NASA Technical Memorandum 107566

SYSTEM/OBSERVER/CONTROLLER IDENTIFICATION TOOLBOX

Jer-Nan Juang, Lucas G. Horta, and Minh Phan

February 1992

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

ACKNOWLEDGEMENT

This is considered as the second version of the original version entitled "System/Observer Realization Toolbox" which was published in July, 1991 and presented at the short course taught by the first author in NASA Langley Research Center, Hampton, Virginia. Since then, we have doubled the number of function files to include the closed-loop identification and backward observer identification. The closed-loop identification allows one to identify not only the open-loop system but also the feedback gain and the effective observer gain. The backward observer identification allows one to smooth the current measurement data from future data. Some of the original function files were completely rewritten to improve the computational speed and reduce the storage memory. In particular, the era function files were completely overhauled.

All the function files were written by the authors who developed the system identification methods shown in this toolbox, including the Eigensystem Realization Algorithm (ERA), the ERA using Data Correlation (ERA/DC), the Observer/Kalman Filter Identification (OKID), and the Observer/Controller Identification (OCID) and the Backward Observer Identification (BOID). However, this toolbox will not come true without the following co-developers of the methods mentioned above.

Richard S. Pappa (ERA), NASA Langley Research Center, USA
Jonathan E. Cooper (ERA/DC), University of Manchester, England
Jan R. Wright (ERA/DC), University of Manchester, England
Richard W. Longman (OKID), Columbia University, USA
Chung-Wen Chen (OKID), North Carolina University, North Carolina, USA

Dr. Jiann-shiun Lew who helped write the functions monera, peradc and svra in the original version is also deeply appreciated.

It takes time to develop a system identification technique useful for structural dynamics and control testing. We probably would not have been able to develop any useful technique had we not had a considerable strong support from our superiors. We thank **Brantley R. Hanks** and **Larry D. Pinson** for providing inspiring working conditions and making our laboratory experience possible in control and system identification of flexible space structures.

INTRODUCTION

System identification is the process of constructing a mathematical model from input and output data for a system under testing, and characterizing the system uncertainties and measurement noises. The mathematical model structure can take various forms depending upon the intended use. The SYSTEM/OBSERVER/CONTROLLER IDENTIFICATION TOOLBOX (SOCIT) is a collection of functions, written in MATLAB[1] language and expressed in M-files, that implements a variety of modern system identification techniques. For an open-loop system, the central features of the SOCIT are functions for identification of a system model and its corresponding forward and backward observers directly from input and output data. The system and its observers are represented by a discrete model. The identified model and observers may be used for controller design of linear systems as well as identification of modal parameters such as dampings, frequencies, and mode shapes. For a closed-loop system, the central features of the SOCIT include identification of an open-loop model, an observer and its corresponding controller gain directly from input and output data. The basic package is capable of:

1- Identifying system, forward and backward observer Markov parameters (pulse responses) from input and output time histories.
2- Constructing a state space model from pulse responses.
3- Identifying a state space model and its corresponding forward and backward observer gains directly from input and output time histories.
4- Identifying a forward observer/Kalman filter gain with a given state space model, and input and output time histories.
5- Computing variance and bias for identified modal parameters using Monte Carlo and perturbation procedures.
6- Computing forward prediction errors and backward smoothing errors for any of the models generated.
7- Identifying a state space model, and its corresponding controller gain and observer/Kalman filter gain directly from input, output and control force time histories.

The unique features of this package are:

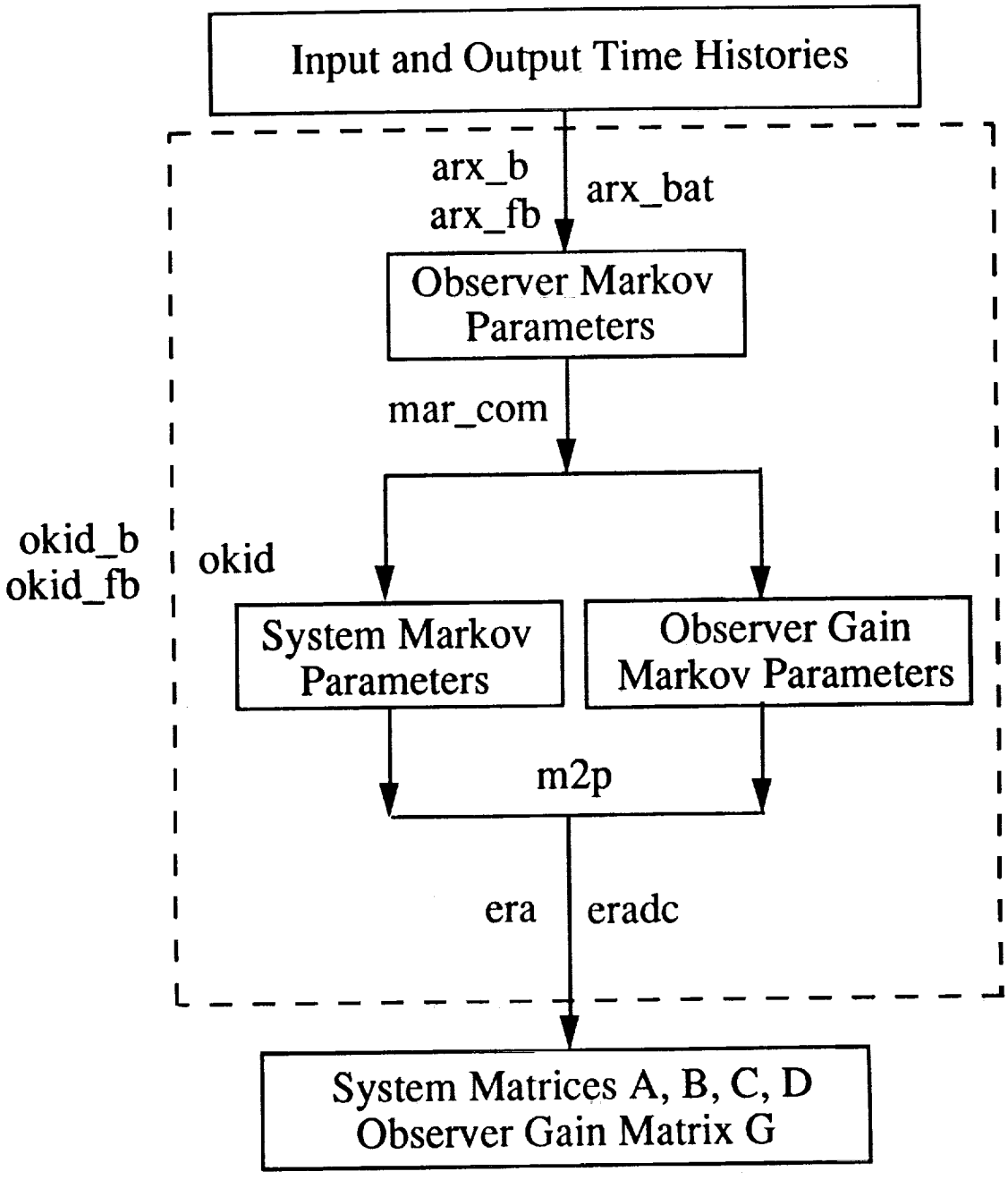
1- No nonlinear programming involved.
2- No a priori noise information required.
3- Guided model order selection.
4- Direct identification of system & observer/Kalman filter.
5- Direct identification of closed-loop controller.
6- Suitable for stable & unstable systems.

Notes

SYSTEM/OBSERVER/CONTROLLER IDENTIFICATION TOOLBOX

Reference

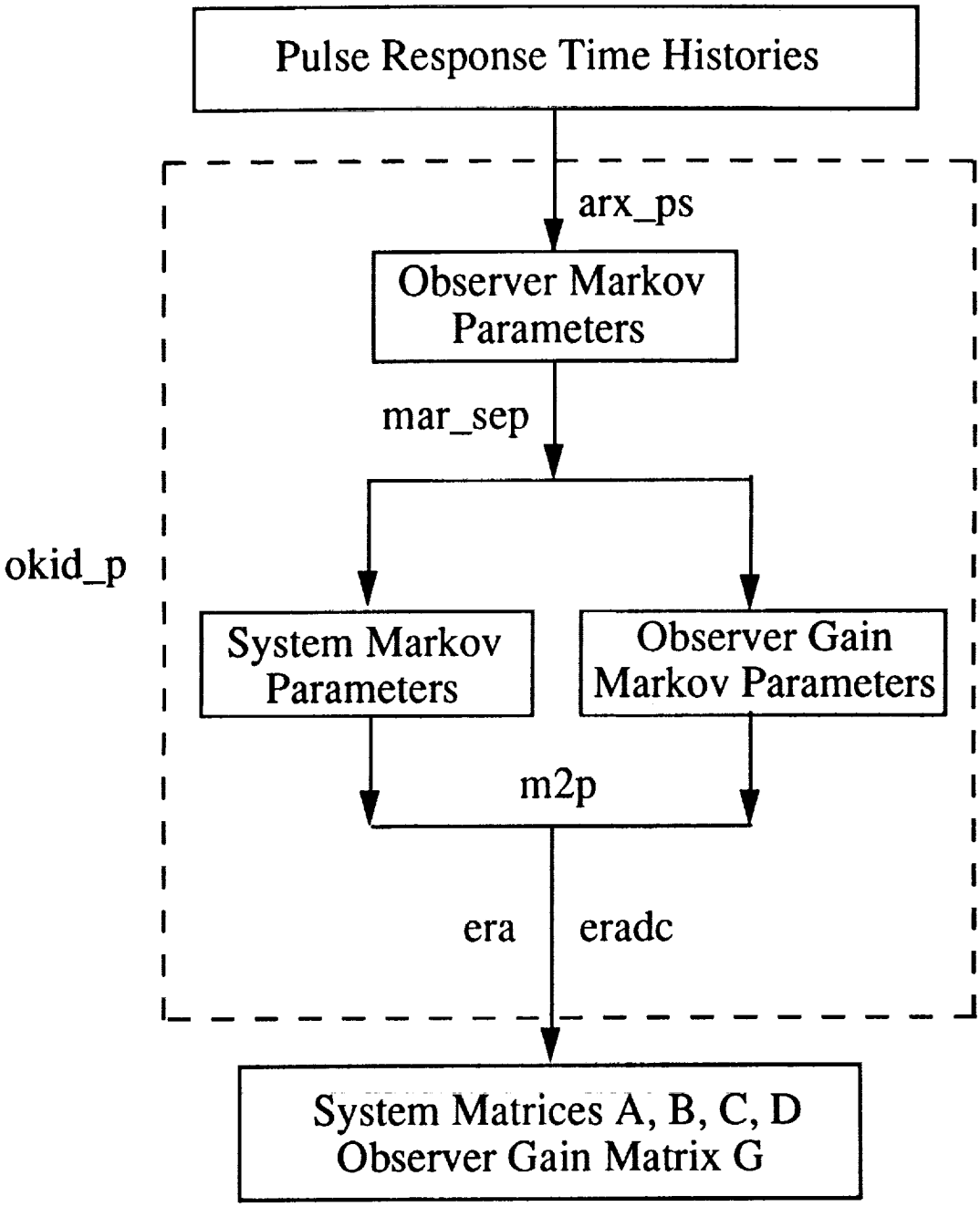
Road Map for Identification of an Open-Loop System
(general input/output data)



Note that Markov parameters also mean pulse response time histories.

Road Map for Identification of an Open-Loop System

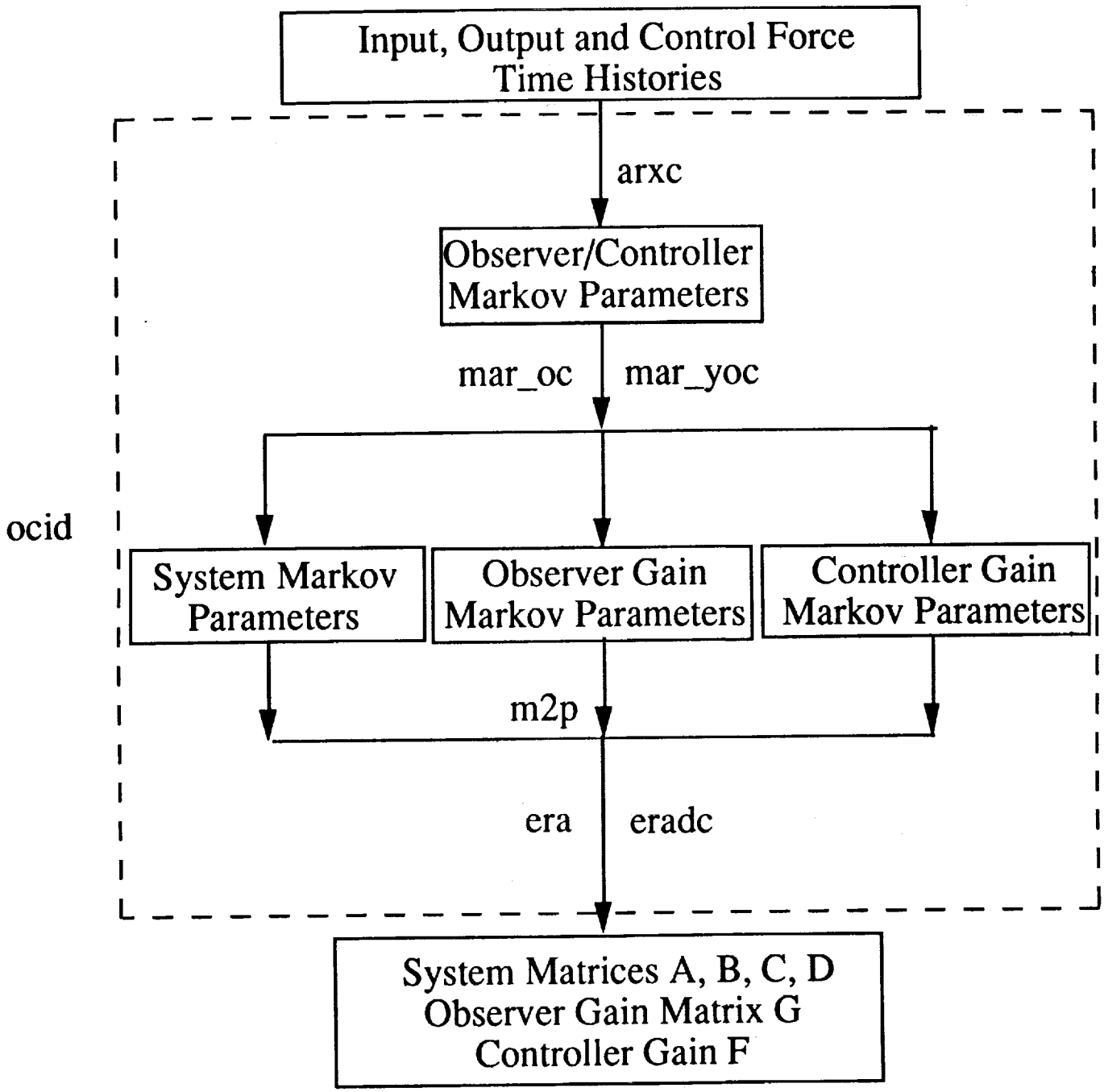
(pulse response time histories)



Note that Markov parameters also mean pulse response time histories.

Road Map for Identification of a Closed-Loop System

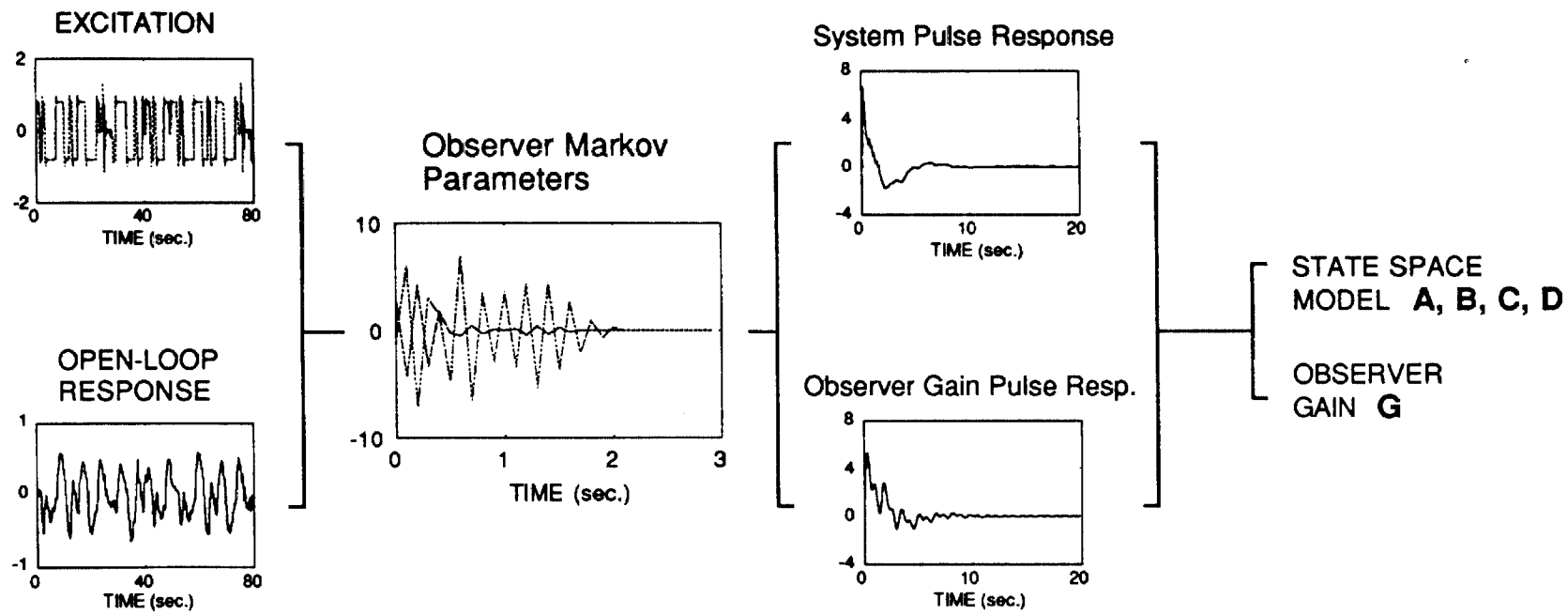
(general input/output data)



Note that Markov parameters also mean pulse response time histories.

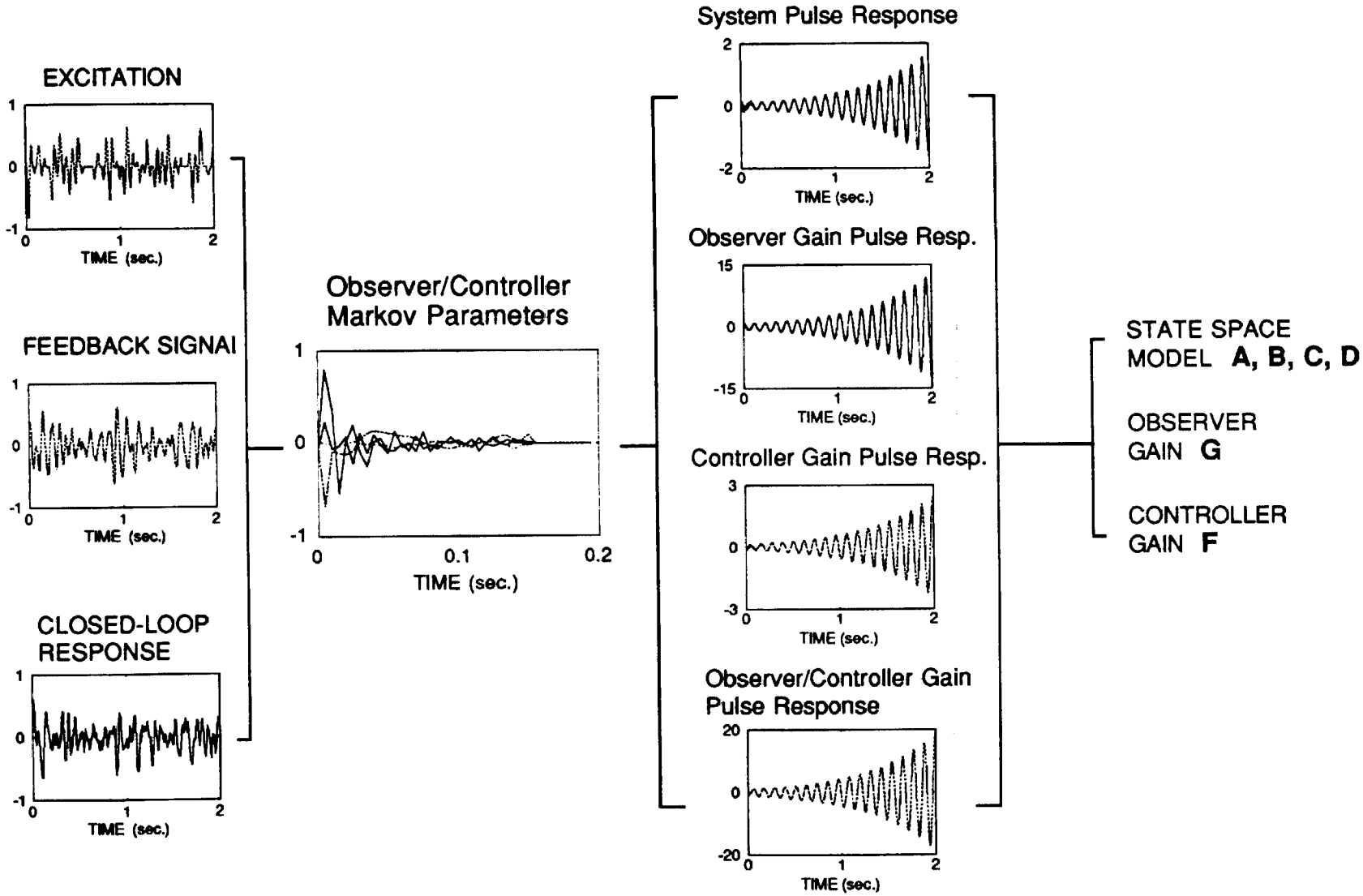
Observer/Kalman Filter Identification (OKID)

(Hubble Space Telescope Flight Test Data)



Closed-Loop Observer/Controller Identification (OCID)

(Aircraft Flutter Test Data)



Notes

Purpose:

Compute observer Markov parameters.

Synopsis:

[*Ybf*]=arx_b(*m,r,u,y,p,ic1*)
[*Yf*]=arx_bat(*m,r,u,y,p,ic2*)
[*Yf,Yb*]=arx_fb(*m,r,u,y,p,ic2*)

Description:

The function computes observer Markov parameters from input/output data. The identified observer system is deadbeat of order p. Given l samples, r inputs, and m outputs, the input matrix u must have dimensions $l \times r$ and output y $l \times m$. Multiple experiments may be used in these functions. In that case, the input matrix u becomes $l \times (rn_e)$ where n_e is the number of experiments, and the output matrix becomes $l \times (mn_e)$. Function arx_bat solves the least squares problem of a forward observer;

$$\underline{y}_f = Y_f V_f$$

where

$$\underline{y}_f = [y(0)\, y(1) \cdots y(l-1)]$$

$$Y_f = [D \quad C\overline{B} \quad C\overline{A}\overline{B} \quad \cdots \quad C\overline{A}^{p-1}\overline{B}]$$

$$V_f = \begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(l-1) \\ & v(0) & v(1) & \cdots & v(l-2) \\ & & \vdots & & \vdots \\ & & v(0) & \cdots & v(l-p-1) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} u(i) \\ y(i) \end{bmatrix}; \quad i = 0,1,\ldots,l-1$$

Here $\overline{A} = A + GC$, $\overline{B} = [B + GD \quad -G]$, in which A, B, C, D are system matrices, and G is the forward observer gain matrix. See function okid for more discussion on the definition of these matrices. The solution is stored in *Yf* of dimension $m \times [(r+m)p + r]$ If the experiment started from rest *ic2(1)*=0, otherwise, *ic2(1)*=1. Once an estimate of the parameters is available the user is given the option to compute the prediction error

$$e_f = \underline{y}_f - Y_f V_f$$

This computation when analyzing long records is time consuming. The square root of the diagonal elements of the inverse correlation matrix are proportional to the parameter variance. A chart depicting these values is plotted along with the prediction error. To bypass the prediction error option, set the second element of *ic2* to one, i.e. *ic2(2)=0.*

Function arx_b solves the least squares problem of a backward observer as follows

$$\underline{y}_b = Y_b V_b$$

where

$$\underline{y}_b = [y(0)\, y(1) \cdots y(l-p-2)]$$

$$Y_b = [D + C\tilde{B} \quad C\tilde{A}^{p-1}[\tilde{G}D \quad -\tilde{G}] \quad C\tilde{A}^{p-2}[\tilde{A}\tilde{B} + \tilde{G}D \quad -\tilde{G}] \quad \cdots \quad C[\tilde{A}\tilde{B} + \tilde{G}D \quad -\tilde{G}]]$$

$$V_b = \begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(l-p-2) \\ v(p) & v(p+1) & v(p+2) & \cdots & v(l-2) \\ \vdots & & \vdots & & \vdots \\ v(1) & v(2) & v(3) & \cdots & v(l-p-1) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} u(i) \\ y(i) \end{bmatrix}; \quad i = 0,1,\ldots,l-1$$

Here, $\tilde{A} = A^{-1} + \tilde{G}C$, $\tilde{B} = -A^{-1}B$, in which A, B, C, D are system matrices, and $\tilde{G}$ is the backward observer matrix. See function okid_b for more discussion on the definition of these matrices. The solution is stored in Yb of dimension $m \times [(r+m)p + r]$. Once an estimate of the parameters is available the user is given the option to compute the smoothing error, instead of the prediction error as in the case for the forward observer.

$$e_b = \underline{y}_b - Y_b V_b$$

The square root of the diagonal elements of the inverse correlation matrix are proportional to the parameter variance. A chart depicting these values is plotted along with the smoothing error. To bypass the smoothing error option, set the the variable ic to one, i.e. *ic1=0*. Note that *ic1* is a scalar whereas *ic2* is a vector with two elements.

Observation of the forward and backward formulations shown above immediately reals that one may simultaneously compute forward and backward observer parameters. Both matrices are very similar in the sense that their lower sides are identical. Function arx_fb solves for backward and forward observer parameters simultaneously. All parameters used above also apply to this function.

Algorithm:

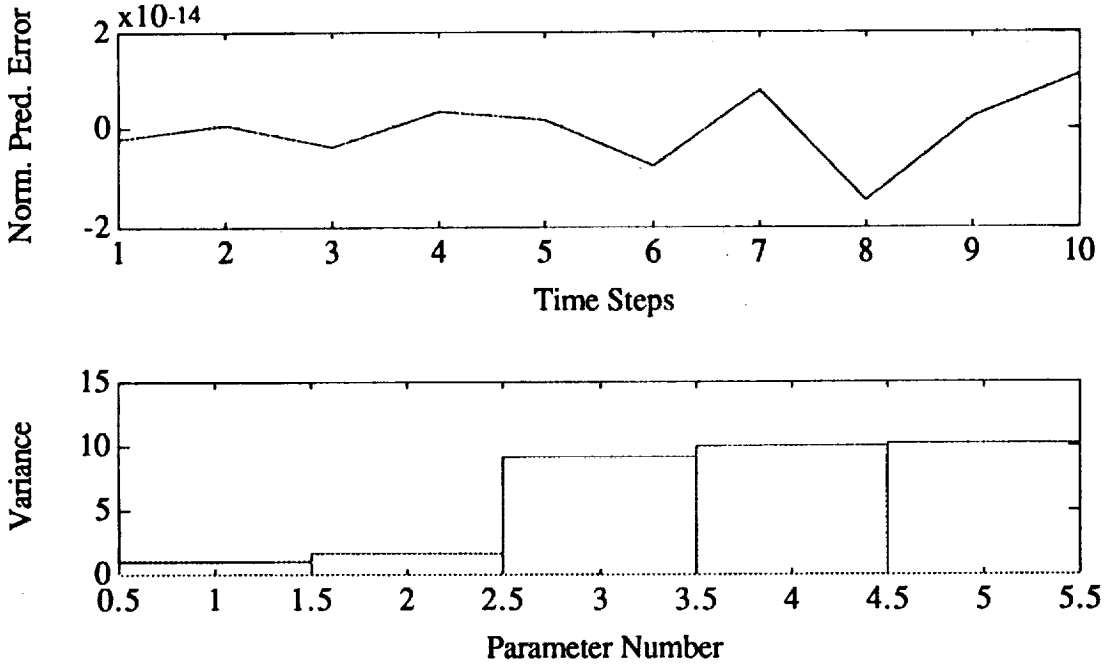
First, the correlation matrices are computed without actually constructing the individual matrices. The parameter estimate is obtained by

$$Y = \underline{y}V^T(VV^T)^+$$

where $(^+)$ refers to pseudo inverse. The pseudo inverse is computed using singular value decomposition.

Example:

```
r=1;m=1;ic=[0 1];index=0;p=2;L=100;
a=[0 -0.16; 1 -1]; b=[0 1]'; c=[0 1]; d=0; G=[0.16 1]';
u=rand(L,m);
y=dlsim(a,b,c,d,u);
psize=r+p*(r+m);
[Y]=arx_bat(m,r,u,y,p,ic);
Compute Prediction Error (1=yes,0=no) =: 1
Square Fitting Error Normalized
1.9097e-29
```



Y = [0.0000 1.0000 -1.0000 0.0000 -0.1600]

See also:

okid, okid_b, okid_fb

Purpose:

Compute combined observer/controller Markov parameters.

Synopsis:

[*Ybar*]=arxc(*y,ufb,ue,p,truncate*)

Description:

This function computes combined observer/controller Markov parameters *Ybar* from feedback control input *ufb*, additive input excitation *ue*, and closed-loop response *y*.

Consider a linear discrete system of the form

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

which is operating in closed-loop. The input to the system, $u(k)$, consists of the feedback signal $u_{fb}(k)$ provided by an *existing* state feedback controller with gain F, and an additive excitation input signal $u_e(k)$

$$u(k) = u_{fb}(k) + u_e(k)$$
$$= -F\hat{x}(k) + u_e(k)$$

The estimated state $\hat{x}(k)$ is provided by an *existing* observer of the form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) - G_d[y(k) - \hat{y}(k)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

The output $y(k)$ is the system closed-loop response due to an excitation $u_e(k)$. The function arxc solves for the observer/controller Markov parameters in *Ybar* which consists of

$$\begin{bmatrix} D \\ 0 \end{bmatrix}, \text{ and } \begin{bmatrix} C \\ -F \end{bmatrix}(A+GC)^{k-1}[B+GD \quad -G], \quad k = 1, 2, \ldots, p$$

where A, B, C, D, and F are of the closed-loop system in operation, and G is another observer gain for the system such that

$$\begin{bmatrix} C \\ -F \end{bmatrix}(A+GC)^{k-1}[B+GD \quad -G] = 0, \quad k = p+1, p+2, \ldots$$

where p is a number specified by the user. The number *truncate* specifies the number of data points to be deleted prior to application of the algorithm. This value is equal to the number of time steps that is expected for the existing observer to converge.

Example:

An example data file is contained in the file xsamp712.

```
load xsamp712
ue(1:600)=[];y(1:600)=[];u(1:600)=[];
[Ybar]=arxc(y,-u,ue,20,300)

Ybar =

 Columns 1 through 7

 -0.0932   0.2513   0.8236  -0.0771   0.3786  -0.0357  -0.5592
  0.1228  -0.0551  -0.7010   0.0188  -0.1409   0.0264  -0.0888

 Columns 8 through 14

  0.0686   0.0149  -0.2200   0.2012   0.1215  -0.0940  -0.1043
  0.0247  -0.1287   0.0462  -0.0793  -0.0044   0.0364  -0.0138

 Columns 15 through 21

 -0.2484   0.0971   0.0189  -0.0101   0.0701  -0.1114  -0.0672
  0.0932  -0.0401   0.0931  -0.0382   0.0814  -0.0203   0.0687

 Columns 22 through 28

  0.1137  -0.0833  -0.0444  -0.0189   0.0115  -0.0695  -0.0910
 -0.0387   0.0449  -0.0070   0.0210  -0.0074   0.0147   0.0149

 Columns 29 through 35

  0.0894   0.1423  -0.0596  -0.0859  -0.0226   0.0041  -0.0683
 -0.0062  -0.0320  -0.0272  -0.0006  -0.0963   0.0263  -0.0956

 Columns 36 through 41

  0.0041   0.0171  -0.0598   0.0512   0.0595  -0.0819
  0.0579  -0.0791   0.0609  -0.0676   0.0661  -0.0833
```

Algorithm:

The observer/controller Markov parameters are computed from feedback control input, additive excitation input, and closed-loop response data. These parameters are used in the function ocid to compute a realization of the system state space matrices, the existing controller gain, and an observer gain.

See also:

ocid, mar_yoc, mar_oc, ryucovar, y_closed, separate

Purpose:

Compute observer Markov parameters from pulse response histories.

Synopsis:

[*d,ys,yo*]=arx_ps(*y,m,p,ic*)

Description:

The function computes observer Markov parameters from pulse response time histories. The identified observer system is deadbeat of order p. Given l samples, r inputs, and m outputs, y is $l \times mr$ .The least squares problem solves the following equations

$$\underline{y} = \overline{Y}V$$

where

$$\underline{y} = [y_1 \quad y_2 \quad \cdots \quad y_r]; \; y_i = [y_i(0) \quad y_i(1) \quad \cdots \quad y_i(l-1)]; \; i = 1,\ldots,r.$$

$$Y = [Y_1 \quad Y_2]; \; Y_1 = [D \quad C(B+GD) \quad C\overline{A}(B+GD) \quad \cdots \quad C\overline{A}^{p-1}(B+GD)];$$

$$Y_2 = [-CG \quad -C\overline{A}G \quad \cdots \quad -C\overline{A}^{p-1}G];$$

$$V = [V_1 \quad V_2 \quad \cdots \quad V_r]; \; V_i = \begin{bmatrix} V_{i1} \\ V_{i2} \end{bmatrix};$$

$$V_{i1} = \begin{bmatrix} u_i(0) & u_i(1) & u_i(2) & \cdots & u_i(l-1) \\ & u_i(0) & u_i(1) & \cdots & u_i(l-2) \\ & & \vdots & & \vdots \\ & & u_i(0) & \cdots & u_i(l-p-1) \end{bmatrix} V_{i2} = \begin{bmatrix} 0 & y_i(0) & y_i(1) & \cdots & y_i(l-2) \\ & & y_i(0) & \cdots & y_i(l-3) \\ & & \vdots & & \vdots \end{bmatrix}$$

$$u_i(0) = \begin{bmatrix} 0 \\ \vdots \\ 1\,(ith element) \\ \vdots \\ 0 \end{bmatrix}; \; u_i(k) = 0; \; k = 1,\ldots,l-1$$

The solution is stored in [*d, ys, yd*] where d is the system transmission matrix D,

$$ys = [C(B+GD) \quad C\overline{A}(B+GD) \quad \cdots \quad C\overline{A}^{p-1}(B+GD)]$$

and

$$yo = [-CG \quad -C\overline{A}G \quad \cdots \quad -C\overline{A}^{p-1}G]$$

The matrix *ys* has dimension $m \times rp$ and *yo* has dimension $m \times mp$.

Algorithm:

First, the correlation matrices are computed without actually constructing the individual matrices. The parameter estimate is obtained by

$$Y = \underline{y} V^T (VV^T)^+$$

where (+) refers to pseudo inverse. The square matrix VV^T has the following special form

$$VV^T = \begin{bmatrix} I_{r(p+1)\times r(p+1)} & \alpha^T_{r(p+1)\times mp} \\ \alpha_{mp\times(p+1)r} & \beta_{mp\times mp} \end{bmatrix}; \quad \beta_{mp\times mp} = \sum_{i=1}^{r} V_{i2} V_{i2}^T$$

$$\alpha_{mp\times(p+1)r} = \begin{bmatrix} 0_{m\times r} & [y_1(0) & \cdots & y_r(0)] & [y_1(1) & \cdots & y_r(1)] & \cdots & [y_1(p-1) & \cdots & y_r(p-1)] \\ & & & & [y_1(0) & \cdots & y_r(0)] & \cdots & [y_1(p-2) & \cdots & y_r(p-2)] \\ & & & & & & & \ddots & & \vdots & \\ & & & & & & & & [y_1(0) & \cdots & y_r(0)] \end{bmatrix}$$

which make the pseudo inverse $(VV^T)^+$ easier as shown below

$$(VV^T)^+ = \begin{bmatrix} I + \alpha^T(\beta - \alpha\alpha^T)^+\alpha & -\alpha^T(\beta - \alpha\alpha^T)^+ \\ -(\beta - \alpha\alpha^T)^+\alpha & (\beta - \alpha\alpha^T)^+ \end{bmatrix}$$

The pseudo inverse $(\beta - \alpha\alpha^T)^+$ is computed using singular value decomposition. Once an estimate of the parameters is available the user is given the option to compute the prediction error

$$e = \underline{y} - YV$$

This computation when analyzing long records is time consuming. The square root of the diagonal elements of the inverse correlation matrix are proportional to the parameter variance. A chart depicting these values is plotted along with the prediction error. To bypass the prediction error option, set *ic=0*.

Example:

This example is to identify observer Markov parameters from the pulse response samples of a three-mass-spring-dashpot system with two inputs and one output.

```
k1=1.0;k2=2.0;k3=3.0;m1=1.0;m2=1.0;m3=1.0;ratio=2*0.005;
K=[k1+k2  -k2   0 ; -k2 k2+k3 -k3; 0   -k3   k3];
Khalf=sqrtm(K);Damp=ratio*Khalf;
Ac=[zeros(3,3) eye(3,3);-K -Damp];Bc=[zeros(3,2);1 0; 0 1; 0 0];C=[zeros(1,5) 1];
dt=1.0;pt=100;p=10;[m,n]=size(C);[n,r]=size(Bc);D=zeros(m,r);
t=[dt:dt:pt*dt]';
[A,B]=c2d(Ac,Bc,dt); y=[];
for i=1:r;
 y=[y dimpulse(A,B,C,D,i,pt)];
end;
[d,ys,yo]=arx_ps(y,m,p,0)
H=mar_sep(ys,yo,d,m,r,10)
```

See also:

mar_sep, okid_p, arx_bat, arx_ps

Purpose:

Transform the complex modal form of a discrete model into a real block diagonal form.

Synopsis:

[*Ab,Bb,Cb*]=bk_diag(*Lambda,B,C*)

Description:

Given a complex diagonal model

$$x(k+1) = \Lambda x(k) + Bu(k);$$
$$y(k) = Cx(k) + Du(k)$$

where

$$\Lambda = diag(\lambda_1,\ldots,\lambda_s,\alpha_{s+1}+j\beta_{s+1},\alpha_{s+1}-j\beta_{s+1},\ldots,\alpha_n+j\beta_n,\alpha_n-j\beta_n)$$

$$B = \begin{bmatrix} b_1 \\ \vdots \\ b_s \\ \alpha_{s+1}+j\beta_{s+1} \\ \alpha_{s+1}-j\beta_{s+1} \\ \vdots \\ \alpha_n+j\beta_n \\ \alpha_n-j\beta_n \end{bmatrix};\quad C = \begin{bmatrix} c_1 & \cdots & c_s & \eta_{s+1}+j\mu_{s+1} & \eta_{s+1}-j\mu_{s+1} & \cdots & \eta_n+j\mu_n & \eta_n-j\mu_n \end{bmatrix}$$

which is a complex model, the function bk_diag transforms this model to the following block-diagonal form

$$x_b(k+1) = A_b x_b(k) + B_b u(k);$$
$$y(k) = C_b x_b(k) + Du(k)$$

where

$$A_b = \begin{bmatrix} \lambda_1 & & & & & & & \\ & \ddots & & & & & & \\ & & \lambda_s & & & & & \\ & & & \alpha_{s+1} & \beta_{s+1} & & & \\ & & & -\beta_{s+1} & \alpha_{s+1} & & & \\ & & & & & \ddots & & \\ & & & & & & \alpha_n & \beta_n \\ & & & & & & -\beta_n & \alpha_n \end{bmatrix}$$

$$B_b = \begin{bmatrix} b_1 \\ \vdots \\ b_s \\ 2\alpha_{s+1} \\ -2\beta_{s+1} \\ \vdots \\ 2\alpha_n \\ -2\beta_n \end{bmatrix}; \quad C_b = [c_1 \quad \cdots \quad c_s \quad \eta_{s+1} \quad \mu_{s+1} \quad \cdots \quad \eta_n \quad \mu_n]$$

All the variables in this block-diagonal form are real rather than complex as in the diagonal form. This function is used in conjunction with function modal to reduce an identified model (stable or unstable) to a stable real block diagonal model for numerical simulations to compare with real data.

Example:

```
rand('normal');
n=7;
am=rand(n,n);bm=rand(n,2);cm=rand(1,n);
[v,lambda]=eig(am);
lambda=diag(lambda);
bm=v\bm; cm=cm*v;
  [lambda,k]=sort(lambda);
  bm=bm(k,:);cm=cm(:,k);
[a,b,c]=bk_diag(lambda,bm,cm)

a =
    1.0188         0         0         0         0         0         0
         0   -0.6935   -1.1752         0         0         0         0
         0    1.1752   -0.6935         0         0         0         0
         0         0         0    0.4642   -1.8936         0         0
         0         0         0    1.8936    0.4642         0         0
         0         0         0         0         0    1.9055   -0.8629
         0         0         0         0         0    0.8629    1.9055
b =
   -3.5006   -0.9259
   -0.2835    1.3741
    1.6978    2.7562
    3.8881    0.6494
   -1.5251   -2.3995
   -3.3324   -0.0417
    2.1830    0.0425

c =
   -0.0730   -0.0421   -0.2797   -0.2737    0.3682   -1.0026   -0.4538
```

See also:

okid_b, okid_fb, modal

Purpose:

Compute the matrix block transposition.

Synopsis:

[*at*]=block_tr(*nrow,nblock,ncol,a,flag*)

Description:

The function performs a 2-way block transposition depending on the input matrix. When *flag* is set to 1 the wide matrix

$$a = [y_0 \quad y_1 \quad \cdots \quad y_{nblock}]$$

is block transposed to

$$at = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{nblock} \end{bmatrix}$$

where each block y_i is a matrix of dimension *nrow* x *ncol*. The reverse operation is obtained when inputing a tall matrix and *flag* = 0.

Example:

```
y0 = [0 1; 2 3];
y1 =[4 5; 6 7];
a=[y0 y1]
a=
   0   1   4   5
   2   3   6   7
[at]=block_tr(2,2,2,a,1)
at=
   0   1
   2   3
   4   5
   6   7
a=block_tr(2,2,2,at,0)
a=
   0   1   4   5
   2   3   6   7
```

See also:

mar_com

Purpose:

Compute pulse response samples using FFT.

Synopsis:

[*ys*]=cpulse(*y*,*u*,*r*,*m*);

Description:

The function cpulse calculates the unit pulse response samples (Markov parameters) from input and output time histories by using a frequency domain approach. The input time histories are required to be sufficiently rich (e.g. random inputs). The system input and output histories *u(t)* and *y(t)* must be stored as follows

$$u=\begin{bmatrix} u_{11}(0) & \cdots & u_{r1}(0) & \cdots & u_{1s}(0) & \cdots & u_{rs}(0) \\ u_{11}(1) & & u_{r1}(1) & & u_{1s}(1) & & u_{rs}(1) \\ u_{11}(2) & \cdots & u_{r1}(2) & \cdots & u_{1s}(2) & \cdots & u_{rs}(2) \\ \vdots & & \vdots & & \vdots & & \vdots \\ u_{11}(l-1) & \cdots & u_{r1}(l-1) & \cdots & u_{1s}(l-1) & \cdots & u_{rs}(l-1) \end{bmatrix}$$

$$y=\begin{bmatrix} y_{11}(0) & \cdots & y_{m1}(0) & \cdots & y_{1s}(0) & \cdots & y_{ms}(0) \\ y_{11}(1) & & y_{m1}(1) & & y_{1s}(1) & & y_{ms}(1) \\ y_{11}(2) & \cdots & y_{m1}(2) & \cdots & y_{1s}(2) & \cdots & y_{ms}(2) \\ \vdots & & & & & & \\ y_{11}(l-1) & \cdots & y_{m1}(l-1) & \cdots & y_{1s}(l-1) & \cdots & y_{ms}(l-1) \end{bmatrix}$$

where r is the number of inputs and m is the number of outputs, and $u_{ij}(t)(y_{ij}(t))$ is the i-th input (output) of the j-th test at discrete time t. The number of experiments s should be greater or equal to r, integer l is required to be even. The ouput of this function is the pulse response histories *ys*.

Example:

The example is to compute the pulse response samples from 5 data sets of random inputs for a single-input and single-output second-order system with sampling interval dt=0.2, pt=512 samples, natural frequency $\omega_n = 1$ and damping factor $\zeta = 0.1$,

```
a=[0  1;-1 -0.2];b=[0; 1];c=[1 0];d=0;
pt=512;dt=0.2;rand('normal');
t=[dt:dt:pt*dt]';
u=rand(pt,5);
for i=1:5,
yt(:,i)=lsim(a,b,c,d,u(:,i),t);end
[y1]=cpulse(yt,u,1,1);
plot(y1),title('Pulse response')
```

Algorithm:

The function cpulse uses a frequency domain approach to compute the pulse response samples (Markov parameters) from output time histories generated from rich input signals. First the *FFT* is applied to calculate the discrete frequency response functions of input $u(t)$ and output $y(t)$. Then the input and output frequency response functions are used to compute the discrete transfer functions $G(z)$, and the pulse response samples are the inverse *FFT* of $G(z)$.

$$Y(z)=G(z)U(z), \qquad Y(t)=\mathrm{FFT}^{-1}(G(z))$$

See also:

pulse

Purpose:

Identify a state-space model from pulse response samples (Markov parameters).

Synopsis:

[*a,b,c,d,sg,eg,mh*]=era(*y,m,r,n,nm,dt*);
[*a,b,c,d,sg,eg,mh*]=eradc(*y,m,r,n,nm,dt*);

Description:

The function era identifies a state-space model of a multi-input and multi-output linear, time-invariant system from pulse response samples (Markov parameters). The pulse response samples are stored as

$$y=\begin{bmatrix} y_{11}(0) & \cdots & y_{m1}(0) & \cdots & y_{1r}(0) & \cdots & y_{mr}(0) \\ y_{11}(1) & \cdots & y_{m1}(1) & \cdots & y_{1r}(1) & \cdots & y_{mr}(1) \\ \vdots & & \vdots & & \vdots & & \vdots \\ y_{11}(l-1) & \cdots & y_{m1}(l-1) & \cdots & y_{1r}(l-1) & \cdots & y_{mr}(l-1) \end{bmatrix}$$

where $y_{ij}(t)$ is the i-th output at discrete time t due to a unit pulse at the j-th input. The system to be identified has r inputs and m outputs. The identified model order is chosen as n. When n is set to zero, user's inputs is required on-line to specify the desired order of an identified model, based on the singular values of a Hankel matrix. Scalar *nm* specifies the number of sample shifts for forming the rows of a Hankel matrix (see the Algorithm section for definition). Integer $nm \times m$ should be greater than the model order n. Scalar *dt* specifies the data sampling interval. [*A,B,C,D,sg,eg,mh*]=era(*y,m,r,n,nm,dt*) returns an n-th order linear, time-invariant identified discrete model:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

Matrix *eg* contains modal parameters of the identified model with damping ratios (%) in the first column and frequencies (Hz) in the second column. The third column of *eg* gives the eigenvalues of the corresponding continuous-time model. Note that the identified discrete model can be easily transformed to a continuous-time model. Vector *sg*, whose elements are singular values of the Hankel matrix, can be used as reference to choose the model order n. The first column of matrix *mh* gives the normalized singular contribution of each identified mode in matrix *eg* to the pulse response samples whereas the second column gives the modal amplitude coherence. The maximum singular value is chosen to normalize the first column of matrix *mh*. These normalized singular values are used to weight the importance of the individual mode to the pulse response samples. Each element in the second column of mh_i (the i-th element of *mh*) is between 0 and 1; $mh_i \rightarrow 1$ indicates that the identified mode eg_i (the i-th eigenvalue) is reliable.

The function eradc is similar to era, but it uses the ERA data correlation method [Juang88] to identify the system model. For a long data lenght, eradc is recommended for use because it takes less memory and computational time to solve for singular values of the Hankel

matrix. In eradc, the size of the data correlation matrix has been minimized to save time in computing its singular values.

Examples:

Example 1:

i) Calculate pulse response samples of a single-input and single-output second-order system with sampling interval dt=0.2, natural frequency $\omega_n = 1$, and damping factor $\zeta = 0.1$.

ii) Use era interactively to identify a model from the pulse response samples from (i) and transform the identified discrete model to a continuous-time model.

iii) Plot the error between the pulse response samples from (i) and the pulse response samples of the identified model.

iv) Display eigenvalue matrix *eg*, singular value vector *sg* and modal amplitude coherence matrix *mh*.

```
a=[0 1;-1 -0.2]; b=[0;1];c=[1 0];d=0;
pt=100;dt=0.2;t=[dt:dt:pt*dt]';
u=zeros(pt,1);u(1,1)=1.0;
y=lsim(a,b,c,d,u,t);
clg
[a1,b1,c1,d1,sg,eg,mh]=era(y,1,1,0,20,dt);
y1=dlsim(a1,b1,c1,d1,u);
error=y-y1;
clg
plot(error),title('Pulse response error')
pause
eg,mh,sg(1:10)
```

Example 2:

i) Calculate pulse response samples from 6 data sets for a single-input and single-output second order system with sampling interval dt=0.4, l=512 samples, natural frequency $\omega_n = 1$, damping factor $\zeta = 0.1$, and noise standard deviation 0.04.

ii) Use eradc interactively to identify a model from the pulse response samples from (i) and transform the identified discrete model to a continuous-time model.

iii) Display eigenvalue matrix *eg*, singular value vector *sg*, and modal amplitude coherence matrix *mh* of the eradc identified model.

iv) Plot the pulse response samples of the original model and the eradc-identified model.

```
a=[0 1;-1 -0.2]; b=[0;1];c=[1 0];d=0;
dt=0.4;pt=512;t=[dt:dt:pt*dt]';
rand('normal');
u=rand(pt,6);
for i=1:6,
yt(:,i)=lsim(a,b,c,d,u(:,i),t);end
y=yt+0.04*rand(pt,6);
yi=cpulse(y,u,1,1);
clg
[a1,b1,c1,d1,sg,eg,mh]=eradc(yi,1,1,0,50,dt);
eg,mh,sg(1:10)
clear u
u=zeros(pt,1);u(1,1)=1.0;
y1=dlsim(a1,b1,c1,d1,u);
y0=lsim(a,b,c,d,u,t);
y=[y0 y1];
clg
plot(y),title('Pulse response')
pause
```

Algorithm:

The function era uses the Eigensystem Realization Algorithm (ERA) from [Juang85], which uses Markov parameters (pulse responses) to form the Hankel matrix

$$H(j-1)=\begin{bmatrix} Y_j & Y_{j+1} & \cdots & Y_{j+\beta} \\ Y_{j+1} & Y_{j+2} & \cdots & Y_{j+\beta+1} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{j+\gamma} & Y_{j+\gamma+1} & \cdots & Y_{j+\gamma+\beta} \end{bmatrix}$$

$$Y_i=\begin{bmatrix} y_{11}(i) & \cdots & y_{1r}(i) \\ y_{21}(i) & \cdots & y_{2r}(i) \\ \vdots & & \vdots \\ y_{m1}(i) & \cdots & y_{mr}(i) \end{bmatrix}$$

where Y_i is the i-th Markov parameter and y_{ij} is the i-th output at discrete time t due to a unit pulse at the j-th input. From the measurement Hankel matrix, ERA uses the SVD of $H(0)$, $H(0)=U\Sigma V^T$, to identify a k-th order discrete state-space model as

$$A_k=\Sigma_k^{-1/2}U_k^T H(1)V_k\Sigma_k^{-1/2}$$
$$B_k=\Sigma_k^{1/2}V_k^T E_r$$
$$C_k=E_m^T U_k\Sigma_k^{1/2}$$
$$D_k=Y(0)$$

where matrix Σ_k is the upper left hand $k\times k$ partition of Σ containing the k largest singular values along the diagonal. Matrices U_k and V_k are obtained from U and V by retaining only the k columns of singular vectors associated with the k singular values. Matrix E_m is a

matrix of appropriate dimension having m columns, all zero except that the top $m \times m$ partition is an identity matrix. E_r is defined analogously.

The function eradc uses a special case of the ERA/DC algorithm from [Juang88]. It starts with the Hankel matrices $H(0)$ and $H(1)$ to generate the block correlation matrices $R(i) = H(i)H(0)^T$. The SVD of $R(0) = U \Sigma V^T$, is applied to identify a kth order model as

$$A_k = \Sigma_k^{-1/2} U_k^T R(1) V_k \Sigma_k^{-1/2}$$

$$B_k = \Sigma_k^{1/2} V_k^T H(0) E_r$$

$$C_k = E_m^T U_k \Sigma_k^{1/2}$$

$$D_k = Y(0)$$

The function eradc uses the block correlation matrices $R(i) = H(i)^T H(0)$ if this matrix is smaller than $R(i) = H(i)H(0)^T$.

Limitations:

The most time consuming step in each algorithm is the singular value decomposition. The number of floating point operations for SVD are roughly a cubic function of the matrix dimension. Also the SVD of a large matrix needs a lot of memory. The size of the SVD matrix in era and eradc is $(nm \times m) \times (\ell - nm - 1 \times r)$ and $(nm \times m) \times (nm \times m)$ [or $(\ell - nm) \times (\ell - nm)$ if this is smaller] respectively where ℓ is the length of the data. In era, the column number of the Hankel matrix may be very large if the data length is large. However, the column number in eradc can be chosen as large as desired, because the size of the data correlation matrix depends only on the number of rows.

See also:

cpulse, okid, okid_b, okid_fb

References:

[1] Juang, J. N. and Pappa, R. S., "An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 5, 1985, pp. 620-627.

[2] Juang, J. N., Cooper, J. E., and Wright J. R., "An Eigensystem Realization Algorithm Using Data Correlation (ERA/DC) for Modal Parameter Identification," *Control Theory and Advanced Technology*, Vol. 4, No. 1, pp. 5-14, 1988.

[3] Lew, J. S., Juang, J. N. and Longman, R. W., "Comparison of Several System Identification Methods for Flexible Structures," *Proceedings of the 32nd Structures, Structural Dynamics, and Materials Conference*, Baltimore, MD, April 1991, pp. 2304-2318.

[4] Juang, J. N., "Mathematical Correlation of Modal Parameter Identification Methods Via System Realization Theory," *International Journal of Analytical and Experimental Modal Analysis*, Vol. 2, No. 1, Jan. 1987, pp.1-18.

Purpose:

Plot the transfer function representation of a discrete time system.

Synopsis:

[*mag,phase*]=freq_pt(*a,b,c,d,p,dt,iu*)

Description:

Given a discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

the function computes the transfer function representation given by

$$G(e^{j\omega\Delta T}) = C(e^{j\omega\Delta T}I - A)^{-1}B + D$$

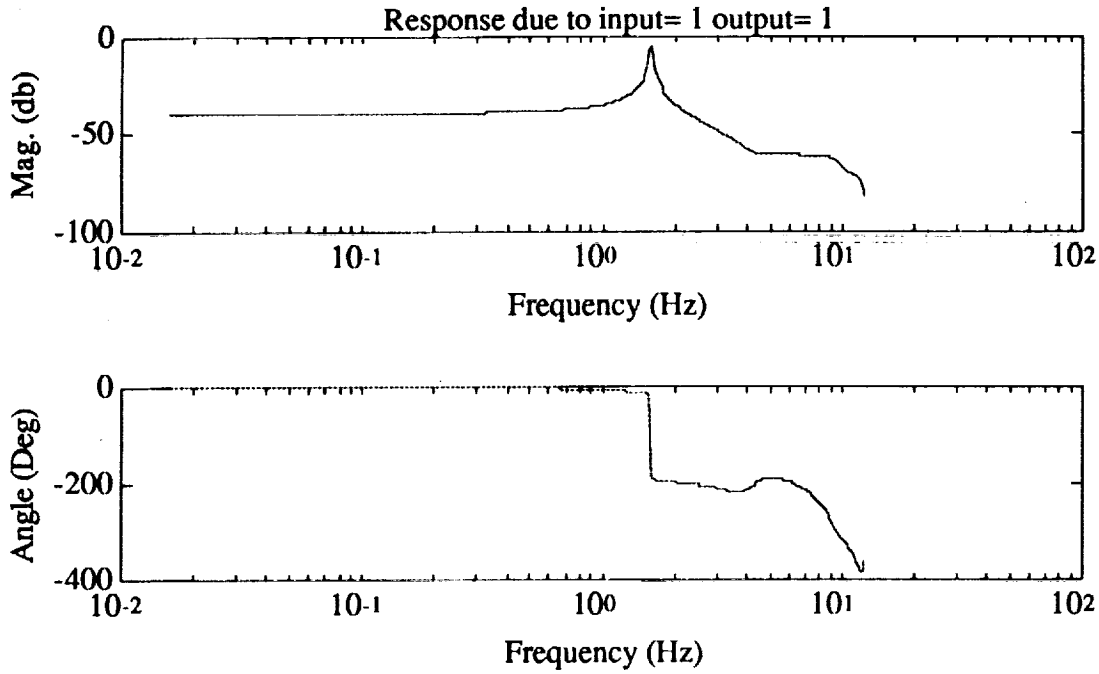
The parameter *p* determines the number of spectral lines plotted, *dt* is the sample time, *iu* is the input number to be plotted. The function [mag,phase]=freq_pt(*A,B,C,D,p,dt,iu*) returns the magnitude (mag) and phase plots of the transfer functions for the *iu*-th input. The function prints the value for the Nyquist frequency. The user is prompted for lowest frequency to plot, and for the upper frequency bound. The upper frequency bound must be given in terms of percentage of the Nyquist frequency. These values are used to define the frequency range. The actual plot scale may be slightly different because it is determined by the plotting function.

Example:

```
a=[0 1;-100 -0.002]; b=[0;1];c=[1 0];d=0;
dt=0.04;pt=1024;t=[dt:dt:pt*dt]';
rand('normal');
u=rand(pt,1);
y=lsim(a,b,c,d,u,t);
y=y+0.0042*rand(pt,1);  %3 percent noise
[a,b,c,d,m]=okid(1,1,dt,u,y,'batch',10);
[mag,phase]=freq_pt(a,b,c,d,300,dt,1);
```

Nyquist frequency (Hz) is =: 12.5

Enter lower frequency to plot (Hz)=: 0.1


Response due to input= 1 output= 1
Mag. (db)
0
-50
-100
10-2
10-1
100
101
102
Frequency (Hz)
Angle (Deg)
0
-200
-400
10-2
10-1
100
101
102
Frequency (Hz)

See also:

Purpose:

Form a Hankel matrix from a sequence of Markov parameters.

Synopsis:

[*D*,*H*]=hankl(*Markov*,*p*)

Description:

Given a discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the pulse response samples are typically stored as

$$Y_p = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_{ri} \end{bmatrix}$$

where y_i = response samples due to a unit pulse at the i-th input. Each y_i has m columns and ℓ rows where ℓ is the length of data. The pulse response samples can be rearranged by using function p2m to the following sequence of system Markov parameters

$$\begin{aligned} Y_s &= \begin{bmatrix} Y_0 & Y_1 & Y_2 & \cdots & Y_{\ell-2} \end{bmatrix} \\ &= \begin{bmatrix} D & CB & CAB & \cdots & CA^{\ell-2}B \end{bmatrix} \end{aligned}$$

Function [*D*,*H*]=hankl(*Ys*,*p*) return the transmission matrix D and a Hankel matrix defined as

$$H(0) = \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_{\ell-p-1} \\ Y_2 & Y_3 & \cdots & Y_{\ell-p} \\ \vdots & \vdots & \ddots & \vdots \\ Y_p & Y_{p+1} & & Y_{\ell-2} \end{bmatrix}$$

Note that all the data pass into the function are used to form the Hankel matrix. The inner products are used in matrix multiplication to reduce computational time. The size of the matrix is $p \times (\ell - p - 1)$. The longer the data length is, the larger the number of rows of the matrix becomes. This function is used in era for identification of system matrices.

See also:

era, eradc, m2p

Purpose:

Form a data correlation matrix from a sequence of Markov parameters.

Synopsis:

[*D*,*R*]=hankl(*Markov*,*p*)

Description:

Given a discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the pulse response samples are typically stored as

$$Y_p = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_{ni} \end{bmatrix}$$

where y_i = response samples due to a unit pulse at the i-th input. Each y_i has m columns and ℓ rows where ℓ is the length of data. The pulse response samples can be rearranged by using function p2m to the following sequence of system Markov parameters

$$\begin{aligned} Y_s &= \begin{bmatrix} Y_0 & Y_1 & Y_2 & \cdots & Y_{\ell-2} \end{bmatrix} \\ &= \begin{bmatrix} D & CB & CAB & \cdots & CA^{\ell-2}B \end{bmatrix} \end{aligned}$$

From this sequence of Markov parameters, define two Hankel matrix as

$$H_0 = \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_{\ell-p-2} \\ Y_2 & Y_3 & \cdots & Y_{\ell-p-1} \\ \vdots & \vdots & \ddots & \vdots \\ Y_p & Y_{p+1} & & Y_{\ell-3} \end{bmatrix}; \quad H = \begin{bmatrix} \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_{\ell-p-2} \\ Y_2 & Y_3 & \cdots & Y_{\ell-p-1} \\ \vdots & \vdots & \ddots & \vdots \\ Y_p & Y_{p+1} & \cdots & Y_{\ell-3} \end{bmatrix} \\ \begin{bmatrix} Y_{p+1} & Y_{p+2} & \cdots & Y_{\ell-2} \end{bmatrix} \end{bmatrix};$$

Function [*D*,*R*]=hankldc(*Ys*,*p*) returns the transmission matrix D and a data correlation matrix R defined as

$$R = HH_0^T$$

Note that all the data pass into the function are used to form the data correlation matrix. The size of this matrix is $(p+1)\times p$ which is independent of the length of the data. This function is used in eradc for identification of system matrices.

See also:

era, eradc, m2p

Purpose:

Identify an Observer/Kalman filter gain matrix from test data for a discrete model to whiten the stochastic residual.

Synopsis:

[*G, Gmar*]=k_abcd(*A,B,C,D,u,y,p*)

Description:

The function k_abcd solves for an observer/Kalman filter gain matrix of a system in the form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) - G[y(k) - \hat{y}(k)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

where $\hat{x}(k)$ is the estimate of the state *x(k)* and $\hat{y}(k)$ is the estimate of *y(k)*. The system has *m* outputs, *r*-inputs, and time samples *dt* apart. The input/output time histories are stored as column matrices. Initially an estimate of the desired observer Markov parameter number *p* must be given for whitening the residuals. It is suggested that *p* is chosen such that the product $p \times m$ is greater than or equal to the order of the system. The identified matrix, *G*, and observer gain Markov parameters, *Gmar* (stored as a column matrix), are returned to the main program. See references for detailed information. To identify a stable observer which whitens the residual between the real output *y(k)* and the estimated output $\hat{y}(k)$, the system matrices *A, B, C, D,* must be reasonably close to the true ones.

Example:

From the test data of a truss structure, use the function to compute a set of system matrices and an observer gain matrix. The order of the system is determined automatically by the function eradc. The residual is further whitened by using the function k_abcd to modify the observer gain matrix. The function y_esti is used to compute the estimated outputs which are then subtracted from the real output to obtain the residual. The following is output taken from a typical run.

```
load sample
[pt,i]=size(u);
[a,b,c,d,m]=okid(n,r,dt,u,y,'batch',20);
[time,y_e]=y_esti(a,b,c,d,m,u,y,dt,pt);
[m1,Cake]=k_abcd(a,b,c,d,u,y,20);
[time,y_e1]=y_esti(a,b,c,d,m1,u,y,dt,pt);
clear a b c d u
res=y-y_e;
[yvt, vvt]=yycovar(n,2,pt,res,1);
clear res
vvt
vvt =
  2.1068e+02  -2.5208e+01   1.3304e+02   7.4350e+00
 -2.5208e+01   1.6014e+02   1.6889e+01   7.4626e+01
  1.3304e+02   1.6889e+01   2.1061e+02  -2.5116e+01
  7.4350e+00   7.4626e+01  -2.5116e+01   1.6015e+02
```

```
res1=y-y_e1;
[yvt, vvt1]=yycovar(res1,2,1);
clear res1
vvt1
vvt1=
  1.2454e+02  -4.8890e+01  -1.9179e+00  -2.9991e+00
 -4.8890e+01   1.3218e+02   2.6610e-01  -1.4728e+01
 -1.9179e+00   2.6610e-01   1.2469e+02  -4.8836e+01
 -2.9991e+00  -1.4728e+01  -4.8836e+01   1.3220e+02
```

The matrix

$$vvt = E\left[\begin{bmatrix} res_1(1) \\ res_2(1) \\ res_1(2) \\ res_2(2) \end{bmatrix} \begin{bmatrix} res_1(1) & res_2(1) & res_1(2) & res_2(2) \end{bmatrix}\right]$$

is the expected value of the auto- and cross-correlation of the residual obtained from the okid identified observer, whereas

$$vvt1 = E\left[\begin{bmatrix} res1_1(1) \\ res1_2(1) \\ res1_1(2) \\ res1_2(2) \end{bmatrix} \begin{bmatrix} res1_1(1) & res1_2(1) & res1_1(2) & res1_2(2) \end{bmatrix}\right]$$

is obtained from the k_abcd identified observer gain using the okid identified system matrices. It is obvious that the residual *res1* is whiter than *res*.

Algorithm:

The algorithm used here is similar to that for the okid which identifies a set of system matrices, *A, B, C, D,* as well as an observer gain matrix, *G*. For given system matrices *A, B, C, D,* the deterministic component of the output can be subtracted out. The observer gain is obtained by whitening the remaining residual. For details see references.

See also:

okid, okid_b, okid_fb, yycovar

References:

[1] Chen, C. W., Huang, J.-K., and Juang, J.-N., "Identification of Linear Stochastic Systems Through Projection Filters," *presented at the AIAA 33rd Structures, Structural Dynamics & Materials Conference*, 1992, Paper No. AIAA-92-2520.

[2] Juang, J.-N., Chen, C. W., and Phan, M., "Estimation of Kalman Filter Gain from Output Residuals" *NASA Technical Memorandum TM-107603*, Langley Research Center, Hampton, VA., March 1992.

Purpose:

Rearrange a Markov parameters sequence in the form of pulse response time histories.

Synopsis:

Y_p=m2p(Y_s,r)

Description:

Given a discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the sequence of system Markov parameters is defined as

$$Y_s = \begin{bmatrix} D & CB & CAB & \cdots & CA^{\ell-2}B \end{bmatrix}$$

The pulse response samples are typically stored as

$$Y_p = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_r \end{bmatrix}$$

where y_i= response samples due to a unit pulse at the i-th input. Each y_i has m columns and ℓ rows where ℓ is the length of data. The sequences Y_S and Y_p are equivalent in the sense that both represent pulse response samples. The function converts Y_S to Y_p. Note that Y_p is the sequence used in the eradc and era functions.

If the combined system/observer Markov parameters (see function mar_com) are used, the input matrix B becomes [B G] and the sequence Y_s becomes

$$Y_s = \begin{bmatrix} D & C[B & G] & CA[B & G] & \cdots & CA^{\ell-2}[B & G] \end{bmatrix}$$

where the input number changes from r to $r+m$. The pulse response samples in Y_p becomes

$$Y_p = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_r & y_{r+1} & y_{r+2} & \cdots & y_{r+m} \end{bmatrix}$$

with m additional $\ell \times m$ matrices due to the observer gain matrix G which is treated as an input matrix. The first r matrices represnt the system pulse response samples and the last m matrices mean the observer gain pulse response samples.

See also:

mar_com, okid, okid_b, okid_fb, p2m

Purpose:

Recover the system Markov parameters from a set of observer Markov parameters.

Synopsis:

H=mar_com(*ybar,r,n_markov*)
H=mar_sep(*y0b,y1b,D̄,r,n_markov*)

Description:

Given an observer of the form

$$x(k+1) = \bar{A}x(k) + \begin{bmatrix} \bar{B} & -G \end{bmatrix} \begin{Bmatrix} u(k) \\ y(k) \end{Bmatrix}$$

$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the sequence of observer Markov parameters previously computed is passed to the function mar_com with

$$\begin{bmatrix} D & C\{\bar{B} & -G\} & C\bar{A}\{\bar{B} & -G\} & \cdots & C\bar{A}^{p}\{\bar{B} & -G\} \end{bmatrix}$$

or to the function mar_sep with

$$y0b = [C\bar{B} \quad C\bar{A}\bar{B} \quad C\bar{A}^2\bar{B} \quad \cdots \quad C\bar{A}^{n\_markov-1}\bar{B}]$$

and

$$y1b = [CG \quad C\bar{A}G \quad C\bar{A}^2G \quad \cdots \quad C\bar{A}^{n\_markov-1}G]$$

There is usually a small number, p, of nonzero observer parameters which is less than *n_markov*. The observer matrices are related to the system matrices by $\bar{A} = A + GC$, $\bar{B} = B + GD$, and D is the direct transmission term which is the same for system and observer. The function computes recursively *n_markov* parameters of the original system and puts them in the form

$$H = \begin{bmatrix} \{D & 0\} & C\{B & -G\} & CA\{B & -G\} & \cdots & CA^{p}\{B & -G\} \end{bmatrix}$$

Algorithm:

The parameters are computed using the following recursive formula

$$CA^k[B \quad -G] = C\bar{A}^k\bar{B} - \sum_{i=0}^{k-1} C\bar{A}^i GCA^{k-i-1}[B \quad -G] - C\bar{A}^k G[D \quad 0]$$

for k=1,2,...*n_markov*

Example:

```
a=[0 -0.16; 1 -1]; b=[0 1]';
c=[0 1]; d=0; G=[0.16 1]';
abar=a+G*c;
bbar=[b+G*d -G];
ybar=[d c*bbar c*abar*bbar];
[H]=mar_com(ybar,1,2)
bg=[b -G];
Hs=[d 0 c*bg c*a*bg]

H =

     0     0   1.0000   -1.0000   -1.0000    0.8400


Hs =

     0     0   1.0000   -1.0000   -1.0000    0.8400
```

See also:

okid, okid_b, okid_fb, okid_p

Purpose:

Compute a specified number of the system, observer, and controller Markov parameters from observer/controller Markov parameters.

Synopsis:

[*H*]=mar_oc(*Ybar,r,m,Ntotal*)

Description:

From a sequence of observer/controller Markov parameters arranged in *Ybar* in the following order

$$\begin{bmatrix} D \\ 0 \end{bmatrix}, \quad and \quad \begin{bmatrix} C \\ -F \end{bmatrix}(A+GC)^{k-1}[B+GD \quad -G], \quad k=1,\ 2,\ ...,\ p$$

the function computes the following sequence of system, observer, and controller Markov parameters

$$\begin{bmatrix} D \\ 0 \end{bmatrix}, \begin{bmatrix} C \\ -F \end{bmatrix}[B \quad -G], \begin{bmatrix} C \\ -F \end{bmatrix}A[B \quad -G], \ ... \ \begin{bmatrix} C \\ -F \end{bmatrix}A^{Ntotal-1}[B \quad -G]$$

that are arranged in the matrix *H* in this order. The scalar *r* denotes the number of inputs, and *m* the number of outputs. *Ntotal* specifies the number of Markov parameters in *H* to be returned to the user. For background information, the sequence of observer/controller Markov parameters are computed from closed-loop excitation data, and the function mar_oc is then used to unscramble this sequence to obtain the system, observer, and controller Markov parameters.

Example:

```
load xsamp712
ue(1:600)=[];y(1:600)=[];u(1:600)=[];
[Ybar]=arxc(y,-u,ue,30,300);
[H]=mar_oc(Ybar,1,1,30)

H =

 Columns 1 through 7

 -0.0689   0.1778   0.8012   0.0572   1.0144   0.1181   0.5774
  0.0351   0.0422  -0.6740  -0.1063  -0.6234  -0.0419  -0.8527

 Columns 8 through 14

  0.0915   0.4184  -0.1442   0.2114   0.0086   0.0914  -0.1890
 -0.1076  -0.6742  -0.0692  -0.6106   0.0527  -0.3847  -0.0236
```

Columns 15 through 21

```
-0.1854  -0.0041  -0.3831  -0.1030  -0.5743  -0.1404  -0.6633
-0.1319   0.1225   0.2654   0.0310   0.5776   0.1159   0.8443
```

Columns 22 through 28

```
-0.0318  -0.7223  -0.0736  -0.7292   0.0206  -0.7679  -0.1022
 0.1240   0.9964   0.0591   1.0904   0.0730   1.1074   0.0075
```

Columns 29 through 35

```
-0.5967   0.0881  -0.4962  -0.0521  -0.2350   0.0872  -0.1258
 1.0717   0.0602   0.8314  -0.0597   0.5758   0.0144   0.2073
```

Columns 36 through 42

```
 0.0442   0.0852   0.0671   0.2845   0.1206   0.5419   0.0896
-0.1010  -0.0944  -0.0799  -0.4433  -0.1145  -0.7440  -0.1510
```

Columns 43 through 49

```
 0.7482   0.1712   0.8393   0.0375   0.9105   0.0575   0.8713
-1.0425  -0.1319  -1.2376  -0.1653  -1.2999  -0.0688  -1.2899
```

Columns 50 through 56

```
 0.0530   0.8104   0.0221   0.6379   0.0741   0.3764  -0.0248
-0.0444  -1.1347  -0.0149  -0.8797   0.0075  -0.5058  -0.0365
```

Columns 57 through 61

```
 0.1169  -0.0111  -0.1443  -0.1061  -0.3409
-0.0961   0.0786   0.3044   0.0666   0.7439
```

Algorithm:

The system, observer, and controller Markov parameters are computed from the observer/controller Markov parameters by a set of recursive equations. If more than p number of system, observer, and controller Markov parameters are required to be solved, the extra observer/controller Markov parameters are set to zero. For further details, see references.

See also:

arxc, mar_yoc, ocid, ryucovar, separate

Reference:

[1] Juang, J. N. and Phan, M., "Identification of System, Observer, and Controller from Closed-Loop Experimental Data," *Presented at the AIAA Guidance, Navigation, and Control Conference,* Hilton Head, South Carolina, Aug. 10-12, 1992.

Purpose:

Compute system, observer, and controller Markov parameters directly from feedback control input *ufb*, additive input excitation *ue*, and output response *y*.

Synopsis:

[*ocs*]=mar_yoc(*y,ufb,ue,p,truncate,Ntotal*)

Description:

The function mar_yoc solves for the Markov parameters

$$\begin{bmatrix} D \\ 0 \end{bmatrix}, \begin{bmatrix} C \\ -F \end{bmatrix}[B \quad -G], \begin{bmatrix} C \\ -F \end{bmatrix}A[B \quad -G], \ldots \begin{bmatrix} C \\ -F \end{bmatrix}A^{Ntotal-1}[B \quad -G]$$

from feedback control input *ufb*, additive input excitation *ue*, and output response *y*. The data is stored as column matrices. For example, for a system with *m* outputs, the closed-loop output data matrix *y* contains *m* columns and as many rows as the number of data points available. The data matrix *ufb* contains the feedback control signal, *ue* contains the additive excitation input signal. The number *p* denotes the number of observer/controller Markov parameters to be solved. The number *truncate* specifies the number of data points to be deleted prior to application of the algorithm. This value is equal to the number of time steps that is expected for the existing observer to converge. *Ntotal* is the total number of Markov parameters to be solved for from *p* identified observer/controller Markov parameters. This is done by setting the extra observer/controller Markov parameters to be zero.

$$\begin{bmatrix} C \\ -F \end{bmatrix}A^{k-1}[B \quad -G] = 0 \quad , \quad k = p+1, \; p+2, \; \ldots$$

These Markov parameters are in the form ready to be used to obtain a state space realization of the system matrices, observer gain, and controller gain.

Example:

An example data file is contained in the file xsamp712.

```
load xsamp712
[ocs]=mar_yoc(y,-u,ue,3,200,4)

ocs =

 Columns 1 through 7

 -0.0689   0.1778   0.8012   0.0572   1.0144   0.1181   0.5774
  0.0351   0.0422  -0.6740  -0.1063  -0.6234  -0.0419  -0.8527
```

Columns 8 through 14

```
 0.0915   0.4184  -0.1442   0.2114   0.0086   0.0914  -0.1890
-0.1076  -0.6742  -0.0692  -0.6106   0.0527  -0.3847  -0.0236
```

Columns 15 through 21

```
-0.1854  -0.0041  -0.3831  -0.1030  -0.5743  -0.1404  -0.6633
-0.1319   0.1225   0.2654   0.0310   0.5776   0.1159   0.8443
```

Columns 22 through 28

```
-0.0318  -0.7223  -0.0736  -0.7292   0.0206  -0.7679  -0.1022
 0.1240   0.9964   0.0591   1.0904   0.0730   1.1074   0.0075
```

Columns 29 through 35

```
-0.5967   0.0881  -0.4962  -0.0521  -0.2350   0.0872  -0.1258
 1.0717   0.0602   0.8314  -0.0597   0.5758   0.0144   0.2073
```

Columns 36 through 42

```
 0.0442   0.0852   0.0671   0.2845   0.1206   0.5419   0.0896
-0.1010  -0.0944  -0.0799  -0.4433  -0.1145  -0.7440  -0.1510
```

Columns 43 through 49

```
 0.7482   0.1712   0.8393   0.0375   0.9105   0.0575   0.8713
-1.0425  -0.1319  -1.2376  -0.1653  -1.2999  -0.0688  -1.2899
```

Columns 50 through 56

```
 0.0530   0.8104   0.0221   0.6379   0.0741   0.3764  -0.0248
-0.0444  -1.1347  -0.0149  -0.8797   0.0075  -0.5058  -0.0365
```

Columns 57 through 61

```
 0.1169  -0.0111  -0.1443  -0.1061  -0.3409
-0.0961   0.0786   0.3044   0.0666   0.7439
```

Algorithm:

The function first computes the observer/controller Markov parameters for the closed-loop system. Then, from the identified observer/controller Markov parameters, the individual system, observer, controller Markov parameters are computed and arranged in the specified form which is ready to be used for realization. The function is a combination of the functions arxc and mar_oc.

See also:

arxc, mar_oc, ocid, ryucovar, separate

Purpose:

match system eigenvalues from identified forward and backward model and provide reduced forward and backward model in modal coordinates.

Synopsis:

[*lamdaf,bmf,cmf,msv_f,lamdab,bmb,cmb,msv_b*]=match(*af,bf,cf,ab,bb,cb*)

Description:

A typical forward state space model has the form

$$x(k+1) = (A+GC)x(k) + \begin{bmatrix} B+GD & -G \end{bmatrix} \begin{Bmatrix} u(k) \\ y(k) \end{Bmatrix}$$

$$y(k) = Cx(k) + Du(k)$$

with m outputs, r inputs, and time samples dt apart. Here A, B, C, D are system matrices and G is referred to as the forward observer gain. The stable modes of the forward model are inside the unit circle, whereas the unstable modes are outside the unit circle.

On the other hand, a typical backward state space model has the form

$$x(k) = (A^{-1} + \tilde{G}C)x(k+1) + \begin{bmatrix} -A^{-1}B & \tilde{G}D & -\tilde{G} \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ y(k) \end{bmatrix}$$

$$y(k) = Cx(k) + Du(k)$$

The stable modes of the backward model are outside the unit circle, whereas the unstable modes are inside the unit circle. All matrices are the same as those described above for the forward model, but here $\tilde{G}$ is referred to as the backward observer gain matrix.

The matrices A, B, C, D, G, and $\tilde{G}$ can be simultaneously identified by the function *okid_fb*. For noise-free data, the A, B, C, D, identified either from the forward model or from the backward model should have an identical input-output map, implying that the identified system eigenvalues are identical. For noisy data, however, the system modes are contaminated by the noises. In addition, there are many computational (spurious) modes in the identified system matrices. As a result, the system matrices identified from both forward and backward models are somewhat different because the system modes are contaminated differently by noises. Note that the forward and backward models are identified by minimizing different residuals due to system uncertainties and measurement noises. Nevertheless, the system modes from both models should be reasonably close, whereas the computational modes may be quite different. Therefore, matching is possible to distinguish the system modes from the computational modes.

Given A, B, C identified for the forward model and A^{-1}, $-A^{-1}B$, C identified for the backward model, function *match* return with a reduced forward model and backward model in modal coordinates. The input parameters *af, bf, cf* represent the forward model matrices A, B, C, whereas *ab, bb, cb* represent A^{-1}, $-A^{-1}B$, C respectively. The output vectors, *lamdaf* and *lamdab*, contain the matched system eigenvalues for the forward and backward models respectively. The matrices *bmf,cmf,bmb,cmb* are the corresponding input and

output matrices in modal coordinates, where the last character *f* means forward and *b* backward. The output vectors *msv_f* and *msv_b* give the *msv* (modal singular value, see function *svpm*) contribution to the pulse response samples for the forward and backward modal parameters, respectively. In addition to the eigenvalue matching, the *msv* contribution is also examined. Those modes which has higher msv contribution than the matched modes are also included in the reduced model. Therefore, both the reduced forward and backward modal models may not be the same in size. In general, the reduced forward modal model is larger in size for stable modes than the reduced backward modal model. On the other hand, the reduced backward modal model may be larger in size for unstable modes than the reduced forward modal model. See references for detailed information. If the identified forward observer gain G and backward observer gain $\bar{G}$ are to be counted in the computation of the *msv* contribution, the input parameters *bf* should be replaced by [*bf gf*] and *bb* by [*bb gb*], where *gf* means G and *gb* means $\bar{G}$.

Example:

From two-input and three-output data of a three-mass-spring-dashpot system with two unstable modes and one stable mode, use okid and okid_b to identify a forward model and backward model, and then match the identified eigenvalues. The input *u* and output *y* data has 1 sec. sampling period, 250 data points and 10% noises.

```
[af,bf,cf,df,gf]=okid(3,2,1,u,y,'batch_lq',3);
[ab,bb,cb,db,gb]=okid_b(3,2,1,u,y,'batch_lq',3);
ab=inv(ab);bb=-ab*bb;
[lamdaf,bmf,cmf,msv_f,lamdab,bmb,cmb,msv_b]=match(af,[bf gf],cf,ab,[bb gb],cb);
n_b=length(lamdab);n_f=length(lamdaf);
eif=deg2hz(lamdaf,dt);
disp([eif(:,1:2) msv_f]);
      -1.4607e-01   2.7570e-01   2.5322e-01
      -1.4607e-01   2.7570e-01   2.5322e-01
      -1.7733e-01   8.0889e-02   1.0000e+00
      -1.7733e-01   8.0889e-02   1.0000e+00
       5.5570e-01   4.4269e-01   6.9444e-02
       5.5570e-01   4.4269e-01   6.9444e-02
lamdab=1.0 ./lamdab;bmb=-diag(lamdab)*bmb;
eib=deg2hz(lamdab,dt);
disp([eib(:,1:2) msv_b]);
      -1.8321e-01   2.7569e-01   2.2238e-01
      -1.8321e-01   2.7569e-01   2.2238e-01
      -1.9672e-01   8.0897e-02   1.0000e+00
      -1.9672e-01   8.0897e-02   1.0000e+00
       3.4786e-01   4.4260e-01   1.2439e-01
       3.4786e-01   4.4260e-01   1.2439e-01
```

See also:

okid, okid_b, okid_fb, svpm

References:

[1] Juang, J.-N. and Phan, M., "Identification of Backward Observer Markov Parameters: Theory and Experiments," *NASA Technical Memorandum TM-107632*, Langley Research Center, Hampton, VA., May 1992.

Purpose:

Compute a reduced stable or unstable model in modal coordinates.

Synopsis:

[*Lambda,Bm,cm*]=modal(*A,B,C,flag*)

Description:

Given the discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the equivalent model in modal coordinates is

$$x_m(k+1) = \Lambda x_m(k) + B_m u(k)$$
$$y = C_m x(k) + Du(k)$$

where Λ is a diagonal matrix containing the eigenvalues, $\lambda_i (i = 1,2,...,n)$, of the state matrix A. Those eigenvalues with their length larger than 1 are known to be unstable modes.

This function returns a complex vector, *Lambda* which contains only either stable modes when flag is set larger or equal to zero or unstable modes otherwise. Corresponding, the input matrix B is transformed to *Bm* and C to *Cm*. Note that the transmission matrix D is coordinate independent. This function is used in okid_b and okid_fb to distinguish identified unstable modes from stable modes for comparison with the identified results from okid. It is used in conjunction with function bk_diag to reduce an identified system model to a stable block diagonal form for numerical simulations to compare with real data.

See also:

okid_b, okid_fb

Purpose:

Estimate variance of the ERA identified parameters.

Synopsis:

[*eg*,*egm*,*vp*,*vw*,*mh*]=monera(*y*,*m*,*r*,*n*,*nm*,*dt*,*ni*,*nos*);

Description:

Monera uses the Monte Carlo approach to calculate the variance of the ERA identified parameters contaminated by noise. The system pulse response samples y are stored as

$$y=\begin{bmatrix} y_{11}(0) & \cdots & y_{m1}(0) & \cdots & y_{1r}(0) & \cdots & y_{mr}(0) \\ y_{11}(1) & & y_{m1}(1) & & y_{1r}(1) & & y_{mr}(1) \\ y_{11}(2) & \cdots & y_{m1}(2) & \cdots & y_{1r}(2) & \cdots & y_{mr}(2) \\ \vdots & & & & & & \\ y_{11}(l-1) & \cdots & y_{m1}(l-1) & \cdots & y_{1r}(l-1) & \cdots & y_{mr}(l-1) \end{bmatrix}$$

where $y_{ij}(t)$ is the i-th output at discrete time t to a unit pulse at the j-th input. The system to be identified has r inputs and m outputs. The identified model order, n, is chosen by the user. Scalar *nm* specifies the row number of the Hankel matrix as described in era. Integers $nm \times m$ should be greater than the model order n. Scalar *dt* specifies the data sampling interval. Scalar *ni* specifies the number of the Monte Carlo runs used to estimate the variance of the ERA identified parameters. Scalar *nos* specifies the standard deviation of the white, zero-mean and Gaussian measurement noise artificially added to the pulse response samples; *nos* should be much smaller than the root mean squared value of the pulse response samples in y.

[*eg*,*egm*,*vp*,*vw*,*mh*]=monera(*y*,*m*,*r*,*n*,*nm*,*dt*,*ni*,*nos*) returns the eigenvalue vector *eg* and the modal amplitude coherence vector *mh* of the ERA identified model from the pulse response samples *y*. The elements of vector *egm* are the mean values of the *ni* set of ERA identified eigenvalues from the pulse response samples with added noise (nos). Vector *vp* (*vw*) is the variance of the ERA identified damping (frequency) corresponding to the eigenvalue vector *eg* from *ni* set of Monte Carlo runs. The variable *mh* is the modal amplitude coherence.

Example:

Calculate the variance of the ERA identified frequencies and dampings from the pulse response samples of a single input, single output second order system with sampling interval dt=0.2, natural frequency $\omega_n = 1$, damping factor $\zeta = 0.1$.

```
a=[0 1;-1 -0.2]; b=[0;1];c=[1 0];d=0;
pt=100;dt=0.2;t=[dt:dt:pt*dt]';
u=zeros(pt,1);u(1,1)=1.0;
y=lsim(a,b,c,d,u,t);
rand('normal')
y=y+0.04*rand(pt,1);
[eg,egm,vp,vw,r]=monera(y,1,1,4,20,dt,50,1.e-5);
eg,egm,r
clg
subplot(211)
bar(vw),title('Frequency variance')
bar(vp),title('Damping Variance')
```

Algorithm:

The function monera uses the Monte Carlo approach [Longman89,91] and data correlation[Juang 88] to estimate the variance of the ERA identified frequencies and dampings. Depending on the number of sensors and the length of data to be used for system realization, monera automatically chooses either eradc or eradct to identify a system model with computational efficiency.

See also:

eradc, peradc

References:

[1] Longman, R. W., Bergman, M. and Juang, J. N., "Variance and Bias Confidence Criteria for ERA Modal Parameter Identification," *Proceedings of the 1988 AAS/AIAA Astrodynamics Specialist Conference*, Minneapolis, Minnesota, August 1988.

[2] Longman, R. W., Lew, J. S., Tseng, D. H. and Juang, J. N., "Variance and Bias Computation for Improved Modal Identification Using ERA/DC," *Proceedings of the 1991 American Control Conference*, Boston, MA, June 1991.

[3] Juang, J. N., Cooper, J. E. and Wright J. R., "An Eigensystem Realization Algorithm Using Data Correlation (ERA/DC) for Modal Parameter Identification," *Control Theory and Advanced Technology*, Vol. 4, No. 1, pp. 5-14, 1988.

Purpose:

Identify system, observer and controller gain matrices from closed-loop test data.

Synopsis:

[*a,b,c,d,g,f*]=ocid(*y,ufb,ue,p,dt,truncate,description*)

Description:

The function ocid solves for the observer/controller Markov parameters of the following system

$$x(k+1) = (A+GC)x(k) + \begin{bmatrix} B+GD & -G \end{bmatrix} \begin{bmatrix} u_{fb}(k)+u_e(k) \\ y(k) \end{bmatrix}$$

$$\begin{bmatrix} y(k) \\ u_{fb}(k) \end{bmatrix} = \begin{bmatrix} C \\ -F \end{bmatrix} x(k) + Du(k)$$

with sampling time intervals *dt* apart. The data is stored as column matrices. For example, for a system with *m* outputs, the closed-loop output data matrix *y* contains *m* columns and as many rows as the number of data points available. The data matrix *ufb* contains the feedback control signal, *ue* contains the additive excitation input signal. The number *p* denotes the number of observer/controller Markov parameters to be solved for. The number *truncate* specifies the number of data points to be deleted prior to application of the algorithm. This value is equal to the number of time steps that is expected for the existing observer to converge. The *description* is a short descriptive tag for the current data set and indicates the computation procedure to be used to compute the observer and controller gain. If the description is set to be 'lq', it means that *a, b, c, d* are realized first, and then *g* and *f* are computed by least-squares (lq). Any other description indicates that *a, b, c, d, g,* and *f* are to be realized simultaneously. The function will first return a plot of singular values for the user to select the desired model order. Once this is done, the function will ask whether or not the user wants to see plots that show the actual responses and reconstructed responses. Finally, a set of realized system matrices, observer gain, and controller gain will be returned.

In general, if *p* observer/controller Markov parameters are to be solved for, then the maximum order of the system that can be recovered is *pm*, where *m* is the number of outputs. The following rules apply with regard to the number of singular values computed. If a flag value of zero is chosen, then the singular value plot will show *(m+r)i* singular values where *i* is the smallest integer such that *(m+r)i* is larger or equal to *pm* where *r* is the number of inputs. If a flag value of one is chosen, then the singular value plot will always show *pm* singular values. In either case, the user can always retain *pm* singular values to obtain a realized system that has the maximum order for a chosen value of *p*.

The following information provides a better understanding of the OCID problem. Consider a linear discrete system of the form

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

which is operating in closed-loop. The input to the system, $u(k)$, consists of two components

$$u(k) = u_{fb}(k) + u_e(k)$$

where $u_{fb}(k)$ denotes the feedback signal provided by an *existing* linear state feedback controller with gain F

$$u_{fb}(k) = -F\hat{x}(k)$$

and $u_e(k)$ denotes an additive excitation input for closed-loop identification. The estimated state $\hat{x}(k)$ is provided by an *existing* observer of the form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) - G_d[y(k) - \hat{y}(k)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

The output $y(k)$ is the system closed-loop response due to an excitation $u_e(k)$. OCID first solves for the Markov parameters

$$D, \text{ and } \begin{bmatrix} C \\ -F \end{bmatrix}(A+GC)^{k-1}[B+GD \quad -G], \quad k = 1, 2, \ldots, p$$

from which a realization of A, B, C, G, and F will be computed and returned to the user. Note that the matrices A, B, C, D are the system matrices and F is the *existing* feedback controller gain as described above. The matrix G , however, is *another* observer gain associated with the identified system A, B, C, D. In general, this observer matrix gain G is *not* the same as the *existing* observer gain G_d of the closed-loop system. The matrix G is an observer gain for the observer given below

$$\tilde{x}(k+1) = A\tilde{x}(k) + Bu(k) - G[y(k) - \tilde{y}(k)]$$
$$\tilde{y}(k) = C\tilde{x}(k) + Du(k)$$

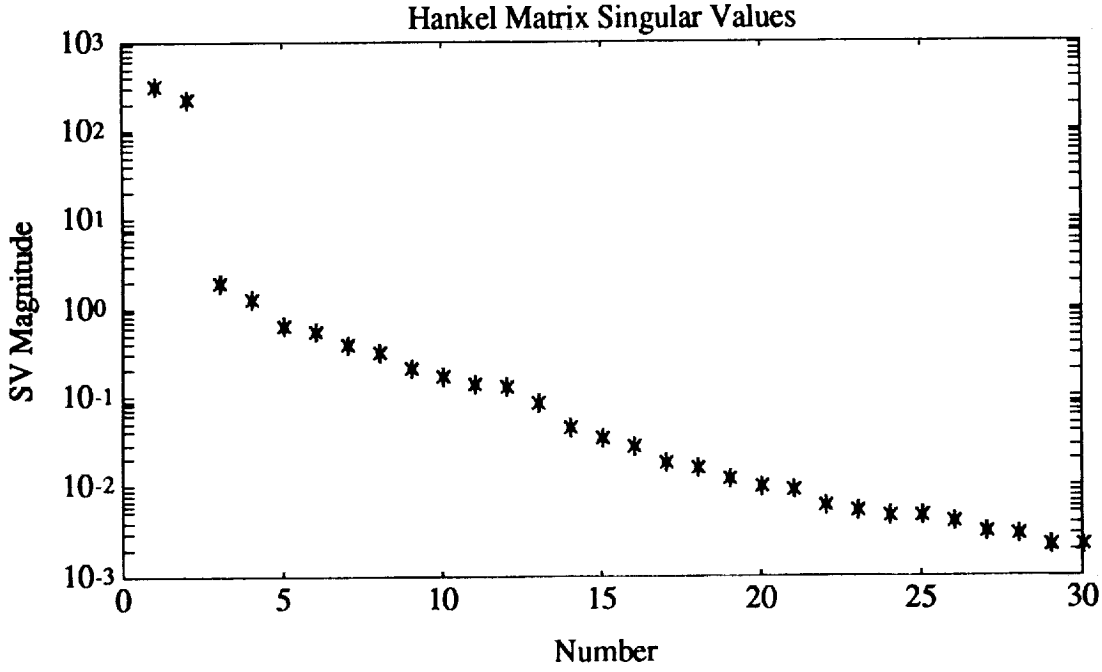
The function returns a, b, c, d, g, and f, which are a realization of A, B, C, D, G, and F.

Example:

Use test data from an aircraft flutter test. The items in italics is information prompted by the function ocid which has to be answered by the user. The rest is just general information returned by the function ocid. The following is output taken from a typical run.

```
load xsamp712
ue(1:600)=[];y(1:600)=[];u(1:600)=[];
[m,n]=size(ue);
time=dt*[0:m-1]';
[a,b,c,d,g,f]=ocid(y,-u,ue,30,dt,300,'ocid');

ERADC is used now.
The Hankel matrix size for ERADC is 30 by 62.
```

Maximum Hankel singular value = 3.213077e+02
Minimum Hankel singular value = 2.251935e-03



Desired Model Order (0=stop)=: 2
Model Describes 98.9109 (%) of Test Data

Damping(%)	Freq(HZ)	Mode SV	MAC
-3.2239e+00	9.1286e+00	1.0000e+00	9.9827e-01
-3.2239e+00	9.1286e+00	1.0000e+00	9.9827e-01

Desired Model Order (0=stop)=: 4
Model Describes 99.4891 (%) of Test Data

Damping(%)	Freq(HZ)	Mode SV	MAC
2.2923e+01	1.1587e+01	5.7107e-02	9.8650e-01
2.2923e+01	1.1587e+01	5.7107e-02	9.8650e-01
-2.8767e+00	8.8912e+00	1.0000e+00	9.9987e-01
-2.8767e+00	8.8912e+00	1.0000e+00	9.9987e-01

Desired Model Order (0=stop)=: 8
Model Describes 99.829 (%) of Test Data

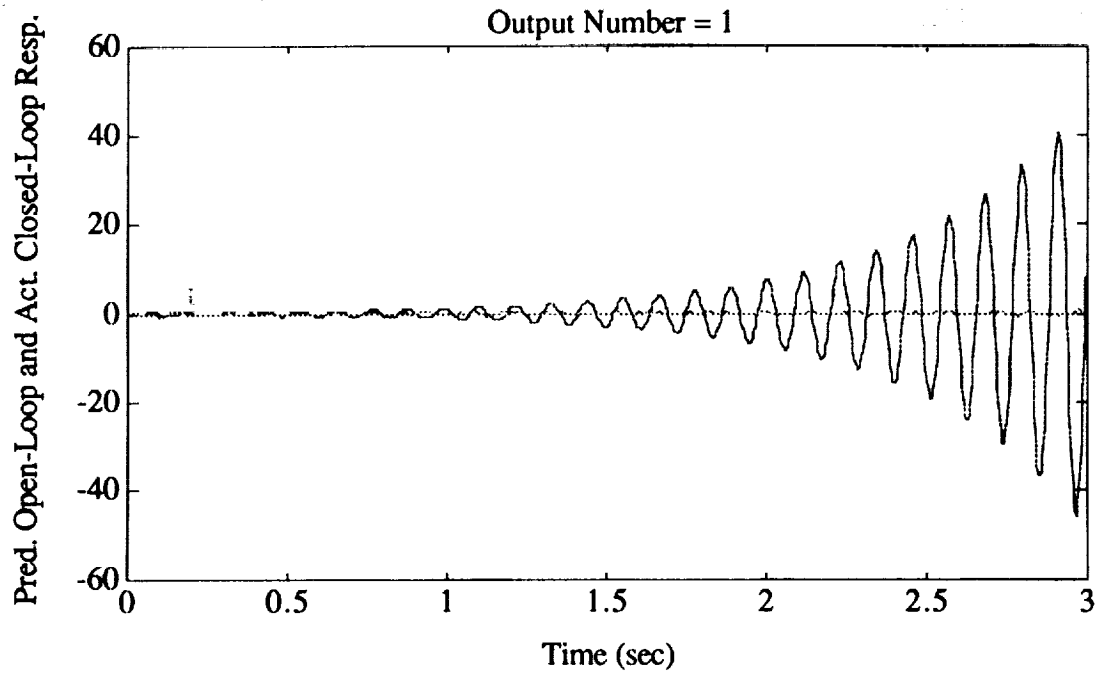
Damping(%)	Freq(HZ)	Mode SV	MAC
1.8507e+01	1.9546e+01	1.2959e-02	9.4607e-01
1.8507e+01	1.9546e+01	1.2959e-02	9.4607e-01
2.9880e+00	8.7176e+01	2.2222e-02	9.8971e-01
2.9880e+00	8.7176e+01	2.2222e-02	9.8971e-01
1.6910e+01	1.1593e+01	9.2265e-02	9.9728e-01
1.6910e+01	1.1593e+01	9.2265e-02	9.9728e-01
-3.3582e+00	8.8459e+00	1.0000e+00	9.9995e-01
-3.3582e+00	8.8459e+00	1.0000e+00	9.9995e-01

Desired Model Order (0=stop)=: 0

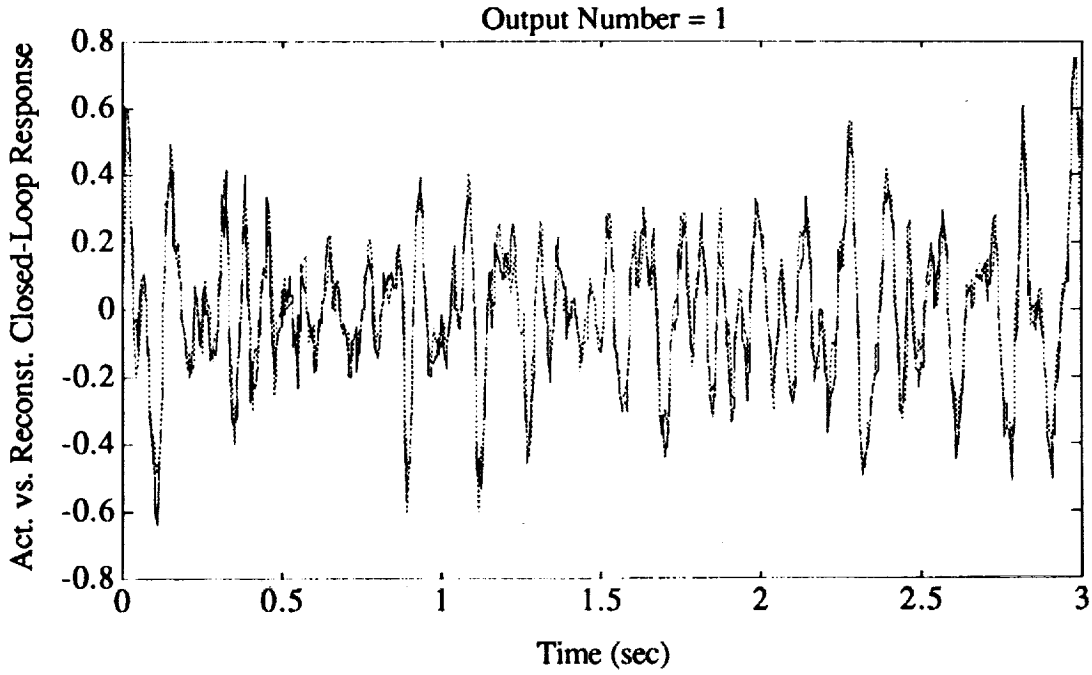
Compare closed-loop reconst. and actual resp. (1=yes,0=no) ?:= 1

Number of sample points to reconstruct ?:= 400

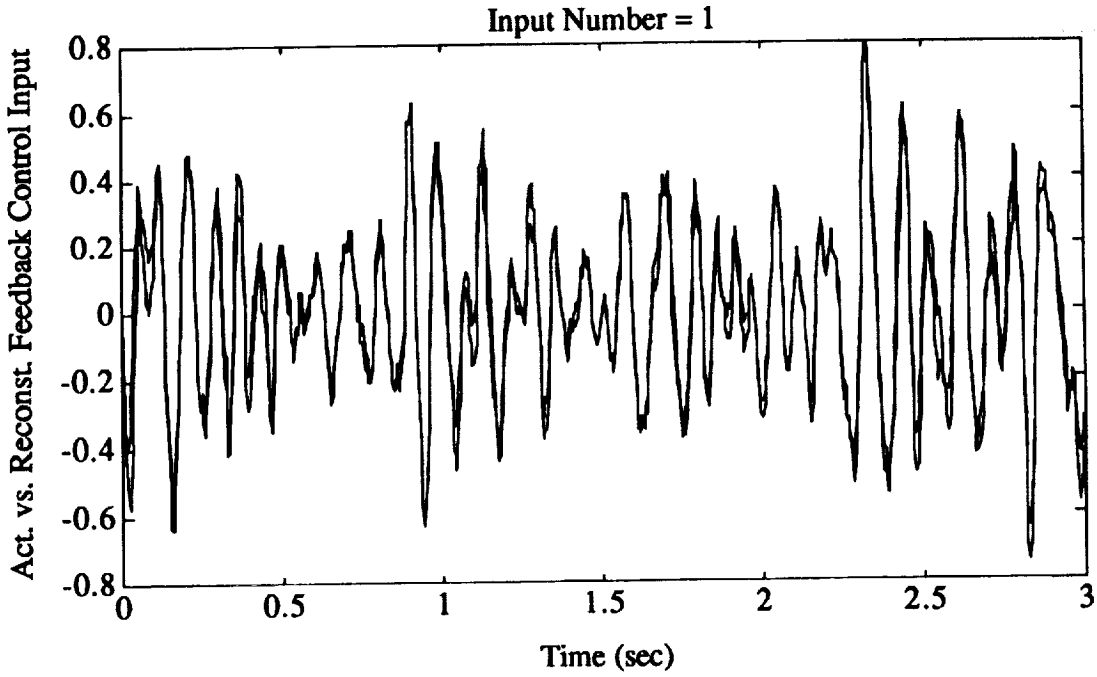
Predicted open-loop and actual closed-loop responses



Actual vs. reconstructed closed-loop responses



Actual vs. reconstructed feedback control input histories



The solid lines represent the real data whereas the dashed lines mean the reconstructed data. The predicted outputs are the reconstructed data from the identified system model only. The estimated outputs are the reconstructed data from the identified observer. It is obvious that the reconstructed data match the real data very well.

Algorithm:

Identification of the observer/controller Markov parameters for the system shown before is obtained using a least-squares solution. The observer/controller Markov parameters are identified first, from which the individual Markov parameters of the system, observer, and controller are computed, and used to obtain a realized state space model and the observer and controller gain matrices. For further details, see references.

See also:

arxc, mar_yoc, mar_oc, ryucovar, y_closed, separate

Reference:

[1] Juang, J. N. and Phan, M., "Identification of System, Observer, and Controller from Closed-Loop Experimental Data," *Presented at the AIAA Guidance, Navigation, and Control Conference,* Hilton Head, South Carolina, Aug. 10-12, 1992.

Purpose:

Identify a state space model and its corresponding observer from test data.

Synopsis:

[*a,b,c,d,g*]=okid(*m,r,dt,u,y,description,p*)
[*a,b,c,d,g*]=okid_b(*m,r,dt,y,description,p*)
[*af,bf,cf,df,gf,ab,bb,cb,db,gb*]=okid_fb(*m,r,dt,u,y,description,p*)
[*a,b,c,d,g*]=okid_p(*m,r,dt,y,description,p*)

Description:

The function [*A,B,C,D,G*]=okid(*m,r,dt,u,y,description,p*) identifies a state space model of the form

$$x(k+1) = (A+GC)x(k) + \begin{bmatrix} B+GD & -G \end{bmatrix} \begin{Bmatrix} u(k) \\ y(k) \end{Bmatrix}$$

$$y(k) = Cx(k) + Du(k)$$

with m outputs, r inputs, and time samples dt apart. Here A, B, C, D are system matrices and G is referred to as the forward observer gain. The input variable *description* is a short descriptive tag for the current data set being analyzed and also serves as a flag which is described latter. The input/output time histories are stored as column matrices. For s experiments, the input matrix u must contain $s \times r$ columns and the output matrix y $s \times m$. The number of rows in the input and output matrices equals the number of sample points. Initially an estimate of the number of observer Markov parameters, p, must be specified. For a given p, the maximum system order that can be identified is the product $p \times m$. The function will prompt the user at various points for information. After computing the observer Markov parameters, the option to compute the prediction error is given. The calculation of observer Markov parameters and the output prediction error is time consuming when analyzing long records, therefore, it should only be used when necessary. For the very first run, the observer Markov parameters and related parameters including p are stored in the data file *dokid_f* for function file okid, *dokid_b* for okid_b, *dokid_fb* for okid_fb and *dokid_p* for okid_p. The user will be prompted if he has already stored these parameters in the data file. If he did, computation of these parameters will be by-passed. A plot of the Hankel matrix singular values is shown to aid selecting the correct system order. The number of non-zero singular values equals the system order. The magnitude of the Hankel matrix singular values, arranged in descending order, measures the state contribution. For noisy data, one has to make a judgement as to how many singular values to retain. After selecting a particular order, the percentage of the response realized by the model is computed using the singular values. In addition, the corresponding frequencies and damping values are listed with the corresponding modal amplitude coherence factors. If the model is acceptable, the computation is completed. The identified matrices A,B,C,D, and G are returned to the main program. See references for detailed information.

The user is recommended to run the batch job for the first time so that he does not have to wait for the computation of observer Markov parameters which may take time for a long data record. The user may then come back to run the same job again interactively and use the existing data record for the observer Markov parameters.

The user is recommended to run the batch job for the first time so that he does not have to wait for the computation of observer Markov parameters which may take time for a long data record. The user may then come back to run the same job again interactively and use the existing data record for the observer Markov parameters.

okid: It simultaneously identifies A, B, C, D and G directly from input/output data. User's interaction is required to determine the order of the system by looking at the singular values plot.

description = *'batch'* ; It performs a batch job. The order of the system is determined internally in the era function file and thus user's interaction is not required.

description = *'lq'* ; It computes the system matrices, A, B, C, and D, first and then the observer gain G using least-squares. Researchers who are interested in identifying the system matrices only are recommended to use this function file.

okid_p: a modified version of okid. It uses pulse response samples, y, to simultaneously identify A, B, C, D and G. No user inputs are required in this function file.

description = *'lq'* ; It computes the system matrices, A, B, C, and D, first and then the observer gain G using least-squares. Researchers who are interested in identifying the system matrices only are recommended to use this function file.

The identified system has the following form

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with its corresponding identified observer as

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) - G[y(k) - \hat{y}(k)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

where $\hat{x}(k)$ is the estimate of the state $x(k)$.

The function [*A*,*B*,*C*,*D*,*G*]=okid_b(*m*,*r*,*dt*,*u*,*y*,*description*,*p*) identifies a state space model of the form

$$x(k) = (A^{-1} + \tilde{G}C)x(k+1) + \begin{bmatrix} -A^{-1}B & \tilde{G}D & -\tilde{G} \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ y(k) \end{bmatrix}$$

$$y(k) = Cx(k) + Du(k)$$

with m outputs, r inputs, and time samples dt apart. All the input and output parameters of this function are the same as those described above for the forward observer. Here A, B, C, D are system matrices and $\tilde{G}$ is referred to as the backward observer gain matrix. See references for detailed information.

okid_b: It simultaneously identifies A, B, C, D and $\tilde{G}$ directly from input/output data. User's interaction is required to determine the order of the system by looking at the singular values plot.

description = *'batch'* ; It performs a batch job. The order of the system is determined internally in the era function file and thus user's interaction is not required.

description = *'lq'* ; It computes the system matrices, A, B, C, and D, first and then the observer gain $\tilde{G}$ using least-squares. Researchers who are interested in identifying the system matrices only are recommended to use this function file.

The identified system is the same as above, whereas the observer becomes

$$\hat{x}(k) = A^{-1}\hat{x}(k+1) + A^{-1}Bu(k) - \tilde{G}[y(k+1) - \hat{y}(k+1)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

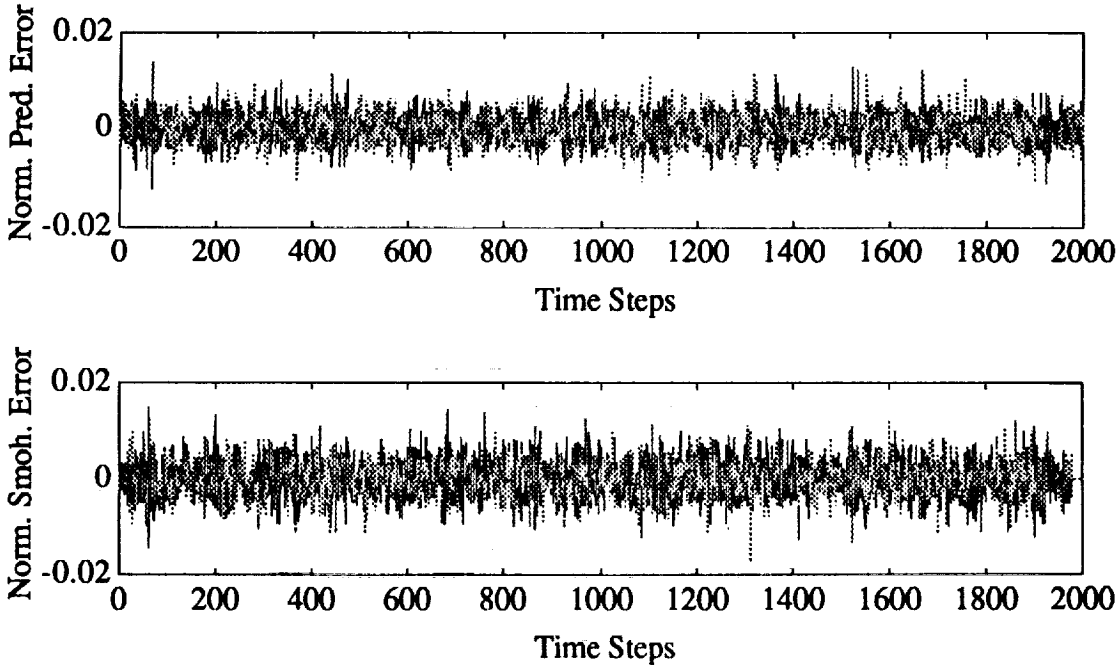
where $\hat{x}(k)$ is the estimate of the state $x(k)$. Note that the backward approach identifies the inverse of the system state matrix whereas the forward approach identifies the system state matrix directly. The advantage of the backward approach is that all the identified spurious modes tend to be stable and the strong system modes to be unstable. Therefore when the identified state matrix is inverted, the strong system modes become stable but the computational modes become unstable. Nevertheless, experiences suggest that the identified results are somewhat underestimated particularly when noises are high. To do numerical simulations, functions modal and bk_diag may be used to reduce the identified model to a stable model in the real domain.

The function [*Af,Bf,Cf,Df,Gf,Ab,Bb,Cb,Db,Gb*]=okid_fb(*m,r,dt,u,y,description,p*) identifies two state space models simultaneously using both the forward and backward approach. The small cases *f* and *b* behind the capital characters means forward and backward respectively. In this function, comparison of stable system modes in terms of frequencies and dampings from these two models is provided to the user for his judgement on how accurate the system modes are. The comparison is based on the error between the forward system eigenvalues and backward system eigenvalues, and their singular value contributions to the pulse response samples. The comparison provides the reduced output forward model [*Af,Bf,Cf,Df,Gf*] and backward model [*Ab,Bb,Cb,Db,Gb*] where the state matrices *Af* and *Ab* are both in block-diagonal form (see function bk_diag). Note that the reduced modal observer may not be stable, since the modal reduction is not optimal in general. In this case, the user is recommended to use function okid to identify a system model and then reduced by other methods such as the optimal projection or balanced coordinates. The function okid_fb is strongly recommeded for those users who do not care about the observer identification. With the same order of observer markov parameters, it is believed that the forward approach provides the identified results with better accuracy. The forward model is in general larger in size than the backward model. The backward approach may miss some system modes particularly with light damping. However, it provides information of strong system modes.

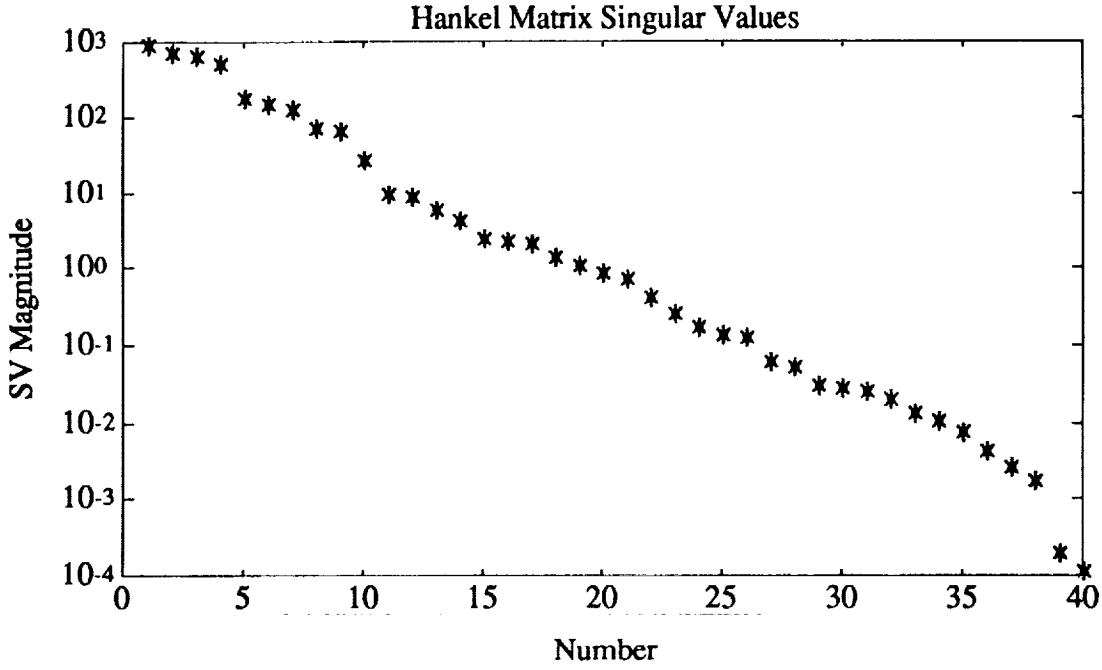
Example :

Use test data from a truss structure. The items in italics is information prompted by the function which has to be answered by the user. The rest is just general information returned by the function. The following is output taken from a typical run.

```
load xsample
[a,b,c,d,m]=okid_fb(n,r,dt,u,y,'okid_fb',20);

Total number of sample  points = 2000
Number of experiments in file = 1
Number of inputs = 2
Number of outputs = 2
Compute Observer Paramters For Data Set Number 1
Time (min) to compute parameters 4.114

Have you run OKID_FB with the same data & P before (1=yes,0=no) ?:= 0

Compute Forward and Backward Error (1=yes,0=no)? =: 1
Compute Prediction Error For Data Set Number 1
Forward Square Fitting Error Normalized
  7.2631e-02  -4.8648e-02
 -4.8648e-02   1.1211e-01
Backward Square Fitting Error Normalized
  1.5161e-01  -4.6554e-02
 -4.6554e-02   1.4184e-01
Time (min) to Markov parameters 0.1517
```



THE FOLLOWING COMPUTES A DISCRETE MODEL FROM A FORWARD OBSERVER.



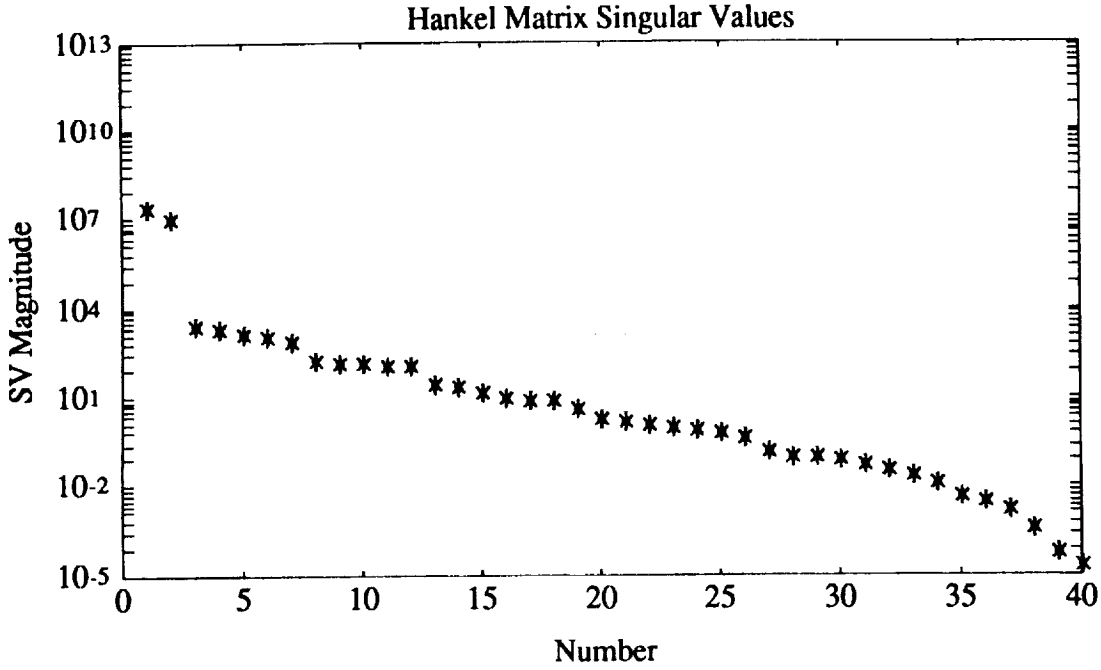
Desired Model Order (0=stop)=: 26

Model Describes 99.9924 (%) of Test Data

Damping(%)	Freq(HZ)	Mode SV	MAC
9.8448e+00	9.7606e+01	3.0499e-03	9.9677e-01
9.8448e+00	9.7606e+01	3.0499e-03	9.9677e-01
7.4294e+00	1.1862e+02	2.2901e-02	9.9957e-01
7.4294e+00	1.1862e+02	2.2901e-02	9.9957e-01
5.1193e+00	1.0573e+02	7.2535e-03	9.9891e-01
5.1193e+00	1.0573e+02	7.2535e-03	9.9891e-01
1.7537e+01	2.2577e+01	2.3866e-03	9.9869e-01
1.7537e+01	2.2577e+01	2.3866e-03	9.9869e-01
1.1365e+01	3.3578e+01	9.4557e-04	9.8801e-01
1.1365e+01	3.3578e+01	9.4557e-04	9.8801e-01
1.7740e+01	1.9096e+01	1.4288e-03	9.9687e-01
1.7740e+01	1.9096e+01	1.4288e-03	9.9687e-01
1.9292e+00	1.1410e+02	4.4511e-02	9.9999e-01
1.9292e+00	1.1410e+02	4.4511e-02	9.9999e-01
3.4358e+00	6.1287e+01	1.6785e-03	9.9893e-01
3.4358e+00	6.1287e+01	1.6785e-03	9.9893e-01
1.4563e+00	4.6591e+01	1.2229e-02	9.9996e-01
1.4563e+00	4.6591e+01	1.2229e-02	9.9996e-01
6.7864e-01	7.4241e+01	4.4470e-03	9.9929e-01
6.7864e-01	7.4241e+01	4.4470e-03	9.9929e-01
4.2429e-01	4.8646e+01	6.9666e-02	1.0000e+00
4.2429e-01	4.8646e+01	6.9666e-02	1.0000e+00
3.1829e-01	7.2733e+00	4.8384e-01	1.0000e+00
3.1829e-01	7.2733e+00	4.8384e-01	1.0000e+00
3.9005e-01	5.8479e+00	1.0000e+00	1.0000e+00
3.9005e-01	5.8479e+00	1.0000e+00	1.0000e+00

Desired Model Order (0=stop)=: 0

THE FOLLOWING COMPUTES A DISCRETE MODEL FROM A BACKWARD OBSERVER.



Desired Model Order (0=stop)=: 26

Model Describes 100 (%) of Test Data

Damping(%)	Freq(HZ)	Mode SV	MAC
8.7805e+00	1.0444e+02	1.5233e-02	9.9862e-01
8.7805e+00	1.0444e+02	1.5233e-02	9.9862e-01
6.9248e+00	1.1904e+02	5.8056e-02	9.9986e-01
6.9248e+00	1.1904e+02	5.8056e-02	9.9986e-01
5.7392e+00	1.0529e+02	2.2203e-02	9.9896e-01
5.7392e+00	1.0529e+02	2.2203e-02	9.9896e-01
1.9648e+00	1.1406e+02	8.8767e-02	1.0000e+00
1.9648e+00	1.1406e+02	8.8767e-02	1.0000e+00
7.0426e+00	3.1259e+01	2.6690e-03	9.8609e-01
7.0426e+00	3.1259e+01	2.6690e-03	9.8609e-01
2.6303e+00	6.1469e+01	2.4191e-03	9.9780e-01
2.6303e+00	6.1469e+01	2.4191e-03	9.9780e-01
4.7085e+00	1.9505e+01	1.6134e-02	9.9995e-01
4.7085e+00	1.9505e+01	1.6134e-02	9.9995e-01
1.8700e+00	4.6542e+01	2.1705e-02	9.9976e-01
1.8700e+00	4.6542e+01	2.1705e-02	9.9976e-01
6.9673e-01	7.4489e+01	4.6185e-03	9.9365e-01
6.9673e-01	7.4489e+01	4.6185e-03	9.9365e-01
4.2593e-01	4.8600e+01	8.9072e-02	9.9999e-01
4.2593e-01	4.8600e+01	8.9072e-02	9.9999e-01
-8.8086e-01	5.8583e+00	1.0000e+00	1.0000e+00
-8.8086e-01	5.8583e+00	1.0000e+00	1.0000e+00
-9.3811e-01	7.3066e+00	3.9005e-01	9.9999e-01
-9.3811e-01	7.3066e+00	3.9005e-01	9.9999e-01
-1.9640e+01	1.4194e+01	3.1534e-02	1.0000e+00
-1.9640e+01	1.4194e+01	3.1534e-02	1.0000e+00

Desired Model Order (0=stop)=: 0

COMPARISON OF FORWARD AND BACKWARD IDENTIFICATION

Forward Identification			Backward Identification		
Damping(%)	Freq(hz)	Mode SV	Damping(%)	Freq(hz)	Mode SV
1.7740e+01	1.9096e+01	1.8532e-03	1.9640e+01	1.4194e+01	3.2081e-02
1.7740e+01	1.9096e+01	1.8532e-03	1.9640e+01	1.4194e+01	3.2081e-02
3.4358e+00	6.1287e+01	3.2029e-03	9.3811e-01	7.3066e+00	3.4642e-01
3.4358e+00	6.1287e+01	3.2029e-03	9.3811e-01	7.3066e+00	3.4642e-01
1.7537e+01	2.2577e+01	3.2734e-03	8.8086e-01	5.8583e+00	1.0000e+00
1.7537e+01	2.2577e+01	3.2734e-03	8.8086e-01	5.8583e+00	1.0000e+00
1.4563e+00	4.6591e+01	1.3625e-02			
1.4563e+00	4.6591e+01	1.3625e-02			
6.7864e-01	7.4241e+01	1.7323e-02			
6.7864e-01	7.4241e+01	1.7323e-02			
9.8448e+00	9.7606e+01	2.8269e-02			
9.8448e+00	9.7606e+01	2.8269e-02			
5.1193e+00	1.0573e+02	5.7957e-02			
5.1193e+00	1.0573e+02	5.7957e-02			
4.2429e-01	4.8646e+01	7.3772e-02			
4.2429e-01	4.8646e+01	7.3772e-02			
7.4294e+00	1.1862e+02	2.0175e-01			
7.4294e+00	1.1862e+02	2.0175e-01			
1.9292e+00	1.1410e+02	3.2723e-01			
1.9292e+00	1.1410e+02	3.2723e-01			
3.1829e-01	7.2733e+00	4.4107e-01			
3.1829e-01	7.2733e+00	4.4107e-01			
3.9005e-01	5.8479e+00	1.0000e+00			
3.9005e-01	5.8479e+00	1.0000e+00			

These are the modes selected from forward and backward models
You should also examine the list of modes from the forward model
to see if there are some other modes left out from the above table.

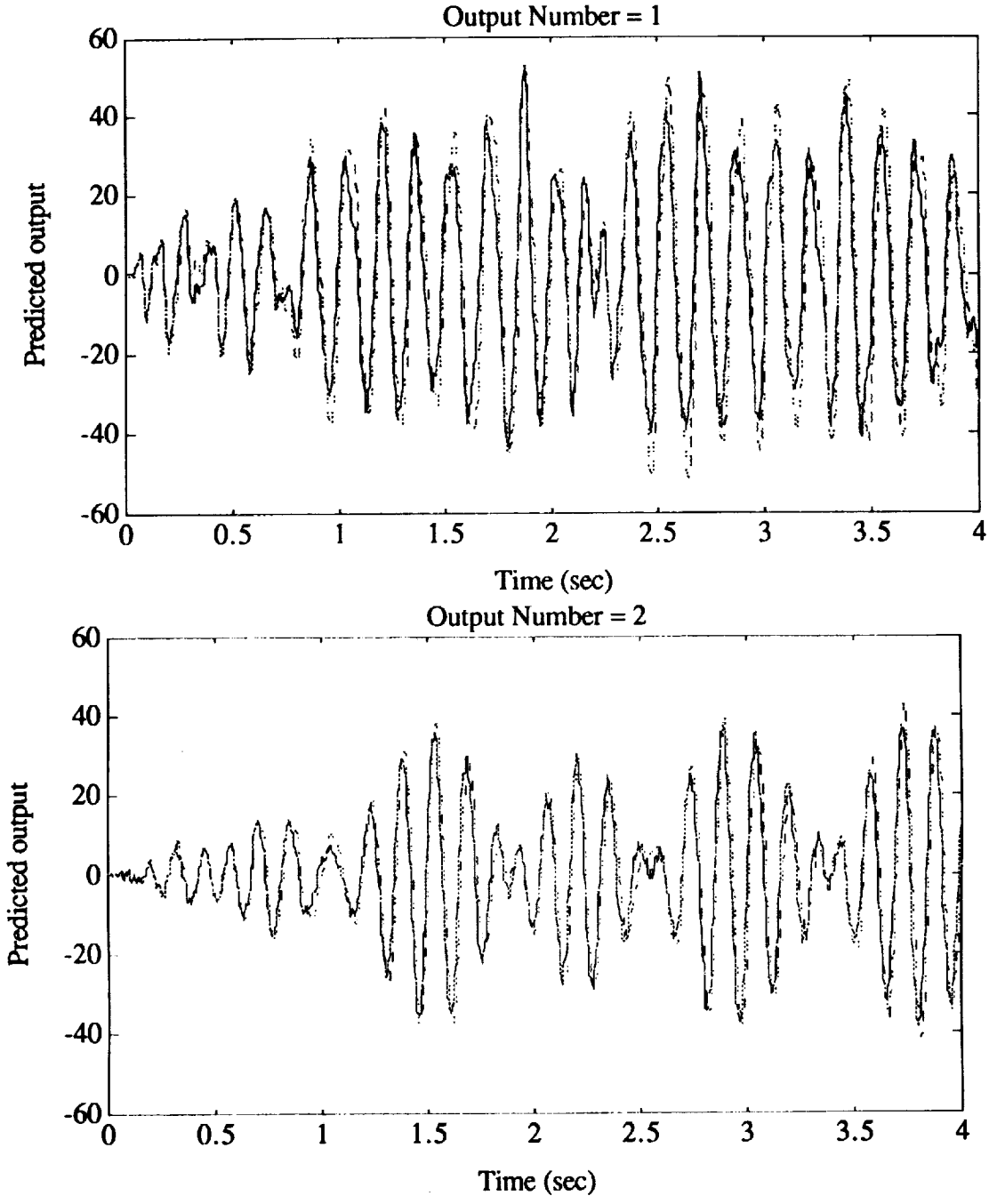
DATA RECONSTRUCTION FROM THE IDENTIFIED FORWARD MODEL

Compare Recons. Output and True output (1=yes,0=no) ?:= 1

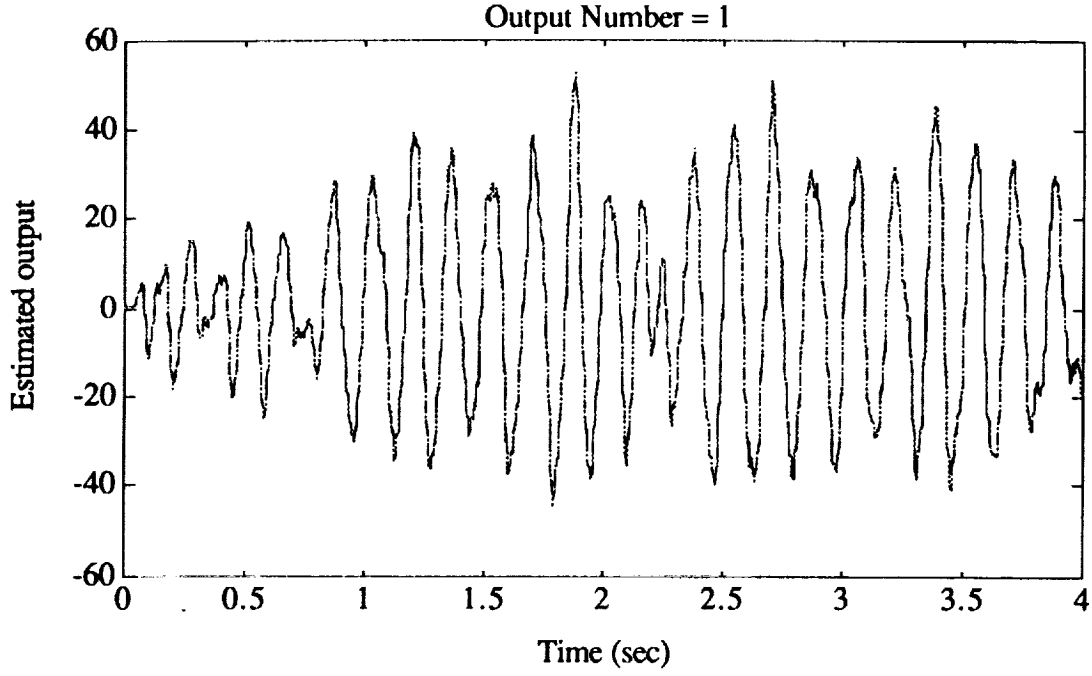
Number of Sample Points to Reconstruct ?:= 1000

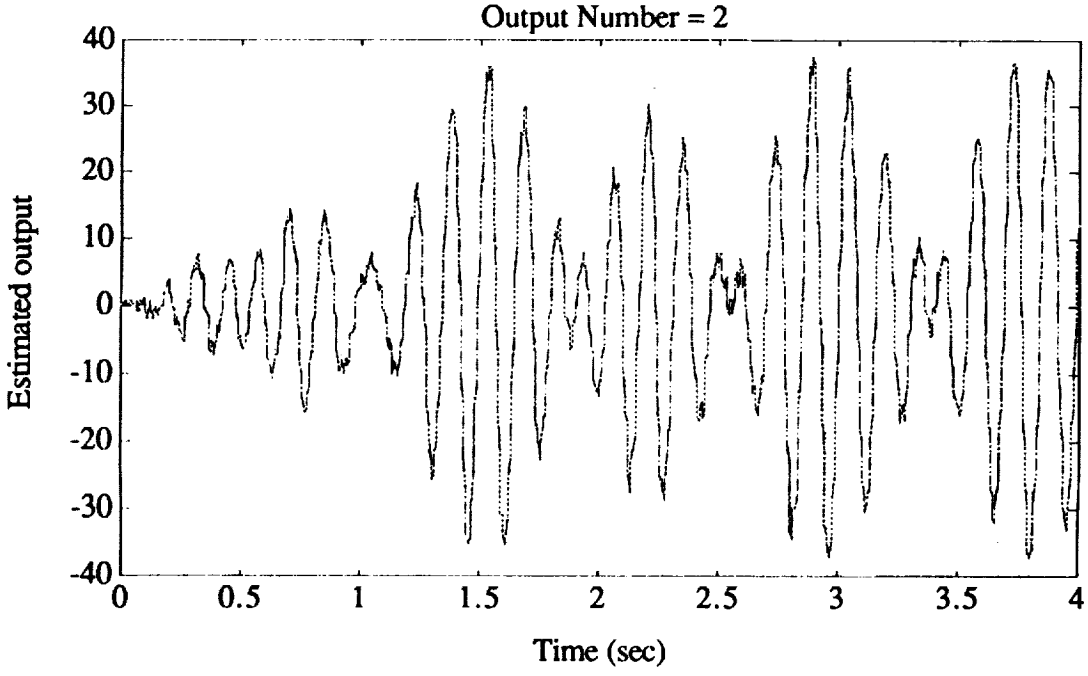
Comparison For Data Set Number No. 1

The following figures show predicted and real outputs



The following figures show estimated and real outputs





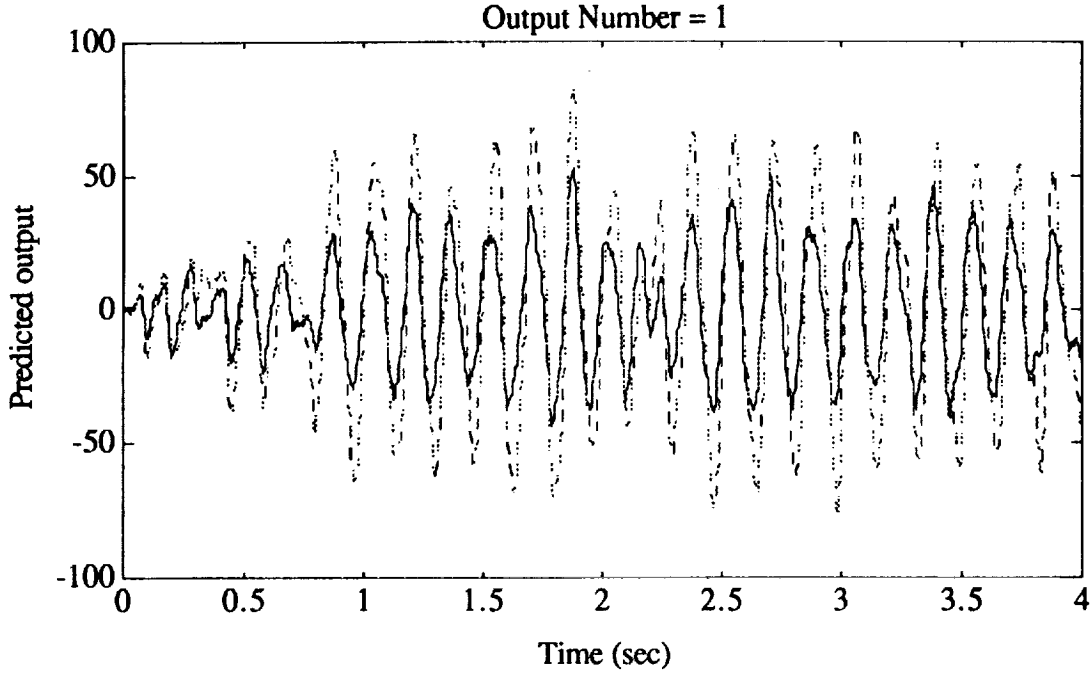
DATA RECONSTRUCTION FROM THE IDENTIFIED BACKWARD MODEL

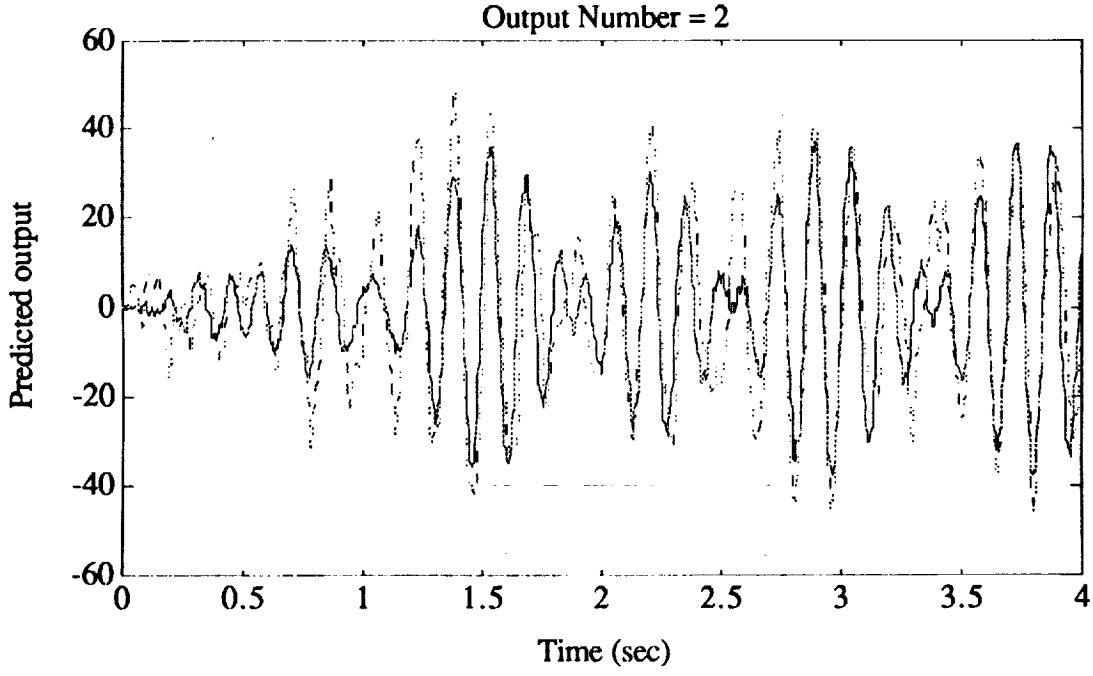
Compare Recons. Output and True output (1=yes,0=no) ?:= 1

Number of Sample Points to Reconstruct ?:= 1000

Comparison For Data Set No. 1

The following figures show smoothed and real outputs





The solid lines in the above figures represent the real data whereas the dashed lines mean the reconstructed data. The predicted outputs are the reconstructed data from the identified system model only. The estimated outputs are the reconstructed data from the identified observer. It is obvious that the reconstructed data from the identified forward observer match the real data much better than that from the identified model only. With the same order chosen which is 26 in this example, the forward results are somewhat better than the backward results (see the reconstructed predicted outputs). In this example, only three stable modes (after the identified state matrix was inverted) are identified from the backward approach.

Algorithm:

Identification of the pulse response (Markov parameters) for the observer system shown before is obtained using singular value decomposition. The forward or backward observer is identified first, as opposed to the system itself, such that the observer has all its poles placed at the origin. From this, the system Markov parameters (pulse response) are recovered and used in system realization. Theoretically, there are only a specific number of independent system Markov parameters for a finite set of observer Markov parameters. Therefore a minimum number of system Markov parameters may be used in the function era or eradc to minimize the computational time in identifying the system. Nevertheless, it seems from experience that a little larger number than the minimum one for the system Markov parameters help a little bit for the identification of system with very low damping. The realization algorithm provides a state space model and permits the evaluation of different system orders. Order selection is guided by the singular values of a Hankel matrix, but for test data it is up to the user to decide. For details see references.

See also:

arx_b, arx_bat, arx_fb, era, eradc, k_abcd, mar_com, match, pred_err, svmp, uy_stack, yucovar

References:

[1] Juang, J.-N., and Pappa, R. S.," An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics,* Vol. 8, No. 5, Sept.-Oct. 1985, pp. 620-627.

[2] Juang, J.-N., Cooper, J. E., and Wright, J. R., "An Eigensystem Realization Algorithm Using Data Correlations (ERA/DC) for Modal Parameter Identification," *Control-Theory and Advanced Technology*, Vol. 4, No. 1, 1988, pp. 5-14.

[3] Juang, J.-N., Horta, L. G., and Longman, R. W., "Input/Output System Identification: Learning From Repeated Experiments," *Mechanics and Control of Large Space Structures* edited by John L. Junkins, AIAA Monograph on Progress in Astronautics and Aeronautics, Vol 29, 1990, pp. 87-99.

[4] Chen, C. W., Huang, J.-K., Phan, M., and Juang, J.-N.,"Integrated System Identification and Modal State Estimation for Control of Large Flexible Space Structures," *Journal of Guidance, Control and Dynamics,* Vol. 15, No. 1, Jan.-Feb. 1992, pp. 88-95.

[5] Phan, M., Juang, J.-N., and Longman, R. W., "Identification of Linear Multivariable Systems from a Single Set of Data by Identification of Observers with Assigned Real Eigenvalues," *Proceedings of the AIAA 32nd Structures, Structural Dynamics & Materials Conference,*

Baltimore, MD., April 8-10, 1991, pp. 2325-2335, and to appear in the *Journal of the Astronautical Sciences.*

[6] Phan, M., Horta, L. G., Juang, J.-N., and Longman, R. W., "Linear System Identification Via an Asymptotically Stable Observer," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, Aug. 1991, pp. 1180-1194, and NASA Technical Paper 3164, 1991, and to appear in the *Journal of Optimization Theory and Application.*

[7] Juang, J.-N., Phan, M., Horta, L. G.,and Longman, L. G., "Identification of Observer and Kalman Filter Markov Parameters: Theory and Experiments," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, Aug. 1991, pp. 1195-1207.

[8] Horta, L. G., Phan, M., Juang, J.-N., Longman, R. W., and Sulla, J., "Frequency Weighted System Identification and Linear Quadratic Controller Design," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, Aug. 1991, pp. 1172-1179, and to appear in the *Journal of Guidance, Control and Dynamics.*

[9] Phan, M., Juang, J.-N., and Longman, R. W., "On Markov Parameters in System Identification," *NASA Technical Memorandum TM-104156,* Langley Research Center, Hampton, VA., Oct. 1991.

[10] Juang, J.-N. and Phan, M., "Identification of Backward Observer Markov Parameters: Theory and Experiments," *NASA Technical Memorandum TM-107632,* Langley Research Center, Hampton, VA., May 1992.

[11] Gawronski, W. and Juang, J. N., "Model Reduction for Flexible Structures," *Advances in Large Scale Systems Dynamics*, Edited by C. T. Leondes, Academic Press, Inc., New York, 1990, pp. 143-222.

[12] Hollkamp, J.J. and Batill, S.M., "Automated Parameter Identification and Order Reduction for Discrete Series Models," *AIAA Journal,* Vol. 29, No. 1, 1991.

[13] Hollkamp, J.J. and Batill, S.M., "Structural identification Using Order Overspecified Time-Series Models," *Journal of Dynamic Systems, Measurement, and control,* To appear.

Purpose:

Rearrange pulse response time histories in the form of Markov parameters sequence.

Synopsis:

Y_o=p2m(Y_p,r)

Description:

Given a discrete model

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

with r inputs and m outputs, the pulse response samples are typically stored as

$$Y_p = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_{\ell} \end{bmatrix}$$

where y_i = response samples due to a unit pulse at the i-th input. Each y_i has m columns and ℓ rows where ℓ is the length of data. The sequence of system Markov parameters is defined as

$$Y_0 = \begin{bmatrix} D & CB & CAB & \cdots & CA^{\ell-2}B \end{bmatrix}$$

The sequences Y_o and Y_p are equivalent in the sense that both represent pulse response samples. The function converts Y_p to Y_o. Note that Y_p is the sequence used in the eradc and era functions.

See also:

m2p, okid, okid_b, okid_fb

Purpose:

Calculate variance and bias of the ERA/DC identified parameters.

Synopsis:

[*eg,vp,vw,bp,bw,sg,vsg,bsg*]=peradc(*y,n,nm,nr,dt*);

Description:

The function peradc calculates the variance and bias of the single-input and single-output eradc identified parameters from pulse response samples. The column vector *y* is the pulse response samples in which the *i*th element of *y* is the unit pulse response at discrete time *i*. Scalar *n* specifies the identified model order. Scalar *nm*(*nr*) specifies the number of rows (columns) of a Hankel matrix. In peradc, *nr* is required to be larger than or equal to *nm*. Scalar *dt* specifies the data sampling interval.

[*eg,vp,vw,bp,bw,sg,vsg,bsg*]=peradc(*y,n,nm,nr,dt*) returns the variance and quadratic bias of the eradc identified parameters. The elements of vector *eg* are the eigenvalues of the eradc identified model. Vector *vp* (*vw*) is the variance of the eradc identified dampings (frequencies) corresponding to eigenvalue vector *eg*. Vector *bp* (*bw*) is the quadratic bias of the eradc identified dampings (frequencies) corresponding to eigenvalue vector *eg*. Vector *sg* is the singular value vector of the correlation matrix in eradc. The elements in vector *vsg* (*bsg*) is the normalized variance (quadratic bias) of the singular values in *sg*.

Example:

Calculate the variance and bias of the eradc identified parameters from the noisy pulse response samples of a single-input single-output second-order system with sampling interval *dt*=0.2, natural frequency $\omega_n = 1$, and damping factor $\zeta = 0.1$.

```
a=[0 1;-1 -0.2]; b=[0;1];c=[1 0];d=0;
pt=100;dt=0.2;t=[dt:dt:pt*dt]';
u=zeros(pt,1);u(1,1)=1.0;
y=lsim(a,b,c,d,u,t);
rand('normal')
y=y+0.04*rand(pt,1);
[eg,vp,vw,bp,bw,sg,vsg,bsg]=peradc(y,4,10,20,dt);
eg
clg
subplot(221)
bar(vw),title('Frequency variance')
bar(vp),title('Damping variance')
bar(bw),title('Frequency bias')
bar(bp),title('Damping bias')
pause
clg
subplot(121)
bar(vsg),title('Variance of singular values')
bar(bsg),title('Quadratic bias of singular values')
```

Algorithm:

The function peradc uses the algorithm from [Longman89,91]. In peradc the perturbation theory is applied to estimate the variance and quadratic bias of the ERA/DC identified parameters.

See also:

monera, eradc

References:

[1] Longman, R. W., Bergman, M. and Juang, J. N., "Variance and Bias Confidence Criteria for ERA Modal Parameter Identification," *Proceedings of the 1988 AAS/AIAA Astrodynamics Specialist Conference*, Minneapolis, Minnesota, August 1988.

[2] Longman, R. W., Lew, J. S., Tseng, D. H. and Juang, J. N., "Variance and Bias Computation for Improved Modal Identification Using ERA/DC," *Proceedings of the 1991 American Control Conference*, Boston, MA, June 1991.

Purpose:

Compute the prediction error based on identified parameters.

Synopsis:

[*errorf,yhatf*]=pred_err(*u,y,Yf,p,flag*)
[*errorb,yhatb*]=pred_err(*u,y,Yb,p,flag*)
[*errorf,yhatf, errorb,yhatb*]=pred_err(*u,y,Yf,Yb,p,flag*)

Description:

Given the input matrix u of dimension l x r, the output matrix y dimension l x m, and an estimate of p forward observer Markov parameters stored in Y_f, the forward prediction error computed in matrix form is

$$e_f = \underline{y}_f - Y_f V_f$$

where

$$\underline{y}_f = [y(0)\, y(1) \cdots y(l-1)]$$

$$Y_f = [D \quad C\overline{B} \quad C\overline{A}\overline{B} \quad \cdots \quad C\overline{A}^{p-1}\overline{B}]$$

$$V_f = \begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(l-1) \\ & v(0) & v(1) & \cdots & v(l-2) \\ & & \vdots & & \vdots \\ & & v(0) & \cdots & v(l-p-1) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} u(i) \\ y(i) \end{bmatrix}; \quad i = 0,1,\ldots,l-1$$

See functions okid and arx_bat for the definition of matrices shown above. The number of samples is l and the system has m outputs and r inputs. The computation is performed within a loop to reduce storage requirements at the expense of computation time. For cases where memory is not a problem *flag*=1 yields faster computation time. The function pred_err returns the prediction error in the vector *errorf* and the estimated measurement y in the vector *yhatf*.

Given an estimate of p backward observer Markov parameters stored in Y_b, the backward smoothing error computed in matrix form is

$$e_b = \underline{y}_b - Y_b V_b$$

where

$$\underline{y}_b = [y(0)\,y(1)\cdots y(l-p-2)]$$

$$Y_b = [D + C\tilde{B} \quad C\tilde{A}^{p-1}[\tilde{G}D \quad -\tilde{G}] \quad C\tilde{A}^{p-2}[\tilde{A}\tilde{B} + \tilde{G}D \quad -\tilde{G}] \quad \cdots \quad C[\tilde{A}\tilde{B} + \tilde{G}D \quad -\tilde{G}]]$$

$$V_b = \begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(l-p-2) \\ v(p) & v(p+1) & v(p+2) & \cdots & v(l-2) \\ \vdots & & \vdots & & \vdots \\ v(1) & v(2) & v(3) & \cdots & v(l-p-1) \end{bmatrix}$$

$$v(i) = \begin{bmatrix} u(i) \\ y(i) \end{bmatrix}; \quad i = 0,1,\ldots,l-1$$

See functions arx_b, arx_fb, okid_b and okid_fb for definition of the matrices shown above. The function pred_erb returns the backward smoothing error in the vector *errorb* and the smoothed measurement *y* in the vector *yhatf*.

Due to the strong similarity between the matrices V_f and V_b, the forward and backward errors may be computed simultaneously. The function pred_efb returns the prediction error in the vector *errorf*, the smoothing error in *errorb*, the estimated measurement *y* in *yhatf*, and the smoothed measurement *y* in *yhatb*.

Example:

```
m=1;r=1;p=2;L=5;flag=1;
a=[0 -0.16; 1 -1]; b=[0 1]'; c=[0 1]; d=0; G=[0.16 1]';
u=rand(L,r);
y=dlsim(a,b,c,d,u);
abar=a+G*c;bbar=[b+G*d -G];
Y=[d c*bbar c*abar*bbar];
[error,yhat]=pred_err(u,y,Y,p,flag);
```

See also:

arx_bat

Purpose:

Compute pulse response histories from general input and output data.

Synopsis:

[*ys,yo*]=pulse(*m,r,dt,u,y,p,n_pulse,description*);

Description:

The function pulse computes the unit pulse response samples (Markov parameters) from input and output time histories by using a time domain approach. The system input and output histories *u(t)* and *y(t)* must be stored as follows

$$u=\begin{bmatrix} u_{11}(0) & \cdots & u_{r1}(0) & \cdots & u_{1s}(0) & \cdots & u_{rs}(0) \\ u_{11}(1) & & u_{r1}(1) & & u_{1s}(1) & & u_{rs}(1) \\ u_{11}(2) & \cdots & u_{r1}(2) & \cdots & u_{1s}(2) & \cdots & u_{rs}(2) \\ \vdots & & \vdots & & \vdots & & \vdots \\ u_{11}(l-1) & \cdots & u_{r1}(l-1) & \cdots & u_{1s}(l-1) & \cdots & u_{rs}(l-1) \end{bmatrix}$$

$$y=\begin{bmatrix} y_{11}(0) & \cdots & y_{m1}(0) & \cdots & y_{1s}(0) & \cdots & y_{ms}(0) \\ y_{11}(1) & & y_{m1}(1) & & y_{1s}(1) & & y_{ms}(1) \\ y_{11}(2) & \cdots & y_{m1}(2) & \cdots & y_{1s}(2) & \cdots & y_{ms}(2) \\ \vdots & & & & & & \\ y_{11}(l-1) & \cdots & y_{m1}(l-1) & \cdots & y_{1s}(l-1) & \cdots & y_{ms}(l-1) \end{bmatrix}$$

where *r* is the number of inputs and *m* is the number of outputs, and $u_{ij}(t)(y_{ij}(t))$ is the *i*-th input (output) of the *j*-th test at discrete time *t*. Multiple experiments are allowed in using this function file. The input variable *p* is the desired number of independent observer Markov parameters to be identified from input and output time histories. Given the desired number *p*, the maximum number of the system order is *p*m*. The input variable *n_pulse* is the desired length of the pulse response time histories to be computed. The input variable *description* is a short descriptive tag for the current data set being analyzed and also serves as a flag. If *description = 'inverse'*, it computes the pulse response histories for the backward model which can be used to realize the inverse of a system state matrix (see description for function okid_b). This function works for stable and unstable systems. All calculations are performed in the time-domain. The output of this function is the system pulse response histories *ys* of dimension n_*pulse* by *mr* and the observer pulse response histories *yo*. of dimension *n_pulse* by *mm*.

Example:

This example is to identify the pulse response time histories of a three-mass-spring-dashpot system from two-input and one-output data.

```
k1=1.0;k2=2.0;k3=3.0;
m1=1.0;m2=1.0;m3=1.0;ratio=2*0.005;
K=[k1+k2  -k2   0 ; -k2 k2+k3 -k3; 0   -k3   k3];
```

```
Khalf=sqrtm(K);Damp=ratio*Khalf;
Ac=[zeros(3,3) eye(3,3);-K -Damp];
Bc=[zeros(3,2);1 0; 0 1; 0 0];
C=[zeros(1,5) 1];
dt=1.0;pt=200;p=10;
[m,n]=size(C);[n,r]=size(Bc);
D=zeros(m,r);
t=[dt:dt:pt*dt]';
[A,B]=c2d(Ac,Bc,dt);
rand('normal');
u=rand(pt,r);y=dlsim(A,B,C,D,u);
[ys,yo]=pulse(m,r,dt,u,y,p,pt,'pulse');
%[ys,yo]=pulse(m,r,dt,u,y,p,pt,'inverse');%for backward only
```

Algorithm:

This is a time-domain approach. First, the forward or backward observer Markov parameters are identified (see arx_bat, arx_b), as opposed to the system itself, such that the corresponding observer has all its poles placed at the origin. From these identified observer Markov parameters, the system pulse response is recovered (see mar_com). For details see references.

See also:

arx_bat,arx_b,mar_com,m2p,okid,okid_b

References:

[1] Chen, C. W., Huang, J.-K., Phan, M., and Juang, J.-N.,"Integrated System Identification and Modal State Estimation for Control of Large Flexible Space Structures," *Journal of Guidance, Control and Dynamics,* Vol. 15, No. 1, Jan.-Feb. 1992, pp. 88-95.

[2] Phan, M., Horta, L. G., Juang, J.-N., and Longman, R. W., "Linear System Identification Via an Asymptotically Stable Observer," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, Aug. 1991, pp. 1180-1194, and NASA Technical Paper 3164, 1991, and to appear in the *Journal of Optimization Theory and Application.*

[3] Juang, J.-N., Phan, M., Horta, L. G.,and Longman, L. G., "Identification of Observer and Kalman Filter Markov Parameters: Theory and Experiments," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, Aug. 1991, pp. 1195-1207.

[4] Phan, M., Juang, J.-N., and Longman, R. W., "On Markov Parameters in System Identification," *NASA Technical Memorandum TM-104156,* Langley Research Center, Hampton, VA., Oct. 1991.

[5] Juang, J.-N. and Phan, M., "Identification of Backward Observer Markov Parameters: Theory and Experiments," *NASA Technical Memorandum TM-107632,* Langley Research Center, Hampton, VA., May 1992.

Purpose:

Compute the left correlation matrix associated with the feedback control input for observer/controller identification.

Synopsis:

[*ufbVt*] = ryucovar(*y*,*ufb*,*ue*,*p*)

Description:

The data is stored as column matrices. For a system with m outputs, the closed-loop output data matrix y contains m columns and as many rows as the number of data points available. The data matrix *ufb* contains the feedback control signal, *ue* contains the additive excitation input signal in the same format as y. Let the given data be arranged as

$$Y = \begin{bmatrix} y(0) & y(1) & y(2) & \cdots & y(N-1) \end{bmatrix}$$

$$U_{fb} = \begin{bmatrix} u_{fb}(0) & u_{fb}(1) & u_{fb}(2) & \cdots & u_{fb}(N-1) \end{bmatrix}$$

$$R = \begin{bmatrix} u_e(0) & u_e(1) & u_e(2) & \cdots & u_e(N-1) \end{bmatrix}$$

The function ryucovar computes the following correlation matrix

$$ufbVt = U_{fb}V^T$$

where the data matrix V is defined as

$$V = \begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(p) & u(p+1) & \cdots & u(N-1) \\ & z(0) & z(1) & \cdots & z(p-1) & z(p) & \cdots & z(N-2) \\ & & z(0) & \cdots & z(p-2) & z(p-1) & \cdots & z(N-3) \\ & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & z(0) & z(1) & \cdots & z(N-p-1) \end{bmatrix}$$

and $z(k)$ is defined as

$$z(k) = \begin{bmatrix} u_{fb}(k) + u_e(k) \\ y(k) \end{bmatrix}$$

As indicated, the number N specifies the number of data points to be used in the computation. N can be less than the total number of data points available in *ufb*, *ue*, and y. The matrix product returned by the function is used in the computation of the observer/controller Markov parameters in the function ocid.

Example:

```
ufb=[1 -2 1 3 1 3 4 5 6]';
y=[0 3 2 1 3 1 4 4 6]';
ue=[-1 0 2 -1 2 3 5 0 7]';
[ufbVt] = ryucovar(y,ufb,ue,2)
ufbVt =

   70   42   23   15   26
```

Algorithm:

To save memory space, the summations involved in the product $U_{fb}V^T$ are performed using inner matrix product multiplication.

See also:

ocid, arxc, mar_yoc, mar_oc, y_closed, separate

Purpose:

Separate a given matrix sequence into two sequences with prescribed formats.

Synopsis:

[*Y*1,*Y*2]=separate(*Y*,*q*,*m*1,*m*2)

Description:

Given a matrix *Y* in the following format

$$Y=[Y_1(1)\quad Y_1(2)\quad Y_2(1)\quad Y_2(2)\quad \cdots\quad \cdots\quad Y_n(1)\quad Y_n(2)]$$

this function returns *Y*1 and *Y*2 which are

$$Y1=[Y_1(1)\quad Y_2(1)\quad \cdots\quad Y_n(1)]$$

$$Y2=[Y_1(2)\quad Y_2(2)\quad \cdots\quad Y_n(2)]$$

Each $Y_i(1)$ has dimensions $q\times m1$, and each $Y_i(2)$ has dimensions $q\times m2$.

Example:

```
Y=rand(2,9)

Y =
 Columns 1 through 7

   0.9304   0.5269   0.6539   0.7012   0.7622   0.0475   0.3282
   0.8462   0.0920   0.4160   0.9103   0.2625   0.7361   0.6326

  Columns 8 through 9

   0.7564   0.3653
   0.9910   0.2470

[Y1,Y2]=separate(Y,2,1,2)
Y1 =

   0.9304   0.7012   0.3282
   0.8462   0.9103   0.6326

Y2 =

   0.5269   0.6539   0.7622   0.0475   0.7564   0.3653
   0.0920   0.4160   0.2625   0.7361   0.9910   0.2470
```

See also:

arxc, mar_yoc, mar_oc, ocid, ryucovar, y_closed

Purpose:

Compute modal observability matrix and singular values of the modal participation to the pulse response samples.

Synopsis:

[*svm, obsm*]=svpm(*lambda,bm,cm,n*)

Description:

Consider the discrete model in the modal coordinates

$$x_m(k+1) = \Lambda x_m(k) + B_m u(k)$$
$$y = C_m x(k) + Du(k)$$

with r inputs and m outputs, where Λ is a diagonal matrix containing the eigenvalues, $\lambda_i (i = 1,2,...,n)$, of the system matrix. The modal observability matrix is computed by

$$obsm = \begin{bmatrix} c_i \\ c_i\lambda_i \\ \vdots \\ c_i\lambda_i^{n-1} \end{bmatrix}$$

Function [*svm, obsm*]=svpm(*lambda,bm,cm,n*) returns the complex modal observability matrix and a normalized singular value vector *svm* (see algorithm) to quantify the importance of each individual mode, for given system eigenvalues in the vector *lambda*, the modal input matrix *bm*, modal output matrix *cm* and the desired length *n*. Note that the maximum singular value is used to normalize the vector *svm*.

Example:

A three-mass-spring-dashpot system from two-input and three-output data is used.

```
k1=1.0;k2=2.0;k3=3.0;
m1=1.0;m2=1.0;m3=1.0;ratio=2*0.005;
K=[k1+k2  -k2   0 ; -k2 k2+k3 -k3; 0   -k3   k3];
Khalf=sqrtm(K);Damp=ratio*Khalf;
Ac=[zeros(3,3) eye(3,3);-K -Damp];
Bc=[zeros(3,2);1 0; 0 1; 0 0];
C=[zeros(3,3) eye(3,3)];
dt=1.0;pt=60;
[A,B]=c2d(Ac,Bc,dt);
[V,lambda]=eig(A);
bm=V\B;cm=C*V;lambda=diag(lambda);
[sv,obsvm] = svpm(lambda,bm,cm,6);
sv'
sv' = 0.2898  0.2898  1.0000  1.0000  0.1506  0.1506
```

Algorithm:

For a linear system, the map from input u to output y can be fully described by the Markov parameter sequence

$$Y_m = \begin{bmatrix} D & C_m B_m & C_m \Lambda B_m & \cdots & C_m \Lambda^{\ell-2} B_m \end{bmatrix}$$

This sequence is coordinate independent and unique. Let the input and output matrices be partitioned as

$$B_m = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}; \; C_m = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}$$

where n is the number of modal coordinates, b_i $(i=1,2,...,n)$ a row vector of length r and c_i a column vector of length m. Each individual Markov parameter can then be written as a combination of n components contributed from different modal coordinates.

$$C_m \Lambda B_m = \sum_{i=1}^{n} c_i \lambda_i b_i$$

Therefore, each coordinate has a sequence of Markov parameters described as follows

$$Y_{mi} = \begin{bmatrix} 0 & c_i b_i & C_i \lambda_i B_i & \cdots & C_i \lambda_i^{\ell-2} B_i \end{bmatrix}; \; i = 1,2,..., n$$

The total Markov parameter sequence becomes

$$Y_m = \sum_{i=0}^{n} Y_{mi}; \; Y_{m0} = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \end{bmatrix}$$

From this representation, it is obvious that each modal coordinate contributes to the pulse response sample by the individual modal sequence Y_{mi}, which can be quantified by taking its maximum singular value, i.e.,

$$svm = \sqrt{|c_i|}(1+|\lambda_i|+|\lambda_i^2|+...+|\lambda_i^{\ell-2}|)\sqrt{|b_i|} \approx \sqrt{|c_i|}\sqrt{|b_i|} \Big/ (1-|\lambda_i|)$$

where $|\lambda_i|$ is assumed to be less than 1.

See also:

era, eradc, match

Purpose:

Identify a state space model from input and output data via a state vector realization algorithm.

Synopsis:

[*a,b,c,d,eg,sg*]=svra(*u,y,n,nm,nr,dt,k*);

Description:

The function svra identifies a state-space model of a multi-input and multi-output linear, time-invariant system from a set of rich input response data. The *i*-th row vector of matrix *u* (*y*) is the system input (output) at discrete time *i*. Scalar *n* specifies the identified model order. Scalar *nm* specifies the number of sample shift in constructing the rows of a Hankel matrix and it is required to be even, whereas *nr* specifies the column number of the Hankel matrix. Integers $nm \times m$(*m* is the number of outputs) and *nr* are required to be not smaller than the chosen model order *n*. Also, $n + nm \times r$(*r* is the number of inputs) needs to be smaller than *nr*. Scalar *dt* denotes the data sampling interval, and the data used for realization starts from the *k*-th discrete time. [*A,B,C,D,eg,sg*]=svra(*u,y,n,nm,nr,dt,k,flag*) returns an *n*th-order linear, time-invariant identified discrete model:

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

Vector *eg* contains modal parameters of the identified model including frequencies (Hz) in the first column and damping ratios (%) in the second column. The third column of *eg* gives the eigenvalues of the corresponding continuous time model. Note the identified discrete model can be easily transformed to a continuous time model. Vector *sg*, whose elements are singular values of the Hankel matrix, can be used as reference to choose the model order *n*.

Examples:

i) Use svra to identify a system model from a set of random input response data for a single input, single output second order system with sampling interval *dt*=0.3, natural frequency $\omega_n = 1$, and damping factor $\zeta = 0.1$.

ii) Transfer the identified discrete model to a continuous-time model and plot the original system output and the output from the svra identified model.

```
a=[0 1;-1 -0.2]; b=[0;1];c=[1 0];d=0;
pt=100;dt=0.4;t=[dt:dt:dt*pt]';
rand('normal');
u=rand(pt,1);
y=lsim(a,b,c,d,u,t);
[a1,b1,c1,d1,sg,eg]=svra(u,y,2,20,40,dt,10);
y1=dlsim(a1,b1,c1,d1,u);
y2=[y y1];
clg
plot(y2),title('Random input response')
pause
error=y-y1;
clg
plot(error),title('Error')
pause
sg
```

Algorithm:

The function svra uses the state vector realization algorithm from [Moonen89 Lew91]. It uses output and input data to form the measurement matrix as

$$H=\begin{bmatrix} u_k & u_{k+1} & \cdots & u_{k+nr-1} \\ y_k & y_{k+1} & \cdots & y_{k+nr-1} \\ u_{k+1} & u_{k+2} & \cdots & u_{k+nr} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+nr} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k+nm-1} & u_{k+nm} & \cdots & u_{k+nm+nr-2} \\ y_{k+nm-1} & y_{k+nm} & \cdots & y_{k+nm+nr-2} \end{bmatrix}$$

where u_i and y_i denote r-dimensional input vector and m-dimensional output vector at time i, respectively. In svra, the SVD of H is used for a state vector realization [Moonen89], i.e. $x_{k+i}, x_{k+i+1}, \ldots, x_{k+j-1}, x_{k+j}$; $j >> i$. It then uses the following equation

$$\begin{pmatrix} x_{k+i+1} & \cdots & x_{k+j} \\ y_{k+i} & \cdots & y_{k+j-1} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x_{k+i} & \cdots & x_{k+j-1} \\ u_{k+i} & \cdots & u_{k+j-1} \end{pmatrix}$$

to identify a state-space model.

References:

[1] Moonen, M., Demoor, B., Vandenberghe, L. and Vandewalle, J., "On- and Off-Line Identification of Linear State-Space Models," *International Journal of Control,* Vol. 49, 1989, pp 219-232.

[2] Lew, J. S., Juang, J. N. and Longman, R. W., "Comparison of Several System Identification Methods for Flexible Structures," *Proceedings of the 32nd Structures, Structural Dynamics and Materials Conference*, Baltimore, MD, April 1991, pp. 2304-2318.

Purpose:

Compute a stacked matrix with inputs and outputs.

Synopsis:

[*vst*]=uy_stack(*u,y,p*)

Description:

Given a set of data with *l* samples, *r* inputs, *m* outputs, and an assumed system order of *p*m*, the function stacks a matrix using the inputs and outputs as follows

$$V=\begin{bmatrix} u(0) & u(1) & u(2) & \cdots & u(l-1) \\ & v(0) & v(1) & \cdots & v(l-2) \\ & & \vdots & & \vdots \\ & & v(0) & \cdots & v(l-p-1) \end{bmatrix}$$

$$v(i)=\begin{bmatrix} u(i) \\ y(i) \end{bmatrix};\quad i=0,1,\ldots,l-1$$

The matrix dimension is $[(r+m)p+r]\times l$

Example:

```
u=[0 1 2 3 4 5]';
y=[6 7 8 9 10 11]';
p=2;
[vst]=uy_stack(u,y,p)
vst =
   0   1   2   3   4   5
   0   0   1   2   3   4
   0   6   7   8   9  10
   0   0   0   1   2   3
   0   0   6   7   8   9
```

See also:

arx_bat

Purpose:

Reconstruct feedback control input and closed-loop response using ocid-identified system, observer gain, and controller gain matrices.

Synopsis:

[*ufb_rec,y_rec*]=y_closed(*A,B,C,D,G,F,y,ufb,ue*)

Description:

The function reconstructs the data histories *ufb* and *y* using the ocid-identified system matrices *A*, *B*, *C*, *D*, identified observer gain *G*, and identified existing controller gain *F*. The reconstructed $\hat{u}_{fb}(k)$, $\hat{y}(k)$ stored as column matrices in *ufb_rec*, *y_rec*, respectively, are computed from the following equations.

$$\hat{x}(k+1) = (A+GC)\hat{x}(k) + (B+GD)\left[u_{fb}(k) + u_e(k)\right] - Gy(k)$$

$$\hat{y}(k) = C\hat{x}(k) + D\left[u_{fb}(k) + u_e(k)\right]$$

$$\hat{u}_{fb}(k) = -F\hat{x}(k)$$

Note that the above state equation equation is the same as the usual observer equation expressed in terms of the prediction error $y(k) - \hat{y}(k)$

$$\begin{aligned}\hat{x}(k+1) &= A\hat{x}(k) + B\left[u_{fb}(k) + u_e(k)\right] - G\left[y(k) - \hat{y}(k)\right] \\ &= A\hat{x}(k) + B\left[u_{fb}(k) + u_e(k)\right] - Gy(k) + G\left\{C\hat{x}(k) + D\left[u_{fb}(k) + u_e(k)\right]\right\} \\ &= (A+GC)\hat{x}(k) + (B+GD)\left[u_{fb}(k) + u_e(k)\right] - Gy(k)\end{aligned}$$

Algorithm:

The function is programmed using a Matlab function dlsim.

See also:

ocid

Purpose:

Compute estimated outputs using an identified observer.

Synopsis:

[*t,yhat*]=y_esti(*a,b,c,d,G,u,y,dt,npts,flag*)

Description:

Any typical observer has the following form

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) - G[y(k) - \hat{y}(k)]$$
$$\hat{y}(k) = C\hat{x}(k) + Du(k)$$

where $\hat{x}(k)$ is the estimate of the state *x(k)* and $\hat{y}(k)$ is the estimate of the output *y(k)*. The system matrices *A*, *B*, *C*, *D*, and the gain matrix *G* may be identified from input data *u(k)* and output data *y(k)* using the function okid. The function [*t,yhat*]=y_esti(*A,B,C,D,G,u,y,dt,npts,flag*) returns a column vector, *t*, for the time period corresponding to the desired number of sample points (*npts*), and an $npts \times m$ matrix, *yhat*, for the estimated outputs. The *flag* is set to 1 for plotting. The user will be prompted with the desired number of sample points to be reconstructed. Plots will be given to show the comparison between the real test outputs and the estimated outputs.

See also:

okid, okid_b, okid_fb, y_pred

Purpose:

Compute predicted outputs using an identified system model.

Synopsis:

[t,yhat]=y_pred(*a,b,c,d,u,y,dt,npts,flag*)

Description:

The identified discrete system with the sampling time, *dt*, has the following form

$$x(k+1) = Ax(k) + Bu(k)$$
$$y(k) = Cx(k) + Du(k)$$

where *x(k)* is the state vector. The system matrices *A, B, C, D,* and the gain matrix may be identified from input data *u(k)* and output data *y(k)* using the function okid. The function *[t,yhat]*=y_pred(*A,B,C,D,u,y,dt,npts,flag*) returns a column vector, *t*, for the time period corresponding to the desired number of sample points (*npts*), and an $npts \times m$ matrix, *yhat*, for the reconstructed outputs. The outputs *y(k)* are used here for comparison with the reconstructed outputs. The *flag* is set to 1 for plotting. The user will be prompted with the desired number of sample points to be reconstructed. Plots will be given to show the comparison between the real test outputs and the reconstructed outputs.

See also:

okid, okid_b, okid_fb, y_esti

Purpose:

Compute the left and right correlation matrices for least squares identification problem.

Synopsis:

[*ybarf,vbarf*]=yucovar(*u,y,p,flag*)
[*ybarb,vbarb*]=yucov_b(*u,y,p*)
[*ybarf,vbarf, ybarb, vbarb*]=yucovfb(*u,y,p*)

Description:

Given l sample points of a system with m outputs, r inputs, and an assumed system order of $p*m$, the input matrix u is of dimension l x r whereas the output matrix y is l x m. The flag is set to 1 for long histories. Let $\underline{y}_b$ be partitioned as

$$\underline{y}_f = [y(0) \quad y(1) \quad y(2) \quad \cdots \quad y(\ell-1)];$$

where each measurement $y(i)(i=1,2,...)$ has the dimension $m \times 1$. The least squares problem for the forward observer Markov parameters is posed as

$$\underline{y}_f V_f^T = Y_f V_f V_f^T$$

The matrices $\underline{y}_f V_f^T$ and $V_f V_f^T$ have the same structure as the cross and auto correlation as seen in the following

$$\underline{y}_f V_f^T = \left[\sum_{i=0}^{l-1} y(i)u^T(i) \quad \sum_{i=0}^{l-2} y(i+1)v^T(i) \quad \cdots \quad \sum_{i=0}^{l-p-1} y(i+p)v^T(i) \right]$$

$$V_f V_f^T = \begin{bmatrix} \sum_{i=0}^{l-1} u(i)u^T(i) & \sum_{i=0}^{l-2} u(i+1)v^T(i) & \sum_{i=0}^{l-3} u(i+2)v^T(i) & \cdots & \sum_{i=0}^{l-p-1} u(i+p)v^T(i) \\ \sum_{i=0}^{l-2} v(i)u^T(i+1) & \sum_{i=0}^{l-2} v(i)v^T(i) & \sum_{i=0}^{l-3} v(i+1)v^T(i) & \cdots & \sum_{i=0}^{l-p-1} v(i+p-1)v^T(i) \\ \sum_{i=0}^{l-3} v(i)u^T(i+2) & \sum_{i=0}^{l-3} v(i)v^T(i+1) & \sum_{i=0}^{l-3} v(i)v^T(i) & \cdots & \sum_{i=0}^{l-p-1} v(i+p-2)v^T(i) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{i=0}^{l-p-1} v(i)u^T(i+p) & \sum_{i=0}^{l-p-1} v(i)v^T(i+p-1) & \sum_{i=0}^{l-p-1} v(i)v^T(i+p-2) & \cdots & \sum_{i=0}^{l-p-1} v(i)v^T(i) \end{bmatrix}$$

where

$$v(i) = \begin{bmatrix} u(i) \\ y(i) \end{bmatrix}; \quad i = 0,1,...,l-1$$

The summations are performed using inner products to reduce computational time.

[*ybarf,vbarf*]=yucovar(*u,y,p,flag*) returns $\underline{y}_f V_f^T$ and $V_f V_f^T$ in *ybarf* and *vbarf* respectively.

For the backward approach, let $\underline{y}_b$ be partitioned as

$$\underline{y}_b = [y(0) \quad y(1) \quad y(2) \quad \cdots \quad y(\ell - p)];$$

where each measurement$y(i)(i=1,2,...)$ has the dimension $m \times 1$. The least squares problem for the backward observer Markov parameters is posed as

$$\underline{y}_b V_b^T = Y_b V_b V_b^T$$

The matrices $\underline{y}_b V_b^T$ and $V_b V_b^T$ have the same structure as the cross and auto correlation as seen in the following

$$\underline{y}_b V_b^T = \left[\sum_{i=0}^{l-p} y(i)u^T(i) \quad \sum_{i=0}^{l-p} y(i)v^T(i+p) \quad \cdots \quad \sum_{i=0}^{l-p} y(i)v^T(i+1) \right]$$

$$V_b V_b^T = \begin{bmatrix} \sum_{i=0}^{l-p} u(i)u^T(i) & \sum_{i=0}^{l-p} u(i)v^T(i+p) & \sum_{i=0}^{l-p} u(i)v^T(i+p-1) & \cdots & \sum_{i=0}^{l-p} u(i)v^T(i+1) \\ \sum_{i=0}^{l-p} v(i+p)u^T(i) & \sum_{i=0}^{l-p} v(i+p)v^T(i+p) & \sum_{i=0}^{l-p} v(i+p)v^T(i+p-1) & \cdots & \sum_{i=0}^{l-p} v(i+p)v^T(i+1) \\ \sum_{i=0}^{l-p} v(i+p-1)u^T(i) & \sum_{i=0}^{l-p} v(i+p-1)v^T(i+p) & \sum_{i=0}^{l-p} v(i+p-1)v^T(i+p-1) & \cdots & \sum_{i=0}^{l-p} v(i+p-1)v^T(i+1) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{i=0}^{l-p} v(i+1)u^T(i) & \sum_{i=0}^{l-p} v(i+1)v^T(i+p-1) & \sum_{i=0}^{l-p} v(i+1)v^T(i+p-1) & \cdots & \sum_{i=0}^{l-p} v(i+1)v^T(i+1) \end{bmatrix}$$

The summations are performed using inner products to reduce computational time.

[*ybarb,vbarb*]=yucov_b(*u,y,p*) returns $\underline{y}_b V_b^T$ and $V_b V_b^T$ in *ybarb* and *vbarb* respectively. Note that there is no *flag* in this function.

The strong similarity among these matrices $\underline{y}_f V_f^T$, $\underline{y}_b V_b^T$, $V_f V_f^T$ and $V_b V_b^T$ suggests the possibility of simultaneously computing them in one single function.

[*ybarf,vbarf, ybarb,vbarb*]=yucovfb(*u,y,p*) returns $\underline{y}_f V_f^T$, $V_f V_f^T$, $\underline{y}_b V_b^T$ and $V_b V_b^T$ in *ybarf, vbarf, ybarb*, and *vbarb*, respectively.

Example:

```
y=[5 6 7 8 9]';
p=2,flag=0;
[ybar,ubar]=yucovar(u,y,p,flag)
ybar =
   80   50  200   26  146
ubar =
   30   20   70   11   56
   20   14   44    8   38
   70   44  174   23  128
   11    8   23    5   20
   56   38  128   20  110
```

See also:

arx_bat, arx_b, arx_fb

Purpose:

Compute the left and right output residual correlation matrices.

Synopsis:

[*yvt,vvt*]=yycovar(*y,p,iexp*)

Description:

Given l sample points of a system with m output residuals (the stochastic part of the system) and an assumed system order of $p*m$, the output residuals y is l x m. The flag is set to 1 for long histories. Let y be partitioned as

$$\underline{y} = [y(0) \quad y(1) \quad y(2) \quad \cdots \quad y(\ell-1)];$$

where each $y(i)(i=1,2,...)$ has the dimension $m \times 1$. The least squares problem for the state estimator Markov parameters which whiten the residual is posed as

$$\underline{y}V^T = YVV^T$$

the matrices $\underline{y}V^T$ and VV^T have the same structure as the cross and auto correlation as seen in the following

$$\underline{y}V^T = \left[\sum_{i=0}^{l-1} y(i+1)y^T(i) \quad \sum_{i=0}^{l-2} y(i+2)y^T(i) \quad \cdots \quad \sum_{i=0}^{l-p-1} y(i+p)y^T(i)\right]$$

$$VV^T = \begin{bmatrix} \sum_{i=0}^{l-2} y(i)y^T(i) & \sum_{i=0}^{l-3} y(i+1)y^T(i) & \sum_{i=0}^{l-4} y(i+2)y^T(i) & \cdots & \sum_{i=0}^{l-p-2} y(i+p-1)y^T(i) \\ \sum_{i=0}^{l-3} y(i)y^T(i+1) & \sum_{i=0}^{l-3} y(i)y^T(i) & \sum_{i=0}^{l-4} y(i+1)y^T(i) & \cdots & \sum_{i=0}^{l-p-2} y(i+p-2)y^T(i) \\ \sum_{i=0}^{l-4} y(i)y^T(i+2) & \sum_{i=0}^{l-4} y(i)y^T(i+1) & \sum_{i=0}^{l-4} y(i)y^T(i) & \cdots & \sum_{i=0}^{l-p-2} y(i+p-3)y^T(i) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{i=0}^{l-p-2} y(i)y^T(i+p) & \sum_{i=0}^{l-p-2} y(i)y^T(i+p-1) & \sum_{i=0}^{l-p-2} y(i)y^T(i+p-2) & \cdots & \sum_{i=0}^{l-p-2} y(i)y^T(i) \end{bmatrix}$$

where

$$\underline{y} = [y(1) \quad y(2) \quad \cdots \quad y(\ell-1)]$$

The summations are performed using inner products to reduce computational time.

[*yvt,vvt*]=yycovar(*y,p,iexp*) returns $\underline{y}V^T$ and VV^T respectively in *yvt* and *vvt*.

Example: Some random numbers are created to verify the computation of this function.

```
rand('normal');
y=rand(2,11);
v=[y(:,1:10)];
y1=y(:,2:11);
y1*v'
ans =
  -4.6867e-01   2.4383e+00
  -1.0534e+00  -3.8444e+00
v*v'
ans =
   2.4726e+00  -2.0860e+00
  -2.0860e+00   1.3314e+01

y=y';
[ybar,ubar]=yycovar(y,1,0)

ybar =
  -4.6867e-01   2.4383e+00
  -1.0534e+00  -3.8444e+00
ubar =
   2.4726e+00  -2.0860e+00
  -2.0860e+00   1.3314e+01
```

See also:

k_abcd, arx_bat

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1992	Technical Memorandum

4. TITLE AND SUBTITLE
System/Observer/Controller Identification Toolbox

5. FUNDING NUMBERS
WU 590-14-21-01

6. AUTHOR(S)
Jer-Nan Juang, Lucas Horta, and Minh Phan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23665-5225

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-107566

11. SUPPLEMENTARY NOTES
Jer-Nan Juang: Langley Research Center, Hampton, VA; Lucas Horta: Langley Research Center, Hampton, VA; Minh Phan: Lockheed Engineering & Sciences Co., Hampton, VA.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited

Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
System Identification is the process of constructing a mathematical model from input and output data for a system under testing, and characterizing the system uncertainties and measurement noises. The mathematical model structure can take various forms depending upon the intended use. The SYSTEM/OBSERVER/CONTROLLER IDENTIFICATION TOOLBOX (SOCIT) is a collection of functions, written in MATLAB language and expressed in M-files, that implements a variety of modern system identification techniques. For an open-loop system, the central features of the SOCIT are functions for identification of a system model and its corresponding forward and backward observers directly from input and output data. The system and its observers are represented by a discrete model. The identified model and observers may be used for controller design of linear systems as well as identification of modal parameters such as dampings, frequencies, and mode shapes. For a closed-loop system, the central features of the SOCIT include identification of an open loop model, an observer and its corresponding controller gain directly from input and output data.

14. SUBJECT TERMS	15. NUMBER OF PAGES
System Identification, Observer Identification, Controller Identification, Modal Testing	86
	16. PRICE CODE: A05

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		