0.4pt0pt 0pt0pt

Contents

Chapter 1

User guide to SMI 2.0

1.1 List of functions in SMI 2.0

The following table gives an overview of the functions in the SMI toolbox. Some function are new to this toolbox. Others appeared in the previous version of the toolbox under an other name. These names have been changed to prevent conflicts with other toolboxes and to provide a more consistent naming. This makes the functions easier to recognize and remember.

function	**description**	**old name**
	Discrete time moesp	
dordom	Ordinary moesp preprocessor	dordom
dordpi	Ppast input moesp preprocessor	dordpi
dordpo	Past output moesp preprocessor	dordpo
dordeiv	Eiv moesp preprocessor	-
dordrs	Reconstructed state moesp preprocessor	dordrs
destac	Estimate A,C	dmodpi,dmodpo
destbd	Estimate B,D	dac2bd,destb
destk	Estimate Kalman gain	dmodpo
destx	Estimate initial state	dinit
dmoesp	Frontend for DT moesp	-
	Continuous time moesp	
cordom	Ordinary moesp preprocessor	-
cordpi	Past input moesp preprocessor	-
cordpo	Past output moesp preprocessor	-
cestac	Estimate A,C	-
cestbd	Estimate B,D	-
cestx	Estimate initial state	-
	Recursive moesp	
drpi	Recursive PI moesp	-
drpo	Recursive PO moesp	-
	SLS optimization	
dss2th	Parameterization of state space system	ss2thon
dth2ss	Reconstruction of state space system	th2sson
dslslin	Optimize DT linear model using SLS	gnlisls
dslswie	Optimize DT wiener model using SLS	gnwisls
dfunlin	Cost-function for dslslin	-
dfunwie	Cost-function for dslswie	-
drslslin	Recursive optimization of DT model using SLS	-
clslin	Optimize CT linear model using SLS	-
cfunlin	Cost-function for dslslin	-
crslslin	Recursive optimization of CT model using SLS	-
	Non causal models	
ncdlsim	Simulate non causal model	-
ncdestac	Estimate A and C for non-causal model	-
ncdestbd	Estimate B and D for non-causal model	-
kroneckf	Calculate Kronecker canonical form	-
	Nonlinear models	
chebest	Estimate MIMO nonlinear model	tchebest
chebsim	Simulate MIMO nonlinear model	tchebest
	Miscellaneous	
prbn	Pseudo random binary sequence	prbn
vaf	Variance accounted for	vaf
shave	Remove peaks and outliers	shave

1.2 Function descriptions

cestac

Purpose

Estimates the matrices A and C of a LTI state space model using the result of the preprocessor routines `cordxx` (`cordom`, `cordpo`, etc.). General model structure:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + Du(t) + v(t) \end{aligned}$$

For more information about the disturbance properties see the help pages for the preprocessor `cordxx` functions.

Syntax

```
[A,C]=cestac(R,n);
```

Inputs

`R`	Data structure obtained from `cordxx`, containing the triangular factor and additional information (such as i/o dimension etc.).
`n`	Order of system to be estimated.

Outputs

`A,C`	Estimated system matrices.

See also

`cordom, cordpi, cordpo, cestbd`

cestbd

Purpose

Estimates the matrices B and D of the state space model

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) + w(t)\\ y(t) &= Cx(t) + Du(t) + v(t)\\ x(0) &= x_0\end{aligned}$$

using the knowledge of the pair A, C. This function can concatenate different input-output data batches, through the matrix R and Rold). B and D and x_0 are calculated by solving a linear least squares problem.

Syntax

```
[B,D,x0,R]=cestbd(u,y,A,C,[fB fD fx],Rold);
[B,D]=cestbd(u,y,A,C);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`A,C`	The estimated system matrices A and C.
`model`	Three element flag vector $\begin{bmatrix} fB & fD & fx \end{bmatrix}$ indicating whether B, D and x_0 should be estimated. The default value is $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$. The matrix B or D can be assumed zero by setting fB or fD to zero. The calculation of x_0 can be omitted by setting fx to zero. However x_0 will not be assumed zero then. It's influence will still be taken into account for the computation of B and D.
`Rold`	R matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments.

Outputs

`B,D`	The estimated system matrices B and D.
`x0`	The estimated initial state of the system.
`R`	Compressed data matrix, storing information on the calculation of the matrices B and D in following `cestbd`. Used when analyzing multiple input-output data sequences.

See also

`cordxx, cestac, cestx`

cestx

Purpose

Estimate the initial state, given the estimated system matrices and a set of input/output data.

Syntax

```
x0=cestx(u,y,Ts,A,B,C,D);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`Ts`	Sampling period of the measured data.
`A,B,C,D`	System matrices.

Outputs

`x0`	Estimated initial state.

See also

`cestbd`

chebest

Purpose

This function estimates a MIMO static nonlinear function between the signals y and z. The function is estimated on the basis of Chebychev polynomials. Before estimating the coefficients of the polynomials the input signal is shifted and scaled to fall within the region [-1,1]. The shifting and scaling factors are included in the parameter vector.

Syntax

```
[thl,ze,Phi]=chebest(y,z,nn);
```

Inputs

`y,z`	Input and output of the nonlinearity.
`nn`	Order of the Chebychev polynomials in the nonlinear function.

Outputs

`thl`	Vector with the parameters of the static nonlinearity.
`ze`	Estimated output, on basis of the model that is obtained.
`Phi`	matrix with the Chebychev functions of y, such that ze = Phi $\times$ thl.

See also

`chebsim, dslswie`

chebsim

Purpose

Simulates a static nonlinear function on the basis of Chebychev polynomials with input y. The coefficients of the Chebychev polynomials are given with the vector thl, and commonly estimated with either `dslswie` or `chebest`.

Syntax

```
[ze,Phi]=chebsim(y,thl)
```

Inputs

`y`	The input to the nonlinearity.
`thl`	Parameter matrix, with coefficients of the nonlinear function.

Outputs

`ze`	Estimated output.
`Phi`	matrix with the Chebychev functions of y, such that ze = Phi $\times$ thl.

See also

`chebest, dslswie`

cordom

Purpose

This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function `cestac` to extract the matrices A and C. Model structure:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

This function does not use an instrumental variable and is comparable to `dordom` for the discrete time case.

Syntax

```
[Sn,R]=cordpo(u,y,t,a,i);
[Sn,R]=cordpo(u,y,t,a,i,Rold);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`t`	Sampling time or time-vector for the sampled data.
`a`	Filter coefficient of the Laguerre filters that are used in the algorithm. When a is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `cordom` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `cestac` for the estimation of A and C or by a next call to `cordom`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`cordpi, cordpo, cestac, cestbd.`

cordpi

Purpose

This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function `cestac` to extract the matrices A and C. Model structure:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Du(t) + v(t)\end{aligned}$$

where $v(t)$ is a finite variance disturbance on the output and is independent of the noise-free input $u(t)$. This function uses the past input instrumental variable and is comparable to `dordpi` for the discrete time case.

Syntax

```
[Sn,R]=cordpi(u,y,t,a,i);
[Sn,R]=cordpi(u,y,t,a,i,Rold);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`t`	Sampling time or time-vector for the sampled data.
`a`	Filter coefficient of the Laguerre filters that are used in the algorithm. When a is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `cordpi` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `cestac` for the estimation of A and C or by a next call to `cordpi`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`cordom`, `cordpo`, `cestac`, `cestbd`.

cordpo

Purpose

This function is a preprocessor function that extracts the column-space of the extended observability matrix of a continuous time LTI system from input/output data. The estimated column-space is used in the function `cestac` to extract the matrices A and C. Model structure:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + Du(t) + v(t) \end{aligned}$$

where $w(t)$ is a Wiener noise signal and $v(t)$ a finite variance disturbance on the output, independent of the noise-free input $u(t)$.

This function uses the past output instrumental variable and is comparable to `dordpo` for the discrete time case.

Syntax

```
[Sn,R]=cordpo(u,y,t,a,i);
[Sn,R]=cordpo(u,y,t,a,i,Rold);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`t`	Sampling time or time-vector for the sampled data.
`a`	Filter coefficient of the Laguerre filters that are used in the algorithm. When a is negative, the anti-causal instrumental variable method is used. Otherwise the causal instrumental variable method is chosen.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `cordpo` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `cestac` for the estimation of A and C or by a next call to `cordpo`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`cordom`, `cordpi`, `cestac`, `cestbd`.

css2th

Purpose

This function converts a continuous time state space model to a parameter vector that describes the model.
Model structure:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) + Ke(t)\\ y(t) &= Cx(t) + Du(t) + e(t)\\ x(0) &= x_0\end{aligned}$$

Syntax

```
[theta,params] = css2th(A,B,C,D,x0,K,partype);
[theta,params] = css2th(A,B,C,D);
[theta,params] = css2th(A,C);
```

Inputs

`A,B,C,D`	System matrices describing the state space system. The B and D matrices are optional. An optional element can be left out or given as an empty matrix to indicate it is not part of the model.
`x0`	Initial condition, This is optional.
`K`	Kalman gain. Also this matrix is optional.
`partype`	This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on'= Output Normal and 'tr'=TRidiagonal.

Outputs

`theta`	Parameters vector describing the system.
`params`	A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether B, D, x_0 or K are present, etc.
`T`	Transformation matrix between the input state space system and the state space system in the form described by theta (output normal or tridiagonal).

See also

`cth2ss, cslslin`

cth2ss

Purpose

This function converts a parameter vector that describes a continuous time state space model in output normal form to the state space matrices of that model.
Model structure:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) + Ke(t)\\ y(t) &= Cx(t) + Du(t) + e(t)\\ x(0) &= x_0\end{aligned}$$

Syntax

```
[A,B,C,D,x0,K] = cth2ss(theta,params)
[A,C] = cth2ss(theta,params)
```

Inputs

`theta`	Parameter vector describing the system.
`params`	A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D, x_0 or K are present, etc.

Outputs

`A,B,C,D`	System matrices describing the state space system in output normal form. If theta does not contain parameters for D, this matrix will be returned as an empty matrix.
`x0`	Initial condition. If theta does not contain parameters for x_0 this vector will be returned as an empty matrix.
`K`	Kalman gain. Same here.

See also

`cth2ss, cslslin`

destac

Purpose

Estimates the A and C matrices of a LTI state space model form using the result of the preprocessor routines `dordxx`. (`dordom`, `dordpo`, etc.)

General model structure:

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

For information about the possible disturbance signals see the help pages for the preprocessor `dordxx` functions.

Syntax

```
[A,C]=destac(R,n);
```

Inputs

`R`	Data structure obtained from a `dordxx` function, containing the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).
`n`	Order of system to be estimated.

Outputs

`A,C`	Estimated system matrices.

See also

`dordom, dordpi, dordpo, dordeiv, dordrs, destbd`

destbd

Purpose

Estimates the matrices B and D of the state space model

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned}$$

using the knowledge of the pair A, C. This function can concatenate different input-output data batches, through the matrix R and R_{old}. B and D are calculated by solving a linear least squares problem. The function can handle errors-in-variables models by using a instrumental variable.

Syntax

```
[B,D]=destbd(u,y,A,C);
[B,D,x0,R,Phi]=destbd(u,y,A,C,[fB fD fx],Rold,iv,niv);
```

Inputs

`u,y`	The input, and output data of the system to be identified.
`A,C`	The state space matrices A and C.
`model`	Three element flag vector $\begin{bmatrix} fB & fD & fx \end{bmatrix}$ indicating whether B, D and x_0 should be estimated. The default value is $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$. The matrix B or D can be assumed zero by setting fB or fD to zero. The calculation of x_0 can be omitted by setting fx to zero. However x_0 will not be assumed zero. It's influence will still be taken into account for the computation of B and D. Variables that are not asked for will be returned as empty matrices.
`Rold`	R matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments.
`iv`	Instrumental variable. For instance, for the errors-in-variables case the iv can be the past output or for closed loop data, the reference input.
`niv`	Three element vector describing the number of lagged IVs used. niv(1) is the number of lagged IVs, niv(2) the number of lagged past outputs, and niv(3) the number of lagged past inputs. If only one element is present, no past data is used. If not given or empty appropriate default values are used. When past data is used as instrumental variable, about half of the input/output data is used for the instrument.

Outputs

`B,D`	The estimated system matrices B and D.
`x0`	The estimated initial state of the system.
`R`	Compressed data matrix, storing information on the calculation of the matrices B and D in following `destbd`. Used when analyzing multiple input-output data sequences.
`Phi`	Regressors matrix that was used for the Least squares estimate.

See also

`destac destx`

destk

Purpose

Estimates the Kalman gain of a innovations model, when the matrices A and C are known. They can be estimated with the functions `dordpo` and `destac`. Model structure:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned}$$

where $w(k)$, $v(k)$ are zero-mean white noise sequences, independent of the noise-free input $u(k)$.

Syntax

```
K=destk(A,B,C,D,R)
K=destk(A,C,R)
```

Inputs

`A,B,C,D`	State space matrices (B and D are optional).
`R`	Data structure that is obtained from `dordpo`. This matrix contains compressed data needed to estimate the Kalman gain. This includes the lower triangular factor and additional information (such as i/o dimension etc.).

Outputs

`K`	Estimated Kalman gain.

See also

```
See also:  dordpo, destac, destbd
```

destx

Purpose

Estimates the initial state of a state space system on the basis of the system matrices and a set of input/output data.

Syntax

```
x0=destx(u,y,A,B,C,D);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`A,B,C,D`	System matrices.

Outputs

`x0`	Estimated initial state.

See also

`destbd`

dfunlin

Purpose

This function implements the cost-function for dslslin It is not meant for stand-alone use.

Syntax

```
epsilon=dfunlin(thn,u,y,params)
```

Inputs

`thn`	Parameter vector describing the system matrices A and C.
`u,y`	The input and output data of the system to be optimized.
`params`	A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D, x_0 or K is present in the model, etc.

Outputs

`epsilon`	Output of the cost-function, which is the square of the error between the output and the estimated output, divided by the number of samples.

See also

`dslslin, dslswie, dfunwie`

dfunwie

Purpose

This function implements the cost-function for dslswie It is not meant for stand-alone use.

Syntax

```
epsilon=dfunwie(thn,u,y,params)
```

Inputs

`thn`	Parameter vector describing the system matrices A, B, C and D.
`u,z`	The input and output data of the system to be optimized.
`params`	A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D or x_0 is present in the model, etc.

Outputs

`epsilon`	Output of the cost-function, which is the square of the error between the output and the estimated output, divided by the number of samples.

See also

`dslswie, dslslin, dfunlin`

dmoesp

Purpose

High level function for the discrete time moesp that gives the user a simple interface to the lower level MOESP functions such as `dordxx` and `destac`. This function calculates the system matrices A, B, C and D from the given input and output data sequences u and y. Model structure:

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

For information about the possible disturbance signals see the help pages for the preprocessor `dordxx` functions.

The user only needs to specify the input and output data. If the order is not given, a order selection dialog is shown where the user can select the order by examining the singular value plot.

Syntax

```
[A,B,C,D]=dmoesp(u,y,maxorder)
[A,B,C,D,K]=dmoesp(u,y,maxorder,order,method)
```

Inputs

`u,y`	Input and Output data sequence.
`maxorder`	Maximum expected order of the system, this value is used as the Hankel dimension parameter in the underlying identification routines.
`order`	Order of the system to be estimated, if not specified a dialog will be presented which allows the user to choose the order.
`method`	Method used for identification. possible options: **“om”** ordinary moesp; **“pi”** past input moesp; **“po”** past output moesp. The default value is past output moesp.

Outputs

`A,B,C,D`	State space matrices describing the estimated model.
`K`	Estimated Kalman gain, only available for past output moesp.

See also

`destac, dordpo, etc`

dordeiv

Purpose

`dordeiv` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument Z. The estimated column-space is used in the function `destac` to extract the matrices A and C. This function implements the errors-in-variables MOESP algorithm which can be used for the errors-in-variables identification problem.
Model structure:

$$\begin{aligned} x(k+1) &= Ax(k) + B\tilde{u}(k) + w(k) \\ y(k) &= Cx(k) + D\tilde{u}(k) + v(k) \end{aligned}$$

with measurements

$$u(k) = \tilde{u}(k) + f(k) \quad \text{and} \quad y(k)$$

where $f(k), w(k)$ and $v(k)$ are zero-mean white noise sequences independent of the input $\tilde{u}(j)$ for $k \geq j$.

The system can be operated under either open-loop or closed-loop. For closed-loop operation, $r(k)$ is an external reference input. For open-loop operation, where $\tilde{u}(k)$ is white noise, `dordpo` will give better results.

Syntax

```
[Sn,R,Z]=dordeiv(u,y,r,i,Zold)
[Sn,R,Z]=dordeiv(u,y,[],i,Zold)
```

Inputs

`u,y`	The input and output data of the system to be identified.
`r`	Closed-loop reference input.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Zold`	Matrix obtained from processing a previous data-batch with `dordrs`.

Outputs

`Sn`	Singular values bearing information on the order of the system
`R`	Data structure used by `destac` for the estimation of A and C. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).
`Z`	Matrix containing information on system which can be used by future calls of `dordeiv`.

See also

`dordom, dordpi, dordpo, dordrs, destac, destbd.`

dordom

Purpose

`dordom` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument R. The estimated column-space is used in the function `destac` to extract the matrices A and C.
This function implements the ordinary MOESP algorithm which can only be used for the noise free identification problem.
Model structure:

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

Syntax

```
[Sn,R]=dordom(u,y,i);
[Sn,R]=dordom(u,y,i,Rold);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `dordom` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `destac` for the estimation of A and C or by a next call to `dordom`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`dordpi, dordpo, dordrs, dordeiv, destac, destbd.`

dordpi

Purpose

`dordpi` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument R. The estimated column-space is used in the function `destac` to extract the matrices A and C. This function implements the past input MOESP algorithm which can be used for the output error identification problem.
Model structure:

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

where $v(k)$ is zero-mean noise of arbitrary color, independent of the noise-free input $u(k)$.

Syntax

```
[Sn,R]=dordpi(u,y,i);
[Sn,R]=dordpi(u,y,i,Rold);
```

Inputs

`u, y`	The input and output data of the system to be identified.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `dordpi` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `destac` for the estimation of A and C or by a next call to `dordpi`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`dordom, dordpo, dordrs, dordeiv, destac, destbd.`

dordpo

Purpose

`dordpo` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument R. The estimated column-space is used in the function `destac` to extract the matrices A and C. This function implements the past output MOESP algorithm which can be used for the innovations model identification problem.
Model structure:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned}$$

where $w(k)$, $v(k)$ are zero-mean white noise sequences, independent of the noise-free input $u(k)$.

Syntax

```
[Sn,R]=dordpo(u,y,i);
[Sn,R]=dordpo(u,y,i,Rold);
```

Inputs

`u, y`	The input and output data of the system to be identified.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `dordpo` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system.
`R`	Data structure used by `destac` for the estimation of A and C or by a next call to `dordpo`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`dordom, dordpi, dordrs, dordeiv, destac, destbd.`

dordrs

Purpose

`dordrs` is a preprocessor function that extracts the column-space of the extended observability matrix from input/output data. Data from different experiments can be concatenated using the extra input argument R. The estimated column-space is used in the function `destac` to extract the matrices A and C. This function implements the reconstructed state MOESP algorithm which can be used for the output error identification problem.
Model structure:

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

where $v(k)$ is zero-mean noise of arbitrary color, independent of the noise-free input $u(k)$.

Syntax

```
[Sn,R]=dordrs(u,y,x,i);
[Sn,R]=dordrs(u,y,x,i,Rold);
```

Inputs

`u, y`	The input and output data of the system to be identified.
`x`	Reconstructed state.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices. This parameter should be chosen larger than the expected system order. The optimal value has to be found by trial and error. Generally twice as large is a good starting value.
`Rold`	Data structure obtained from processing a previous data-batch with `dordrs` containing the same items as R.

Outputs

`Sn`	Singular values bearing information on the order of the system
`R`	Data structure used by `destac` for the estimation of A and C or by a next call to `dordrs`. This matrix contains the triangular factor, estimated column-space and additional information (such as i/o dimension etc.).

See also

`dordom, dordpi, dordpo, dordeiv, destac, destbd.`

drpi

Purpose

Estimates in a recursive way the matrices of an LTI state space model in innovation form using the output of the dordpi, destac and destbd routines as initial guess.

Model structure:

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

where $v(k)$ is zero-mean white noise sequence, independent of the noise-free input $u(k)$.

Syntax

```
[A,B,C,D]= drpi(u,y,i,n,beta,Ai,Bi,Ci,Di,Ri)
```

Inputs

`u,y`	The input and output data of the system to be identified.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices such as ws used in `dordpi` for the creation of R_i
`n`	Order of system to be estimated.
`beta`	Forgetting parameter for recursive algorithm ($0 < \beta < 1$).
`Ai,Bi,Ci,Di`	Initial estimate of state space matrices. These matrices can for instance be obtained by using `dordpi/destac/destbd` on the first few samples.
`Ri`	Triangular factor from `dordpi`.

Outputs

`A,B,C,D`	The estimated system matrices.

See also

`drpo, dordpi, destac, destbd`

drpo

Purpose

Estimates in a recursive way the matrices of an LTI state space model in innovation form using the output of the dordpo and dmodpo routines as initial guess. Model structure:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned}$$

where $w(k)$, $v(k)$ is zero-mean white noise sequences, independent of the noise-free input $u(k)$.

Syntax

```
[A,B,C,D]= drpo(u,y,i,n,beta,Ai,Bi,Ci,Di,Ri)
```

Inputs

`u,y`	The input and output data of the system to be identified.
`i`	The dimension parameter that determines the number of block rows in the processed Hankel matrices such as ws used in `dordpo` for the creation of R_i
`n`	Order of system to be estimated.
`beta`	Forgetting parameter for recursive algorithm ($0 < \beta < 1$).
`Ai,Bi,Ci,Di`	Initial estimate of state space matrices. These matrices can for instance be obtained by using `dordpo/destac/destbd` on the first few samples.
`Ri`	Triangular factor from `dordpo`.

Outputs

`A,B,C,D`	The estimated system matrices

See also

`drpi, dordpo, destac, destbd`

drslslin

Purpose

This function performs a recursive update of a discrete time state space system,

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

using the Separable Least Squares technique. For this, only the initial estimates of A and C are needed. The same function can also be used to initialize the matrices needed to start up the recursion.

Syntax

```
[A,B,C,D,hist] = drslslin(u,y,A,C)
[A,B,C,D,hist] = drslslin(u,y,A,C,K,model,partype,options,slsstate)
[slsstate] = drslslin(u,y,A,C,K,model,partype,'init')
```

Inputs

`A,C`	Initial estimate of the state space matrices A and C
`u,y`	The input and output data of the system to be optimized.
`model`	Vector [fB,fD,fx,fK] specifying whether the matrix B, D, the initial state x_0 and the Kalman filter gain K should be estimated. Default is $\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$. It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.
`options`	This vector consists of parameters that can be used to influence the optimization process. options(1) Display parameter (Default:0). 1 displays some results. options(2) is the step size parameter. options(3) is a forgetting factor of the nonlinear param. options(4) is a forgetting factor of the linear param.

Outputs

`A,B,C,D`	Estimated state space matrices.
`x0`	Initial state. If it is not estimated, this vector will be returned as an empty matrix.
`K`	Kalman gain. Same here.
`hist`	History of the recursion the first column is the value of the cost-function at every step. the other columns are the estimated parameters at every step.

See also

`dslslin`

dslslin

Purpose

Performs a Least Squares optimization of a discrete time linear state space system system with model structure:

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

First, the state space matrices are parameterized. In order to minimize the number of parameters, the Separable Least Squares technique is used. This allows the cancellation of the parameters of B and D, such that only the parameters that describe A and C need to be optimized. The parameterized model is optimized with the leastsq function from the MATLAB optimization toolbox. If needed also the initial state and a Kalman gain can be optimized.

Syntax

```
[A,B,C,D]=dslslin(u,y,A,C)
[A,B,C,D,x0,K,options] = dslslin(u,y,A,C,K,model,partype,options)
```

Inputs

`u,y`	The input and output data of the system to be optimized.
`A,C`	Initial estimates of the system matrices A, and C.
`model`	Vector with flags that specify the kind of model to use. model(1) specifies if matrix B should be estimated. model(2) specifies if matrix D should be estimated. model(3) specifies if the initial state x_0 should be estimated. model(4) specifies if the Kalman filter gain K is estimated. Default is $\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$. It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.
`partype`	This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on'= Output Normal and 'tr'=TRidiagonal.
`options`	Input parameters that are passed on directly to the optimization function from the Optimization Toolbox. See foptions for more information.

Outputs

`A,B,C,D`	System matrices of the optimized linear model. If B or D is not estimated, it will be returned as an empty matrix.

`x0`	Estimate of the initial state. If the x_0 matrix is not estimated, it will be returned empty.
`K`	Estimate of the initial state. Same here.
`options`	Output parameters from the Optimization Toolbox. See foptions.

See also

```
    See also:  leastsq, foptions
```

dslswie

Purpose

Separable Least Squares optimization of a wiener model. Model structure:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \\ z(k) &= f(y(k)) \end{aligned}$$

First, the state space matrices are parameterized. In order to minimize the number of parameters, the Separable Least Squares technique is used. This allows the cancellation of the parameters of the nonlinearity, such that only the parameters that describe the linear part need to be optimized. The parameterized model is optimized with the leastsq function from the MATLAB optimization toolbox.

Syntax

```
[A,B,C,D,x0,thl,options] = dslswie(u,z,A,B,C,D,x0,nn,model,options)
```

Inputs

`u,z`	The input and output data of the system to be optimized.
`A,B,C,D`	Initial estimates of the linear part of the wiener model.
`x0`	Initial condition, This is optional.
`nn`	The order of the Chebychev polynomials in the static nonlinearity.
`model`	Vector with flags that specify the kind of model to use. model(1) specifies if B should be estimated. model(2) specifies if D should be estimated. model(3) specifies if the initial state x_0 should be estimated. Default is $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$. It is recommended only to estimate the initial state if the data length is short compared to the largest time constant of the system.
`partype`	This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on'= Output Normal and 'tr'=TRidiagonal.
`options`	Input parameters that are passed on directly to the optimization function from the Optimization Toolbox. See foptions for more information.

Outputs

`A,B,C,D`	System matrices of the optimized linear model. If the B or D matrix is not estimated, it will be zero.

`x0`	Estimated initial state. If x_0 is not estimated it will be returned as an empty matrix.
`options`	Output parameters from the Optimization Toolbox. See foptions.

See also

`foptions, dslslin`

dss2th

Purpose

This function converts a discrete time state space model to a parameter vector that describes the model Model structure:

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

Syntax

```
[theta,T,params] = dss2th(A,C,partype)
[theta,T,params] = dss2th(A,B,C,D,partype)
[theta,T,params] = dss2th(A,B,C,D,x0,K,partype)
```

Inputs

`A,B,C,D`	System matrices describing the state space system. The B and D matrices are optional and can be left out or given as an empty matrix to indicate it is not part of the model.
`x0`	Initial condition, This is optional.
`K`	Kalman gain. Also this matrix is optional.
`partype`	This parameter specifies the type of parameterization that is used to parameterize the state space model. Two types of parameterization are supported: 'on'= Output Normal and 'tr'=TRidiagonal.

Outputs

`theta`	Parameters vector describing the system.
`params`	A structure that contains the dimension of the system, such as the order, the number of inputs, whether D, x_0 or K is present, etc.
`T`	Transformation matrix between the input state space system and the state space system in the form described by theta. (the one that is constructed by `dth2ss`

See also

`dth2ss`

dth2ss

Purpose

This function converts a parameter vector that describes a discrete time state space model in output normal form to the state space matrix of that model. Model structure:

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

Syntax

```
[A,B,C,D,x0,K] = dth2ss(theta,params)
[A,C] = dth2ss(theta,params)
```

Inputs

`theta`	Parameter vector describing the system.
`params`	A structure that contains the dimension parameters of the system, such as the order, the number of inputs, whether D, x0 or K is present in the model, etc.
`T`	Transformation matrix to be applied to the state space system that is constructed from theta. This transformation might come from the function `dss2th` and can be used to reconstruct the original state space matrices that were given to `dss2th`.

Outputs

`A,B,C,D`	System matrices describing the state space system in output normal form. If theta does not contain the parameters for a matrix, this matrix will be returned as an empty matrix.
`x0`	Initial state. If theta does not contain parameters for x_0, this vector will be returned as an empty matrix.
`K`	Kalman gain. Same here.

See also

`dss2th`

kronekf

Purpose

Calculates the Kronecker canonical form from a regular pencil A,B such that

$$\begin{aligned} A_a &= QAZ = \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix} \\ B_b &= QBZ = \begin{bmatrix} I & 0 \\ 0 & B_k \end{bmatrix} \end{aligned}$$

where the eigenvalues of As and Bs are split according to alpha and option

Syntax

```
[AA,BB,Q,Z,na,nb]=kronekf(A,B,alpha,option)
```

Inputs

`A,B`	The given matrix pencil.
`alpha`	The value used to split the eigenvalues of the pencil A and B. Default value is 1.
`option`	This parameter can be either 'circle' or 'halfplane'. 'circle' sorts the eigenvalues in A_k to be absolutely smaller than alpha, and those in B_k to be larger than alpha. 'halfplane' sorts the eigenvalues in A_k to have smaller real part than alpha, and those in B_k to have larger real part. Default is 'circle'.

Outputs

`AA,BB`	The pencil in canonical form.
`Q`	The required left transformations.
`Z`	The required right transformations.
`na`	Dimension of the matrix A_k.
`nb`	Dimension of the matrix B_k.

See also

`ncdestac`

ncdestac

Purpose

Estimates the system matrices A_c, A_a, C_c and C_a of a non-causal LTI state space model using the output of one of the `dordxx` preprocessor routines. The model is divided in a causal part and an anti-causal part. The causal part is stable and can be simulated forward in time. The anti-causal part can only be simulated stably backward in time. The other matrices are estimated with `ncdestbd`. The model can be simulated with `ncdlsim`.

Model structure:

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c u(k) \quad \text{(causal part)} \\ x_a(k-1) &= A_a x_a(k) + B_a u(k) \quad \text{(anti-causal part)} \\ y(k) &= C_c x_a(k) + C_a x_a(k) + Du(k) \end{aligned}$$

All preprocessor functions can be used for the causal case as well as the non-causal case. For information about the possible disturbance signals see the help pages for the preprocessor `dordxx` functions.

Syntax

```
[Ac,Aa,Cc,Ca]=ncdmod(R,n)
```

Inputs

`R`	Data structure obtained from a `dordxx` function, containing the triangular factor and additional information (such as i/o dimension etc.).
`n`	Order of system to be estimated.

Outputs

`Ac,Aa,Cc,Ca`	Estimated system matrices.

See also

`ncdestbd, ncdlsim`

ncdestbd

Purpose

Estimates the matrices B_c, B_a and D of a non-causal state space model, using the knowledge of the matrices A_c, A_a, C_c and C_a. This function is able to concatenate different data batches, through the matrices R, R_{old}.
Model structure:

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c u(k) \quad \text{(causal part)} \\ x_a(k-1) &= A_a x_a(k) + B_a u(k) \quad \text{(anti-causal part)} \\ y(k) &= C_c x_a(k) + C_a x_a(k) + D u(k) \end{aligned}$$

B_c, B_a and D are calculated by solving a linear least squares problem. The influence of the initial state x_0 in the data batches is compensated for in the solution. x_{0_c} and x_{0_a} can also be estimated.

Syntax

```
[Bc,Ba,D,x0c,x0a,R]=ncdestbd(u,y,Ac,Aa,Cc,Ca,model,Rold,iv,niv);
[Bc,Ba,D]=ncdestbd(u,y,Ac,Aa,Cc,Ca);
```

Inputs

`u,y`	The input and output data of the system to be identified.
`Ac,Ae` `Cc,Ca`	The estimated system matrices A_c, A_e, C_c and C_a of the non-causal state space system matrices.
`model`	Two element vector $\begin{bmatrix} fB & fD & fx \end{bmatrix}$ indicating whether B_c, B_a, D and x_{0_c}, x_{0_a} should be estimated. Default value is $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$.
`Rold`	R matrix obtained from previous data batch. This variable can be used to process data in batches, or to combine data from different experiments.
`iv`	Instrumental variable. For instance, for the errors-in-variables case the iv can be the past output or for closed loop data, the reference input.
`niv`	Three element vector describing the number of lagged IVs used. niv(1) is the number of lagged IVs, niv(2) the number of lagged past outputs, and niv(3) the number of lagged past inputs. If only one element is present, no past data is used. If not given or empty appropriate default values are used. When past data is used as instrumental variable, about half of the input/output data is used for the instrument.

Outputs

`Bc,Ba,D`	The estimated system matrices B_c, B_a and D.
`x0c,x0c`	The estimated initial state of the system.
`R`	Compressed data matrix, storing information on the calculation of the matrices B_c, B_a and D in a following `ncdestbd`. Used when analyzing multiple input-output data sequences.

See also

`ncdlsim, ncdestac`

ncdlsim

Purpose

This function simulates a non-causal, unstable or unproper system. The model is divided in a causal part, and an anti-causal part. The causal part is stable and can be simulated forward in time. The anti-causal part can only be simulated stably backward in time. The system can be given in the following forms Non causal state space form:

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c u(k) \text{(causal part)} \\ x_a(k+1) &= A_a x_a(k) + B_a u(k) \text{(anti-causal part)} \\ y(k) &= C_c x_c(k) + C_a x_a(k) + Du(k) \end{aligned}$$

Unstable state space form:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned}$$

Non-proper transfer function:

$$a_n y(k+n) + ... + a_0 y(k) = b_m u(k+m) + ... + b_1 u(k+1) + b_0 u(k)$$

Syntax

```
[y,x]=ncdlsim(Ac,Aa,Bc,Ba,Cc,Ca,D,u,xc0,xa0)
[y,x]=ncdlsim(A,B,C,D,u,x0)
[y,x]=ncdlsim(num,den,u)
```

Inputs

`Ac,Aa,Bc,Ba`	State space matrices representing a non-causal
`Cc,Ca,D`	state space model.
`A,B,C,D`	State space matrices representing a possibly unstable state space model.
`num,den`	Numerator and denominator of possibly non-proper transfer function.
`u`	Input to the system.
`xc0,xa0`	Initial states of causal and anti-causal part of the non-causal state space model.
`x0`	Initial state of the state space model.

Outputs

`y`	Simulated output.

`x`	Simulated state of the state space model. The first n_c columns are the states of the causal part, the rest are the states of the anti-causal part.

See also

`ncdestac, ncdestbd, dlsim`

orderselect

Purpose

Gives an estimate of the order of a system based on the singular values. Two methods are available: “manual” and “largest-gap”. The manual method shows a semi-logarithmic plot of the singular values and lets the user manually choose the order by inspection. The largest-gap method simply selects the order to be the one after which the largest gap occurs in the semi-logarithmic plot.

Syntax

```
order=orderselect(S,method)
```

Inputs

`S`	Vector with singular values, obtained from one of the preprocessor functions `dordxx`(`dordom`, `dordpi`, etc).
`method`	Method for selecting the order. Possible methods are: “manual” and “largest-gap”. Default is “manual”.

Outputs

`order`	Selected model order.

See also

`moesp, dordpo, dordpi`

prbn

Purpose

Produces a binary sequence, with values 0 and 1. The chance of switching from level is given by the parameter 'rate'. Rate=0 gives a constant value 0. Rate=1 gives a signal that changes constantly between 0 and 1. Any value in between results in a random binary sequence. This kind of test signal has been described in [**?**]

Syntax

```
y=prbn(N,rate);
```

Inputs

`N`	Number of points.
`rate`	Chance of the signal changing level at every sample. Default is 0.5.

Outputs

`y`	Random binary noise.

See also

shave

Purpose

This function is used for reducing spikes from a measured signal. The spikes of the signal are 'shaved' as follows:

- From the signal a trend is computed using a fourth-order Butterworth filter.
- The standard deviation of the trend-corrected, clipped signal is computed.
- Detection band is defined by the trend plus and minus a certain factor times the standard deviation. All samples that are outside this band are replaced using linear interpolation. This 'shaving' method has been described in [**?**].

Syntax

```
shave(Signal);
Shaved_signal=shave(Signal);
Shaved_signal=shave(Signal,factor,Wn,lo_lim,up_lim);
```

Inputs

`Signal`	Signal to be 'shaved' (column vector)
`factor`	Multiplication factor which determines the width of the detection band. When the detection is poor, you should change this factor. This argument is optional. Its default value is 2.
`Wn`	Cut-off frequency of the low-pass filter used for trend determination. It must be in the range 0.0 < `Wn` < 1.0, with 1.0 corresponding to half the sample rate. This argument is optional. Its default value is 0.01.
`lo_lim, up_lim`	If these arguments are present, the signal is clipped to a minimum value of `lo_lim` and a maximum value of `up_lim` before the 'shaving' starts.

Outputs

`Shaved_signal`	'Shaved' signal.

See also

tlsim

Purpose

Simulation of the time response of LTI continuous-time systems to arbitrary inputs. The input and simulated output are represented by sampled signals. Unlike lsim, where the sampling is assumed to be zero order or first order hold, the sampling in tlsim is assumed to be trapezoidal or Tustin's (hence the t in tlsim).

Syntax

```
[y,x]=tlsim(A,B,C,D,u,Ts,x0)
```

Inputs

`A,B,C,D`	State space matrices of the LTI system.
`u`	Input signal.
`Ts`	Sampling time.
`x0`	Initial state of the system.

Outputs

`y`	Output of the LTI system.
`x`	State of the LTI system.

See also

`lsim, dlsim`

vaf

Purpose

Compute the percentage Variance Accounted For (VAF) between two signals. the VAF is calculated as:

$$1 - \frac{\text{variance}(y - y_{est})}{\text{variance}(y)} \times 100\%$$

The VAF of two signals that are the same is 100%. If they differ, the VAF will be lower. When y and y_{est} are matrices, the VAF is calculated for every column in y and y_{est}. The VAF is often used to verify the correctness of a model, by comparing the real output with the estimated output of the model.

Syntax

```
v= vaf(y, ye)
```

Inputs

`y`	Reference signal, often the real output
`ye`	Second signal, often the estimated output of a model

Outputs

`v`	VAF, computed for the two signals

See also