t)}{p(\mathbf{y};\boldsymbol{\theta}_C^t)}d\mathbf{z}\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.18)\\
&= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}}\int_{\mathbf{z}} \frac{p(\boldsymbol{\alpha}|\mathbf{z};\boldsymbol{\theta}_C^t)p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_C^t)p(\mathbf{z};\boldsymbol{\theta}_C^t)}{p(\mathbf{y};\boldsymbol{\theta}_C^t)}d\mathbf{z}\ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta}))d\boldsymbol{\alpha} + const. && (6.19)
\end{aligned}$$

[3]To see this, consider the simple example that $\boldsymbol{\alpha} = [\alpha_1,\alpha_2]^T$ with $p(\alpha_1,\alpha_2) = p(\alpha_1)p(\alpha_2)$, $p(\alpha_1) = \mathcal{N}(\alpha_1;0,1)$, $p(\alpha_2) = \mathcal{N}(\alpha_2;0,1)$, and $z = \alpha_1 + \alpha_2$. Now, $p(z)$ is well defined since it is simply the convolution of $p(\alpha_1)$ and $p(\alpha_2)$. However, $p(\boldsymbol{\alpha},z)$ is degenerate in the sense the "density" is restricted to values for which $z = \alpha_1 + \alpha_2$. Similarly, $p(\boldsymbol{\alpha}|z) = \frac{p(\boldsymbol{\alpha},z)}{p(z)}$ as well as $p(z|\boldsymbol{\alpha}) = \frac{p(\boldsymbol{\alpha},z)}{p(\boldsymbol{\alpha})}$ are degenerate for the same reason. Thus, they act as a sort of sampling that squeezes the domain of $\boldsymbol{\alpha}$ into the domain of z since z is deterministically derived from $\boldsymbol{\alpha}$. In a sense they have a "sampling property" similar to that of the generalised Dirac delta function.

$$= \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{\alpha}} \int_{\mathbf{z}} p(\boldsymbol{\alpha}|\mathbf{z};\boldsymbol{\theta}_C^t) p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}_C^t) d\mathbf{z} \ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta})) d\boldsymbol{\alpha} + const. \tag{6.20}$$

$$= \arg\max_{\boldsymbol{\theta}} \int_{\mathbf{z}} p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}_C^t) \int_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathbf{z};\boldsymbol{\theta}_C^t) \ln(p(\mathbf{y}|\boldsymbol{\alpha};\boldsymbol{\theta})) d\boldsymbol{\alpha} d\mathbf{z} + const. \tag{6.21}$$

$$= \arg\max_{\boldsymbol{\theta}} \int_{\mathbf{z}} p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}_C^t) \ln(p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta})) d\mathbf{z} + const. \tag{6.22}$$

$$= \arg\max_{\boldsymbol{\theta}} \int_{\mathbf{z}} p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}_C^t) \ln(p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta})) d\mathbf{z} \tag{6.23}$$

$$= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z}|\mathbf{y},\boldsymbol{\theta}_C^t}[\ln(p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}))] \tag{6.24}$$

where the constant is due to $p(\boldsymbol{\alpha};\boldsymbol{\theta}_I)$ being constant w.r.t. elements of $\boldsymbol{\theta}_o$.

Now since we are using GAMP which is trying to (indirectly) find the true posteriors $p(\boldsymbol{\alpha}|\mathbf{y},\boldsymbol{\theta}_C)$ and $p(\mathbf{z}|\mathbf{y},\boldsymbol{\theta}_C)$, we only have the GAMP approximated posteriors in Equations (3.4) and (3.8), respectively, available. Thus, these GAMP approximations are used in computing the expectation in the EM update [28], i.e., the E-step becomes approximate[4]. We then have the final GAMP EM channel parameter recursions

$$\boldsymbol{\theta}_I^{t+1} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t}[\ln(p(\boldsymbol{\alpha};\boldsymbol{\theta}))] \tag{6.25}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t}[\ln(p(\alpha_j;[\boldsymbol{\theta}]_j))] \tag{6.26}$$

$$\boldsymbol{\theta}_o^{t+1} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{z}|\mathbf{y},\mathbf{v},\mathbf{o},\boldsymbol{\theta}_o^t}[\ln(p(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}))] \tag{6.27}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathbb{E}_{\mathbf{z}|\mathbf{y},\mathbf{v},\mathbf{o},\boldsymbol{\theta}_o^t}[\ln(p(y_i|z_i;[\boldsymbol{\theta}]_i))] \tag{6.28}$$

where we have used the separability properties of the in- and output channels as described in Equations (1.16) and (1.17). Note that the GAMP approximated posteriors $p(\mathbf{z}|\mathbf{y},\mathbf{v},\mathbf{o},\boldsymbol{\theta}_o^t)$ and $p(\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t)$ by definition are separable (as everything else in the in- and output channels). Also note that one may use several GAMP iterations to find $p(\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t)$ and $p(\mathbf{z}|\mathbf{y},\mathbf{v},\mathbf{o},\boldsymbol{\theta}_o^t)$ for each EM update in Equations (6.26), (6.28) [28].

Furthermore, Schniter and Vila [28] as well as Krzakala et al. [5] use a "complete" partial M-step in the sense that the elements of $\boldsymbol{\theta}_C$ are updated one at a time, i.e. they essentially use the expectation/conditional maximization (ECM) algorithm [57]. Thus, in the following, our focus is on finding recursions based on Equations (6.26) and (6.28) for one parameter (one element of $\boldsymbol{\theta}_C$) at a time. When using such a scheme, the ordering of the updates of the elements of $\boldsymbol{\theta}_C$ become important since all parameter updates should be based on the latest value of all other parameters. The particular choice of update order may be arbitrary, but the all updates must be based on the most recent values of all other parameters that they depend on. Note, though, that Schniter and Vila [28] use the GAMP approximated posteriors $p(\mathbf{z}|\mathbf{y},\mathbf{v},\mathbf{o},\boldsymbol{\theta}_o^t)$ and $p(\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t)$ based on iteration t for the all parameter updates. That is, they do not recompute the GAMP posteriors between the parameter updates.

[4] It is not clear whether or not this falls under the framework of partial E-steps from [56] guaranteed to increase the likelihood.

6.2.1 EM Updates for Common Output Channels

We now state EM updates for several commonly used GAMP output channels.

6.2.1.1 AWGN Output Channel

For the AWGN output channel given in Equations (3.57) and (3.59) with noise variance σ^2, Krzakala suggested the following EM recursion on the noise variance [5] (Eq. 77)

$$(\sigma^2)^{t+1} = \frac{\sum_i \frac{(y_i - o_i^{t+1})^2}{\left(1+\frac{1}{(\sigma^2)^t}v_i^{t+1}\right)^2}}{\sum_i \frac{1}{1+\frac{1}{(\sigma^2)^t}v_i^{t+1}}} \tag{6.29}$$

Manoel and Tramel suggested the following implementation[5] of the EM recursion in Equation (6.29):

$$(\sigma^2)^{t+1} = (\sigma^2)^t \frac{\sum_i \left(\frac{y_i - o_i^{t+1}}{(\sigma^2)^t + v_i^{t+1}}\right)^2}{\sum_i ((\sigma^2)^t + v_i^{t+1})^{-1}} \tag{6.30}$$

which in matrix-vector + Numpy broadcast notation is:

$$\sigma_t^2 = \sigma_{t\text{-}1}^2 \frac{\sum \left(\frac{\mathbf{y} - \mathbf{o}_t}{\sigma_{t\text{-}1}^2 + \mathbf{v}_t}\right)}{\sum (\sigma_{t\text{-}1}^2 + \mathbf{v}_t)^{-1}} \tag{6.31}$$

Note that $\mathbf{v}_t$ becomes scalar when using the sum approximation GAMP described in Section 4 and Algorithm 3. Thus, one must make sure to implement the denominator sum in Equation (6.31) such that it acts as if $\mathbf{v}_t$ was a vector (a sum over m elements - not just a single element).

Schniter and Vila suggested the following EM recursion on the noise variance [28] (Eq. (27)), [39] (Eq. (38))

$$(\sigma^2)^{t+1} = \frac{1}{m}\sum_i (|y_i - \tilde{z}_i^{t+1}|^2 + \tilde{z}_i^{t+1}) \tag{6.32}$$

However, in [40] it is reported that the closely related expression

$$(\sigma^2)^{t+1} = \frac{1}{m}(||\mathbf{y} - \mathbf{A}\tilde{\boldsymbol{\alpha}}^{t+1}||_2^2 + \sum_i [|\mathbf{A}|^{\circ 2}\tilde{\boldsymbol{\alpha}}^{t+1}]_i) \tag{6.33}$$

is supposed to yield improved performance in low SNR cases (SNR $<$ 10 dB). Note how this expression is an extension of Equation (6.1).

6.2.1.2 AWLN Output Channel

For the AWLN output channel given in Equations (3.62) and (3.63) with rate parameter λ, Vila and Schniter suggested the following EM recursion on the rate parameter [40] (Eq. (52))

$$\lambda^{t+1} = \frac{m}{\sum_i \left(\Phi_{\mathcal{N}}\left(\frac{-\hat{z}_i}{v_i}\right)\left(\hat{z}_i + \sqrt{v_i}\frac{\phi_{\mathcal{N}}\left(\frac{-\hat{z}_i}{\sqrt{v_i}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\hat{z}_i}{\sqrt{v_i}}\right)}\right) - \Phi_{\mathcal{N}}\left(\frac{\hat{z}_i}{v_i}\right)\left(\hat{z}_i - \sqrt{v_i}\frac{\phi_{\mathcal{N}}\left(\frac{-\hat{z}_i}{\sqrt{v_i}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\hat{z}_i}{\sqrt{v_i}}\right)}\right)\right)} \tag{6.34}$$

for

$$\hat{\mathbf{z}} \coloneqq \mathbf{A}\boldsymbol{\alpha} - \mathbf{y} \tag{6.35}$$

[5] See: `https://github.com/eric-tramel/SwAMP-Demo/blob/master/python/amp.py`

6.2.2 EM Updates for Common Input Channels

We now state EM updates for several commonly used GAMP input channels.

6.2.2.1 General i.i.d. Sparse Input Channel

As noted in [28], one may find a general expression for the EM update of the signal density τ in the general sparse i.i.d. input prior in Equation (3.29). In particular, we have

$$\tau^{t+1} = \arg\max_{\tau\in[0,1]} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t}[\ln\left(p(\alpha_j;\tau,\boldsymbol{\theta}_{I\setminus\tau}^t)\right)] \tag{6.36}$$

$$= \arg\max_{\tau\in[0,1]} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) \ln\left((1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})\right) d\alpha_j \tag{6.37}$$

Setting the derivative of the objective equal to zero, we obtain

$$0 = \frac{\partial}{\partial\tau} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) \ln\left((1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})\right) d\alpha_j \tag{6.38}$$

$$= \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) \frac{\partial}{\partial\tau} \ln\left((1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})\right) d\alpha_j \tag{6.39}$$

where we have used Leibniz's integral rule to exchange differentiation and integration. Leibniz's integral rule requires the integrand and its partial derivative w.r.t. τ to be continuous in both α_j and τ which is not strictly true for the above objective due to the Dirac delta function. However, as noted in [40], one may justify its use by considering an approximation of the Dirac delta function $\delta_{\text{Dirac}}(\alpha) \approx \mathcal{N}(\alpha,0,\epsilon)$ for a fixed arbitrarily small $\epsilon > 0$. For the partial derivative, we have

$$\frac{\partial}{\partial\tau} \ln\left((1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})\right) = \frac{\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau}) - \delta_{\text{Dirac}}(\alpha_j)}{(1-\tau)\delta_{\text{Dirac}}(\alpha_j) + \tau\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})} \tag{6.40}$$

$$= \frac{\frac{\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})}{\delta_{\text{Dirac}}(\alpha_j)} - 1}{(1-\tau) + \tau\frac{\varphi(\alpha_j;\boldsymbol{\theta}_{I\setminus\tau})}{\delta_{\text{Dirac}}(\alpha_j)}} \tag{6.41}$$

$$= \begin{cases} \frac{-1}{1-\tau}, & \alpha_j = 0 \\ \frac{1}{\tau}, & \alpha_j \neq 0 \end{cases} \tag{6.42}$$

Following [28], we may define a closed ball $\mathcal{B}_\epsilon \coloneqq [-\epsilon,\epsilon]$ and its complement $\overline{\mathcal{B}}_\epsilon = \mathbb{R}\setminus\mathcal{B}_\epsilon$ which may be used to evaluate the integral in Equation (6.39) as $\epsilon \to 0$. Then using the GAMP approximated posterior in Equation (3.37), we get

$$0 = \sum_{j=1}^{n} \left(\frac{-1}{1-\tau} \int_{\alpha_j\in\mathcal{B}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) d\alpha_j + \frac{1}{\tau} \int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) d\alpha_j \right) \tag{6.43}$$

$$= \sum_{j=1}^{n} \left(\frac{-1}{1-\tau}(1-\pi(r_j,s_j,\boldsymbol{\theta}_I^t)) \int_{\alpha_j\in\mathcal{B}_\epsilon} \delta_{\text{Dirac}}(\alpha_j) d\alpha_j + \right.$$

$$\left. \frac{1}{\tau}\pi(r_j,s_j,\boldsymbol{\theta}_I^t) \int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j;\boldsymbol{\theta}_I^t) d\alpha_j \right) \tag{6.44}$$

$$= \sum_{j=1}^{n} \left(\frac{-1}{1-\tau}(1-\pi(r_j,s_j,\boldsymbol{\theta}_I^t)) + \frac{1}{\tau}\pi(r_j,s_j,\boldsymbol{\theta}_I^t) \right) \tag{6.45}$$

$$= \sum_{j=1}^{n} \left(\frac{-\tau}{1-\tau}(1-\pi(r_j,s_j,\boldsymbol{\theta}_I^t)) + \pi(r_j,s_j,\boldsymbol{\theta}_I^t) \right) \tag{6.46}$$

$$= \frac{-\tau n}{1-\tau} - \frac{-\tau}{1-\tau}\sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) + \sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.47}$$

$$= -\tau n + \tau\sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) + \sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) - \tau\sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.48}$$

$$= -n\tau + \sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.49}$$

with $\pi(r_j, s_j, \boldsymbol{\theta}_I^t)$ defined in Equation (3.41). Note that in going from Equation (6.44) to Equation (6.45), we have assumed that $\varphi(\alpha_j; \boldsymbol{\theta}_I^t)$ is well behaved at $\alpha_j = 0$ such that

$$\int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) d\alpha_j = \int_{\alpha_j} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) d\alpha_j = 1 \tag{6.50}$$

Solving for τ, we get the final expression for its EM update

$$\tau^{t+1} = \frac{1}{n}\sum_{j=1}^{n}\pi(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.51}$$

The update in Equation (6.51) is intuitively pleasing since it states that the signal density τ is the average of the posterior signal density (posterior support probabilities) $\pi(r_j, s_j, \boldsymbol{\theta}_I^t)$. Also note that since $\pi(r_j, s_j, \boldsymbol{\theta}_I^t) \in [0; 1], \forall j$, we have that $\tau^{t+1} \in [0; 1]$.

6.2.2.2 General Weighted Sparse Input Channel

An EM update of the common signal density τ in the GWS input prior in Equation (3.44) may be found using similar derivations as those used for the general i.i.d. sparse input channel detailed in Section 6.2.2.1. In particular, we have

$$\tau^{t+1} = \arg\max_{\tau\in[0,1]} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t}[\ln\Big(p(\alpha_j; \tau, \boldsymbol{\theta}_{I\setminus\tau}^t)\Big)] \tag{6.52}$$

$$= \arg\max_{\tau\in[0,1]} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \ln\big((1-w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})\big)\, d\alpha_j \tag{6.53}$$

Setting the derivative of the objective equal to zero, we obtain

$$0 = \frac{\partial}{\partial\tau}\sum_{j=1}^{n}\int_{\alpha_j} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \ln\big((1-w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})\big)\, d\alpha_j \tag{6.54}$$

$$= \sum_{j=1}^{n}\int_{\alpha_j} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \frac{\partial}{\partial\tau} \ln\big((1-w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})\big)\, d\alpha_j \tag{6.55}$$

where we again have used Leibniz's integral rule to exchange differentiation and integration. For the partial derivative, we have

$$\frac{\partial}{\partial\tau} \ln\big((1-w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})\big)$$

$$= \frac{w_j\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau}) - w_j\delta_{\text{Dirac}}(\alpha_j)}{(1-w_j\tau)\delta_{\text{Dirac}}(\alpha_j) + w_j\tau\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})} \tag{6.56}$$

$$= \frac{w_j\frac{\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})}{\delta_{\text{Dirac}}(\alpha_j)} - w_j}{(1-w_j\tau) + w_j\tau\frac{\varphi(\alpha_j; \boldsymbol{\theta}_{I\setminus\tau})}{\delta_{\text{Dirac}}(\alpha_j)}} \tag{6.57}$$

$$= \begin{cases} \frac{-w_j}{1-w_j\tau}, & \alpha_j = 0 \\ \frac{1}{\tau}, & \alpha_j \neq 0 \end{cases} \tag{6.58}$$

Again, using the closed ball $\mathcal{B}_\epsilon := [-\epsilon, \epsilon]$ and its complement $\overline{\mathcal{B}}_\epsilon = \mathbb{R} \setminus \mathcal{B}_\epsilon$ in evaluating the integral in Equation (6.55) as $\epsilon \to 0$ and using the GAMP approximated posterior in Equation (3.45), we get

$$0 = \sum_{j=1}^{n} \left(\frac{-w_j}{1 - w_j\tau} \int_{\alpha_j \in \mathcal{B}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) d\alpha_j + \frac{1}{\tau} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) d\alpha_j \right) \tag{6.59}$$

$$= \sum_{j=1}^{n} \left(\frac{-w_j}{1 - w_j\tau} (1 - \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)) + \frac{1}{\tau} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \right) \tag{6.60}$$

$$= \sum_{j=1}^{n} \left(\frac{-w_j\tau}{1 - w_j\tau} (1 - \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)) + \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \right) \tag{6.61}$$

$$= \sum_{j=1}^{n} \left(\frac{-w_j\tau}{1 - w_j\tau} - \frac{-w_j\tau}{1 - w_j\tau} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) + \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \right) \tag{6.62}$$

$$= \sum_{j=1}^{n} \left(-w_j\tau + w_j\tau\pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) + \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) - w_j\tau\pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \right) \tag{6.63}$$

$$= -\tau \sum_{j=1}^{n} w_j + \sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.64}$$

with $\pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)$ defined as in Equation (3.46). Solving for τ, we get the final expression for its EM update:

$$\tau^{t+1} = \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)}{\sum_{j=1}^{n} w_j} \tag{6.65}$$

Note that the GWS τ EM update in Equation (6.65) reduces to the general sparse channel τ EM updates in Equation (6.51) for the choice of weights $\forall j, w_j = 1$.

In order for the GAMP updates to be stable, the GAMP posterior must remain a proper density which requires that $w_j\tau^{t+1} \in [0; 1]$, $\forall j$ as described in Section 3.5. If the requirement on the choice of weights is $w_j \in [0; 1]$, $\forall j$, then one must also require that $\tau^{t+1} \in [0; 1]$. Now, consider the case of having just a single element in the coefficient vector. The τ EM update then becomes

$$\tau^{t+1} = \frac{\pi_1^{\mathrm{w}}(r_1, s_1, \boldsymbol{\theta}_I^t)}{w_1} \in [0; \frac{1}{w_1}] \tag{6.66}$$

Thus, if the GAMP posterior support probability $\pi_1^{\mathrm{w}}(r_1, s_1, \boldsymbol{\theta}_I^t)$ is large and we have chosen a small w_1, we may end up with $\tau^{t+1} > 1$. That is, if there is a significant mismatch between our prior belief about the support probability (expressed by w_1 and the actual posterior support probability $\pi_1^{\mathrm{w}}(r_1, s_1, \boldsymbol{\theta}_I^t)$, it may potentially violate the requirement that $\tau^{t+1} \in [0; 1]$. In generalising this result to arbitrary length vectors we may consider

$$\tau^{t+1} = \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)}{\sum_{j=1}^{n} w_j} = \frac{\frac{1}{n}\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)}{\frac{1}{n}\sum_{j=1}^{n} w_j} \tag{6.67}$$

That is, if the average GAMP posterior support probability becomes larger than our prior average belief about the support probabilities (expressed by the w_j's), it may violate the requirement that $\tau^{t+1} \in [0; 1]$. Note that, as discussed in Section 6.2.2.1, this problem is not present if $w_j = 1$, $\forall j$.

At least two strategies for handling this violation can be identified

1. We may force $\tau^{t+1} = 1$ whenever $\tau^{t+1} > 1$. This may be interpreted as forcing the prior belief on the support probabilities. Note that since τ models the overall average sparsity of the signal, forcing $\tau \leq 1$ has the effect of forcing the average sparsity in the next GAMP iteration to be no larger than the average of the weights.

2. We may adjust the weights towards $w_j = 1$, $\forall j$ according to some scheme detailing the weights update. This strategy allows for the weighted model to adapt towards a non-weighted model, if the data suggest such a change.

One scheme for adjusting the weights towards $w_j = 1, \forall j$ is to apply Algorithm 4 which attempts to increase all weights such that $\tau = 1$. Worst case using this scheme, we get $w_j = 1, \forall j$ after n iterations of the while loop. In a practical implementation of Algorithm 4, it may be beneficial to introduce some other stop criterion.

Algorithm 4 - A scheme for adjusting the weights towards $w_j = 1, \forall j$ in the general weighted sparse GAMP input channel.

```
while τ > 1 do
    for j ∈ 1, ..., n do
        w_j = τ w_j
        if w_j > 1 then
            w_j = 1
        end if
    end for
    τ = (Σ_{j=1}^{n} π_j^w(r_j, s_j, θ_I^t)) / (Σ_{j=1}^{n} w_j)
end while
```

6.2.2.3 EM updates of other channel parameters in the GWS channel

The derivation of EM updates for channel parameters in a general sparse input channel as detailed in [28], [39], [45] may be summarised as follows:

1. Pick a parameter to update.
2. Write down the single step EM-update as in e.g. Equation (6.53).
3. Take the partial derivative, set equal to zero, and apply Leibniz's integral rule to interchange integration and differentiation (assuming that this interchange is valid) as in e.g. Equation (6.55).
4. Compute the partial derivative as in e.g. Equation (6.58).
5. Handle the discontinuity at zero by splitting the integration and treating a closed ball around zero separately from the remaining domain as in e.g. Equations (6.59).
6. Compute the integrals and solve for the parameter of interest to obtain the EM update as in e.g. Equation (6.65).

For this recipe to work for other channel parameters than τ, the partial derivative must conform to a certain structure. Specifically, consider the channel defined by $p(\alpha_j; \boldsymbol{\theta}_I) = (1 - w_j\tau)\delta_{\text{Dirac}}(\alpha_J) + w_j\tau f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j, \boldsymbol{\theta}_I))$ for some well-behaved functions f_1 and f_2. This channel has partial derivatives for each of the k parameters in $\boldsymbol{\theta}_I$

$$\frac{\partial}{\partial\theta_k}\ln(p(\alpha_j;\boldsymbol{\theta}_I)) = \frac{\partial}{\partial\theta_k}\ln((1-w_j\tau)\delta_{\text{Dirac}}(\alpha_J) + w_j\tau f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))) \tag{6.68}$$

$$= \frac{\frac{\partial}{\partial\theta_k}f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))}{(1-w_j\tau)\delta_{\text{Dirac}}(\alpha_J) + w_j\tau f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))} \tag{6.69}$$

$$= \begin{cases} 0, & \alpha_j = 0 \\ \frac{\frac{\partial}{\partial\theta_k}f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))}{w_j\tau f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))}, & \alpha_j \neq 0 \end{cases} \tag{6.70}$$

Now consider the case in which f_1 and f_2 have structure such that

$$0 = \sum_{j=1}^{n}\int_{\alpha_j\in\mathcal{B}_\epsilon} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)0d\alpha_j + \int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)\frac{\frac{\partial}{\partial\theta_k}f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))}{w_j\tau f_1(\boldsymbol{\theta}_I)\exp(f_2(\alpha_j,\boldsymbol{\theta}_I))}d\alpha_j \tag{6.71}$$

$$= \sum_{j=1}^{n} \int_{\alpha_j \in \mathcal{B}_\epsilon} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) 0 d\alpha_j + \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)(f_3(\alpha_j) - \theta_k) d\alpha_j \tag{6.72}$$

for some function f_3 which does not depend on θ_k. In this case we find that

$$0 = \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)(f_3(\alpha_j) - \theta_k) d\alpha_j \tag{6.73}$$

$$= \sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t)(f_3(\alpha_j) - \theta_k) d\alpha_j \tag{6.74}$$

$$\theta_k^{t+1} = \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) f_3(\alpha_j) d\alpha_j}{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) d\alpha_j} \tag{6.75}$$

$$= \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \int_{\alpha_j} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) f_3(\alpha_j) d\alpha_j}{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)} \tag{6.76}$$

$$= \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \int_{\alpha_j} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) f_3(\alpha_j) d\alpha_j}{\tau^{t+1} \sum_{j=1}^{n} w_j} \tag{6.77}$$

where we have assumed that all the integrands are well-behaved at zero and that τ is the first parameter to be updated. From Equation (6.77), we find that this EM update separates into a slab-part only element ($\int_{\alpha_j} \varphi_{\alpha_j|\mathbf{y};s_j,r_j,\boldsymbol{\theta}_I^t}(\alpha_j; \boldsymbol{\theta}_I^t) f_3(\alpha_j) d\alpha_j$) and the posterior support probabilities $\pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t)$. Furthermore, since it turns out that these slab-part elements are typically computed in the GAMP channel update, all elements needed in the EM-update are already available following a GAMP channel update which allows for efficient implementations of the EM update. This is the case for the sparse Bernoulli-Gauss channel as shown in [28] and detailed in Section 6.2.2.5. The above imposed structure on the partial derivatives is somewhat limiting in terms of the possible $\varphi(\alpha_j; [\boldsymbol{\theta}_I]_j)$ that one may consider. However, as is done in some updates in [28] as well as for the sparse Bernoulli-Laplace input channel detailed in Section 6.2.2.5, one may apply various approximations to impose this structure.

An implementation in which the GWS τ EM update is decoupled from the slab-part EM updates is slightly more tricky than the corresponding channel updates since we assume common channel parameters shared across all n coefficients effectively requiring a reduction from all n elements to a single element. However, if the slab-part channel has the structure discussed above, one may store the relevant channel update elements and reuse those in the common EM updates which still provides an efficient implementation in which the GWS framework may be easily combined with different slab-part priors.

6.2.2.4 I.i.d. Sparse Bernoulli-Laplace input channel

For the i.i.d. BL input channel given in Equations (3.127) and (3.129), we consider the following parameter update order: τ, λ, μ. For the EM updates of the signal density parameter τ, we may (via Equation (3.41)) use the general result in Equation (6.51)

$$\tau^{t+1} = \frac{1}{n} \sum_{j=1}^{n} \pi(r_j, s_j, \boldsymbol{\theta}_I^t) \tag{6.78}$$

$$= \frac{1}{n} \sum_{j=1}^{n} \frac{\tau^t(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)}{(1-\tau^t)\mathcal{N}(0; r_j, s_j) + \tau^t(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)} \tag{6.79}$$

for $\underline{\mathcal{Z}}_I, \bar{\mathcal{Z}}_I$ given in Equations (3.93), (3.94), respectively, and with appropriately indexed r_j, s_j in $\underline{\mathcal{Z}}_I, \bar{\mathcal{Z}}_I$.

Using some of the ideas from [40] (which address a AWLN output channel) and [41] (which address an elastic net input channel), we may find the EM updates for the remaining parameters,

λ, μ. For the EM update of λ, we have

$$\lambda^{t+1} = \arg\max_{\lambda>0} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t}[\ln\left(p(\alpha_j;\lambda,\boldsymbol{\theta}_{I\setminus\lambda}^t)\right)] \tag{6.80}$$

$$= \arg\max_{\lambda>0} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t) \ln\Big((1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp\big(-\lambda|\alpha_j-\mu^t|\big)\Big) d\alpha_j \tag{6.81}$$

Setting the derivative of the objective equal to zero, we obtain

$$0 = \frac{\partial}{\partial\lambda}\sum_{j=1}^{n}\int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)\ln\left((1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp\big(-\lambda|\alpha_j-\mu^t|\big)\right)d\alpha_j \tag{6.82}$$

$$= \sum_{j=1}^{n}\int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)\frac{\partial}{\partial\lambda}\ln\left((1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp\big(-\lambda|\alpha_j-\mu^t|\big)\right)d\alpha_j \tag{6.83}$$

Where we have used Leibniz's integral rule to exchange differentiation and integration using the same argument as for Equation (6.39). Now for the partial derivative, we have

$$\frac{\partial}{\partial\lambda}\ln\left((1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp\big(-\lambda|\alpha_j-\mu^t|\big)\right) \tag{6.84}$$

$$= \frac{\tau^{t+1}\frac{1}{2}\exp(-\lambda|\alpha_j-\mu^t|) - \tau^{t+1}\frac{\lambda}{2}\exp(-\lambda|\alpha_j-\mu^t|)\,|\alpha_j-\mu^t|}{(1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp(-\lambda|\alpha_j-\mu^t|)} \tag{6.85}$$

$$= \frac{\frac{\tau^{t+1}}{2}\exp(-\lambda|\alpha_j-\mu^t|)\,(1-\lambda|\alpha_j-\mu|)}{(1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \lambda\frac{\tau^{t+1}}{2}\exp(-\lambda|\alpha_j-\mu^t|)} \tag{6.86}$$

$$= \frac{\frac{\frac{\tau^{t+1}}{2}\exp(-\lambda|\alpha_j-\mu^t|)(1-\lambda|\alpha_j-\mu|)}{(1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j)}}{1+\frac{\lambda\frac{\tau^{t+1}}{2}\exp(-\lambda|\alpha_j-\mu^t|)}{(1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j)}} \tag{6.87}$$

$$= \begin{cases} 0, & \alpha_j = 0 \\ \frac{1}{\lambda} - |\alpha_j-\mu^t|, & \alpha_j \neq 0 \end{cases} \tag{6.88}$$

Following [28], we may define a closed ball $\mathcal{B}_\epsilon \coloneqq [-\epsilon,\epsilon]$ and its complement $\overline{\mathcal{B}}_\epsilon = \mathbb{R}\setminus\mathcal{B}_\epsilon$ which may be used to evaluate the integral in Equation (6.83) as $\epsilon\to 0$

$$0 = \sum_{j=1}^{n}\int_{\alpha_j} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)\frac{\partial}{\partial\lambda}\ln\left((1-\tau^{t+1})\delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1}\frac{\lambda}{2}\exp\big(-\lambda|\alpha_j-\mu^t|\big)\right)d\alpha_j \tag{6.89}$$

$$= \sum_{j=1}^{n}\left(\int_{\alpha_j\in\mathcal{B}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)0\,d\alpha_j + \int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)(\frac{1}{\lambda}-|\alpha_j-\mu^t|)d\alpha_j\right) \tag{6.90}$$

$$= \sum_{j=1}^{n}\int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)(\frac{1}{\lambda}-|\alpha_j-\mu^t|)d\alpha_j \tag{6.91}$$

$$= \sum_{j=1}^{n}\left(\frac{1}{\lambda}\int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)d\alpha_j - \int_{\alpha_j\in\overline{\mathcal{B}}_\epsilon} p(\alpha_j|\mathbf{y},s_j,r_j,\boldsymbol{\theta}_I^t)|\alpha_j-\mu^t|d\alpha_j\right) \tag{6.92}$$

$$\tag{6.93}$$

$$= \sum_{j=1}^{n} \left(\frac{1}{\lambda} \pi(r_j, s_j, \boldsymbol{\theta}_I^t) - \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) |\alpha_j - \mu^t| d\alpha_j \right) \tag{6.94}$$

$$= \frac{1}{\lambda} n \tau^{t+1} - \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) |\alpha_j - \mu^t| d\alpha_j \tag{6.95}$$

Solving for λ, we get

$$\lambda^{t+1} = \frac{n \tau^{t+1}}{\sum_{j=1}^{n} \int_{\overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) |\alpha_j - \mu^t| d\alpha_j} \tag{6.96}$$

The integral in the denominator in Equation (6.96) is exactly the expectation from Equation (3.127) with four exceptions:

1. The Dirac delta contribution should be left out since the integration is over $\overline{\mathcal{B}}_\epsilon$ instead of the entire real line. This is, however, not important since the Dirac delta contribution turns out to be zero anyway.
2. The integral involves the expectation of $|\alpha_j - \mu^t|$ instead of α_j. Thus, applying a shift $\check{\alpha}_j = \alpha_j - \mu^t$ eliminates the contribution from a non-zero μ^t.
3. The absolute value in $|\alpha_j - \mu^t|$ must be addressed. Specifically, after having done the shift $\check{\alpha}_j = \alpha_j - \mu^t$, one must handle the absolute value $|\check{\alpha}_j|$ in (what corresponds to) Equation (3.102) correctly by changing the sign of the integral over the negative part of the real line.
4. The integration in (what corresponds to) (3.102) is done from $-\infty$ to 0^- and 0^+ to ∞. However, since the GAMP posterior $p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)$ without the Dirac delta contribution at $\alpha_j = 0$ is well behaved, this makes no difference and we may integrate over the entire real line.

Thus, we have

$$\begin{aligned} &\int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) |\alpha_j - \mu^t| d\alpha_j \\ &= \tau^t \left(\frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) - \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) \right) \end{aligned} \tag{6.97}$$

for

$$\check{r}_j = r_j - \mu^t \tag{6.98}$$

$$\bar{r}_j = \check{r}_j - \lambda^t s \tag{6.99}$$

$$\underline{r}_j = \check{r}_j + \lambda^t s_j \tag{6.100}$$

and $\mathcal{Z}_I$, $\underline{\mathcal{Z}}_I$, and $\bar{\mathcal{Z}}_I$ as defined in Equations (3.92), (3.93), (3.94) (with appropriate indices on the variables), respectively. Finally, we have the EM update for λ

$$\lambda^{t+1} = \frac{n \tau^{t+1}}{\sum_{j=1}^{n} \tau^t \left(\frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) - \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) \right)} \tag{6.101}$$

For the EM update of μ, we have

$$\mu^{t+1} = \arg\max_{\mu} \sum_{j=1}^{n} \mathbb{E}_{\boldsymbol{\alpha}|\mathbf{y},\mathbf{s},\mathbf{r},\boldsymbol{\theta}_I^t} [\ln \left(p(\alpha_j; \mu, \boldsymbol{\theta}_{I \setminus \mu}^t) \right)] \tag{6.102}$$

$$\begin{aligned} &= \arg\max_{\mu} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \ln \Big(\\ &\quad (1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1} |\alpha_j - \mu|\big) \Big) d\alpha_j \end{aligned} \tag{6.103}$$

The partial derivative with respect to μ of the integrand in Equation (6.103) is unfortunately not continuous (even if using the same argument for the Delta function as in Equation (6.39)) due to the absolute value causing problems at $\alpha_j = \mu$. Thus, we may not apply Leibniz's integral rule. However, in order to derive a reasonable EM update for μ, we apply the quadratic approximation $|\alpha_j - \mu| \approx (\alpha_j - \mu)^2$. Now, using Leibniz's integral rule, taking the partial derivative of the objective, and setting equal to zero, we obtain

$$\begin{aligned}
0 &= \frac{\partial}{\partial \mu} \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \ln \Bigg((1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) \\
&\quad + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \Bigg) d\alpha_j && (6.104) \\
&= \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \frac{\partial}{\partial \mu} \ln \Bigg((1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) \\
&\quad + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \Bigg) d\alpha_j && (6.105)
\end{aligned}$$

For the partial derivative, we have

$$\begin{aligned}
&\frac{\partial}{\partial \mu} \ln \left((1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \right) d\alpha_j && (6.106) \\
&= \frac{\tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \left(-2\lambda^{t+1}(\alpha - \mu)(-1)\right)}{(1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp(-\lambda^{t+1}(\alpha_j - \mu)^2)} && (6.107) \\
&= \frac{\tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \left(2\lambda^{t+1}(\alpha - \mu)\right)}{(1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp(-\lambda^{t+1}(\alpha_j - \mu)^2)} && (6.108) \\
&= \frac{\frac{\tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\left(-\lambda^{t+1}(\alpha_j - \mu)^2\right) (2\lambda^{t+1}(\alpha - \mu))}{(1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j)}}{1 + \frac{\tau^{t+1} \frac{\lambda^{t+1}}{2} \exp(-\lambda^{t+1}(\alpha_j - \mu)^2)}{(1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j)}} && (6.109) \\
&= \begin{cases} 0, & \alpha_j = 0 \\ 2\lambda^{t+1}(\alpha_j - \mu), & \alpha_j \neq 0 \end{cases} && (6.110)
\end{aligned}$$

Again, using the closed ball $\mathcal{B}_\epsilon$, we may evaluate the integral in Equation (6.105) as $\epsilon \to 0$

$$\begin{aligned}
0 &= \sum_{j=1}^{n} \int_{\alpha_j} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \frac{\partial}{\partial \mu} \ln \Bigg((1 - \tau^{t+1}) \delta_{\text{Dirac}}(\alpha_j) \\
&\quad + \tau^{t+1} \frac{\lambda^{t+1}}{2} \exp\big(-\lambda^{t+1}(\alpha_j - \mu)^2\big) \Bigg) d\alpha_j && (6.111) \\
&= \sum_{j=1}^{n} \Bigg(\int_{\alpha_j \in \mathcal{B}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) 0 d\alpha_j && (6.112) \\
&\quad + \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) (2\lambda^{t+1}(\alpha_j - \mu)) d\alpha_j \Bigg) && (6.113) \\
&= \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) (\alpha_j - \mu) d\alpha_j && (6.114) \\
&= \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \alpha_j d\alpha_j - \mu \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) d\alpha_j && (6.115) \\
&= \sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \alpha_j d\alpha_j - \mu \pi(r_j, s_j, \boldsymbol{\theta}_I^t) && (6.116)
\end{aligned}$$

Thus, we have the following update of μ

$$\mu^{t+1} = \frac{\sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \alpha_j d\alpha_j}{\pi(r_j, s_j, \boldsymbol{\theta}_I^t)} \tag{6.117}$$

$$= \frac{\sum_{j=1}^{n} \int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \alpha_j d\alpha_j}{n\tau^{t+1}} \tag{6.118}$$

The numerator in Equation (6.118) is the expectation from Equation (3.127) with the exception of leaving out $\alpha_j = 0$ in the integration. However, since the remaining part of the GAMP posterior $p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t)$ (leaving out the Dirac delta contribution) is well-behaved, we find that

$$\int_{\alpha_j \in \overline{\mathcal{B}}_\epsilon} p(\alpha_j | \mathbf{y}, s_j, r_j, \boldsymbol{\theta}_I^t) \alpha_j d\alpha_j \tag{6.119}$$

$$= \tau^t \left(\mu^t \frac{(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)}{\mathcal{Z}_I} + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \right) \tag{6.120}$$

for

$$\check{r}_j = r_j - \mu^t \tag{6.121}$$

$$\bar{r}_j = \check{r}_j - \lambda^t s \tag{6.122}$$

$$\underline{r}_j = \check{r}_j + \lambda^t s_j \tag{6.123}$$

and $\mathcal{Z}_I$, $\underline{\mathcal{Z}}_I$, and $\bar{\mathcal{Z}}_I$ as defined in Equations (3.92), (3.93), (3.94) (with appropriate indices on the variables), respectively. Finally, we have the EM update for μ

$$\mu^{t+1} = \frac{\sum_{j=1}^{n} \tau^t \left(\mu^t \frac{(\underline{\mathcal{Z}}_I + \bar{\mathcal{Z}}_I)}{\mathcal{Z}_I} + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_I} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \right)}{n\tau^{t+1}} \tag{6.124}$$

Based on the results in Equations (6.101) and (6.124), we may identify the following Laplace EM updates to be used in the GWS EM update framework described in Section 6.2.2.2

$$\lambda^{t+1} = \frac{\tau^{t+1} \sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \left(\frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) - \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) \right)} \tag{6.125}$$

$$\mu^{t+1} = \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \left(\mu^t + \frac{\underline{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\underline{r}_j - \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{-\underline{r}_j}{\sqrt{s_j}}\right)} \right) + \frac{\bar{\mathcal{Z}}_I}{\mathcal{Z}_{\varphi_j}} \left(\bar{r}_j + \sqrt{s_j} \frac{\phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)}{\Phi_{\mathcal{N}}\left(\frac{\bar{r}_j}{\sqrt{s_j}}\right)} \right) \right)}{\tau^{t+1} \sum_{j=1}^{n} w_j} \tag{6.126}$$

6.2.2.5 I.i.d. Sparse Bernoulli-Gauss Input Channel

For the i.i.d. BG input channel given in Equations (3.154) and (3.155), Krzakala et al. suggested the following EM recursions on the channel parameters [5] (Eqs. (74), (78), (79))[6]

$$\tau^{t+1} = \frac{\sum_{j=1}^{n} \frac{\frac{1}{\bar{\theta}^t} + \frac{1}{s_j^{t+1}}}{\frac{r_j^{t+1}}{s_j^{t+1}} + \frac{\bar{\theta}^t}{\tilde{\theta}^t}} \bar{\alpha}_j^{t+1}}{\sum_{j=1}^{n} \left[1 - \tau^t + \frac{\tau^t}{\sqrt{\tilde{\theta}^t}} \sqrt{\frac{1}{\bar{\theta}^t} + \frac{1}{s_j^{t+1}}} \exp\left(\frac{\left(\frac{r_j^{t+1}}{s_j^{t+1}} + \frac{\bar{\theta}^t}{\tilde{\theta}^t} \right)^2}{2\left(\frac{1}{\bar{\theta}^t} + \frac{1}{s_j^{t+1}} \right)} - \frac{(\bar{\theta}^t)^2}{2\tilde{\theta}^t} \right) \right]^{-1}} \tag{6.127}$$

$$\bar{\theta}^{t+1} = \frac{\sum_{j=1}^{n} \bar{\alpha}_j^{t+1}}{n\tau^{t+1}} \tag{6.128}$$

$$\tilde{\theta}^{t+1} = \frac{\sum_{j=1}^{n} (\tilde{\alpha}_j^{t+1} + (\bar{\alpha}_j^{t+1})^2)}{n\tau^{t+1}} - (\bar{\theta}^{t+1})^2 \tag{6.129}$$

Note that Krzakala et al. do not make it clear (in either of [5], [6]) whether or not the iteration dependence on the channel parameters (and thereby the ordering of the updates) are as given in Equations (6.127) - (6.129). In [5] it is noted that the following heuristic rules should be used together with the Equations (6.127), (6.128), (6.129)

- If the variance, $\tilde{\theta}^{t+1}$, becomes negative, it should be set to zero.
- If the signal density, τ^{t+1}, becomes larger than the undersampling ratio δ, it should be set to δ.
- A damping of 0.5 should be used on all EM updates. That is, the updated parameter values should be taken to be the mean of the values of the updates in Equations (6.127), (6.128), (6.129) and the respective previous values.

Schniter and Vila suggested the following EM recursions on the channel parameters for the i.i.d. BG input channel [28] (Eqs. (34), (41), (47)), [39] (Eqs. (19), (25), (32))

$$\tau^{t+1} = \frac{1}{n} \sum_j \ell(r_j^{t+1}, s_j^{t+1}; \tau^t, \bar{\theta}^t, \tilde{\theta}^t) \tag{6.130}$$

$$\bar{\theta}^{t+1} = \frac{1}{n\tau^{t+1}} \sum_{j=1}^{n} \ell(r_j^{t+1}, s_j^{t+1}; \tau^{t+1}, \bar{\theta}^t, \tilde{\theta}^t) g(r_j^{t+1}, s_j^{t+1}; \tau^{t+1}, \bar{\theta}^t, \tilde{\theta}^t) \tag{6.131}$$

$$\tilde{\theta}^{t+1} = \frac{1}{n\tau^{t+1}} \sum_{j=1}^{n} \ell(r_j^{t+1}, s_j^{t+1}; \tau^{t+1}, \bar{\theta}^{t+1}, \tilde{\theta}^t) \Big(|\bar{\theta}^{t+1} - g(r_j^{t+1}, s_j^{t+1}; \tau^{t+1}, \bar{\theta}^{t+1}, \tilde{\theta}^t)|^2 + h(r_j^{t+1}, s_j^{t+1}; \tau^{t+1}, \bar{\theta}^{t+1}, \tilde{\theta}^t) \Big) \tag{6.132}$$

Where ℓ, g, h are as defined in Equations (3.171), (3.166), and (3.167), respectively.

Based on the results in Equations (6.131) and (6.132), we may identify the following Gauss EM

[6] Eq. (74) in [6] lacks a set of parentheses to be equal to Equation (6.127). However, Eqs. (71) and (72) in [6] suggest that Eq. (74) is to be interpreted as in Equation (6.127).

updates to be used in the GWS EM update framework described in Section 6.2.2.2

$$\bar{\theta}^{t+1} = \frac{\sum_{j=1}^{n} \left(\pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \frac{\frac{\bar{\theta}_j^t}{\tilde{\theta}_j^t} + \frac{r_j}{s_j}}{\frac{1}{\tilde{\theta}_j^t} + \frac{1}{s_j}} \right)}{\tau^{t+1} \sum_{j=1}^{n} w_j} \tag{6.133}$$

$$\tilde{\theta}^{t+1} = \frac{\sum_{j=1}^{n} \pi_j^{\mathrm{w}}(r_j, s_j, \boldsymbol{\theta}_I^t) \left(\left| \bar{\theta}^{t+1} - \left(\frac{\frac{\bar{\theta}_j^{t+1}}{\tilde{\theta}_j^t} + \frac{r_j}{s_j}}{\frac{1}{\tilde{\theta}_j^t} + \frac{1}{s_j}} \right) \right|^2 + \frac{1}{\tilde{\theta}_j^t + \frac{1}{s_j}} \right)}{\tau^{t+1} \sum_{j=1}^{n} w_j} \tag{6.134}$$

$$\tag{6.135}$$

6.3 Parameter Initialisation

Since the GAMP and EM algorithms are only guaranteed to converge to local optima, proper parameter initialisation is important. The various proposed ways to initialise the GAMP states are reproduced in the respective GAMP implementations in Algorithms 1, 2, and 3. Below we summarise the various proposed EM initialisations.

6.3.1 EM Initialisation of the AWGN Output Channel Parameters

For the AWGN output channel EM update in Equation (6.32), Vila and Schniter [28] (Eq. (71)) suggested the following initialisation

$$\sigma_0^2 = \frac{||\mathbf{y}||_2^2}{m(\mathrm{SNR}^0 + 1)} \tag{6.136}$$

For some assumed true signal-to-noise ratio $\mathrm{SNR}^0 = \frac{||\mathbf{A}\boldsymbol{\alpha}||_2^2}{||\mathbf{e}||_2^2}$. In lack of knowledge of the true signal-to-noise ratio, $\mathrm{SNR}^0 = 100$ is proposed.

6.3.2 EM Initialisation of the AWLN Output Channel Parameters

For the AWLN output channel EM update in Equation (6.34), Vila and Schniter [40] (Eq. (67)) suggested the following initialisation

$$\lambda_0 = 1 \tag{6.137}$$

6.3.3 EM initialisation of the i.i.d. Sparse Bernoulli-Gauss Input Channel Parameters

For the i.i.d. BG input channel EM updates in Equations (6.127) - (6.129), Krzakala et al. [5] (Eq. (80)) suggested the following initialisation

$$\tau_0 = \frac{\delta}{10} \tag{6.138}$$

$$\bar{\theta}_0 = 0 \tag{6.139}$$

$$\tilde{\theta}_0 = \frac{||\mathbf{y}||_2^2}{\tau_0 ||\mathbf{A}||_F^2} \tag{6.140}$$

Vila and Schniter [39] (Eqs. (39), (40)), [28] (Eqs. (70), (71)) suggested the following initialisation

$$\tau_0 = \delta \rho_{\mathrm{SE}}(\delta) \tag{6.141}$$

$$\bar{\theta}_0 = 0 \tag{6.142}$$

$$\tilde{\theta}_0 = \frac{||\mathbf{y}||_2^2 - m\sigma_0^2}{\tau_0 ||\mathbf{A}||_F^2} \tag{6.143}$$

when using the AWGN output channel EM initialisation in Equation (6.136) and with ρ_{SE} the theoretical LASSO phase transition curve [2] given by

$$\rho_{\text{SE}}(\delta) = \max_{c>0} \frac{1 - \frac{\zeta}{\delta}[(1+c^2)\Phi_{\mathcal{N}}(-c) - c\phi_{\mathcal{N}}(c)]}{1 + c^2 - 2[(1+c^2)\Phi_{\mathcal{N}}(-c) - c\phi_{\mathcal{N}}(c)]} \tag{6.144}$$

for $\zeta = 2$ (sparse signed vectors). When using the AWLN output channel, Schniter and Vila suggested using $\sigma_0^2 = 1$ [40] (Eq. (67)).

6.3.4 EM Initialisation of the i.i.d. Sparse Bernoulli-Laplace Input Channel Parameters

For the i.i.d. BL input channel EM updates in Equations (6.127), (6.101), and (6.124) when used with an AWGN output channel, we suggest the following initialisation

$$\tau_0 = \delta\rho_{\text{SE}}(\delta) \tag{6.145}$$

$$\mu_0 = 0 \tag{6.146}$$

$$\lambda_0 = \sqrt{\frac{2}{\frac{||\mathbf{y}||_2^2 - m\sigma_0^2}{\tau_0 ||\mathbf{A}||_F^2}}} \tag{6.147}$$

That is, an initialisation based on Equations (6.141)-(6.143) but with λ initialised based on the variance of a Laplace distributed random variable being $\frac{2}{\lambda^2}$.

7 GAMP Software

Various software packages include implementations of the GAMP algorithms described in this note. Here we briefly describe a few of them.

7.1 Magni GAMP Implementation

Magni is a Python package which enables reconstruction of undersampled Atomic Force Microscopy (AFM) images [58]. The `magni.cs.reconstruction.gamp` and `magni.cs.reconstruction.amp` subpackages, which are part of Magni version $\geq$ 1.6.0, provide an implementation of GAMP in Python by the authors of the present tech report. The Magni package is fully documented, has an extensive test suite, makes use of an input validation framework [59], and comes with tools for aiding in making computational results reproducible [60]. Related links are:

- Online documentation: `http://magni.readthedocs.io/en/latest/`
- Official source repository: `http://dx.doi.org/10.5278/VBN/MISC/Magni`
- GitHub repository: `https://github.com/SIP-AAU/Magni`

7.1.1 Magni GAMP Overview

The `magni.cs.reconstruction.amp` subpackage provides an implementation of the AMP algorithm by Donoho/Maleki/Monatnari described in Section 2.1. As of Magni version 1.7.0, the subpackage consists of the following modules:

- `_algorithm`: The base algorithm implementation which is available through `magni.cs.reconstruction.amp.run`.
- `_config`: Configuration module available through `magni.cs.reconstruction.amp.config` for choosing stop criterion, max iterations, threshold, etc.
- `stop_criterion`: Implementations of the stop criteria discussed in Section 5.1.
- `threshold_operator`: Implementations of threshold operators for DMM AMP as discussed in Section 2.1.
- `util`: Utilities for use in the AMP algorithm.

The `magni.cs.reconstruction.gamp` subpackage provides an implementation of the MMSE GAMP algorithm detailed in Algorithm 1 and the MMSE GAMP with Rangan sum approximations detailed in Algorithm 3. For both algorithms it also offers the damping option from [20]. As of Magni version 1.7.0, the GAMP subpackage consists of the following modules:

- `_algorithm`: The base algorithm implementation which is available through `magni.cs.reconstruction.gamp.run`.
- `_config`: Configuration module available through `magni.cs.reconstruction.gamp.config` for choosing stop criterion, max iterations, in- and output channels, etc.
- `channel_initialisation`: Implementations of the in- and output channel EM initialisations described in Section 6.3.
- `input_channel`: Implementations of the input channels discussed in Sections 3.1, 3.4, 3.5, and 3.7 with EM updates as discussed in Section 6.2.2.
- `output_channel`: Implementations of the output channels discussed in Sections 3.2 and 3.6 with EM updates as discussed in Section 6.2.1
- `stop_criterion`: Implementations of the stop criteria discussed in Section 5.1.

7.2 GAMPMatlab Implementation

The GAMPMatlab Toolbox is an implementation of GAMP in MATLAB [61]. The GAMPMatlab Toolbox is maintained by Phillip Schniter and Sundeep Rangan and has contributions from several coauthors of the various GAMP algorithms and extensions. Related links are:

- Online documentation: `http://gampmatlab.wikia.com/wiki/Generalized_Approximate_Message_Passing`
- Official source repository: `https://sourceforge.net/projects/gampmatlab/`
- SVN repository: `svn.code.sf.net/p/gampmatlab/code/`

7.3 BPCS AMP Implementation

The BPCS AMP package is another MATLAB based implementation of AMP which has been developed by Jean Barbier in connection with his PhD studies [62]. Related links are:

- GitHub repository: `https://github.com/jeanbarbier/BPCS/`

7.4 Vampyre

Vampyre is a joint collaboration on a Python implementation of GAMP algorithms by many of the authors of the works on GAMP. It has yet to kick-off, but may potentially become the reference GAMP implementation.

- GitHub repository `https://github.com/GAMPTeam/vampyre`

8 GAMP Extensions

A significant number of works on various extensions of the GAMP algorithm for specific applications have been published. Here, as a reference, we list (in no particular order) some of these works.

- Markov-tree / Markov-random-field priors [33], [63], [64].
- Learning based priors [65]
- Phase retrieval [66].
- Hyperspectral image unmixing [67], [68].
- Linearly constrained non-neagtive sparse signals [40], [69].
- Non-stationary signals [70], [71].
- Multiple measurement vectors [72], [73].
- Classification and feature selection [74], [75].
- Low rank matrix completion [76].
- Spatially coupled structured operators [77].
- Total Variation like prior [78]
- Analysis compressive sensing [79].
- Quantized measurements [80], [81].
- Magnetic resonance imaging [82].

References

[1] S. Rangan, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, Jul. 31 – Aug. 5, 2011, pp. 2168–2172. doi:10.1109/ISIT.2011.6033942

[2] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, p. 18914–18919, Nov. 2009. doi:10.1073/pnas.0909892106

[3] ——, "Message Passing Algorithms for Compressed Sensing: I. Motivation and Construction," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5. doi:10.1109/ITWKSPS.2010.5503193

[4] ——, "Message Passing Algorithms for Compressed Sensing: II. Analysis and Validation," in *IEEE Information Theory Workshop (ITW)*, Cairo, Egypt, Jan. 6 – 8, 2010, p. 5. doi:10.1109/ITWKSPS.2010.5503228

[5] F. Krzakala, M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices," *Journal of Statistical Mechanics: Theory and Experiment*, vol. P08009, pp. 1–57, Aug. 2012. doi:10.1088/1742-5468/2012/08/P08009

[6] F. Krzakala, M. Mézard, F. Sausset, Y. F. Sun, and L. Zdeborová, "Statistical-Physics-Based Reconstruction in Compressed Sensing," *Physical Review X*, vol. 2, no. 2, pp. (021 005–1)–(021 005–18), May 2012. doi:10.1103/PhysRevX.2.021005

[7] D. Donoho and J. Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical Transactions of the Royal Society A*, vol. 367, no. 1906, pp. 4273–4293, Nov. 2009. doi:10.1098/rsta.2009.0152

[8] L. Zdeborová and F. Krzakala, "Statistical physics of inference: Thresholds and algorithms," Jul. 2016, arxiv:1511.02476v4.

[9] A. Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, ch. 9, pp. 394–438.

[10] M. Bayati and A. Montanari, "The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011. doi:10.1109/TIT.2010.2094817

[11] M. Bayati, M. Lelarge, and A. Montanari, "Universality In Polytope Phase Transitions And Message Passing Algorithms," *The Annals of Applied Probability*, vol. 25, no. 2, pp. 753–822, Feb. 2015. doi:10.1214/14-AAP1010

[12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001. doi:10.1109/18.910572

[13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1st ed., ser. Morgan Kaufmann Series in Representation and Reasoning. San Francisco, California, USA: Morgan Kaufmann Publishers, Inc., 1988.

[14] F. Krzakala, A. Manoel, and E. W. Tramel, "Variational Free Energies for Compressed Sensing," in *IEEE International Symposium on Information Theory (ISIT)*, Honolulu, Hawaii, USA, Jun. 29 – Jul. 4, 2014, pp. 1499–1503. doi:10.1109/ISIT.2014.6875083

[15] J. T. Parker, "Approximate Message Passing Algorithms for Generalized Bilinear Inference," Ph.D. dissertation, Graduate School of The Ohio State University, 2014.

[16] S. Rangan, "Generalized Approximate Message Passing for Estimation with Random Linear Mixing," Aug. 2012, arXiv:1010.5141v2.

[17] M. Bayati, M. Lelarge, and A. Montanari, "Universality in Polytope Phase Transitions and Iterative Algorithms," in *IEEE International Symposium on Information Theory (ISIT)*, Cambridge, Massachusetts, USA, Jul. 1 – 6, 2012, pp. 1643–1647. doi:10.1109/ISIT.2012.6283554

[18] A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling," *Information and Inference*, vol. 2, no. 2, pp. 115–144, Dec. 2013. doi:10.1093/imaiai/iat004

[19] C. Rush and R. Venkataramanan, "Finite-Sample Analysis of Approximate Message Passing," in *IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, Jul. 10 – 15, 2016, pp. 755–759. doi:10.1109/ISIT.2016.7541400

[20] S. Rangan, P. Schniter, and A. Fletcher, "On the Convergence of Approximate Message Passing with Arbitrary Matrices," in *IEEE International Symposium on Information Theory (ISIT)*, Honolulu, Hawaii, USA, Jun. 29 – Jul. 4, 2014, pp. 236–240. doi:10.1109/ISIT.2014.6874830

[21] B. Cakmak, O. Winther, and B. H. Fleury, "S-AMP: Approximate Message Passing for General Matrix Ensembles," in *IEEE Information Theory Workshop (ITW)*, Hobart, Tasmania, Australia, Nov. 2 – 5, 2014, pp. 192–196. doi:10.1109/ITW.2014.6970819

[22] ——, "S-AMP for Non-linear Observation Models," in *IEEE International Symposium on Information Theory (ISIT)*, Hong Kong, China, Jun. 15 – 19, 2015, pp. 2807–2811. doi:10.1109/ISIT.2015.7282968

[23] S. Rangan, A. K. Fletcher, P. Schniter, and U. S. Kamilov, "Inference for Generalized Linear Models via Alternating Directions and Bethe Free Energy Minimization," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 676–697, Jan. 2017. doi:10.1109/TIT.2016.2619373

[24] J. Ma and L. Peng, "Orthogonal AMP," *IEEE Access*, vol. 5, pp. 2020–2033, Jan. 2017. doi:10.1109/ACCESS.2017.2653119

[25] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed Points of Generalized Approximate Message Passing with Arbitrary Matrices," in *IEEE International Symposium on Information Theory (ISIT)*, Istanbul, Turkey, Jul. 7–12, 2013, pp. 664–668. doi:10.1109/ISIT.2013.6620309

[26] H. Monajemi, S. Jafarpour, M. Gavish, S. C. . Collaboration, and D. L. Donoho, "Deterministic matrices matching the compressed sensing phase transitions of Gaussian random matrices," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 4, p. 1181–1186, Jan. 2013. doi:10.1073/pnas.1219540110

[27] Y. Wu and S. Verdú, "Optimal Phase Transitions in Compressed Sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6241–6263, Oct. 2012. doi:10.1109/TIT.2012.2205894

[28] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013. doi:10.1109/TSP.2013.2272287

[29] H. Rauhut, "Compressive Sensing and Structured Random Matrices," in *Theoretical Foundations and Numerical Methods for Sparse Recovery*, ser. Radon Series on Computational and Applied Mathematics, M. Fornasier, Ed. De Gruyter, 2010, vol. 9, pp. 1–92.

[30] T. T. Do, L. Gan, N. H. Nguyen, and T. D. Tran, "Fast and Efficient Compressive Sensing Using Structurally Random Matrices," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 139–154, Jan. 2012. doi:10.1109/TSP.2011.2170977

[31] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborová, "Adaptive Damping and Mean Removal for the Generalized Approximate Message Passing Algorithm," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, Apr. 19 – 24, 2015, pp. 2021–2025. doi:10.1109/ICASSP.2015.7178325

[32] S. Som and P. Schniter, "Compressive Imaging Using Approximate Message Passing and a Markov-Tree Prior," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3439–3448, Jul. 2012. doi:10.1109/TSP.2012.2191780

[33] P. Schniter, "Turbo Reconstruction of Structured Sparse Signals," in *44th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New Jersey, USA, Mar. 17 – 19, 2010, p. 6. doi:10.1109/CISS.2010.5464920

[34] A. Maleki and A. Montanari, "Analysis of Approximate Message Passing Algorithm," in *44th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New York, USA, Mar. 17–19, 2010. doi:10.1109/CISS.2010.5464887

[35] M. R. Andersen, "Sparse inference using approximate message passing," Master's thesis, Technical University of Denmark, Department of Applied Mathematics and Computer Science, Matematiktorvet, Building 303B, DK-2800 Kgs. Lyngby, Denmark, 2014.

[36] I. Daubechies, M. Defrise, and C. D. Mol, "An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004. doi:10.1002/cpa.20042

[37] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Mar. 2009. doi:10.1137/080716542

[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends(R) in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jul. 2011. doi:10.1561/2200000016

[39] J. Vila and P. Schniter, "Expectation-Maximization Bernoulli-Gaussian Approximate Message Passing," in *Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 6 – 9 2011, pp. 799–803. doi:10.1109/ACSSC.2011.6190117

[40] J. P. Vila and P. Schniter, "An Empirical-Bayes Approach to Recovering Linearly Constrained Non-Negative Sparse Signals," *IEEE Transactions on Signal Processing*, vol. 62, no. 18, pp. 4689–4703, Sep. 2014. doi:10.1109/TSP.2014.2337841

[41] J. Ziniel, "Message Passing Approaches to Compressive Inference Under Structured Signal Priors," Ph.D. dissertation, Graduate School of The Ohio State University, 2014.

[42] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, 2nd ed., ser. Wiley Series in Probability and Mathematical Statistics. Wiley-Interscience, 1994, vol. 1.

[43] D. R. Barr and E. T. Sherrill, "Mean and Variance of Truncated Normal Distributions," *The American Statistician*, vol. 53, no. 4, pp. 357–361, Nov. 1999. doi:http://doi.org/10.2307/2686057

[44] M. R. Zaghloul, "On the calculation of the Voigt line profile: a single proper integral with a damped sine integrand," *Monthly Notices of the Royal Astronomical Society*, vol. 375, no. 3, pp. 1043–1048, Mar. 2007. doi:10.1111/j.1365-2966.2006.11377.x

[45] J. Vila and P. Schniter, "Expectation-Maximization Gaussian-Mixture Approximate Message Passing," in *46th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New Jersey, USA, Mar. 21 – 23, 2012, p. 6. doi:10.1109/CISS.2012.6310932

[46] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.

[47] A. Maleki and D. L. Donoho, "Optimally Tuned Iterative Reconstruction Algorithms for Compressed Sensing," *IEEE Journal Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 330–341, Apr. 2010. doi:10.1109/JSTSP.2009.2039176

[48] F. Caltagirone, L. Zdeborová, and F. Krzakala, "On Convergence of Approximate Message Passing," in *IEEE International Symposium on Information Theory (ISIT)*, Honolulu, Hawaii, USA, Jun. 29 – Jul. 4, 2014. doi:10.1109/ISIT.2014.6875146

[49] A. Manoel, F. Krzakala, E. W. Tramel, and L. Zdeborová, "Swept Approximate Message Passing for Sparse Estimation," in *32nd International Conference on Machine Learning (ICML)*, Lille, France, Jul. 6 – 11, 2015, p. 1123–1132.

[50] J. P. Vila, "Empirical-Bayes Approaches to Recovery of Structured Sparse Signals via Approximate Message Passing," Ph.D. dissertation, Graduate School of The Ohio State University, 2015.

[51] U. Kamilov, S. Rangan, A. Fletcher, and M. Unser, "Approximate Message Passing with Consistent Parameter Estimation and Applications to Sparse Learning," in *Advances in Neural Information Processing Systems (NIPS) 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Lake Tahoe, California, USA: MIT Press, Dec. 3 – 6, 2012, pp. 2447–2455.

[52] U. S. Kamilov, S. Rangan, A. K. Fletcher, and M. Unser, "Approximate Message Passing With Consistent Parameter Estimation and Applications to Sparse Learning," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2969–2985, May 2014. doi:10.1109/TIT.2014.2309005

[53] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, Feb. 2012. doi:10.1109/TIT.2011.2173241

[54] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[56] R. M. Neal and G. E. Hinton, "A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants," in *Learning in Graphical Models*, M. I. Jordan, Ed. MIT Press, 1999, pp. 355–368. doi:10.1007/978-94-011-5014-9_12

[57] X.-L. Meng and D. B. Rubin, "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, Jun. 1993. doi:10.2307/2337198

[58] C. S. Oxvig, P. S. Pedersen, T. Arildsen, J. Østergaard, and T. Larsen, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images," *Journal of Open Research Software*, vol. 2, no. 1, p. e29, Oct. 2014. doi:10.5334/jors.bk

[59] P. S. Pedersen, C. S. Oxvig, J. Østergaard, and T. Larsen, "Validating Function Arguments in Python Signal Processing Applications," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 106–113.

[60] C. S. Oxvig, T. Arildsen, and T. Larsen, "Storing Reproducible Results from Computational Experiments using Scientific Python Packages," in *Proceedings of the 15th Python in Science Conference*, Austin, Texas, USA, Jul. 11 – 17, 2016, pp. 45–50.

[61] J. Ziniel, S. Rangan, and P. Schniter, "A Generalized Framework for Learning and Recovery of Structured Sparse Signals," in *IEEE Statistical Signal Processing Workshop (SSP)*, Ann Arbor, Michigan, USA, Aug. 5 – 8, 2012, pp. 325–328. doi:10.1109/SSP.2012.6319694

[62] J. Barbier, "Statistical Physics And Approximate Message Passing Algorithms for Sparse Linear Estimation Problems in Signal Processing and Coding Theory," Ph.D. dissertation, École Normale Supérieure, 2015.

[63] S. Som, L. C. Potter, and P. Schniter, "Compressive Imaging using Approximate Message Passing and a Markov-Tree Prior," in *Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 7 – 10, 2010, pp. 243–247. doi:10.1109/ACSSC.2010.5757509

[64] S. Som and P. Schniter, "Approximate Message Passing for Recovery of Sparse Signals with Markov-Random-Field Support Structure," in *International Conference on Machine Learning (ICML)*, Bellevue, Washington, USA, Jul. 2, 2011.

[65] E. W. Tramel, A. Dremeau, and F. Krzakala, "Approximate message passing with restricted Boltzmann machine priors," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 7, pp. 1–15, Jul. 2016. doi:10.1088/1742-5468/2016/07/073401

[66] P. Schniter and S. Rangan, "Compressive Phase Retrieval via Generalized Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1043–1055, Feb. 2015. doi:10.1109/TSP.2014.2386294

[67] J. Vila, P. Schniter, and J. Meola, "Hyperspectral Image Unmixing via Bilinear Generalized Approximate Message Passing," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XIX, Proceedings of the SPIE*, S. S. Shen and P. E. Lewis, Eds., vol. 8743, Baltimore, Maryland, USA, Apr. 29, 2013, pp. (87 430Y–1)–(87 430Y–9). doi:10.1117/12.2015859

[68] ——, "Hyperspectral Unmixing via Turbo Bilinear Approximate Message Passing," *IEEE Transactions on Computational Imaging*, vol. 1, no. 3, pp. 143 – 158, Sep. 2015. doi:10.1109/TCI.2015.2465161

[69] J. Vila and P. Schniter, "An Empirical-Bayes Approach to Recovering Linearly Constrained Non-Negative Sparse Signals," in *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, St. Martin, France, Dec. 15 – 18, 2013, pp. 5–8. doi:10.1109/CAMSAP.2013.6713993

[70] J. Ziniel, L. C. Potter, and P. Schniter, "Tracking and Smoothing of Time-Varying Sparse Signals via Approximate Belief Propagation," in *Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 7 – 10, 2010, pp. 808–812. doi:10.1109/ACSSC.2010.5757677

[71] J. Ziniel and P. Schniter, "Dynamic Compressive Sensing of Time-Varying Signals Via Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5270–5284, Nov. 2013. doi:10.1109/TSP.2013.2273196

[72] ——, "Efficient Message Passing-Based Inference in the Multiple Measurement Vector Problem," in *Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, California, USA, Nov. 6 – 9, 2011, pp. 1447–1451. doi:10.1109/ACSSC.2011.6190257

[73] ——, "Efficient High-Dimensional Inference in the Multiple Measurement Vector Problem," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 340–354, Jan. 2013. doi:10.1109/TSP.2012.2222382

[74] J. Ziniel, P. Schniter, and P. Sederberg, "Binary Linear Classification and Feature Selection via Generalized Approximate Message Passing," in *48th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, New Jersey, USA, Mar. 19 – 21, 2014, pp. 1–6. doi:10.1109/CISS.2014.6814160

[75] E. Byrne and P. Schniter, "Sparse Multinomial Logistic Regression via Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5485–5498, Nov. 2016. doi:10.1109/TSP.2016.2593691

[76] S. Rangan and A. K. Fletcher, "Iterative Estimation of Constrained Rank-One Matrices in Noise," in *IEEE International Symposium on Information (ISIT)*, Cambridge, Massachusetts, USA, Jul. 1 – 6, 2012, pp. 1246–1250. doi:10.1109/ISIT.2012.6283056

[77] J. Barbier, C. Schulke, and F. Krzakala, "Approximate message-passing with spatially coupled structured operators, with applications to compressed sensing and sparse superposition codes," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2015, no. 5, p. P05013, May 2015. doi:10.1088/1742-5468/2015/05/P05013

[78] J. Barbier, E. W. Tramel, and F. Krzakala, "Scampi: a robust approximate message-passing framework for compressive imaging," in *International Meeting on High-Dimensional Data-Driven Science*, ser. Journal of Physics: Conference Series, vol. 699, Kyoto, Japan, Dec. 14 – 17, 2015, pp. 1–13. doi:10.1088/1742-6596/699/1/012013

[79] M. Borgerding, P. Schniter, J. Vila, and S. Rangan, "Generalized Approximate Message Passing for Cosparse Analysis Compressive Sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, Apr. 19 – 24, 2015, pp. 3756–3760. doi:10.1109/ICASSP.2015.7178673

[80] U. Kamilov, V. K. Goyal, and S. Rangan, "Generalized Approximate Message Passing Estimation from Quantized Samples," in *4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, San Juan, Puerto Rico, Dec. 13 – 16, 2011, pp. 365–368. doi:10.1109/CAMSAP.2011.6136027

[81] ——, "Optimal Quantization for Compressive Sensing under Message Passing Reconstruction," in *IEEE International Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, Jul. 31 – Aug. 5, 2011, pp. 459–463. doi:10.1109/ISIT.2011.6034168

[82] K. Sung, B. L. Daniel, and B. A. Hargreaves, "Location Constrained Approximate Message Passing for Compressed Sensing MRI," *Magnetic Resonance in Medicine*, vol. 70, no. 2, p. 370–381, Aug. 2013. doi:10.1002/mrm.24468

Dataset H

Typical Reconstructions of Undersampled AFM images

This dataset contains a collection of typical reconstructions of undersampled AFM images from the simulation study detailed in Chapter 7. An even larger collection of typical reconstructions (which includes the figures from this dataset) is part of the "Extra Figures" supplementary material available at doi:10.5278/252861471.

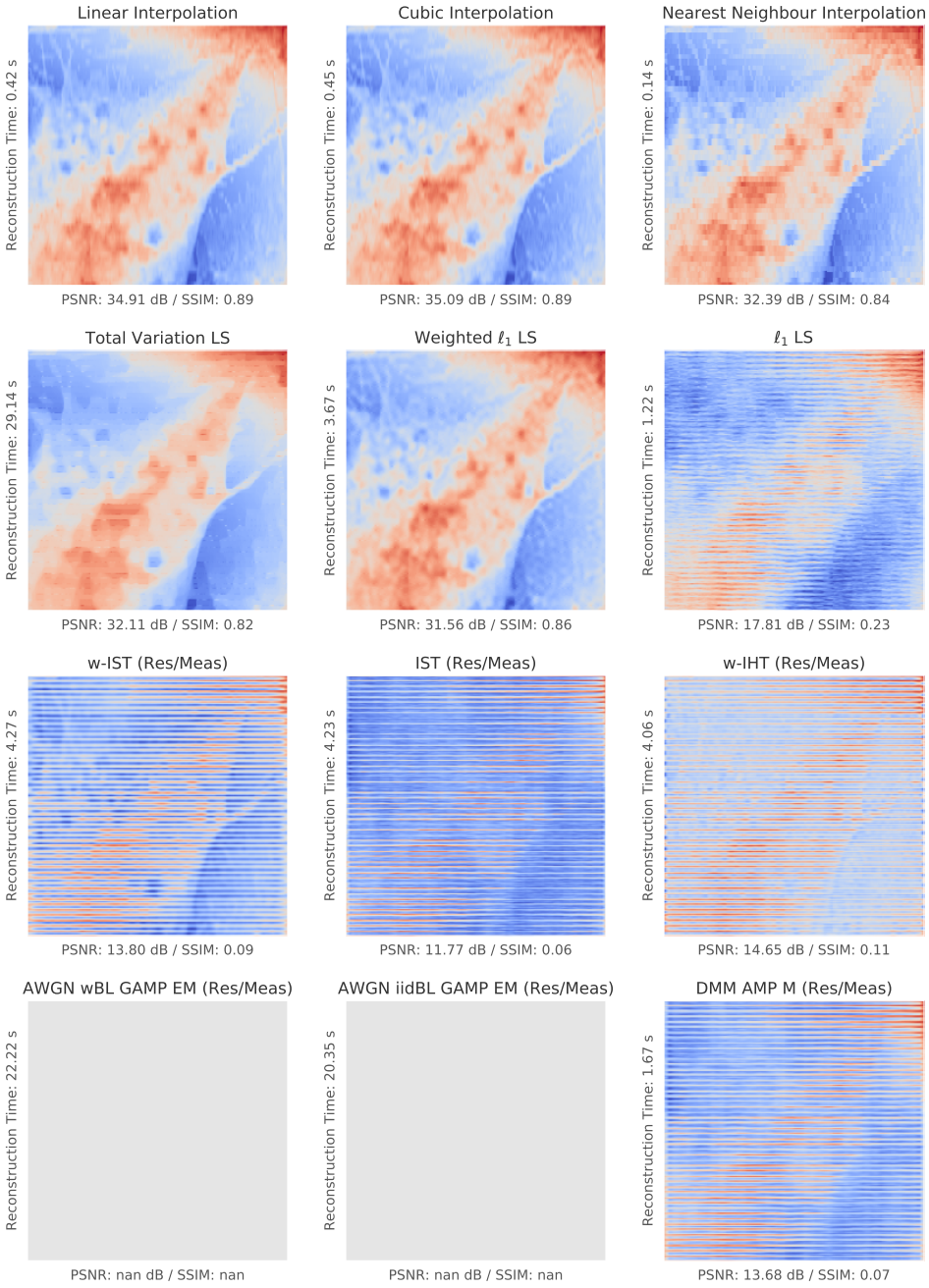


Figure H.1: Typical reconstructions of the first AFM image shown in Figure 7.1 when using uniform line sampling and an undersampling ratio of $\delta = 0.10$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

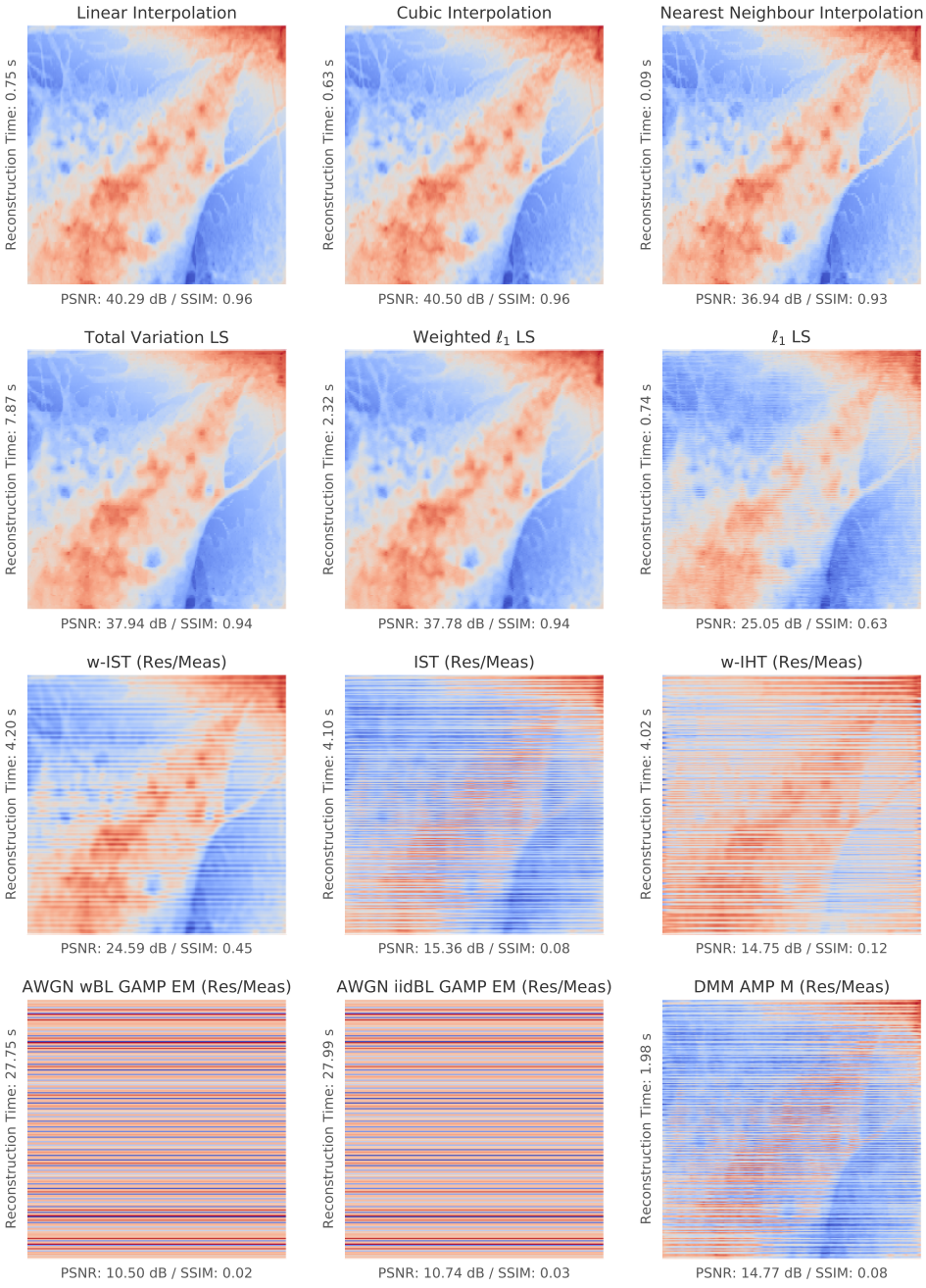


Figure H.2: Typical reconstructions of the first AFM image shown in Figure 7.1 when using uniform line sampling and an undersampling ratio of $\delta = 0.20$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

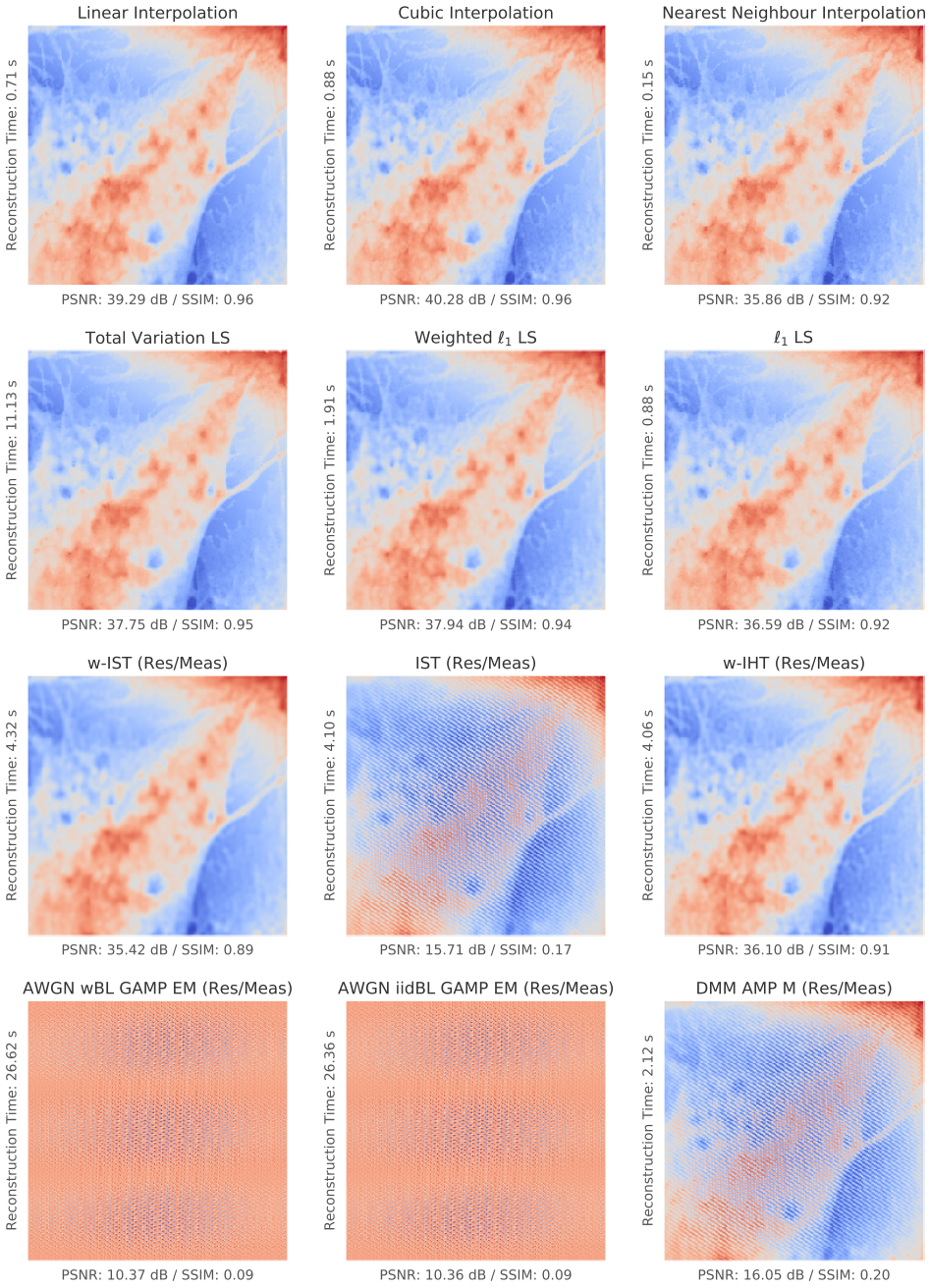


Figure H.3: Typical reconstructions of the first AFM image shown in Figure 7.1 when using rotated uniform line sampling and an undersampling ratio of $\delta = 0.15$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

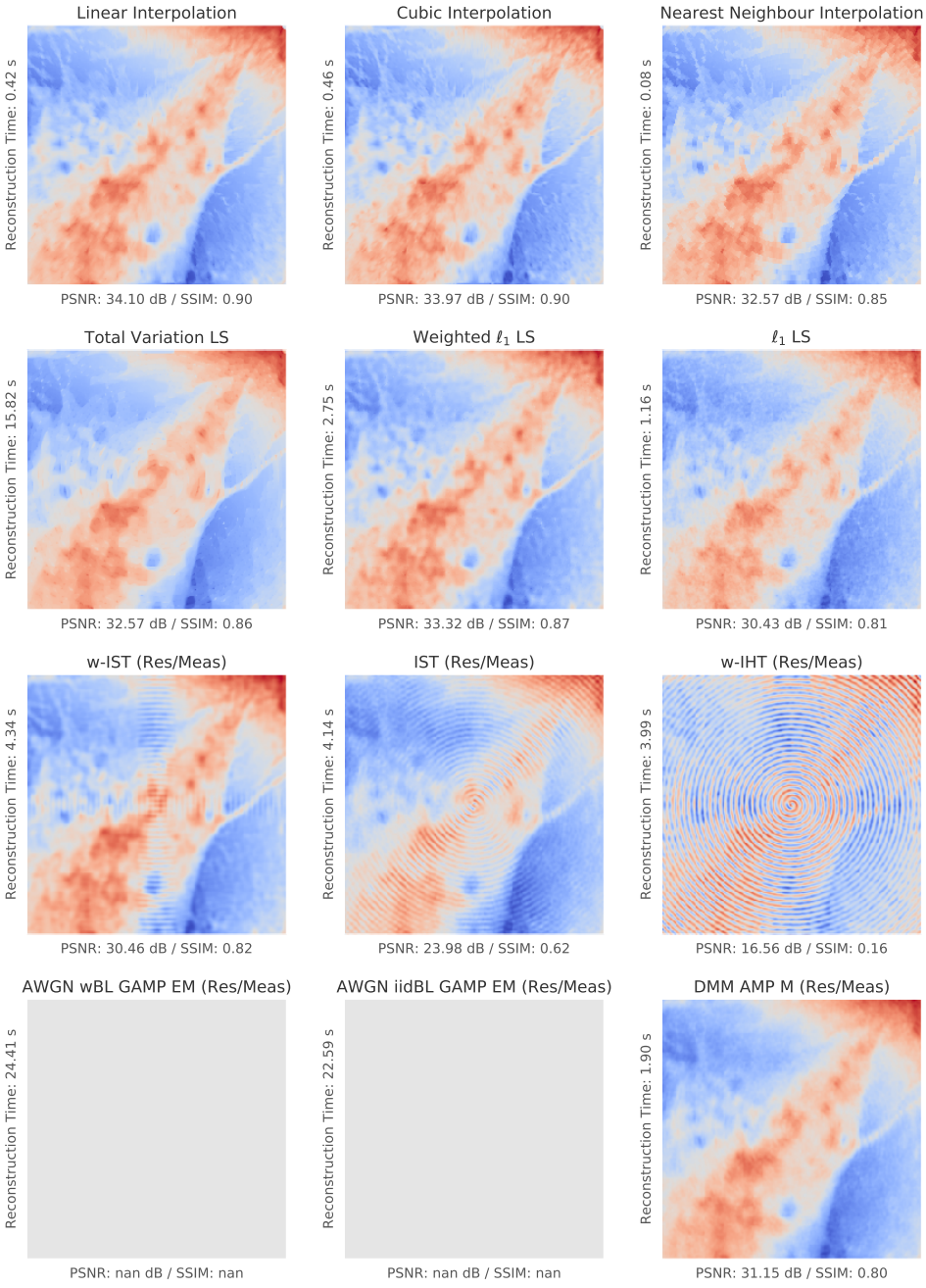


Figure H.4: Typical reconstructions of the first AFM image shown in Figure 7.1 when using spiral (including corners) sampling and an undersampling ratio of $\delta = 0.15$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

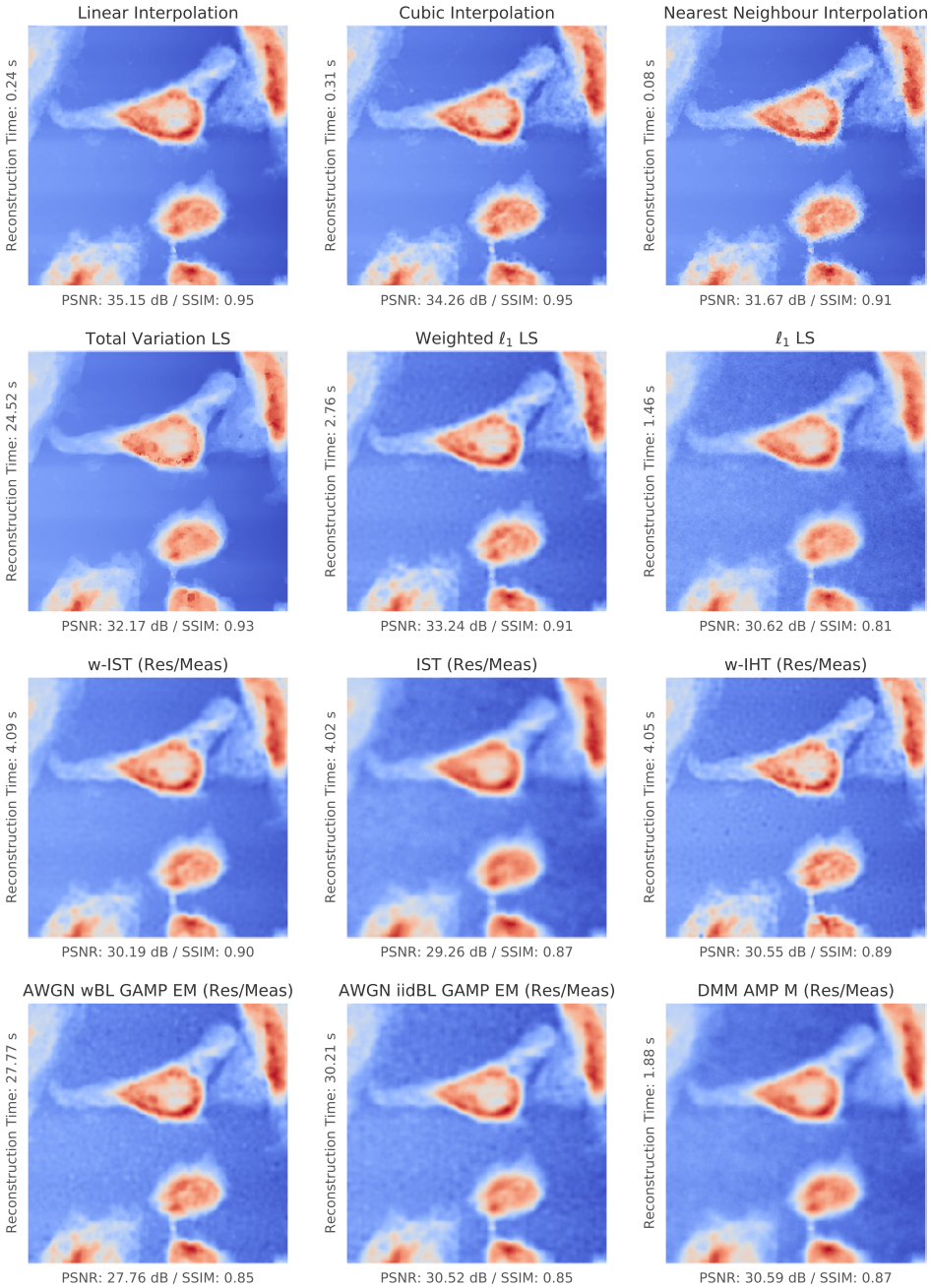


Figure H.5: Typical reconstructions of the second AFM image shown in Figure 7.1 when using random pixel sampling and an undersampling ratio of $\delta = 0.10$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

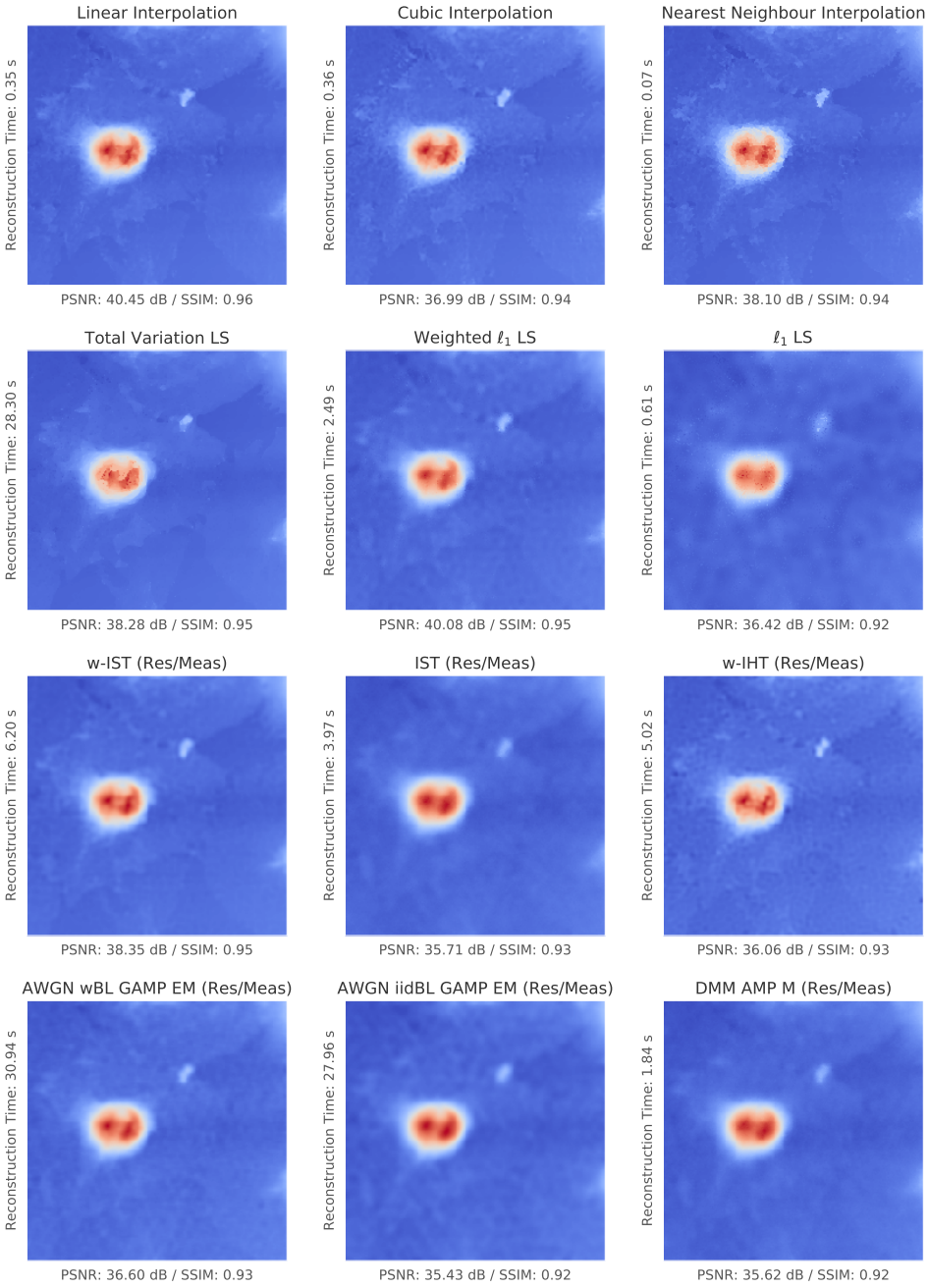


Figure H.6: Typical reconstructions of the third AFM image shown in Figure 7.1 when using random pixel sampling and an undersampling ratio of $\delta = 0.10$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

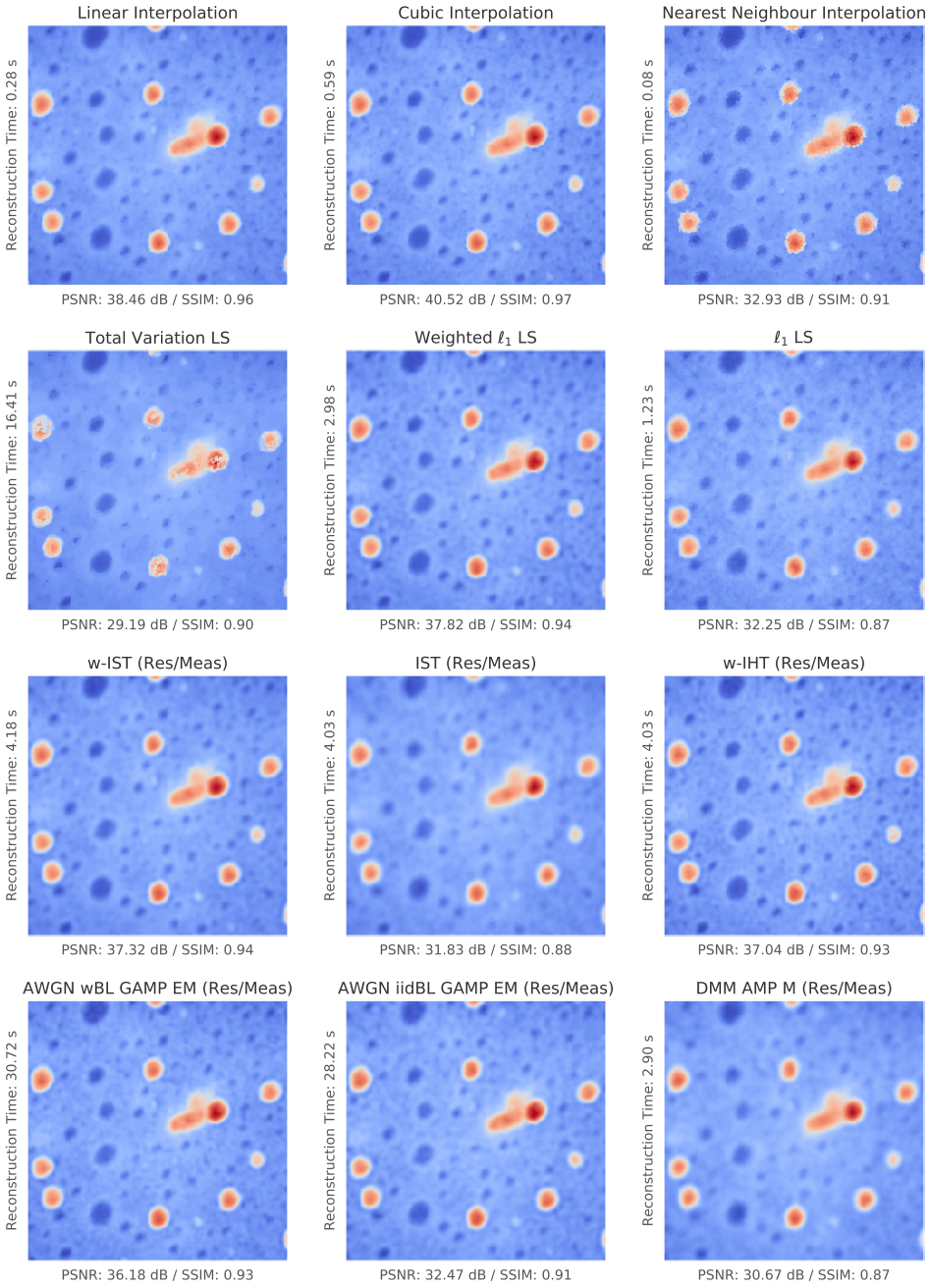


Figure H.7: Typical reconstructions of the eleventh AFM image shown in Figure 7.1 when using random pixel sampling and an undersampling ratio of $\delta = 0.10$. The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

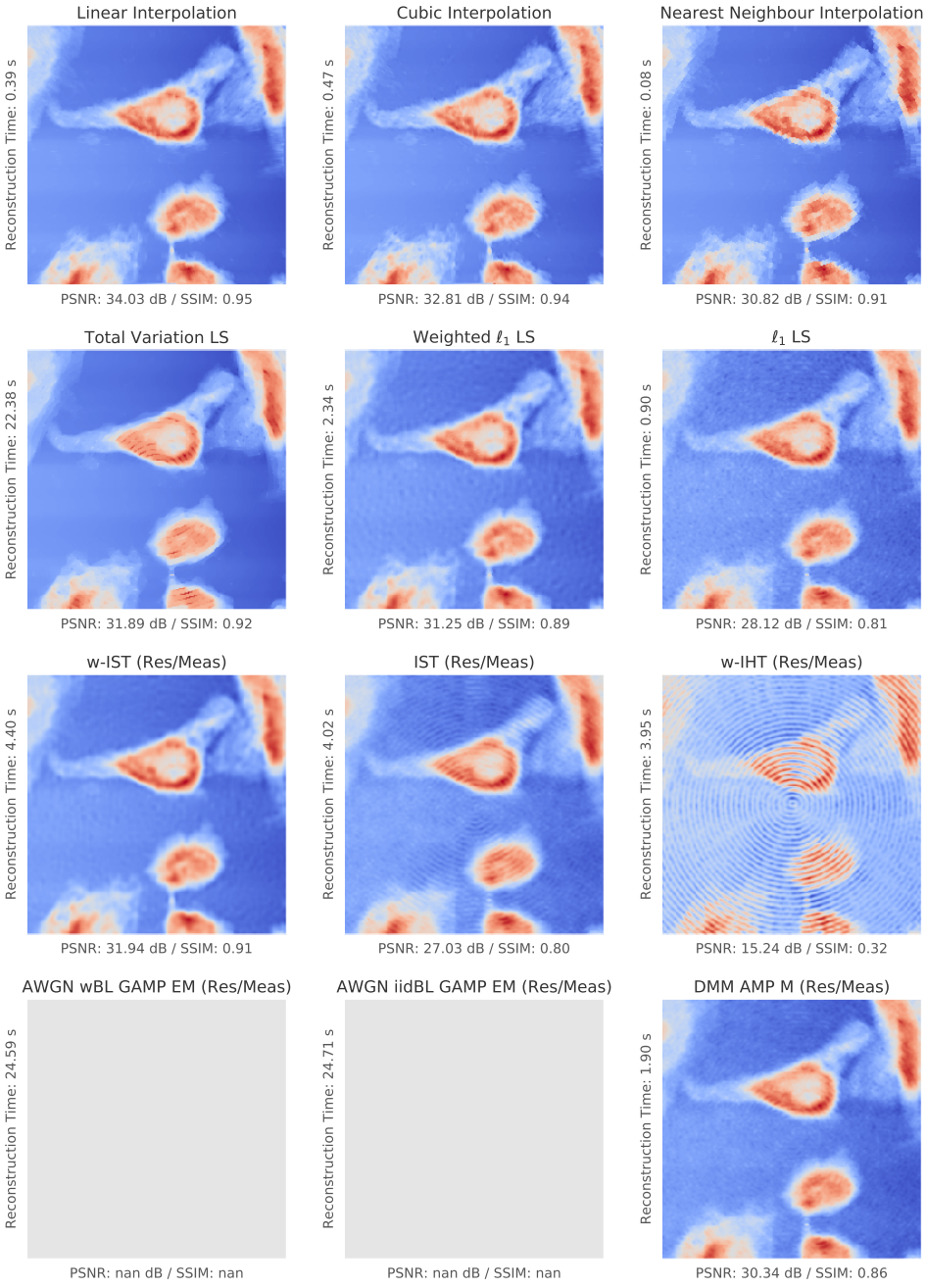


Figure H.8: Typical reconstructions of the second AFM image shown in Figure 7.1 when using spiral (including corners) sampling and an undersampling ratio of $\delta = 0.15$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

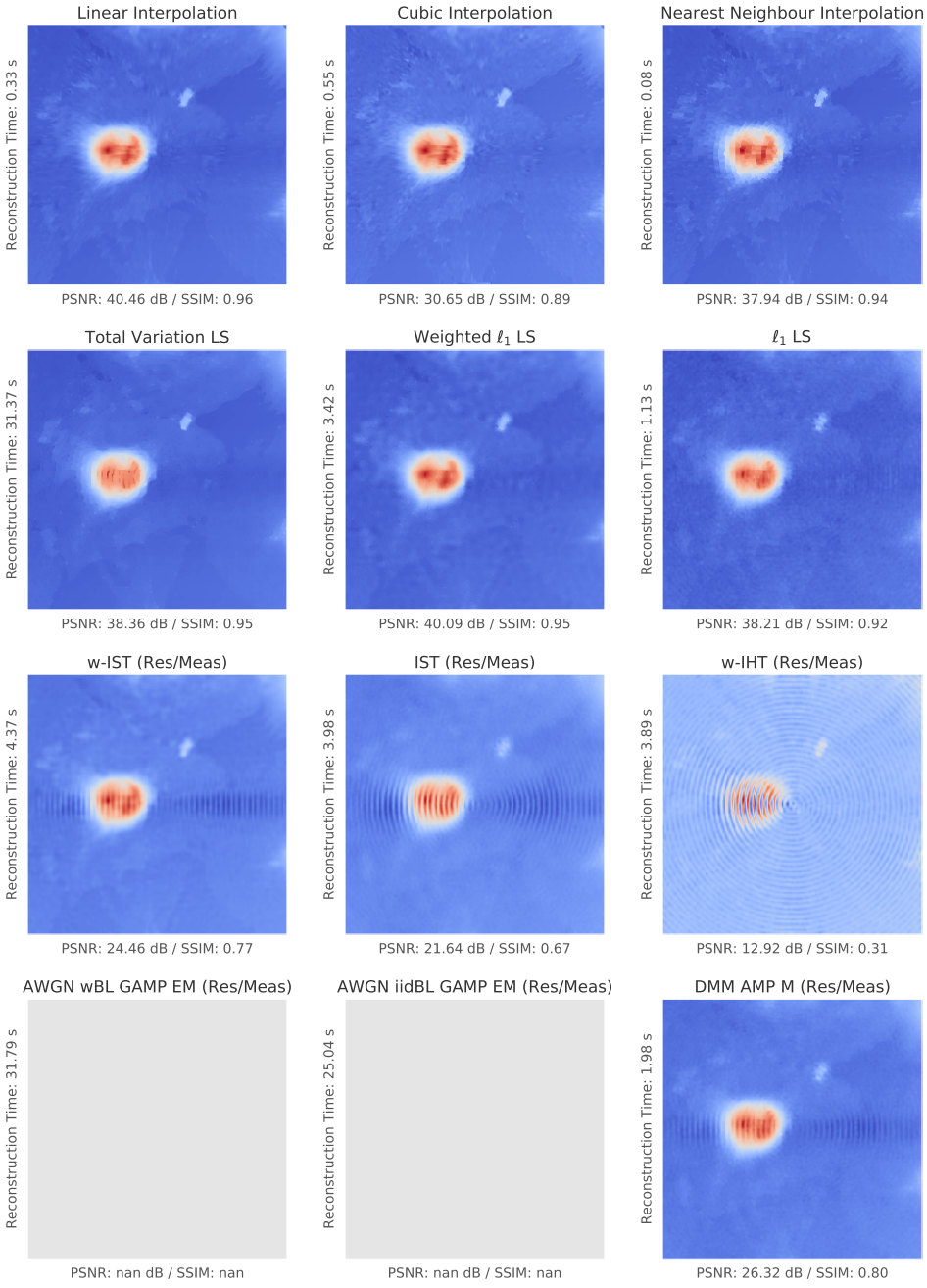


Figure H.9: Typical reconstructions of the third AFM image shown in Figure 7.1 when using spiral (including corners) sampling and an undersampling ratio of $\delta = 0.15$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

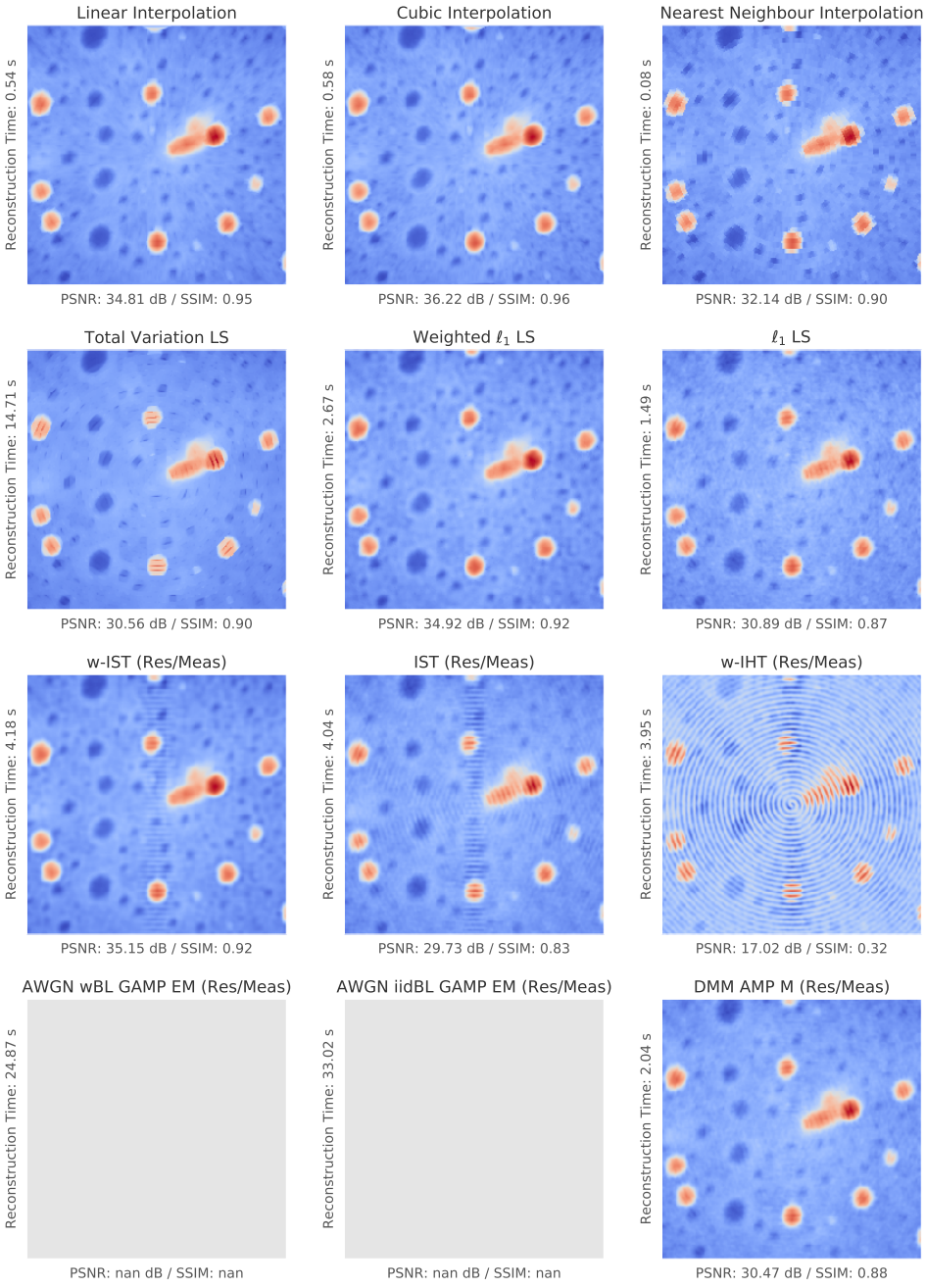


Figure H.10: Typical reconstructions of the eleventh AFM image shown in Figure 7.1 when using spiral (including corners) sampling and an undersampling ratio of $\delta = 0.15$. In this setting, the GAMP algorithm diverges yielding a solution of all NaNs (not a number). The reader is encouraged to study the details in this figure using the electronic version of this thesis available at doi:10.5278/vbn.phd.engsci.00158.

ISSN (online): 2446-1628
ISBN (online): 978-87-7112-951-9

AALBORG UNIVERSITY PRESS