Structural Time Domain Identification Toolbox User's Guide

Andersen, P.; Kirkegaard, Poul Henning; Brincker, Rune

Publication date:
1997

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

Link to publication from Aalborg University

Citation for published version (APA):
Andersen, P., Kirkegaard, P. H., & Brincker, R. (1997). *Structural Time Domain Identification Toolbox User's Guide*. Dept. of Building Technology and Structural Engineering, Aalborg University. Fracture and Dynamics Vol. R9701 No. 97

INSTITUTTET FOR BYGNINGSTEKNIK
DEPT. OF BUILDING TECHNOLOGY AND STRUCTURAL ENGINEERING
AALBORG UNIVERSITET • AAU • AALBORG • DANMARK

FRACTURE & DYNAMICS
PAPER NO. 97

Version 2.0 for use with MATLAB® version 4.2

P. ANDERSEN, P.H. KIRKEGAARD, R. BRINCKER

STRUCTURAL TIME DOMAIN IDENTIFICATION TOOLBOX USER'S GUIDE

NOVEMBER 1997 ISSN 1395-7953 R9701

Structural Time Domain IdentificationToolbox

Version 2.0 for use with MATLAB® version 4.2

User's Guide

By Palle Andersen, Poul Henning Kirkegaard & Rune Brincker

The software described in this document is developed by:

Assistant professor P. Andersen
Associated professor Poul Henning Kirkegaard
Associated professor Rune Brincker

Department of Building Technology and Structural Engineering.
Aalborg University
Sohngaardsholmsvej 57
DK-9000 Aalborg.
Denmark.

Comments and corrections will be appreciated.
Please E-mail these to Palle Andersen: i6pa@civil.auc.dk

Structural Time Domain Identification Toolbox User's Guide
Fracture and Dynamics, No. 97.
ISSN 1395-7953 R9701, 1997

Printed at Aalborg University

Table of Contents

1 Introduction

This manual describes the *Structural Time Domain Identification* toolbox for use with MATLAB. This version of the toolbox has been developed using the PC-based MATLAB version 4.2c, but is compatible with prior versions of MATLAB and UNIX-based versions. The routines of the toolbox are the so-called m-files that can be executed from the MATLAB command line prompt or built into other m-files.

The primary purpose of the toolbox is system identification of stochastic excited linear and time-invariant systems. In some cases though, it is possible to include measured excitation into the identification process as well. One of the primary results of the system identification is the estimation of modal parameters. In this toolbox the estimated modal parameters are the natural eigenfrequencies, the damping ratios and the scaled complex mode shapes. The identification of these parameters can be obtained by different identification techniques.

1.1 State Space Representation

The identification of the modal parameters can be based on the calibration of a state space system on the basis of measurements. In case of ambient excited structures only the system response will be available. In this case the dynamic behaviour of a linear and time-invariant system subjected to stationary ambient excitation can be modelled by the following system

$$\begin{aligned} x(t_{k+1}) &= Ax(t_k) + Ke(t_k) \ , \qquad e(t_k) \in NID(0,\Lambda) \\ y(t_k) &= Cx(t_k) + e(t_k) \end{aligned}$$

where the dimensions and the parameters of the matrices A, C and K, together with the innovations $e(t_k)$, are adjusted in order to describe the measured system response $y(t_k)$ optimally.

1.2 ARMAV Representation

The innovation system can also be represented in polynomial form by the so-called Auto-Regressive Moving Average Vector model

$$\begin{aligned} & y(t_k) + A_1 y(t_{k-1}) + \dots + A_n y(t_{k-n}) = \\ & e(t_k) + C_1 e(t_{k-1}) + \dots + C_n e(t_{k-n}) \ , \qquad e(t_k) \in NID(0,\Lambda) \end{aligned}$$

In this case, the auto-regressive coefficient matrices and the moving average coefficient matrices, together with the innovations $e(t_k)$, are adjusted in order to describe the measured system response $y(t_k)$ optimally.

The innovation state space system and the ARMAV model are equivalent descriptions of a stationary excited linear and time-invariant system. For a p-variate system the relation between the state space dimension m and the order n of the ARMAV model is $np = m$.

1.3 Modal Parameter Estimation

The modal parameters of a linear and time-invariant system are obtained by a modal decomposition of the transition matrix A

$$\Psi^{-1}A\Psi = \mu \,, \quad \mu = diag\{\mu_j\} \qquad j = 1,\, 2,\, \ldots \,,\, m$$

where Ψ is the complex modal matrix, whose columns are the eigenvectors of the transition matrix. μ is a diagonal matrix of associated eigenvalues. The complex mode shapes are extracted from the complex modal matrix as

$$\Phi = C\Psi$$

where Φ is a matrix, whose columns are the scaled complex mode shapes. By means of the sampling interval T, the natural eigenfrequencies f_j and the damping ratios ζ_j are extracted from the discrete-time eigenvalues μ_j of the underdamped structural modes as

$$\left.\begin{aligned} \lambda_j &= \frac{log\,(\mu_j)}{T} \\ f_j &= \frac{|\lambda_j|}{2\pi} \\ \zeta_j &= -\frac{Re\,(\lambda_j)}{|\lambda_j|} \end{aligned}\right\} \qquad j = 1,\, 2,\, \ldots \,,\, m$$

If the identified system is described by and ARMAV model this model is converted into state space before the modal decomposition is performed.

1.4 Organizing the Results

One of the most important problems that must be solved by a good system identification software is, how to organize all the results that are generated through a system identification session. In this context, it is important to distinguish between primary and secondary results. Primary results are defined as results obtained directly from an identification algorithm. These results will typically be:

- *Model structure information.*
- *Estimated model parameters.*
- *Estimated uncertainties of the model parameters.*
- *Estimated innovation covariance matrix.*
- *Performance criteria.*

The model structure information will typically be a set of parameters describing the dimensions of the identified model, and which of the parameters of the model have been estimated and which of them have been held constant. Other primary results are the estimated uncertainties of the estimated model parameters. These uncertainties are based upon the Hessian of the criterion function and must therefore be estimated inside the identification algorithm. This also accounts for the estimated innovation covariance matrix, which is based upon the prediction errors calculated from the final parameter estimates and the measurements, and the performance measures such as the AIC and FPE criteria. Based on these primary results any secondary result can be calculated.

Typical secondary results are the modal parameters and the spectral densities of the response of the model. The decisions about whether the model is optimal or not can also be characterized as a secondary result, since this decision is based on analysis of different primary results. Common to all secondary results is that they can be calculated at any time if the primary results are available. It is therefore unnecessary to store these results. In the toolbox the storage of the primary results is performed by organizing these in one matrix structure, which is called a DDS structure matrix. DDS is an abbreviation of Data Dependent Systems which is a synonym for system identification based on measured data. The content of this matrix is presented in table 1.1.

The DDS Structure Matrix
Model structure information
Index of adjustable parameters
Adjustable and non-adjustable model parameters
Estimated covariance matrix of adjustable model parameters
Estimated innovation covariance matrix
Number of measurement channels
Number of measurements in each channel
Scaling matrix of the measurements
Sampling interval
Final loss (Final value of the criterion function)
Akaike's Final Prediction Error Criterion (FPE)
ID number for the routine that created the structure
Time and date for the creation of the structure

Table 1.1: Information stored in the DDS structure matrix.

The creation of the structure is performed automatically by the identification routines, and the access to the different elements of the structure is provided by different functions. A call to an identification routine in the MATLAB environment will typically look like

```
» dds = function( y, ddsinit )
```

where `y` is the measured output and `ddsinit` is an initial DDS structure containing initial model parameters and model structure information. The primary results of the identification are then returned in `dds`. On the basis of the primary results stored in `dds` the model validation routines are typically called by

```
» valid_result = function( dds, y )
```

and secondary results, such as the modal parameters, are typically obtained by

```
» second_result = function( dds )
```

2 An Introductory Example

The best way to introduce the toolbox is by considering an example. The following example is an edited version of the demonstration m-file `demostdi.m` located in the toolbox. This file can be executed from the MATLAB command line prompt as

```
» demostdi
```

2.1 Simulating System Response

This file demonstrates some of the routines of the toolbox. In order to generate some response of some system, a continuous-time Gaussian white noise excited 3 degree-of-freedom system is constructed.

The system is described by the following stiffness matrix `K`, viscous (non-proportional) damping matrix `C`, and mass matrix `M`.

```
K =
    350   -150      0
   -150    450   -300
      0   -300    400

C =
     0.7000    -0.2000          0
    -0.2000     0.4000    -0.2000
          0    -0.2000     0.8000

M =
       1      0      0
       0      1      0
       0      0      1
```

The continuous-time Gaussian white noise excitation is described by the intensity matrix `W` given by

```
W =
       2      0      0
       0      2      0
       0      0      2
```

On the basis of this information a covariance equivalent discrete-time ARMAV(2,1) model can be established. The simulated samples of the response obtained from such a model has the correct statistical properties in case of Gaussian distributed continuous-time system response.

The conversion to discrete time is performed by the following command line call :

```
» T=0.075;
» dds=armav21(M,C,K,T,W);
```

where `T` is the sampling interval. The structure matrix `dds` contains all the necessary primary information about the constructed ARMAV model. This information can be furnished to the screen using the command line call:

```
» showdds(dds);
```

The output from this call is the following information:

```
This discrete-time 3-channel ARMAV(2,1) model was created by the
command ARMAV21 on the 24/9 1997 at 09:46.
It was based on 0 samples in each channel, and a sampling interval
of 0.075 seconds.
Loss function: 0, Akaikes FPE: 0
The parameters and their standard deviations given as imaginary
parts are:
A =
  Columns 1 through 7
1       0       0   -0.3769   -0.5539   -0.1014    0.9486
0       1       0   -0.5630   -0.1967   -1.0919    0.0118
0       0       1   -0.0973   -1.0737   -0.3301    0.0005

  Columns 8 through 9
0.0177    -0.0020
0.9726     0.0087
0.0201     0.9403

C =
1       0       0    0.3401   -0.0350    0.0034
0       1       0   -0.0378    0.3752   -0.0774
0       0       1    0.0045   -0.0721    0.3581

The innovation covariance matrix is:
Lam =
  1.0e-003 *
    0.3283     0.0652     0.0114
    0.0652     0.3167     0.1217
    0.0114     0.1217     0.3214

The scaling matrix of the response is:
Scale =
     1      0      0
     0      1      0
     0      0      1
```

The system response can now be simulated using the following command line call :

```
» p=3;
» N=3000;
» y=ddssim(dds,randn(N,p));
```

The number of simulation in each of the three outputs is 3000. Noise can then be added in order to simulate real measurements even more :

```
» y=y+0.1*mean(std(y)')*randn(N,p);
```

The added noise is zero-mean Gaussian white noise with a standard deviation equal to 10 per cent of the averaged standard deviations of the noise-free simulations.

2.2 Model Estimation

The modal parameters of the simulated system can then be estimated from the measurements by the estimation of an adequate ARMAV(2,2) model. The estimation will be based on the Prediction Error Method. This method requires a good initial estimate which can be provided by the stochastic subspace ARMAV estimator. This estimator can be invoked by the following command line call:

```
» ddsinit=armavnn(y,20,2,T);
```

Again the information stored in `ddsinit` can be viewed:

```
» showdds(ddsinit);
```

```
This discrete-time 3-channel ARMAV(2,2) model was created by the
command ARMAVNN on the 24/9 1997 at 09:52.
It was based on 3000 samples in each channel, and a sampling
interval of 0.075 seconds.
Loss function: 0.0000937, Akaikes FPE: 0.00009597
The parameters and their standard deviations given as imaginary
parts are:
A =
  Columns 1 through 7
1       0       0  -0.3743    -0.7761    -0.1522     0.9524
0       1       0  -0.3950    -0.1948    -1.0534     0.0138
0       0       1  -0.0676    -1.1153    -0.3295     0.0025

  Columns 8 through 9

    0.0293   -0.0081
    0.9654    0.0100
    0.0165    0.9478

C =
  Columns 1 through 7

1       0       0   0.0495    -0.2883     0.0255     0.2254
0       1       0  -0.1257     0.2070    -0.3766    -0.0214
0       0       1   0.0217    -0.4388     0.1427     0.0144

  Columns 8 through 9

   -0.0958    0.0072
    0.2391   -0.0805
   -0.0971    0.2178

The innovation covariance matrix is:
Lam =
    0.0721    0.0124    0.0049
    0.0124    0.0383    0.0147
    0.0049    0.0147    0.0416

The scaling matrix of the response is:
Scale =
    0.0953         0         0
         0    0.1355         0
         0         0    0.1304
```

Based on this initial model information as well as on the system response the Prediction Error Method can be used to iteratively improve the estimate of the state space system.
The prediction error method for ARMAV models is applied in the following manner:

```
» ddsarmav=armav(y,ddsinit);
```

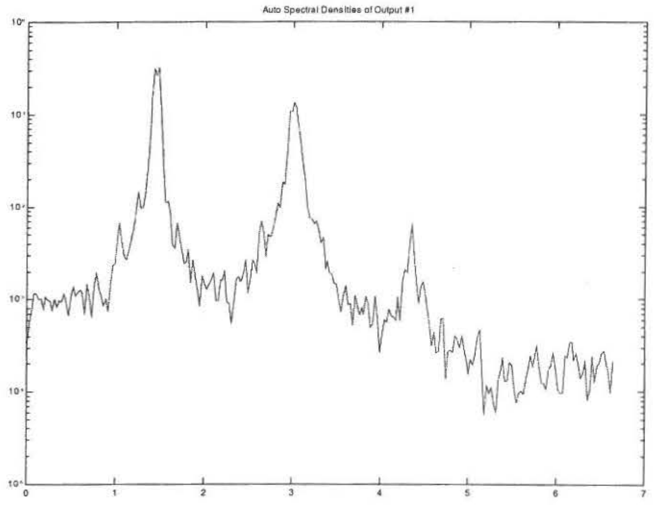
This routine will then iteratively reduce the prediction errors of the model. In other words, the differences between the measurements and the system response predicted by the estimated model.

2.3 Model Validation

One way to validate the performance of the ARMAV model is to compare the spectral densities obtained from the model and from FFT. Such comparative plots are obtained from the command line call:

```
» speccmp(ddsarmav,y);
```

An example of such a plot is shown below.



2.3 Modal Parameter Estimation

Assuming that the performance of the estimated model is satisfactory the modal parameters can be estimated. This can be accomplished by the following command line call:

```
» [phi,f,z]=modal(ddsarmav,1);
```

with `phi` being a matrix whose columns are the scaled complex mode shapes. `f,z` are vectors of associated natural eigenfrequencies and damping ratios.

These modal parameters can be compared with the ones of the simulation system by the following command line calls:

```
» [phi0,f0,z0]=modal(dds,1);
» disp(phi0);

   0.5934 + 0.0041i  -1.9207 + 0.0194i   0.4387 + 0.0052i
   1.0583 + 0.0078i   0.1322 + 0.0027i  -1.1905 + 0.0184i
   1.0000             1.0000             1.0000

» disp(phi)

   0.5948 + 0.0039i  -1.9402 + 0.0442i   0.4337 + 0.0160i
   1.0626 + 0.0071i   0.1252 + 0.0025i  -1.1951 + 0.0103i
   1.0000             1.0000             1.0000

» disp([f0 100*z0]);

    1.4454    1.8261
    3.0211    1.9241
    4.3794    1.5225

» disp([f 100*z]);

[f 100*z] =
    1.4508    1.7884
    3.0246    1.7607
    4.3818    1.6321
```

which reveal a good correspondence of all estimated and simulated modal parameters.

3 Installation

This toolbox is installed by copying the files of the installation disk to the harddisk. We recommend that the files are installed in the MATLAB directory and that the installation directory is named `stdi`. Afterwards this directory must be added to the MATLAB search path by editing the m-file `matlabrc.m` located in the MATLAB directory.

Having installed the toolbox correctly it should be possible to execute the routine `initstdi`. This routine initialize the default parameters used by the routines of the toolbox. Further, it configures the font size of the graphical plotting facilities of the toolbox. All settings are saved in the file `stdi.mat` which must be located in a directory present in the MATLAB search path. If the routine is not called initially, a default `stdi.mat` will be generated automatically. The values of the `stdi.mat` file can always be edited by calling `initstdi`.

`initstdi` is a menu-driven routine and is easy to use. The following options are available

```
1. Retrieve Default Configuration
2. Edit Plot and Axis Font Sizes
3. Edit PEM Optimization Parameters
4. Edit Sampling Interval
H. Help
S. Save and Quit
Q. Quit
```

The default configuration (1) sets all parameters and font sizes to defaults values. The font sizes can be edited by the option (2). The algorithms implementing the prediction error method for different model structures also need to be configured which is performed by the option (3). If a particular sampling interval is used frequently this can be set by the option (4). Help is provided by the option (H) and the configuration is saved using option (S). After saving the routine quits. This can also be accomplished without saving using option (Q).

4 Reference

This section contains detailed descriptions of all the functions in the Structural Time Domain Identification Toolbox. It begins with a list of functions grouped by a subject area and continues with the *Reference* entries in alphabetical order. Information is also available through the online Help facility.

For ease of use, most functions have several default arguments. The Synopsis first lists the function with the necessary input arguments and then with all possible input arguments. The functions can be used with any number of arguments in between these extremes. The rule is that missing, trailing arguments are given default values, as defined in the manual or by the routine `initstdi`. Default values are also obtained by entering the arguments as the empty matrix [ ].

MATLAB does not require that you specify all the output arguments; those not specified are not returned. For functions with several output arguments in the Structural Time Domain Identification Toolbox, missing arguments are, as a rule, not computed, in order to save time.

Non-parametric estimation	
`fftcor`	Correlation function estimate for a data matrix.
`fftcov`	Covariance function estimate for a data matrix.
`fftspec`	Power spectrum estimation of a data matrix.

Parameter Estimation	
`arv`	ARV models of signals using a least-squares approach.
`armav`	ARMAV models of signals using the Prediction Error Method.
`armavnn`	ARMAV models of signals using the stochastic subspace technique
`armavgls`	ARMAV models of signals using a generalized least-squares approach.
`na4sid`	Estimation of a stochastic state space realization of signals using the stochastic subspace technique.
`sspem`	Estimation of a stochastic state space realization of signals using the Prediction Error Method.
`initpem`	Generate initial information for `sspem`

Simulation and Prediction	
`dds2pe`	Computes the prediction errors of an estimated model.
`dds2pred`	Computes the one-step ahead prediction of an estimated model.
`ddssim`	Simulate the response of a model.

Model Structure Creation	
`armav21`	DDS structure of a covariance equivalent ARMAV(2,1) model.
`mck2dds`	DDS structure of a second-order continuous-time system.
`par2dds`	DDS structure created from matrix polynomial parameters.
`poly2dds`	DDS structure from given matrix polynomials.
`ss2dds`	DDS structure of a given state space realization.

Model Conversions	
`dds2cov`	Covariance function of a linear system excited by Gaussian white noise.
`dds2frf`	Frequency response function and noise spectral densities.
`dds2par`	Parameters of the model contained in the DDS structure.
`dds2poly`	Computes the matrix polynomials associated with a given model.
`dds2ss`	Convert the DDS structure to a state space realization.
`modal`	Modal decomposition of a model.

Model Presentation	
`dds2spec`	Plots frequency response functions and noise spectral densities.
`showdds`	Furnishes information about the model to the screen.

Model Validation	
`dds2pe`	Plot prediction errors of the estimated model.
`dds2pred`	Plot predicted and measured response of the estimated model.
`speccmp`	Compare spectral densities of several estimated models and measurements.

Manipulating Model Structures	
`t2dds`	Set sampling interval.
`dds2dds`	Internal conversion of the DDS structure.

Information Extraction	
`dds2id`	System identification information from a model.
`dds2t`	Returns the sampling interval on which a model is based.
`dim/dimss`	Dimensions of the model and the number of channels.
`modal`	Modal parameters of the model.
`isdds`	Checks if a matrix is an DDS structure.
`isddscs`	Checks if an DDS structure contains a continuous-time model.
`isddsss`	Checks if an DDS structure contains a state space realization.

Assessing Model Uncertainty	
`dds2par`	Covariance matrix of estimated parameters.
`sdmodal`	Estimated uncertainties of the modal parameters.

armav

Purpose

ARMAV models of signals using the Prediction Error Method.

Synopsis

```
dds=armav(y,NN)
dds=armav(y,NN,maxiter,tol,maxsize,T,lim,nocovar,noscale,app)
```

Description

The parameters of the full polynomial ARMAV model structure

$$A(q)y(t_k) = C(q)e(t_k)$$

are estimated using a prediction error method [1].

`y` is a matrix with the time series data of each channel stored columnwise. NN contains initial value and structure information. The `dds` structure matrix is returned with the resulting parameter estimates, together with estimated covariance matrices. When no initial parameter estimates are available, enter `NN` as `NN=[na nc]`, the orders of the above model. With an initial estimate available in `ddsinit`, a DDS structure matrix of standard format, enter `NN=ddsinit`. Then the criterion minimization is initialized at `ddsinit`.

The optional argument `app` is a string that controls the search-scheme to use. If `app='GN'` a Gauss-Newton search-scheme is applied. If `app='ML'` the search-scheme will be of the Marquardt-Levenberg type. The optional auxiliary arguments `maxiter`, `tol`, `maxsize`, `T`, `nocovar` and `noscale` are explained under `auxlsvar`.

Algorithm

If `approach='GN'` a robustified quadratic prediction error criterion is minimized using an iterative damped Gauss-Newton algorithm. The Gauss-Newton vector is bisected up to ten times until a lower value of the criterion function is found. The iterations are terminated when `maxiter` is reached, when the Gauss-Newton search gradient has a norm less than `tol`, or when a lower value of the criterion cannot be found.

If `approach='ML'` the same robustified quadratic prediction error criterion is minimized using an iterative Marquardt-Levenberg algorithm. The iterations are terminated when `maxiter` is reached, when the Marquardt-Levenberg search gradient vector has a norm less than `tol`, or when a lower value of the criterion cannot be found.

In order to obtain an initial guess of the parameters an initialization using a two-stage linear least-squares algorithm is used. This initialization is allowed to use ten iterations before terminating.

The cutoff value for the robustification is based on the parameter `lim` as well as on the estimated standard deviation of the residuals from the current parameter estimate. The returned loss function and innovation covariance matrix are the non-robustified ones.

A stability test of the predictor is performed, so as to assure that only models corresponding to stable predictors are tested. Generally, the moving average matrix polynomial must have all its roots inside the unit circle.

Information about the minimization is furnished to the screen. Currently estimated parameters as well as values of the current and previous criterion functions are given. The norm of either the Gauss-Newton or the Marquardt-Levenberg search gradient is also displayed. The number in the upper left corner is the number of times the length of the search vector is adjusted. If configured for it, a graphical PEM Information Window will appear and present the information as well. This is controlled by the routine `initstdi`.

If configure for it, the routine will save all information about the identification after each iteration. In this way the currently obtained results are saved if a breakdown of the system should occur or if CRTL-C is hit. This is controlled by the routine `initstdi`. If the routine is stopped before completion, the data can be restored by the routine `tmp2dds`.

See Also

`armavnn`, `armavgls`, `sspem`, `auxlsvar`, `ddsstruc`, `tmp2dds`, `initstdi`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

armav21

Purpose

Generate a discrete-time covariance equivalent ARMAV model of a continuous-time second order linear system excited by Gaussian white noise.

Synopsis

```
dds = armav21(M,C,K,T)
dds = armav21(M,C,K,T,W,disp)
```

Description

The parameters of the p-variate ARMAV(2,1) model structure

$$A(q)y(t_k) = C(q)e(t_k)$$

are calculated on the basis of equivalence of the covariance function of the linear continuous-time time-invariant Gaussian white noise excited system, described by the differential equation

$$M\ddot{y}(t) + C\dot{y}(t) + Ky(t) = w(t) , \qquad w(t) \in NID(\mathbf{0}, W)$$

and the discrete system. `M`, `C` and `K` are the $p \times p$ mass, viscous damping and stiffness matrices, respectively [1,2]. `W` is the intensity matrix of the continuous-time zero-mean Gaussian white noise $w(t)$, which by default is the identity matrix. `T` is the sampling interval. If `T` is larger than 0.5 times the smallest eigenperiod (the inverse of a natural eigenfrequency), a warning will be given and `T` is altered to 90% of this eigenperiod times 0.5. By default the covariance equivalence is established for the displacements. However, by setting the boolean `disp` to 0 it is established for the velocities instead. In this case any simulated response of the ARMAV(2,1) model will correspond to the velocities of the continuous-time system.

Example

Generate 1000 samples of a covariance equivalent response of a p-variate linear system, defined by a $p \times p$ mass matrix `M`, a $p \times p$ damping matrix `C`, and a $p \times p$ stiffness matrix `K`. Assume that the system is excited by a zero-mean Gaussian white noise excitation described by the intensity matrix `W`, and that the system is sampled using a sampling interval `T`.

The displacements are simulated as

```
dds = armav21(M,C,K,T,W);
y = ddssim(dds,randn(1000,p));
```

and the velocities as

```
        dds = armav21(M,C,K,T,W,0);
        ydot = ddssim(dds,randn(1000,p));
```

See Also

`ddssim, ddsstruc`

Reference

[1] P. Andersen, R. Brincker & PH. Kirkegaard: *Theory of Covariance Equivalent ARMAV Models of Civil Engineering Structures.* 14th International Modal Analysis Conference, Dearborn, USA, 1996.

[2] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models.* Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

armavgls

Purpose

ARMAV models of signals using a generalized least-squares approach.

Synopsis

```
dds = armavgls(y,NN)
dds = armavgls(y,NN,maxiter,tol,maxsize,T,alpha,noscale)
```

Description

The parameters of the full polynomial ARMAV model structure

$$A(q)y(t_k) = C(q)e(t_k)$$

are estimated using a two-stage iterative linear least-squares method, [1,2].

`y` is a matrix with the time series data of each channel stored columnwise. NN contains initial value and structure information. The `dds` structure matrix is returned with the resulting parameter estimates. When no initial parameter estimates are available, enter NN as NN=`[na nc]`, the orders of the above model. With an initial estimate available in `ddsinit`, a DDS structure matrix of standard format enter NN=`ddsinit`. Then the criterion minimization is initialized at `ddsinit`.

The optional auxiliary arguments `maxiter`, `tol`, `maxsize`, `T` and `noscale` are explained under `auxlsvar`.

Algorithm

The difference between parameter estimates is minimized iteratively using a two-stage least-squares algorithm. Each iteration starts by determine the innovations using the latest parameter estimates. On the basis of the current innovations and the previous ones an external input is determined. This external input is used in a multivariable linear least-squares ARX model to obtain new parameter estimates. The mixture of the current innovations and the previous ones in the external input are of the following form

$$u(t) = (1-\alpha)e_{cur}(t) + \alpha e_{prev}(t)$$

where $u(t)$ is the external input, $e(t)$ the innovation at time step t, and α is equal to `alpha`. The reason for this mixing with the previous innovations is due to the convergence properties of the algorithm. Without this mixing the algorithm would have difficulties converging when being close to minimum. The value of `alpha` should always be between zero and one, and is by default 0.3.

The iterations are terminated when `maxiter` is reached, or when the norm of the differences of each of the moving average parameters in two consecutive iterations, is less than `tol`.

Information about the minimization is furnished to the screen. Currently estimated parameters as well as the norm of the differences of each of the currently and previously estimated moving average parameters.

See Also

`armav, armavnn, auxlsvar, ddsstruc`

Reference

[1] Giorcelli, E., Garibaldi, L., Fasana, A. & Riva, A.: *Modal Analysis and System Identification using ARMAV models*. Proc. 12. International Modal Analysis Conference, Hawaii, USA, 1994

[2] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

armavnn

Purpose

Stochastic Subspace Identification of a full-polynomial ARMAV(n,n) model.

Synopsis

```
dds = armavnn(y,m)
[dds,R] = armavnn(y,m,n,T,algo,posreal,maxsize,noscale)

[dds,R] = armavnn(R)
[dds,R] = armavnn(R,n,algo,posreal)
```

Description

The parameters of the full polynomial ARMAV model structure

$$A(q)y(t_k) = C(q)e(t_k)$$

are estimated using stochastic subspace method, [1]

`y` is a matrix with the time series data of each channel stored columnwise. The `dds` structure matrix is returned with the resulting parameter estimates. `m` is the half of the number of block rows in the Hankel matrix constructed from the data. `n` is the order of the auto-regressive and moving average polynomials. If not specified, the order must be determined using a plot of the singular values. After the right order there should be a significant gap between the singular values.

The recommended order is indicated by a green colour. By clicking on an order with the left mouse button the performance of this particular model can be evaluated. This evaluation is based on a comparison of the maximum singular values of the spectral densities of the data and the model. The selected model will be indicated with a red colour. To select a particular dimension click on it with the right mouse button. If the one-step ahead predictor of the selected model is unstable, this will be indicated. If this is the case, the estimated model cannot be used as initial estimate for the prediction error methods.

The string argument `algo` controls how the row space of the Hankel matrix containing future data is weighted and orthogonal projected onto the row space of the Hankel matrix containing past data.

If `algo='CVAQSVD'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Canonical Variate Algorithm [1]. The estimation of the system matrices is based on the Quotient Singular Value Decomposition (QSVD), [2].

If `algo='CVA'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Canonical Variate Algorithm. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach. By default `algo='CVA'`.

If `algo='PC'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Principal Component algorithm, [1,3]. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach.

If `algo='UPC'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Unweighted Principal Component algorithm, [1]. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach.

If the boolean argument `posreal` is true (`posreal=1`) a positive real solution of the steady-state covariance matrix is enforced. This is a necessary requirement for a correct estimation of the moving average and the innovation covariance matrix. If `posreal` is false (`posreal=0`), which it is by default, a positive real solution is not guaranteed and it might result in, e.g. bad spectral density plots or simulations. A warning will be given if the solution is not positive real.

If different orders `n` will be tried for the same `y` and `m`, the matrix `R` contains the necessary information needed for subsequent calls. For these subsequent calls the routine should be called in one of the following ways:

```
[dds,R] = armavnn(R)
[dds,R] = armavnn(R,n,algo,posreal)
```

Again if `n` is not specified, the order must be determined the singular values plot. Finally, please note that `m` and `n` must satisfy the inequality: `m >= n`. In practice for noisy data `m` should be much larger than `n`.

The optional auxiliary arguments `maxsize`, `T` and `noscale` are explained under `auxlsvar`.

Algorithm

On the basis of the user supplied information the routine estimates a stochastic state space realization using the algorithms described in [1,2]. Since the realization is guaranteed minimal, it can be effectively converted into an ARMAV(n,n) model using the approach described in [4].

See Also

`armav, armavgls, na4sid, sspem, auxlsvar, ddsstruc`

Reference

[1] Van Overschee, P. & B. De Moor: *Subspace Identification for Linear Systems. Theory, Implementation, Applications.* Kluwer Academic Publishers, ISBN 0-7923-9717-7, 1996.

[2] Van Overschee, P. & B. De Moor: *Subspace Algorithms for the Stochastic Identification Problem.* Automatica, Vol. 29, No. 3, pp. 649-660, 1993.

[3] Aoki, M.: *State Space Modeling of Time Series.* 2nd Ed., Springer Verlag, 1990.

[4] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models.* Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

Purpose

ARV models of signals using a least-squares approach.

Synopsis

```
dds = arv(y,na)
dds = arv(y,na,maxsize,T,nocovar,noscale)
```

Description

The parameters of the multivariate full-polynomial ARV model structure

$$A(q)y(t_k) = e(t_k)$$

are estimated using a least-squares method that solves a series of smaller subsystems in a memory and speed efficient way, [1].

Each column of the *N x ny* matrix `y` corresponds to the time series data of one of the *ny* channels. `na` defines the model order. The `dds` structure matrix is returned with the resulting parameter estimates, together with estimated covariance matrices.

The optional auxiliary arguments `maxsize`, `T`, `nocovar` and `noscale` are explained under `auxlsvar`.

See Also

`armav, armavgls, armavnn, auxlsvar, ddsstruc`

Reference

[1] L. Ljung: *System Identification - Theory for the User*. Prentice-Hall, Englewood Cliffs, 1987, chapter 7.

auxlsvar

Purpose

Describe auxiliary variables `maxiter`, `tol`, `lim`, `maxsize` and `T`.

Synopsis

```
help auxlsvar
```

Description

Most of the optimization functions have an optional argument `maxsize` that allows a tradeoff between memory usage and speed. Several functions allow the sampling period `T` to be specified. The iterative search procedures in `armavgls`, `armav` and `sspem` are controlled by `maxiter` and `tol`. The search procedure in `armav` and `sspem` is further controlled by `lim`.

`maxsize`: No matrix formed by the function is allowed to contain more than `maxsize` elements. Instead, the algorithms split the calculations into `for`-loops, which are slower. The default value of `maxsize` is set by editing `inistdi`. On small machines, it is `maxsize=4096`. The main use of `maxsize` is to limit variable sizes when the algorithms run out of memory.

`T`: Specifying the sampling period `T` gives correct frequency scales on frequency domain plots, and correct time scales on domain plots. Further, the eigenfrequencies obtained from `modal`, and standard deviations of eigenfrequencies obtained from `sdmodal` will be correctly scaled. The default value is `T=1`.

`maxiter`: This variable determines the maximum number of iterations performed during search for a minimum. The default value is `maxiter=10`. `maxiter=0` returns the results of the initial estimate.

`tol`: In `armav` and `sspem` the iterations are continued until the norm of the search gradient is less than `tol`. The iterations also terminate when the algorithm fails to find a lower value of the criterion function and when the maximum number of iterations are reached. In `armavgls` the iterations are continued until the norm of the differences of each of the moving average parameters in two consecutive iterations is less than `tol`. The iterations also terminate when the maximum number of iterations are reached. The default value is `tol=0.0001`.

`lim`: This variable determines how the criterion is modified from quadratic to one that gives linear weight to large errors. See reference for a more precise definition. Default is `lim=0`, which means that a non-robustified (truly quadratic) criterion is used. If the criterion needs to be robustified a good starting value is `lim=1.6`.

nocovar: (Boolean) - If true (nocovar = 1) the algorithm will not estimate and store the covariance matrix of the adjustable parameters. By default (nocovar = 0), i.e. false and in this case the algorithm will estimate and store the covariance matrix.

noscale: (Boolean) - If true (noscale = 1) the algorithm will not scale each of the output measurement channels with its sampled standard deviation. By default (noscale = 0), i.e. false and in this case the algorithm will scale the output measurements.

Default values of these parameters are obtained either by omitting trailing arguments or by entering them as the empty matrix [ ]. The default values can be altered using the routine initstdi.

dds2cov

Purpose

Covariance function of a linear system excited by Gaussian white noise.

Synopsis

```
S = dds2cov(dds,n)
```

Description

The routine returns the covariance matrix `S` at time lags specified in the vector `n` of the response of a white noise excited system, described by the structure matrix `dds`. This routine only works for stochastic systems without deterministic input [1,2].

Example

Generate a covariance equivalent ARMAV model on the basis of a continuous time linear time-invariant system, defined by a $p \times p$ mass matrix `M`, a $p \times p$ viscous damping matrix `C`, and a $p \times p$ stiffness matrix `K`. Assume zero mean Gaussian white noise excitation described by an intensity matrix `W`. The system is sampled using a sampling interval `T`. Calculate the covariance matrix at discrete time lags 0, 1, and -1 of the response of the ARMAV model

```
dds = armav21(M,C,K,T,U);
S = dds2covd(dds,[0 1 -1]);
S0 = submat(S,1);
S1 = submat(S,2);
S_1 = submat(S,3);
```

See Also

`armav21, ddsstruc`

Reference

[1] Aoki, M.: *State Space Modeling of Time Series*. 2nd. Ed. Springer Verlag, 1990.

[2] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

dds2dds

Purpose

Multipurpose routine for internal conversion of DDS structures.

Synopsis

```
ddspoly = dds2dds('SS2POLY',ddsss)
ddsss = dds2dds('POLY2SS',ddspoly)
ddsd = dds2dds('CS2DS',ddsc,T)
```

Description

`ddspoly = dds2dds('SS2POLY',ddsss)` converts a DDS structure `ddsss` containing a state space realization into a new structure `ddspoly` that contains a full-polynomial model structure instead.
`ddsss = dds2dds('POLY2SS',ddspoly)` makes the opposite conversion.

`ddsd = dds2dds('CS2DS',ddsc,T)` converts a DDS structure `ddsc` containing a continuous-time state space realization into a new structure `ddsd` that contains a discrete-time structure. The conversion scheme is based on the zero order hold equivalence technique.

See Also

`dds2poly, dds2ss, ddsstruc`

dds2frf

Purpose

Frequency response function and noise spectral densities.

Synopsis

```
[G,NSP] = dds2frf(dds,f)
```

Description

G is returned as the frequency response function estimate, and NSP (if specified) as the noise spectrum, corresponding to the dds model. If the system is free of noise, NSP is returned as the empty matrix. If dds describes a time series model, G is returned as its spectrum and NSP will returned as the empty matrix. Both discrete and continuous time models are handled.

G is stored as G=[G(0) G(1) .. G(N)] and NSP likewise. Use submat to extract individual matrices or mat2vec to extract individual channels. The routine dds2spec uses this routine to plot the frequency response function and spectral densities.

Example

Given a time series model represented by a dds model structure. Let's make a semilogarithmical plot of the auto spectral density of the first channel:

```
        N=512;
        T=dds2t(dds);

        %Natural frequency 0 to Nyquist
        f=0.5*[0:N]/N/T;

        G = dds2frf(dds,f);
        G11 = mat2vec(G,1,1);
        semilogy(f,abs(G11))
```

If we want to investigate the DC-term this can be obtained as

```
        DC = submat(G,1)
```

See Also

dds2spec, submat, mat2vec

dds2id

Purpose

System identification information from a model.

Synopsis

```
[V,FPE,AIC,N,T] = dds2id(dds)
```

Description

This routine returns the PEM loss function, `V`, Akaike's Final Prediction Error criterion, `FPE`, and Akaike's Information Criterion, `AIC`. These parameters can be used to evaluate the performance of the estimate. These parameters can be extracted from all identification routines even though they are not of the PEM type. Also returned are the number of samples, N, and the sampling interval `T`.

See Also

`showdds, dds2t, dim, dimss, ddsstruc`

dds2par

Purpose

Returns the parameters of a given model.

Synopsis

```
[par,lam,P] = dds2par(dds)
```

Description

If the `dds` structure contains a polynomial model, `par` consists of the parameters of the autoregressive polynomial, the external input polynomial, and the moving average polynomial. The parameters are stored by stacking all columns of the coefficient matrices on top of each other. `Lam` is the innovation covariance matrix, and `P` is the covariance matrix of the estimated parameters. `dds` is the model structure matrix.

If the `dds` structure describes a state space realization `par` consists of the adjustable parameters of the stacked column of the state matrices ***A***, ***B***, ***C***, ***D***, and ***K***.

Example

Consider a `dds` structure defined by a two-channel autoregressive matrix polynomial `A`, and a two-channel moving average matrix polynomial `C`, as

```
A=[eye(2) [1 3;-5 2] [5 8;4 2]]
C=[eye(2) [5 -6;1 -2]]
```

by creating the `dds` structure the parameters can be extracted,

```
dds = poly2dds(A,[],C);
par = dds2par(dds)

par =
      1
     -5
      3
      2
      5
      1
     -6
     -2
```

See Also

`ddsstruc, dim, dimss, dds2ss, dds2poly`

dds2pe

Purpose

Compute or plot the prediction errors of an estimated model.

Synopsis

```
dds2pe(dds,z)
e = dds2pe(dds,z)
```

Description

`dds2pe(dds,z)` plots the prediction errors of the model structure `dds`. `z` is a matrix of output and input data on which the calculations are based. If the `dds` structure describes a time series model then `z=y`, where `y` is the output data. If the `dds` structure describes a model having external input then `z=[y u]`, where u is the external input exciting the model. The routine plots all power spectral densities between zero and the Nyquist frequency of the prediction errors. It also plots all correlation functions up to time lag 20, together with the 95% confidence interval for being a realization of independent distributed stochastic variables, [1].

`e = dds2pe(dds,z)` returns the calculated prediction errors in `e`. In this case the routine will not plot the results.

See Also

`ddsstruc, ddssim, dds2pred`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

dds2poly

Purpose

Compute the matrix polynomials associated with a given model.

Synopsis

```
[A,B,C,LAM] = dds2poly(dds)
```

Description

If the `dds` structure contains a polynomial model, `A`, `B`, and `C` are returned as the autoregressive, external input, and moving average matrix polynomials, respectively. Each of these polynomials are stored as a row of matrix coefficients. In order to extract individual matrix coefficients, use `submat`. If `nk` denotes the number of delays between external input and output, the external input matrix polynomial will have `nk` leading matrices containing zeros only. `lam` is the innovation covariance matrix.

If the `dds` structure describes a state space system, the system matrices are converted according to the approach derived in [1]. In the discrete-time case all ending zero matrices are removed, i.e. matrices of the highest power in the backward shift operator. In the continuous-time case all leading zero matrices are removed, i.e. the matrices in the highest power of the differential operator. `lam` is the innovation covariance matrix.

See Also

```
ddsstruc, dim, dds2ss, dds2par
```

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

dds2pred

Purpose

Compute or plot the one-step ahead prediction of an estimated model.

Synopsis

```
dds2pred(dds,z)
yhat = dds2pred(dds,z)
```

Description

`dds2pred(dds,z)` plots the measured response given in `z` and predicted response of the model structure `dds` [1,2]. The FFT-based spectral densities of both measured and predicted response are also plotted. `z` is a matrix of output and input data on which the calculations are based. If the `dds` structure describes a time series model then `z=y`, where `y` is the output data. If the `dds` structure describes a model having external input then `z=[y u]`, where u is the external input exciting the model. The routine plots the measured and predicted response. Also, it plots all power spectral densities between zero and the Nyquist frequency of both measured and predicted response. In all cases the measured response is plotted behind the predicted.

`yhat=dds2pred(dds,z)` returns the predicted response in `yhat`. In this case the routine will not plot the results.

See Also

`ddsstruc, ddssim, dds2pe`

Reference

[1] L. Ljung: *System Identification - Theory for the User*. Prentice-Hall, Englewood Cliffs, 1987.

[2] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

Purpose

Plot frequency response function and noise spectral densities.

Synopsis

```
dds2spec(option,dds)
dds2spec(option,dds,f)
```

Description

If option = 'PSD' all absolute values of the frequency response functions of the system contained in the `dds` structure matrix are plotted. If the system also includes a noise term its auto and cross spectral densities are also plotted. If `dds` describes a time series model all absolute values of the auto and cross spectral densities of the response of this model are plotted, [1].

If option = 'SVD' all singular values of the frequency response functions of the system contained in the `dds` structure matrix are plotted. If the system also includes a noise term, the singular values of its spectral densities are also plotted. If `dds` describes a time series model, the singular values of the spectral densities of the response of this model are plotted, [1].

The input argument `f` should be a vector of frequencies to be plotted. This is especially applicable for the continuos-time models. By default 256 frequency points between 0 and the Nyquist frequency are plotted. The Nyquist frequency is determined on the basis of the absolute value of sampling interval contained in the `dds` structure.

See Also

`ddsstruc, dds2frf`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

dds2ss

Purpose

Convert the DDS structure to an innovation state space realization.

Synopsis

```
[A,B,C,D,K,X0,lam] = dds2ss(dds)
[A,B,C,D,K,X0,lam] = dds2ss(dds,dobal)
```

Description

Converts the dds model structure matrix into the following innovation state space system

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) + Ke(t_k)$$

$$y(t_k) = Cx(t_k) + Du(t_k) + e(t_k)$$

where A is the state space matrix, B is the external input matrix, K is the Kalman gain matrix, C is the observation matrix, and D is the auxiliary observation matrix. X0 is default initial values of the state vector, and lam is the innovation covariance matrix. By default the returned realization is of the Observability canonical form, [2]. However, by setting the boolean dobal to 1 (true), the returned realization will be balanced according to [3]. By default dobal is 0 (false).

Example

Given a time series model represented by a dds model structure, and given some measurements y; it is very easy to convert to innovation state space form, in order to obtain the predicted response yhat and the prediction errors e:

```
[A,B,C,D,K,X0] = dds2ss(dds);

% B and D will be empty because
% dds describes a time series model

Xhat = ltitr(A-K*C,K,y,X0);
yhat = (C*Xhat')';
e = y-yhat;
```

See Also

ddsstruc, dds2par, dds2pred, dds2pe

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

[2] T. McKelvey: *SSID - A MATLAB Toolbox for Multivariable State-Space model Identification*. Proc. SYSID'94, Copenhagen, 1994.

dds2t

Purpose

Return the sampling interval on which a model is based.

Synopsis

```
T = dds2t(dds)
```

Description

`T` is the sampling interval of the model `dds`.

See Also

`ddsstruc, t2dds`

ddssim

Purpose

Simulate the response of a model.

Synopsis

```
y = ddssim(dds,z)
[y,e] = ddssim(dds,z)
```

Description

Simulate the response y of an input-output model

$$A(q)y(t_k) = B(q)u(t_k) + C(q)e(t_k)\ , \qquad e(t_k) \in NID(\mathbf{0},\Lambda)$$

or an innovation state space realization

$$\begin{aligned} x(t_{k+1}) &= Ax(t_k) + Bu(t_k) + Ke(t_k)\ , \qquad e(t_k) \in NID(\mathbf{0},\Lambda) \\ y(t_k) &= Cx(t_k) + Du(t_k) + e(t_k) \end{aligned}$$

described by the dds structure matrix. If dds describes a time series model, i.e. ***B*** is not present in any of the above model descriptions, z is equal to a sequence of independent Gaussian distributed random numbers, i.e. z=e. If the model also contains the external input term, z also contains the external (or deterministic) input u, i.e. z=[u e].

The random numbers in a are correlated and scaled in order to have a covariance matrix equal to Λ. The innovation sequence obtained from this operation is on output given in e.

Example

Given a time series model represented by a dds model structure, let's simulate a sequence of 1000 samples. Assume that the model has four outputs. In this case the simulated response ysim and the innovations e can be obtained as

```
z = randn(1000,4);
[ysim,e] = ddssim(dds,z);
```

See Also

ddsstruc, dds2pred

ddsstruc

Purpose

Description of the DDS structure matrix.

Synopsis

```
help ddsstruc
```

Description

The DDS (Data Dependent System) structure matrix stores information about a given model, derived from the polynomial model

$$A(q)y(t_k) = B(q)u(t_k) + C(q)e(t_k)$$

or the innovation state space model structure

$$\begin{aligned} x(t_{k+1}) &= Ax(t_k) + Bu(t_k) + Ke(t_k) \\ y(t_k) &= Cx(t_k) + Du(t_k) + e(t_k) \end{aligned}$$

It stores the following general information:

Number of output channels.
Innovation covariance matrix.
Loss function.
Akaike's Final Prediction Error (FPE).
Sampling interval.
Number of measurements used for the estimation.
An identifier for the routine that generated it.
The time it was generated.
Model parameters.
Estimated covariance matrix of the model parameters.

In the case of the polynomial model the DDS structure is organized in the following way.

Row *1*, **Column** *1 to 17:*

na: Auto regressive polynomial order
nb: External input polynomial order
nc: Moving average polynomial order
nk: Number of delays between external input and output
ny: Number of output channels
T: Sampling period (Default *T=1*)
type: Type identifier of identification routine
fpe: Akaike's FPE criterion
N: Number of measurements
date: (*year, month, day*) of creation
time: (*hour, minute*) of creation
V: Loss function that corresponds to the estimated parameters
nu: Number of input channels
nocovar: Boolean. 1 -No covariance matrix of the parameters is stored.
0 - Estimated covariance matrix of the parameters is stored.

Row *2*, **Column** *1 to* ny^2*:*

lam: Innovation covariance matrix (Default the zero matrix)

Row *2*, **Column** ny^2+1 *to* $2ny^2$

scl: Output scaling matrix (Default the identity matrix)

Row *3*, **Column** *1 to* $(na+nc)*ny^2+nb*nu*ny$*:*

par: Parameters of the estimated autoregressive and moving average polynomials.

Row *4 to* $3+(na+nc)*ny^2+nb*nu*ny$*:*, **Column** *1 to* $(na+nc)*ny^2+nb*nu*ny$*:*

(if nocovar=0)
P: Covariance matrix of estimated parameters (Default zero matrix)

In the case of the state space description the DDS structure is organized in the following way.

Row *1*, **Column** *1 to 16:*

na: State space dimension, i.e. $dim(\boldsymbol{A})$
nb: Number of columns of the external input matrix $\boldsymbol{B}$
nc: Number of rows of the observation matrix $\boldsymbol{C}$
nk: Number of columns of the Kalman gain matrix $\boldsymbol{K}$
ny: Number of output channels $(ny=nc)$
T: Sampling period (Default $T=1$)
type: Type identifier of identification routine
fpe: Akaike's FPE criterion
N: Number of measurements
date: (*year, month, day*) of creation
time: (*hour, minute*) of creation
V: Loss function that corresponds to the estimated parameters
nocovar: Boolean. 1 - No covariance matrix of the parameters is stored.
0 - Estimated covariance matrix of the parameters is stored.

Row *2*, **Column** *1 to* ny^2*:*

lam: Innovation covariance matrix (Default the zero matrix)

Row *2*, **Column** ny^2+1 *to* $2ny^2$

scl: Output scaling matrix (Default the identity matrix)

Row *2*, **Column** $2ny^2$ to $3ny^2$

$\boldsymbol{x}_0$: Initial state vector (Default a vector of zeroes)

Row *3* **Column** *1 to* $(na^2+na{\cdot}nb+nc{\cdot}na+nc{\cdot}nb+na{\cdot}nk)$*:*

par: Parameters of the state space system, stored as $par = [\boldsymbol{A}(:)\ \boldsymbol{B}(:)\ \boldsymbol{C}(:)\ \boldsymbol{D}(:)\ \boldsymbol{K}(:)]^T$

Row *4* **Column** *1*:

Nest: Number of estimated parameters in the model

Row *4* **Column** *2 to Nest*+1:

Indx. Index of estimated parameters of par

Row *5* **Column** *1 to Nest*: (if nocovar=0)

P: Covariance of estimated parameters

dim

Purpose

Extract the dimensions of the matrix polynomials describing the DDS structure.

Synopsis

```
n = dim(dds)
[na,nb,nc,nk,ny,nu] = dim(dds)
```

Description

The dimensions of the polynomial model form of the DDS structure

$$A(q)y(t_k) = B(q)u(t_k - nk) + C(q)e(t_k)$$

are extracted.

na, nb and nc is the autoregressive, external input, and moving average order, respectively. ny is the number of output channels, and nu is the number of input channels. nk is the number of delays between the external input and the system response. If only one output arguments is specified then n is returned as na times ny (the state space dimension).

See Also

ddsstruc, dimss, dds2meas

dimss

Purpose

Extract the dimensions of the state space system that describes the DDS structure.

Synopsis

```
[na,nb,nc,nk] = dimss(dds)
```

Description

The dimensions of the innovation state space system

$$\begin{aligned} x(t_{k+1}) &= \boldsymbol{A}x(t_k) + \boldsymbol{B}u(t_k) + \boldsymbol{K}e(t_k) \\ y(t_k) &= \boldsymbol{C}x(t_k) + \boldsymbol{D}u(t_k) + e(t_k) \end{aligned}$$

are extracted.

`na` is the dimension of the square state matrix $\boldsymbol{A}$. `nb` is the number of columns of the external input matrix $\boldsymbol{B}$. `nc` is the number of rows (or output) of the observation matrix $\boldsymbol{C}$, and `nk` is the number of columns of the Kalman gain matrix $\boldsymbol{K}$.

The dimensions of each of the matrices describing the system are the:

$\dim(\boldsymbol{A}) = na \times na$
$\dim(\boldsymbol{B}) = na \times nb$
$\dim(\boldsymbol{C}) = nc \times na$
$\dim(\boldsymbol{D}) = nc \times nb$
$\dim(\boldsymbol{K}) = na \times nk$

See Also

`ddsstruc, dim, dds2meas`

Purpose

Correlation and covariance function estimate for a data matrix.

Synopsis

```
C = fftcor(y,n)
G = fftcov(y,n)
```

Description

`C = fftcor(y,n)` estimates the one-sided correlation functions from lag 0 to lag `n-1`. On output, `C` is the correlation matrices for each lag are stored in row starting with lag 0.
`G = fftcov(y,n)` estimates the one-sided covariance functions from lag 0 to lag `n-1`. On output, `G` is the covariance matrices for each lag are stored in row starting with lag 0.

In both cases y is a matrix storing the data to be processed. The channels are stored column-wise. `n` is the number of returned submatrices in `C` and `G`. Use `submat` to extract individual correlation or covariance matrices, and use `mat2vec` to extract a specific element from each matrix.

Algorithm

The routines calculates an unbiased estimate of the auto and cross correlation functions using FFT combined with a Bartlett triangular lag window. An unbiased estimate is achieved by extending each data segment by padding zeroes at the ends up to the double size. This procedure ensures that the estimate is only biased by the window. Because of this the bias can be removed by multiplication by the inverse window.

The covariance function is obtained on the basis of the correlation function which is scaled using a zero lag covariance estimate obtained by using the routine `cov`.

See Also

`fftspec, cov, submat, mat2vec`

fftspec

Purpose

Power spectrum estimation of a data matrix.

Synopsis

```
P = fftspec(y)
[P,f] = fftspec(y,nfft,noverlap,T)
```

Description

Estimate the Power Spectral Density of signals stored columnwise in `y` using Welch's averaged periodogram method. The signals of `y` are divided into overlapping sections, each of which is detrended and windowed, then zero padded to length `nfft`. The overlap is specified by `noverlap`.

`P` is returned as the multivariate spectral densities. Each frequency is described by a square spectral density matrix. These matrices are stored in a row starting with the DC term. Use `submat` to extract individual spectral density matrices and `mat2vec` to extract a specific element from each spectral density matrix.

Given a sampling period `T` the vector `f` is returned as the frequencies at which the PSD is estimated. If `T` is not specified the frequencies lie between 0 and 1.

The default values for the parameters are `nfft = 256` (or size(y,1) whichever is smaller), `noverlap = 0`, `T = 0.5` (i.e. Nyquist frequency is 1)

If no output arguments are specified, the routine plots the auto spectral densities and the absolute values of the cross-spectral densities.

See Also

`fftcor, fftcov, submat, mat2vec`

initpem

Purpose

Generate initial information for `sspem`.

Synopsis

```
ddsinit = initpem(dds)
ddsinit = initpem(dds,operation,elements)
```

Description

The structure matrix `dds` of the standard format contains the initial information to use. The DDS structure is on output returned in `ddsinit` together with the information about which parameters are adjustable. `operation` is a string that controls how the routine creates `ddsinit`. This string
has the following options:

`operation = MATRIX`: A manual option for selecting adjustable parameters with `MATRIX` indicating which of the system matrices to manipulate (one of `'A','B','C','D'` or `'K'`). When using this option `elements` is an n x 2 matrix that contains the row and column numbers of the elements to be marked adjustable. If the argument `elements` is omitted all the elements of the system matrix are marked adjustable.

`operation = 'auto'`: Using this option the routine will return `ddsinit` as a transformed observability canonical state space realization and automatically generate the index of adjustable parameters in the model. NOTE that this option is the default one and that it must be used if the DDS structure contains a polynomial model.

`operation = 'unlock'`: This option unlocks all parameters. In other words, it makes all parameters adjustable.

`operation = 'lock'`: This option locks all parameters. In other words, it makes all parameters non-adjustable.

It should be noted that any covariance matrix of prior adjustable parameters in the DDS structure is deleted using all options except `operation = 'unlock'`.

See Also

`sspem, ddsstruc`

initstdi

Purpose

Configuration and initialization of default parameters.

Synopsis

```
initstdi
```

Description

`initstdi` edits the default parameters stored in the `stdi.mat` configuration file. Should this file get lost the routine can also be used to reconstruct it.

The routine is menu driven. Currently values as well as default values of the editable parameters are presented. The following options are available

```
1. Retrieve Default Configuration
2. Edit Plot and Axis Font Sizes
3. Edit PEM Optimization Parameters
4. Edit Sampling Interval
H. Help
S. Save and Quit
Q. Quit
```

The default configuration (1) sets all parameters and font sizes to defaults values. The font sizes can be edited by the option (2). The algorithms implementing the prediction error method for different model structures also need to be configured which is performed by the option (3). If a particular sampling interval is used frequently this can be set by the option (4). Help is provided by the option (H) and the configuration is saved using option (S). After saving the routine quits. This can also be accomplished without saving using option (Q).

See Also

```
auxlsvar
```

isdds

Purpose

Check if a matrix is a DDS structure.

Synopsis

```
B = isdds( A )
```

Description

`B` is a boolean. If `A` is a DDS structure then `B=1`, otherwise `B=0`.

isddscs

Purpose

Check if a DDS structure contains a continuous-time system.

Synopsis

```
B = isddscs(dds)
```

Description

`B` is a boolean. If `dds` contains a continuous-time state space description then `B=1`, otherwise `B=0`.

isddsss

Purpose

Check if a DDS structure contains a state space realization

Synopsis

```
B = isddsss(dds)
```

Description

`B` is a boolean. If `dds` contains a state space description then `B=1`, otherwise `B=0`.

mat2vec

Purpose

Extract a specific element from all submatrices stored in a row.

Synopsis

```
y = mat2vec(x,r,c)
y = mat2vec(x,r,c,n)
```

Description

`x` is a row vector of square matrices. This routine extracts the `r,c` element of each of the submatrices, where `r` is the row index and `c` the column index. All extracted elements are stored in the row vector `y`. If the submatrices are not square, the number of columns of each submatrix is specified in `n`.

Example

Consider the following row matrix A consisting of three square submatrices

```
A = [ [1 2;5 6] [5 8;10 5] [0 7;1,6] ]

A =
       1     2     5     8     0     7
       5     6    10     5     1     6
```

Extract element 1,2 of each submatrix and store all extracted elements in `B`

```
B = mat2vec(A,1,2)

B =
       2     8     7
```

See Also

`submat, dds2spec`

Purpose

Convert from a continuous-time second order differential system into a DDS structure.

Synopsis

```
dds = mck2dds(M,C,K)
dds = mck2dds(M,C,K,S,OUTPUT)
```

Description

Convert the second order linear time-invariant continuous-time system described the differential equation

$$\boldsymbol{M}\ddot{\boldsymbol{y}}(t) + \boldsymbol{C}\dot{\boldsymbol{y}}(t) + \boldsymbol{K}\boldsymbol{y}(t) = \boldsymbol{S}\boldsymbol{u}(t)$$

where `M`, `C` and `K` are the mass, damping and stiffness matrices, respectively, into a continuous-time state space realization stored in the `dds` structure matrix. `S` is a selection or resolution matrix, which by default is set to the identity matrix.

The `OUTPUT` argument is a character that controls what type of output the state space system gives. If `OUTPUT='d'` the output is displacements, if `OUT-PUT='v'` the output is velocities, and if `OUTPUT='a'` the output is accelerations.

See Also

`armav21, ss2dds`

modal

Purpose

Modal decomposition and modal parameter extraction of a model.

Synopsis

```
[psi,mu] = modal(dds,'MDCMP')

[phi,freq,zeta] = modal(dds)
[phi,freq,zeta] = modal(dds,norm)
```

Description

Depending on the input arguments, this routine returns either a modal decomposition or modal parameters of a model stored in a DDS structure.

`[psi,mu] = modal(dds,'MDCMP')` performs a modal decomposition of the model stored in `dds`. On output `psi` is the complex modal matrix. The columns of this matrix are the eigenvectors of the state space matrix. The corresponding eigenvalues are returned in mu, [1,2].

`[phi,freq,zeta] = modal(dds)` returns the modal parameters of the model stored in `dds`. The scaled mode shapes are returned as the columns of the matrix `phi`. By default, the normalization of the mode shapes is determined by the routine `eig`. However, by supplying an extra boolean argument `norm` the normalization can be performed with respect to the bottom row of `phi`. In this case `norm` is set 1 (true). By default `norm` is 0 (false).

`freq` is a vector of the corresponding natural eigenfrequencies scaled with respect to the sampling interval of the model. `zeta` is a vector of corresponding damping ratios.

All modal parameters are sorted so that the natural eigenfrequencies are stored in ascending order.

See Also

`ddsstruc, sdmodal`

Reference

[1] Pandit, S.M.: *Modal and Spectrum Analysis: Data Dependent Systems in State Space*. pp. 231-234, John Wiley & Sons, 1991.

[2] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

mpcheck

Purpose

Check the consistency of a matrix polynomial.

Synopsis

```
msg = mpcheck(pol)
msg = mpcheck(pol,ismonic,ncolsub)
```

Description

`pol` is a matrix polynomial stored as a row of square matrices. `msg = mpcheck(pol)` checks the consistency of `pol`. `pol` must not have fewer columns than rows. The number of columns divided with the number of rows must equal an integer. By default the routine assumes and check that the matrix polynomial is monic, i.e. has a leading identity matrix. The checking of this can be disabled by setting the boolean `ismonic` to zero. The routine also by default assumes that the number of columns in each matrix coefficient is equal to its number of rows. This can be changed by specification of `ncolsub`.

See Also

`mpfilt, mpconv, mpdeconv`

mpconv, mpdeconv

Purpose

Multivariate convolution and deconvolution, or matrix polynomial multiplication and division.

Synopsis

```
c = mpconv(a,b)
[q,r] = mpdeconv(b,a)
```

Description

a and b are row vectors of square matrices. mpconv(a,b) convolves a and b. The matrix convolution sum is

$$c(n+1) = \sum_{k=0}^{N-1} a(k+1)b(n-k)$$

where N is the maximum sequence length. [q,r] = mpdeconv(b,a) deconvolves a out of b, using long division. The result (quotient) is returned in matrix q and the remainder in matrix r so that b = conv(q,a) + r. If a and b are row vectors of matrix polynomial coefficients, then to convolve them is equivalent to multiplying the two polynomials, and deconvolution is polynomial division. The result of dividing b by a is quotient q and remainder r.

Example

Consider two monic 2-channel matrix polynomials a and b, defined as

```
a=[eye(2) [1 3;-5 2] [5 8;4 2]];
b=[eye(2) [5 -6;1 -2]];
```

the convolution (or polynomial multiplication) is

```
c=mpconv(a,b)

c =
     1     0     6    -3    40    11     1    28
     0     1    -4     0    15     1    -3     4
```

Use deconvolution to divide a back out again

```
[q,r]=mpdeconv(c,a)

q =
     1     0     5    -6
     0     1     1    -2
```

```
r =
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

mpfilt

Purpose

Filtering of multivariate linear systems.

Synopsis

```
y = mpfilt(b,a,x)
[y,zf] = mpfilt(b,a,x,zi)
```

Description

`mpfilt` filters data using a multivariate digital filter based on state space filtering. The filter handles both multivariate FIR and multivariate IIR filters. Access to initial and final conditions is available.

`y = mpfilt(b,a,x)` filters the data in the matrix `x` with the filter described by the vectors of matrix polynomial coefficients given in matrices `a` and `b` to create the matrix `y`. Each column of `x` and `y` corresponds to a channel of data. The operations performed by `mpfilt` are described in time domain by the difference equation

```
y(n,:)=submat(b,1)*x(n,:)+submat(b,2)*x(n-1,:)+...+submat(b,nb+1)*x(n-nb,:)
                            -submat(a,2)*y(n-1,:)-...-submat(a,na+1)*y(n-na,:)
```

If the first coefficient matrix of `a` is different from the identity matrix, all filter coefficients will be normalized by pre-multiplying the coefficient matrices with the inverse of the first coefficient matrix. When use is made with two left-hand arguments, `mpfilt` returns the final conditions on the states

```
        [y,zf] = mpfilt(b,a,x)
```

When use is made with an extra right-hand argument, initial conditions on the states are specified

```
        y = mpfilt(b,a,x,zi)
```

The size of the initial/final conditions must be *2 x max(na,nb)*p*, where *p* is the number of channels.

Example

Consider two monic 2-channel matrix polynomials `a` and `b`, defined as

```
        a=[eye(2) [1 3;-5 2] [5 8;4 2]],b=[eye(2) [5 -6;1 -2]]
```

and an input matrix

```
        x=[1 1;0 0;0 0]
```

Filter x using b and a as

```
        y=mpfilt(b,a,x)
```

to yield

```
        y =
             1      1
            -5      2
           -14    -35
```

See Also

`filter, ltitr, mpcheck, mpconv, mpdeconv`

mporder

Purpose

Return the order and the number of channels of a matrix polynomial.

Synopsis

```
[n,p] = mporder(a)
[n,p] = mporder(a,ncolsub)
```

Description

`a` is a matrix polynomial stored in a row of matrix polynomial coefficients. `[n,p] = mporder(a)` returns the order of the polynomial in `n` and the number of channels in `p`. By default the routine assumes that the number of columns in each matrix coefficient is equal to its number of rows. This can be changed by specification of `ncolsub`.

Example

Consider a monic 2-channel matrix polynomial `a` defined as

```
a=[eye(2) [1 3;-5 2] [5 8;4 2]]

a =
      1      0      1      3      5      8
      0      1     -5      2      4      2
```

Determine the polynomial order

```
[n,p]=mporder(a)
```

to yield

```
n =
      2

p =
      2
```

Purpose

Stochastic Subspace Identification of an innovation state space realization.

Synopsis

```
dds = na4sid(y,m)
[dds,R] = na4sid(y,m,n,T,algo,posreal,maxsize,noscale)

[dds,R] = na4sid(R)
[dds,R] = na4sid(R,n,algo,posreal)
```

Description

The parameters of the stochastic innovation state space system

$$\begin{aligned} x(t_{k+1}) &= Ax(t_k) + Ke(t_k)\,, \qquad e(t_k) \in NID(\mathbf{0},\Lambda) \\ y(t_k) &= Cx(t_k) + e(t_k) \end{aligned}$$

are estimated using stochastic subspace state space identification, [1].

`y` is a matrix with the time series data of each channel stored columnwise. `n` is the desired state space dimension, and `m` is equal to the half number of block rows in the Hankel matrix constructed from the data. `n` is the state space dimension. If not specified, it must be determined using a plot of the singular values. After the right state space dimension there should be a significant gap between the singular values.

The recommended state space dimension is indicated by a green colour. By clicking on a dimension with the left mouse button the performance of this particular realization can be evaluated. This evaluation is based on a comparison of the maximum singular values of the spectral densities of the data and the realization. The selected state space dimension will be indicated with a red colour. To select a particular dimension click on it with the right mouse button. If the one-step ahead predictor of the selected realization is unstable, this will be indicated. If this is the case, the estimated realization cannot be used as initial estimate for the Prediction Error Methods.

The string argument `algo` controls how the row space of the Hankel matrix containing future data is weighted and orthogonal projected onto the row space of the Hankel matrix containing past data.

If `algo='CVAQSVD'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Canonical Variate Algorithm, [1]. The estimation of the system matrices is based on the Quotient Singular Value Decomposition (QSVD), [2].

If `algo='CVA'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Canonical Variate Algorithm. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach, [1]. By default `algo='CVA'`.

If `algo='PC'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Principal Component algorithm, [1,3]. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach.

If `algo='UPC'` the weighting of the orthogonal projection of the row space future data onto the row space of the past data makes the algorithm correspond to the Unweighted Principal Component algorithm, [1]. The estimation of the system matrices is based on a combined QR and Singular Value Decomposition (SVD) approach.

If the boolean argument `posreal` is true (`posreal=1`) a positive real solution of the steady-state covariance matrix is enforced. This is a necessary requirement for a correct estimation of the moving average and the innovation covariance matrix. If `posreal` is false (`posreal=0`), which it is by default, a positive real solution is not guaranteed and it might result in, e.g. bad spectral density plots or simulations. A warning will be given if the solution is not positive real.

If different state space dimensions `n` will be tried for the same `y` and `m` the matrix `R` contains the necessary information needed for subsequent calls. For these subsequent calls the routine should be called in one of the following ways:

```
[dds,R] = na4sid(R)
[dds,R] = na4sid(R,n,algo,posreal)
```

Again if `n` is not specified, it must be determined the singular values plot. Finally, please note that `m` and `n` must satisfy the inequality: `m` >= `np`, where `p` is the number of columns of `y`. In practice for noisy data `m` should be much larger than `np`.

The optional auxiliary arguments `maxsize`, `T` and `noscale` are explained under `auxlsvar`.

Algorithm

On the basis of the user supplied information the routine estimates a stochastic state space realization using the algorithms described in [1,2].

See Also

`armavnn, sspem, ssmbh, armav, armavgls, auxlsvar, ddsstruc`

Reference

[1] Van Overschee, P. & B. De Moor: *Subspace Identification for Linear Systems. Theory, Implementation, Applications.* Kluwer Academic Publishers, ISBN 0-7923-9717-7, 1996.

[2] Van Overschee, P. & B. De Moor: *Subspace Algorithms for the Stochastic Identification Problem.* Automatica, Vol. 29, No. 3, pp. 649-660, 1993.

[3] Aoki, M.: *State Space Modeling of Time Series.* 2nd Ed., Springer Verlag, 1990.

par2dds

Purpose

Create a DDS structure from parameters of a polynomial model.

Synopsis

```
dds = par2dds(par,na,nb,nc,nk,ny)
dds = par2dds(par,na,nb,nc,nk,ny,nu,lam,P)
```

Description

If the model is of a polynomial form

$$A(q)y(t_k) = B(q)u(t_k) + C(q)e(t_k)$$

then `par` is a vector of stacked columns of the polynomial coefficients of the autoregressive polynomial $A(q)$, the external input polynomial $B(q)$, and the moving average polynomial $C(q)$. `na` is the autoregressive polynomial order, `nb` is the external input polynomial order, and `nc` is the moving average polynomial order. `nk` is the number of delays between external input and output. `ny` is the number of outputs, and `nu` is the number of inputs. `lam` is the innovation covariance matrix and `P` is the covariance matrix that corresponds to `par`. If the model is a state space realization, use `ss2dds` instead.

See Also

`ddsstruc, dds2par, ss2dds`

poly2dds

Purpose

Create a DDS structure from matrix polynomials.

Synopsis

```
dds = poly2dds(A)
dds = poly2dds(A,B,C,nu,nk,lam,P)
```

Description

`A` is an autoregressive matrix polynomial, `B` is an external input matrix polynomial, and `C` is a moving average matrix polynomial. `nu` is number of input channels. This should only be specified if the number of input is different from the number of output. `nk` is the number of delays between the external input and the output. `lam` is the innovation covariance matrix. `P` is the covariance matrix of the estimated parameters stored in the format that corresponds to the parameter vector returned from `dds2par`. The routine returns as `dds` the DDS structure matrix.

See Also

`ddsstruc, dds2par, dds2poly`

sdmodal

Purpose

Estimated uncertainties of modal parameters.

Synopsis

```
[sdphi,sdfreq,sdzeta] = sdmodal(dds)
[sdphi,sdfreq,sdzeta,Pphi,Pfreq,Pzeta] = sdmodal(dds)
```

Description

The standard deviations and covariance matrices of the modal parameters of a discrete-time model are estimated using numerical differentiation.

The standard deviations of the scaled mode shapes are returned as the matrix `sdphi` and organized just as the mode shapes returned by modal. In other words, each column contains the standard deviations of a scaled mode shape. For numerical stability the standard deviations are returned for mode shapes that are normalized with respect to their bottom element.

`sdfreq` is a vector of standard deviations of the corresponding natural eigenfrequencies. `sdzeta` is a vector of standard deviations of the corresponding damping ratios.

Supplied with the extra output arguments, sdmodal returns `Pphi` as the full covariance matrix of the scaled mode shapes, `Pfreq` as the full covariance matrix of the natural eigenfrequencies, and `Pzeta` as the full covariance matrix of the damping ratios.

Algorithm

The covariance matrices of the modal parameters are obtained by a linear transformation of the covariance matrix of the estimated parameters, [1]. This linear transformation corresponds to a Taylor expansion of the mean values of the modal parameters. This transformation involves the construction of a transformation matrix. This matrix is constructed using numerical differentiation. Having obtained the covariance matrices of the modal parameters, the standard deviations are calculated as the square roots of the diagonal elements.

See Also

`ddsstruc, modal`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

showdds

Purpose

Display the information in the DDS structure matrix on the screen.

Synopsis

```
showdds(dds)
```

Description

If `dds` contains a input-output model, this routine displays the nonempty matrix polynomials of the model `dds`, together with their standard deviations given as fake imaginary parts.

If instead the `dds` structure contains a state space description, the nonempty state space matrices are displayed.

Also displayed are:

Type of model
Number of channels
Name of routine that created the structure
Creation time
Number of samples used for the creation
Sampling interval
Loss function
Akaike's Final Prediction Error Criterion (FPE)
Innovation covariance matrix.
Scaling matrix of the output measurements.

See Also

`ddsstruc`

speccmp

Purpose

Compare spectral densities of one or several DDS structures with response data.

Synopsis

```
speccmp(dds,y)
speccmp(project,n)
speccmp(project,n,y)
```

Description

`speccmp(dds,y)` compares the absolute values of the auto and cross spectral densities of the DDS structure matrix `dds`, and response data `y`, where each column of `y` is a measurement channel. The auto and cross spectral densities of `y` are calculated using FFT.

`speccmp(project,n)` compares several DDS structures with each other. The DDS structures to be compared are saved under the variable name `dds` in *.MAT files. The names of the files are concatenations of the string project and an integer starting from 1 and going to `n`, as:

```
for i=1:n
        file = [project,int2str(i)]
end
```

`speccmp(project,n,y)` compares the DDS structures with response data in `y`.

This routine only works for discrete time series models.

See Also

`ddsstruc, dds2spec, spectrum`

ss2dds

Purpose

Create a DDS structure of an innovation state space realization.

Synopsis

```
dds = ss2dds(A,B,C)
dds = ss2dds(A,B,C,D,K,X0,T,lam,P,indx)
```

Description

Convert from an innovation state space system of the type

$$\begin{aligned} \boldsymbol{x}(t_{k+1}) &= \boldsymbol{A}\boldsymbol{x}(t_k) + \boldsymbol{B}\boldsymbol{u}(t_k) + \boldsymbol{K}\boldsymbol{e}(t_k)\,, \qquad \boldsymbol{e}(t_k) \in NID(\boldsymbol{0},\boldsymbol{\Lambda}) \\ \boldsymbol{y}(t_k) &= \boldsymbol{C}\boldsymbol{x}(t_k) + \boldsymbol{D}\boldsymbol{u}(t_k) + \boldsymbol{e}(t_k) \end{aligned}$$

where `A` is the state space matrix, `B` is the external input matrix, `K` is the Kalman gain matrix, `C` is the observation matrix, and `D` is the direct term matrix, into a `dds` structure matrix. Initial conditions of the state vector $\boldsymbol{x}(t_0)$ are supplied in `X0`. The innovation covariance matrix Λ is supplied in `lam`, [1].

Assume that the parameters of the realization are assembled in one vector θ by stacking all columns of $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ and $\boldsymbol{K}$ on top of each other. If a covariance matrix of the adjustable parameters exists then it can be supplied in `P`. In this case `indx` is entered as a vector of coordinates of the locations of the adjustable parameters in θ.

See Also

`ddsstruc, poly2dds, dds2ss`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

sscheck

Purpose

Check the dimensional consistency of a state space system.

Synopsis

```
msg = sscheck(A,B,C,D,K,X0)
```

Description

Check the dimensional consistency of the state space system

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) + Ke(t_k)$$

$$y(t_k) = Cx(t_k) + Du(t_k) + e(t_k)$$

If the dimensions are consistent `msg` is returned as the empty matrix, otherwise it will be returned as a string with an error message.

See Also

`dds2ss`

Purpose

Estimate a stochastic state space realization using Matrix Block Hankel factorization.

Synopsis

```
dds = ssmbh(y,n)
dds = ssmbh(y,n,T,noscale)
```

Description

The parameters of the innovation state space system

$$\begin{aligned} x(t_{k+1}) &= Ax(t_k) + Ke(t_k) \\ y(t_k) &= Cx(t_k) + e(t_k) \end{aligned}$$

are estimated using Matrix Block Hankel factorization, [1,2].

`y` is a matrix with the time series data of each channel stored columnwise. `n` is the desired state space dimension. The `dds` structure matrix is returned with the resulting parameter estimates.

The optional auxiliary arguments `T` and `noscale` are explained under `auxlsvar`.

See Also

`ddsstruc, na4sid, sspem`

Reference

[1] Aoki, M. *State Space Modeling of Time Series*. 2nd. Ed., Springer Verlag, 1990.

[2] Hoen, C.: *System Identification of Structures Excited by Stochastic load Process*. Ph.D. Thesis, The University of Trondheim, Norway, 1991.

sspem

Purpose

Prediction error estimate of an innovation state space realization.

Synopsis

```
dds=sspem(y,ddsi)
dds=sspem(y,ddsi,maxiter,tol,maxsize,T,lim,nocovar,noscale,app)
```

Description

The parameters of a realization of the innovation state space system

$$\begin{aligned} x(t_{k+1}) &= \boldsymbol{A}x(t_k) + \boldsymbol{B}u(t_k) + \boldsymbol{K}e(t_k)\,, \qquad e(t_k) \in NID(\boldsymbol{0},\Lambda) \\ y(t_k) &= \boldsymbol{C}x(t_k) + \boldsymbol{D}u(t_k) + e(t_k) \end{aligned}$$

are estimated using a prediction error method, [1].

`y` is a matrix with the time series data of each channel stored columnwise. `ddsi` contains initial parameter estimates and structure information stored in a DDS structure. This structure is obtained as output from `initpem`. The criterion minimization is then initialized at `ddsi`. The `dds` structure matrix is returned with the resulting parameter estimates, together with estimated covariance matrices.

The optional argument `app` is a string that controls which search-scheme to use. If `app='GN'` a Gauss-Newton search-scheme is applied. If `app='ML'` the search-scheme will be of the Marquardt-Levenberg type. The optional auxiliary arguments `maxiter`, `tol`, `maxsize`, `T`, `nocovar` and `noscale` are explained under `auxlsvar`.

Algorithm

If `app='GN'` a robustified quadratic prediction error criterion is minimized using an iterative damped Gauss-Newton algorithm. The Gauss-Newton vector is bisected up to ten times until a lower value of the criterion function is found. The iterations are terminated when `maxiter` is reached, when the Gauss-Newton search gradient has a norm less than `tol`, or when a lower value of the criterion cannot be found.

If `app='ML'` the same robustified quadratic prediction error criterion is minimized using an iterative Marquardt-Levenberg algorithm. The iterations are terminated when `maxiter` is reached, when the Marquardt-Levenberg search gradient vector has a norm less than `tol`, or when a lower value of the criterion cannot be found.

In order to obtain an initial guess of the parameters an initialization using a two-stage linear least-squares algorithm is used. This initialization is allowed to use ten iterations before terminating.

The cutoff value for the robustification is based on the parameter `lim` as well as on the estimated standard deviation of the residuals from the current parameter estimate. The returned loss function and innovation covariance matrix are the non-robustified ones.

A stability test of the predictor is performed to assure that only models corresponding to stable predictors are tested. Generally, the moving average matrix polynomial must have all its roots inside the unit circle.

Information about the minimization is furnished to the screen. Currently estimated parameters as well as values of the current and previous criterion functions are given. The norm of either the Gauss-Newton or the Marquardt-Levenberg search gradient is also displayed. The number in the upper left corner is the number of times the length of the search vector is adjusted. If configured for it, a graphical PEM Information Window will appear and present the information as well. This is controlled by the routine `initstdi`.

If configure for it, the routine will save all information about the identification after each iteration. In this way the currently obtained results are saved if a breakdown of the system should occur or if CRTL-C is hit. This is controlled by the routine `initstdi`. If the routine is stopped before completion, the data can be restored by the routine `tmp2dds`.

See Also

`armav, na4sid, auxlsvar, ddsstruc, tmp2dds, initstdi`

Reference

[1] Andersen, P.: *Identification of Civil Engineering Structures using Vector ARMA Models*. Ph.D. thesis, Aalborg University, Denmark, ISSN 1395-7953 R9724, 1997.

submat

Purpose

Extract a submatrix from a row of matrices.

Synopsis

```
Asub = submat(A,indx)
Asub = submat(A,indx,ncol)
```

Description

`A` is a row of submatrices each with `ncol` columns. `Asub` is the submatrix number `indx` from the left of `A`. All submatrices must have the same number of columns. If the submatrices are square, i.e. the number of columns of each of the submatrices are equal to the number of rows of `A`, it is not necessary to specify `ncol`.

Example

Consider a two-channel monic matrix polynomial `phi` defined as

```
phi=[eye(2) [1 3;-5 2] [5 8;4 2]]
```

Extract the second polynomial coefficient as `Asub`

```
Asub = submat(phi,2)

Asub =
     1     3
    -5     2
```

See Also

`mat2vec, mporder, mpcheck`

Purpose

Set the sampling interval in a DDS structure.

Synopsis

```
ddsnew = t2dds(dds,T)
```

Description

All functions that create the model descriptions in the DDS format sets the sampling period `T` in their last argument. For convenience `t2dds` offers an alternative to set it directly, i.e. `ddsnew` is equal to `dds` except that `ddsnew` is updated with `T`. If `dds` describes a continuous-time state space realization, and if for some purpose it is desirable to alter the default sampling period (`T=-1`), the new sampling period must be supplied with a minus sign. This routine cannot be used for conversion between continuous-time and discrete-time systems.

See Also

`ddsstruc, dds2t`

tmp2dds

Purpose

Construct a DDS structure from a temporarily PEM backup file.

Synopsis

```
dds = tmp2dds
```

Description

If saving of temporarily backup files is enabled using `initstdi`, this routine will construct a DDS structure on the basis of the information stored in one of these files. Afterwards, the temporarily backup file will be deleted.

These particular backup files contain information about the status of the currently completed nonlinear PEM iteration of the routines `sspem` and `armav`. If a system failure or a CRTL-C keyboard hit interrupt the optimization, this file will contain the necessary information needed in order to construct a DDS structure.

If backup files exist both for `sspem` and `armav` and if the information of both files is desired, this routine has to be called twice.

See Also

`initstdi, armav, sspem`

Department of Building Technology and Structural Engineering
Aalborg University, Sohngaardsholmsvej 57, DK 9000 Aalborg
Telephone: +45 9635 8080 Telefax: +45 9814 8243