B

MATLAB® Command Summaries

MATLAB is a software package that is nearly a de facto standard for linear systems analysis. It was written *by* linear systems researchers and practitioners *for* linear systems researchers and practitioners. It therefore has included in it many specialized commands that general-purpose mathematical packages do not contain.

Throughout this book, certain terms have been marked with an *M* superscript (such as $rank^M$). These are terms that describe linear algebraic or control systems operations and procedures that are easily performed with the aid of MATLAB. Given in this appendix are the "help" texts for the commands that are indexed in the book. These descriptions are the responses that a user will receive upon typing `help <command>` at the command prompt.

MATLAB is a registered trademark of The MathWorks, Inc. For further information, users can contact

The Mathworks, Inc.
24 Prime Park Way
Natick, MA 01760-1500
Phone: (508) 647-7000
Fax: (508) 647-7001
Email: info@mathworks.com

There are also a great many textbooks and other references on the use of MATLAB in linear systems analysis and control system design (as well as in other specialties in science and engineering). Some of these references are included at the end of this appendix.

Command Reference

ACKER Pole placement gain selection using Ackermann's formula.

```
   K = ACKER(A,B,P)  calculates the feedback gain matrix K such that
   the single input system


           .
           x = Ax + Bu

   with a feedback law of  u = -Kx  has closed loop poles at the
   values specified in vector P, i.e.,  P = eig(A-B*K).

   Note: This algorithm uses Ackermann's formula.  This method
   is NOT numerically reliable and starts to break down rapidly
   for problems of order greater than 10, or for weakly controllable
   systems.  A warning message is printed if the nonzero closed-loop
   poles are greater than 10 from the desired locations specified
   in P.

   See also  PLACE.

   Algorithm is from page 201 of:
   Kailath, T.  "Linear Systems", Prentice-Hall, 1980.

   Copyright (c) 1986-96 by The MathWorks, Inc.
```

BALREAL Gramian-based balancing of state-space realizations.

```
   SYSb = BALREAL(SYS) returns a balanced state-space
   realization of the reachable, observable, stable system SYS.

   [SYSb,G,T,Ti] = BALREAL(SYS) also returns a vector G containing
   the diagonal of the Gramian of the balanced realization.  The
   matrices T is the state transformation xb = Tx used to convert SYS
   to SYSb, and Ti is its inverse.

   If the system is normalized properly, small elements in the balanced
   Gramian G indicate states that can be removed to reduce the model
   to lower order.

   See also  MODRED, GRAM, SSBAL.
   J.N. Little 3-6-86
   Revised 12-30-88
       Alan J. Laub 10-30-94
       P. Gahinet 6-27-96
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.2 $  $Date: 1996/10/18 15:16:05 $

   Reference:
   [1] Laub, A.J., M.T. Heath, C.C. Paige, and R.C. Ward,
       ``Computation of System Balancing Transformations and Other
       Applications of Simultaneous Diagonalization Algorithms,''
       IEEE Trans. Automatic Control, AC-32(1987), 115--122.
```

```
C2D  Conversion of continuous-time systems to discrete time.
   SYSD = C2D(SYSC,TS,METHOD)  converts the continuous system SYSC
   to a discrete-time system SYSD with sample time TS.  The string
   METHOD selects the discretization method among the following:
      'zoh'       Zero-order hold on the inputs.
      'foh'       Linear interpolation of inputs (triangle appx.)
      'tustin'    Bilinear (Tustin) approximation.
      'prewarp'   Tustin approximation with frequency prewarping.
                  The critical frequency Wc is specified last as in
                  C2D(SysC,Ts,'prewarp',Wc)
      'matched'   Matched pole-zero method (for SISO systems only).
   The default is 'zoh' when METHOD is omitted.

   If SYS is a delay-free state-space model with initial state x0, the
   matching initial state for its FOH discretization is  x0-G*u(1,:)
   where the matrix G is given by   [SYSD,G] = C2D(SYSC,TS,'foh').

   See also D2C, D2D.

   Other syntax
   C2D  Conversion of state space models from continuous to discrete time.
   [Phi, Gamma] = C2D(A,B,T)  converts the continuous-time system:


        .
        x = Ax + Bu

   to the discrete-time state-space system:

        x[n+1] = Phi * x[n] + Gamma * u[n]

   assuming a zero-order hold on the inputs and sample time T.

   See also D2C.

   J.N. Little 4-21-85
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.4 $  $Date: 1996/08/28 21:49:08 $


CANON  Canonical state-space realizations.
   CSYS = CANON(SYS,TYPE)  computes a canonical state-space
   realization CSYS of the LTI model SYS.  The string TYPE
   selects the type of canonical form:
     'modal'    :  Modal canonical form where the system
                   eigenvalues appear on the diagonal.
                   The state matrix A must be diagonalizable.
     'companion':  Companion canonical form where the characteristic
                   polynomial appears in the right column.

   [CSYS,T] = CANON(SYS,TYPE)  also returns the state transformation
   matrix T such that z = Tx where z is the new state.  This syntax
   is only meaningful when SYS is a state-space model.

   The modal form is useful for determining the relative controll-
   ability of the system modes.  Note: the companion form is ill-
   conditioned and should be avoided if possible.
```

```
See also  SS2SS, CTRB, and CTRBF.
```

```
Clay M. Thompson  7-3-90
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.2 $  $Date: 1996/10/18 15:16:11 $


CARE  Solve continuous-time algebraic Riccati equations.
   [X,L,G,RR] = CARE(A,B,Q,R,S,E)  computes the unique symmetric
   stabilizing solution X of the continuous-time algebraic Riccati
   equation
                                -1
      A'XE + E'XA - (E'XB + S)R  (B'XE + S') + Q = 0

   or, equivalently,
                          -1             -1                      -1
      F'XE + E'XF - E'XBR  B'XE + Q - SR  S' = 0  with  F:=A-BR  S'.

   When omitted, R,S and E are set to the default values R=I, S=0,
   and E=I.  Additional optional outputs include the gain matrix
               -1
          G = R  (B'XE + S') ,

   the vector L of closed-loop eigenvalues (i.e., EIG(A-B*G,E)),
   and the Frobenius norm RR of the relative residual matrix.

   [X,L,G,REPORT] = CARE(A,B,Q,...,'report')  turns off error
   messages and returns a success/failure diagnosis REPORT instead.
   The value of REPORT is
     * -1 if Hamiltonian matrix has eigenvalues too close to jw axis
     * -2 if X=X2/X1 with X1 singular
     * the relative residual RR when CARE succeeds.

   [X1,X2,L,REPORT] = CARE(A,B,Q,...,'implicit')  also turns off
   error messages, but now returns matrices X1,X2 such that X=X2/X1
   and [X1;X2] has orthonormal columns.  REPORT=0 indicates success.

   See also  DARE.

   Author(s): Alan J. Laub (1993)  (laub@ece.ucsb.edu)
              with key contributions by Pascal Gahinet, Cleve Moler,
              and Andy Potvin
   Revised: 94-10-29, 95-07-20, 96-01-09, 8-21-96
   Copyright (c) 1986-96 by Alan J. Laub and The MathWorks, Inc.
   $Revision: 1.9 $

   Assumptions: E is nonsingular, Q=Q', R=R' with R nonsingular, and
                the associated Hamiltonian pencil has no eigenvalues
                on the imaginary axis.
   Sufficient conditions to guarantee the above are stabilizability,
   detectability, and [Q S;S' R] >= 0, with R > 0.

   Reference: W.F. Arnold, III and A.J. Laub, ``Generalized Eigenproblem
           Algorithms and Software for Algebraic Riccati Equations,''
           Proc. IEEE, 72(1984), 1746--1754.
```

CDF2RDF Complex diagonal form to real block diagonal form.
[V,D] = CDF2RDF(V,D) transforms the outputs of EIG(X) (where X is real) from complex diagonal form to a real diagonal form. In complex diagonal form, D has complex eigenvalues down the diagonal. In real diagonal form, the complex eigenvalues are in 2-by-2 blocks on the diagonal. Complex conjugate eigenvalue pairs are assumed to be next to one another.

See also EIG, RSF2CSF.

J.N. Little 4-27-87
Based upon M-file from M. Steinbuch, N.V.KEMA & Delft Univ. of Tech.

$Revision: 5.3 $ $Date: 1996/05/10 19:15:48 $

CHOL Cholesky factorization.
CHOL(X) uses only the diagonal and upper triangle of X.
The lower triangular is assumed to be the (complex conjugate) transpose of the upper. If X is positive definite, then R = CHOL(X) produces an upper triangular R so that R'*R = X.
If X is not positive definite, an error message is printed.

[R,p] = CHOL(X), with two output arguments, never produces an error message. If X is positive definite, then p is 0 and R is the same as above. But if X is not positive definite, then p is a positive integer.
When X is full, R is an upper triangular matrix of order q = p-1 so that R'*R = X(1:q,1:q).
When X is sparse, R is an upper triangular matrix of size q-by-n so that the L-shaped region of the first q rows and first q columns of R'*R agree with those of X.

See also CHOLINC.

$Revision: 5.4 $ $Date: 1996/09/04 19:20:15 $
Built-in function.

COMPAN Companion matrix.
COMPAN(P) is a companion matrix of the polynomial with coefficients P.

$Revision: 5.1 $ $Date: 1996/01/01 23:53:56 $

COND Condition number with respect to inversion.
COND(X) returns the 2-norm condition number (the ratio of the largest singular value of X to the smallest). Large condition numbers indicate a nearly singular matrix.

COND(X,P) returns the condition number of X in P-norm:

NORM(X,P) * NORM(INV(X),P).

```
where P = 1, 2, inf, or 'fro.'

   See also CONDEST, CONDEIG, NORM, NORMEST.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.7 $  $Date: 1996/04/04 16:49:41 $


CONV Convolution and polynomial multiplication.
   C = CONV(A, B) convolves vectors A and B.  The resulting
   vector is length LENGTH(A)+LENGTH(B)-1.
   If A and B are vectors of polynomial coefficients, convolving
   them is equivalent to multiplying the two polynomials.

   See also XCORR, DECONV, CONV2, FILTER, and CONVMTX in the Signal
            Processing Toolbox.

   J.N. Little 4-21-85
   Revised 9-3-87 JNL
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.4 $  $Date: 1996/10/28 22:46:30 $


CROSS  Vector cross product.
   C = CROSS(A,B) returns the cross product of the vectors
   A and B.  That is, C = A x B.  A and B must be 3 element
   vectors.

   C = CROSS(A,B) returns the cross product of A and B along the
   first dimension of length 3.

   C = CROSS(A,B,DIM), where A and B are N-D arrays, returns the cross
   product of vectors in the dimension DIM of A and B. A and B must
   have the same size, and both SIZE(A,DIM) and SIZE(B,DIM) must be 3.

   Note: Use sum(A.*B) to compute the dot product.

   Clay M. Thompson
   updated 12-21-94, Denise Chen
   Copyright © 1984-96 by The MathWorks, Inc.
   $Revision: 5.9 $  $Date: 1996/04/03 02:53:39 $


CTRB   Form the controllability matrix.
   CO = CTRB(A,B)  returns the controllability matrix [B AB A^2B …].

   CO = CTRB(SYS)  returns the controllability matrix of the
   state-space system SYS with realization (A,B,C,D).  This is
   equivalent to  CTRB(sys.a,sys.b)

   See also  CTRBF.

   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.1 $  $Date: 1996/06/28 18:21:21 $
```

CTRBF Controllability staircase form.

```
   [ABAR,BBAR,CBAR,T,K] = CTRBF(A,B,C) returns a decomposition
   into the controllable/uncontrollable subspaces.

   [ABAR,BBAR,CBAR,T,K] = CTRBF(A,B,C,TOL) uses tolerance TOL.

   If Co=CTRB(A,B) has rank r <= n, then there is a similarity
   transformation T such that

   Abar = T * A * T' ,  Bbar = T * B ,  Cbar = C * T'
   and the transformed system has the form

          | Anc    0 |           | 0 |
   Abar =  ----------  ,  Bbar =  ---  ,  Cbar = [Cnc| Cc].
          | A21   Ac |           |Bc |
                                              -1          -1
   where (Ac,Bc) is controllable, and Cc(sI-Ac)Bc = C(sI-A)B.

   See also  CTRB, OBSVF.

   Author : R.Y. Chiang  3-21-86
   Revised 5-27-86 JNL
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.2 $  $Date: 1996/08/09 14:23:31 $

   This M-file implements the Staircase Algorithm of Rosenbrock, 1968.
```

DARE Solve discrete-time algebraic Riccati equations.

```
   [X,L,G,RR] = DARE(A,B,Q,R,S,E)  computes the unique symmetric
   stabilizing solution X of the discrete-time algebraic Riccati
   equation
                                        -1
    E'XE = A'XA - (A'XB + S)(B'XB + R)  (A'XB + S)' + Q

   or, equivalently (if R is nonsingular)
                                  -1            -1                -1
    E'XE = F'XF - F'XB(B'XB + R)  B'XF + Q - SR  S'  with  F:=A-BR  S'.

   When omitted, R,S and E are set to the default values R=I, S=0,
   and E=I.  Additional optional outputs include the gain matrix
                      -1
          G = (B'XB + R)  (B'XA + S'),

   the vector L of closed-loop eigenvalues (i.e., EIG(A-B*G,E)),
   and the Frobenius norm RR of the relative residual matrix.

   [X,L,G,REPORT] = DARE(A,B,Q,...,'report')  turns off error
   messages and returns a success/failure diagnosis REPORT instead.
   The value of REPORT is
     * -1 if symplectic pencil has eigenvalues too close to unit circle,
     * -2 if X=X2/X1 with X1 singular
     * the relative residual RR when DARE succeeds.

   [X1,X2,L,REPORT] = DARE(A,B,Q,...,'implicit')  also turns off
   error messages, but now returns matrices X1,X2 such that X=X2/X1
   and [X1;X2] has orthonormal columns.  REPORT=0 indicates success.
```

```
See also  CARE.

      Author(s): Alan J. Laub (1993)  (laub@ece.ucsb.edu)
           with key contributions by Pascal Gahinet, Cleve Moler,
           and Andy Potvin
   Revised: 94-10-29, 95-07-20, 95-07-24, 96-01-09
       Copyright (c) 1986-96 by Alan J. Laub and The MathWorks, Inc.
       $Revision: 1.8 $

   Assumptions: E is nonsingular, Q=Q', R=R', and the associated
                symplectic pencil has no eigenvalues on the unit circle.
   Sufficient conditions to guarantee the above are stabilizability,
   detectability, and [Q S;S' R] >= 0.

   Reference: W.F. Arnold, III and A.J. Laub, ``Generalized Eigenproblem
              Algorithms and Software for Algebraic Riccati Equations,''
              Proc. IEEE, 72(1984), 1746--1754.
```

DECONV Deconvolution and polynomial division.

```
   [Q,R] = DECONV(B,A) deconvolves vector A out of vector B.  The result
   is returned in vector Q and the remainder in vector R such that
   B = conv(A,Q) + R.

   If A and B are vectors of polynomial coefficients, deconvolution
   is equivalent to polynomial division.  The result of dividing B by
   A is quotient Q and remainder R.

   See also CONV.

   J.N. Little 2-6-86
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/01/01 21:44:52 $
```

DET Determinant.

```
   DET(X) is the determinant of the square matrix X.

   Use COND instead of DET to test for matrix singularity.

   See also COND.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.2 $  $Date: 1996/05/10 16:00:06 $
   Built-in function.
```

DIAG Diagonal matrices and diagonals of a matrix.

```
   DIAG(V,K) when V is a vector with N components is a square matrix
   of order N+ABS(K) with the elements of V on the K-th diagonal. K = 0
   is the main diagonal, K > 0 is above the main diagonal and K < 0
   is below the main diagonal.
```

```
   DIAG(V) is the same as DIAG(V,0) and puts V on the main diagonal.

   DIAG(X,K) when X is a matrix is a column vector formed from
   the elements of the K-th diagonal of X.

   DIAG(X) is the main diagonal of X. DIAG(DIAG(X)) is a diagonal matrix.

   Example
      m = 5;
      diag(-m:m) + diag(ones(2*m,1),1) + diag(ones(2*m,1),-1)
   produces a tridiagonal matrix of order 2*m+1.

   See also SPDIAGS, TRIU, TRIL.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.2 $  $Date: 1996/03/29 20:02:29 $
   Built-in function.
```

DLQR Linear-quadratic regulator design for discrete-time systems.

```
   [K,S,E] = DLQR(A,B,Q,R,N)  calculates the optimal gain matrix K
   such that the state-feedback law  u[n] = -Kx[n]  minimizes the
   cost function

         J = Sum {x'Qx + u'Ru + 2*x'Nu}

   subject to the state dynamics   x[n+1] = Ax[n] + Bu[n].

   The matrix N is set to zero when omitted.  Also returned are the
   Riccati equation solution S and the closed-loop eigenvalues E:
                                  -1
    A'SA - S - (A'SB+N)(R+B'SB) (B'SA+N') + Q = 0,   E = EIG(A-B*K).


   See also  DLQRY, LQRD, LQGREG, and DARE.

   Author(s): J.N. Little 4-21-85
   Revised    P. Gahinet  7-24-96
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.4 $  $Date: 1996/11/05 16:13:12 $
```

DLYAP Discrete Lyapunov equation solver.

```
    X = DLYAP(A,Q) solves the discrete Lyapunov equation:

        A*X*A' - X = Q

    See also  LYAP.

    J.N. Little 2-1-86, AFP 7-28-94
    Copyright (c) 1986-96 by The MathWorks, Inc.
    $Revision: 1.2 $  $Date: 1996/10/18 15:16:26 $

 How to prove the following conversion is true.  Re: show that if
        (1) Ad X Ad' + Cd = X             Discrete lyaponuv eqn
        (2) Ac = inv(Ad + I) (Ad - I)     From dlyap
        (3) Cc = (I - Ac) Cd (I - Ac')/2  From dlyap
```

```
Then
        (4) Ac X + X Ac' + Cc = 0          Continuous lyapunov
```

```
Step 1) Substitute (2) into (3)
         Use identity 2*inv(M+I) = I - inv(M+I)*(M-I)
                                 = I - (M-I)*inv(M-I) to show
         (5) Cc = 4*inv(Ad + I)*Cd*inv(Ad' + I)
 Step 2) Substitute (2) and (5) into (4)
 Step 3) Replace (Ad - I) with (Ad + I -2I)
         Replace (Ad' - I) with (Ad' + I -2I)
 Step 4) Multiply through and simplify to get
         X -inv(Ad+I)*X -X*inv(Ad'+I) +inv(Ad+I)*Cd*inv(Ad'+I) = 0
 Step 5) Left multiply by (Ad + I) and right multiply by (Ad' + I)
 Step 6) Simplify to (1)
```

DOT Vector dot product.

```
   C = DOT(A,B) returns the scalar product of the vectors A and B.
   A and B must be vectors of the same length.  When A and B are both
   column vectors, DOT(A,B) is the same as A'*B.

   DOT(A,B), for N-D arrays A and B, returns the scalar product
   along the first non-singleton dimension of A and B. A and B must
   have the same size.

   DOT(A,B,DIM) returns the scalar product of A and B in the
   dimension DIM.

   See also CROSS.

   Clay M. Thompson
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.6 $
```

EIG Eigenvalues and eigenvectors.

```
   E = EIG(X) is a vector containing the eigenvalues of a square
   matrix X.

   [V,D] = EIG(X) produces a diagonal matrix D of eigenvalues and a
   full matrix V whose columns are the corresponding eigenvectors so
   that X*V = V*D.

   [V,D] = EIG(X,'nobalance') performs the computation with balancing
   disabled, which sometimes gives more accurate results for certain
   problems with unusual scaling.

   E = EIG(A,B) is a vector containing the generalized eigenvalues
   of square matrices A and B.

   [V,D] = EIG(A,B) produces a diagonal matrix D of generalized
   eigenvalues and a full matrix V whose columns are the
   corresponding eigenvectors so that A*V = B*V*D.

   See also CONDEIG.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/05/10 16:01:07 $
   Built-in function.
```

ESTIM Form estimator given estimator gain.

```
   EST = ESTIM(SYS,L)  produces an estimator EST with gain L for
   the outputs and states of the state-space model SYS, assuming
   all inputs of SYS are stochastic and all outputs are measured.
   For a continuous system
                .
      SYS:   x = Ax + Bw ,   y = Cx + Dw   (with w stochastic),

   the resulting estimator
                .
       x_e  = [A-LC] x_e + Ly

      |y_e| = |C| x_e
      |x_e|   |I|

   generates estimates x_e and y_e of x and y.  ESTIM behaves
   similarly when applied to discrete-time systems.

   EST = ESTIM(SYS,L,SENSORS,KNOWN)  handles more general plants
   SYS with both deterministic and stochastic inputs, and both
   measured and non-measured outputs.  The index vectors SENSORS
   and KNOWN specify which outputs y are measured and which inputs
   u are known, respectively.  The resulting estimator EST uses
   [u;y] as input to produce the estimates [y_e;x_e].

   You can use pole placement techniques (see PLACE) to design
   the estimator (observer) gain L, or use the Kalman filter gain
   returned by KALMAN or KALMD.

   See also  REG, PLACE, KALMAN, KALMD, LQGREG.

   Clay M. Thompson 7-2-90
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.2 $  $Date: 1996/10/18 15:15:51 $
```

EXPM   Matrix exponential.

```
   EXPM(X) is the matrix exponential of X.  EXPM is computed using
   a scaling and squaring algorithm with a Pade approximation.

   Although it is not computed this way, if X has a full set
   of eigenvectors V with corresponding eigenvalues D, then
   [V,D] = EIG(X) and EXPM(X) = V*diag(exp(diag(D)))/V.

   EXPM1, EXPM2 and EXPM3 are alternative methods.

   EXP(X) (that's without the M) does it element-by-element.

   See also EXPM1, EXPM2, EXPM3, LOGM, SQRTM, FUNM.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/05/10 16:01:45 $
   Built-in function.
```

EYE Identity matrix.
EYE(N) is the N-by-N identity matrix.

EYE(M,N) or EYE([M,N]) is an M-by-N matrix with 1's on
the diagonal and zeros elsewhere.

EYE(SIZE(A)) is the same size as A.

See also ONES, ZEROS, RAND, RANDN.

$Revision: 5.2 $ $Date: 1996/03/29 20:03:29 $
Built-in function.

FUNM Evaluate general matrix function.
F = FUNM(A,'fun') for a square matrix argument A, evaluates the
matrix version of the function specified by 'fun'. For example,
FUNM(A,'sin') is the matrix sine. For matrix exponentials,
logarithms and square roots, use EXPM(A), LOGM(A) and SQRTM(A)
instead.

FUNM uses a potentially unstable algorithm. If A is close to a
matrix with multiple eigenvalues and poorly conditioned eigenvectors,
FUNM may produce inaccurate results. An attempt is made to detect
this situation and print a warning message. The error detector is
sometimes too sensitive and a message is printed even though the
the computed result is accurate.

[F,ESTERR] = FUNM(A,'fun') does not print any message, but returns
a very rough estimate of the relative error in the computed result.

If A is symmetric or Hermitian, then its Schur form is diagonal and
FUNM is able to produce an accurate result.

S = SQRTM(A) and L = LOGM(A) use FUNM to do their computations,
but they can get more reliable error estimates by comparing S*S
and EXPM(L) with A. E = EXPM(A) uses a completely different
algorithm.

See also EXPM, SQRTM, LOGM.

C.B. Moler 12-2-85, 7-21-86, 7-11-92, 5-2-95.

$Revision: 5.7 $ $Date: 1996/05/10 16:02:21 $

Parlett's method. See Golub and VanLoan (1983), p. 384.

GRAM Controllability and observability gramians.
Wc = GRAM(SYS,'c') returns the controllability gramian of
the state-space system SYS.

Wo = GRAM(SYS,'o') returns its observability gramian.

In both cases, the state-space model SYS should be stable.
The gramians are computed by solving the Lyapunov equations:

```
  *  A*Wc + Wc*A' + BB' = 0  and   A'*Wo + Wo*A + C'C = 0
     for continuous-time systems
            dx/dt = A x + B u  ,   y = C x + D u

  *  A*Wc*A' - Wc + BB' = 0  and   A'*Wo*A - Wo + C'C = 0
     for discrete-time systems
        x[n+1] = A x[n] + B u[n] ,  y[n] = C x[n] + D u[n].

See also  BALREAL, CTRB, OBSV.


J.N. Little 3-6-86
P. Gahinet  6-27-96
Copyright (c) 1986-96 by The MathWorks, Inc.
$Revision: 1.2 $  $Date: 1996/10/18 15:17:53 $

Laub, A., "Computation of Balancing Transformations", Proc. JACC
  Vol.1, paper FA8-E, 1980.
```

HANKEL Hankel matrix.

```
HANKEL(C) is a square Hankel matrix whose first column is C and
whose elements are zero below the first anti-diagonal.

HANKEL(C,R) is a Hankel matrix whose first column is C and whose
last row is R.

Hankel matrices are symmetric, constant across the anti-diagonals,
and have elements H(i,j) = P(i+j-1) where P = [C R(2:END)]
completely determines the Hankel matrix.

See also TOEPLITZ.

J.N. Little 4-22-87
Revised 1-28-88 JNL
Revised 2-25-95 Jim McClellan
Copyright (c) 1984-96 by The MathWorks, Inc.
$Revision: 5.5 $  $Date: 1996/08/08 21:13:27 $
```

INITIAL Initial condition response of state-space models.

```
INITIAL(SYS,X0) plots the undriven response of the state-space
system SYS with initial condition X0 on the states.  This
response is characterized by the equations
                    .
  Continuous time:  x = A x ,  y = C x ,  x(0) = x0

  Discrete time:  x[k+1] = A x[k],  y[k] = C x[k],  x[0] = x0 .

The time range and number of points are chosen automatically.

INITIAL(SYS,X0,TFINAL)  simulates the time response from t = 0
to the final time t = TFINAL.  For discrete-time systems with
unspecified sample time, TFINAL should be the number of samples.
```

```
   INITIAL(SYS,X0,T)  specifies a time vector T to be used for
   simulation.  For discrete systems, T should be of the form
   0:Ts:Tf where Ts is the sample time of the system.  For continuous
   systems, T should be of the form 0:dt:Tf where dt will become the
   sample time of a discrete approximation of the continuous system.

   INITIAL(SYS1,SYS2,...,X0,T)  plots the response of multiple LTI
   systems SYS1,SYS2,... on a single plot.  The time vector T is
   optional.  You can also specify a color, line style, and marker
   for each system, as in  initial(sys1,'r',sys2,'y--',sys3,'gx',x0).

   When invoked with left hand arguments,
       [Y,T,X] = INITIAL(SYS,X0,...)
   returns the output response Y, the time vector T used for simulation,
   and the state trajectories X.  No plot is drawn on the screen.  The
   matrix Y has LENGTH(T) rows and as many columns as outputs in SYS.
   Similarly, X has LENGTH(T) rows and as many columns as states.

   See also  IMPULSE, STEP, LSIM.

    Clay M. Thompson  7-6-90
    Revised ACWG 6-21-92
    Revised AFP 9-21-94,  PG 4-25-96
    Copyright (c) 1986-94 by The MathWorks, Inc.
    $Revision: 1.4 $  $Date: 1996/10/18 15:16:41 $
```

INV Matrix inverse.

```
   INV(X) is the inverse of the square matrix X.
   A warning message is printed if X is badly scaled or
   nearly singular.

   See also SLASH, PINV, COND, CONDEST, NNLS, LSCOV.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/05/10 16:03:33 $
   Built-in function.
```

KALMAN Continuous or discrete Kalman estimator

```
   [KEST,L,P] = KALMAN(SYS,Qn,Rn,Nn)  designs a Kalman estimator
   KEST for the continuous or discrete LTI plant SYS.  For a
   continuous plant
        .
        x = Ax + Bu + Gw            {State equation}
        y = Cx + Du + Hw + v        {Measurements}

   with known inputs u, process noise w, measurement noise v, and
   noise covariances

        E{ww'} = Qn,     E{vv'} = Rn,     E{wv'} = Nn,

   the estimator KEST has input [u;y] and generates optimal
   estimates y_e,x_e of y,x by:
          .
         x_e  = Ax_e + Bu + L(y - Cx_e - Du)
```

```
|y_e| = | C | x_e + | D | u
|x_e|   | I |       | 0 |
```

```
Type HELP DKALMAN for details on the discrete-time counterpart.

   The LTI system SYS contains the plant data (A,[B G],C,[D H]),
   and Nn is set to zero when omitted.  The Kalman estimator KEST
   is continuous when SYS is continuous, discrete otherwise.
   Also returned are the estimator gain L and the steady-state
   error covariance P.  In continuous time with H=0, P solves the
   Riccati equation
                              -1
        AP + PA' - (PC'+G*N)R  (CP+N'*G') + G*Q*G' = 0 .


   [KEST,L,P] = KALMAN(SYS,Qn,Rn,Nn,SENSORS,KNOWN)  handles more
   general plants SYS where the known and stochastic inputs u,w
   are mixed together, and not all outputs are measured.  The
   index vectors SENSORS and KNOWN then specify which outputs y
   of SYS are measured and which inputs u are known.  All other
   inputs are assumed stochastic.

   See also  KALMD, ESTIM, LQGREG, CARE, DARE.

   Author(s): P. Gahinet  8-1-96
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.5 $  $Date: 1996/11/05 16:13:17 $


KRON   Kronecker tensor product.
   KRON(X,Y) is the Kronecker tensor product of X and Y.
   The result is a large matrix formed by taking all possible
   products between the elements of X and those of Y.   For
   example, if X is 2 by 3, then KRON(X,Y) is

      [ X(1,1)*Y  X(1,2)*Y  X(1,3)*Y
        X(2,1)*Y  X(2,2)*Y  X(2,3)*Y ]

   If either X or Y is sparse, only nonzero elements are multiplied
   in the computation, and the result is sparse.

   Paul L. Fackler, North Carolina State, 9-23-96
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.4 $


LQE Linear quadratic estimator design.
   For the continuous-time system:
        .
        x = Ax + Bu + Gw            {State equation}
        z = Cx + Du + v             {Measurements}

   with process noise and measurement noise covariances:

        E{w} = E{v} = 0,  E{ww'} = Q,  E{vv'} = R, E{wv'} = 0

   L = LQE(A,G,C,Q,R) returns the gain matrix L such that the
   stationary Kalman filter:
```

```
      .
      x = Ax + Bu + L(z - Cx - Du)
```

produces an LQG optimal estimate of x. The estimator can be formed with ESTIM.

[L,P,E] = LQE(A,G,C,Q,R) returns the gain matrix L, the Riccati equation solution P which is the estimate error covariance, and the closed loop eigenvalues of the estimator: E = EIG(A-L*C).

[L,P,E] = LQE(A,G,C,Q,R,N) solves the estimator problem when the process and sensor noise is correlated: E{wv'} = N.

J.N. Little 4-21-85
Revised Clay M. Thompson 7-16-90

LQR Linear-quadratic regulator design for continuous-time systems.

[K,S,E] = LQR(A,B,Q,R,N) calculates the optimal gain matrix K such that the state-feedback law u = -Kx minimizes the cost function

$$J = \int \{x'Qx + u'Ru + 2x'Nu\}\, dt$$

subject to the state dynamics $\dot{x} = Ax + Bu$.

The matrix N is set to zero when omitted. Also returned are the Riccati equation solution S and the closed-loop eigenvalues E:

$$SA + A'S - (SB+N)R^{-1}(B'S+N') + Q = 0\,, \quad E = \mathrm{EIG}(A-B*K)\,.$$

See also LQRY, DLQR, LQGREG, CARE, and REG.

Author(s): J.N. Little 4-21-85
Revised P. Gahinet 7-24-96

$Revision: 1.4 $ $Date: 1996/11/05 16:13:08 $

LSIM Simulation of the time response of LTI systems to arbitrary inputs.

LSIM(SYS,U,T) plots the time response of the LTI system SYS to the input signal described by U and T. The time vector T consists of regularly spaced time samples and U is a matrix with as many columns as inputs and whose i-th row specifies the input value at time T(i). For instance,

```
        t = 0:0.01:5;   u = sin(t);   lsim(sys,u,t)
```

simulates the response of SYS to u(t) = sin(t) during 5 seconds.

In discrete time, U should be sampled at the same rate as the system (T is then redundant and can be omitted or set to the empty matrix). In continuous time, the sampling period T(2)-T(1) should be chosen small enough to capture the details of the input signal. The time vector T is resampled when intersample oscillations may occur.

LSIM(SYS,U,T,X0) specifies an additional nonzero initial state X0

```
(for state-space systems only).

LSIM(SYS1,SYS2,...,U,T,X0)  simulates the response of multiple LTI
systems SYS1,SYS2,... on a single plot.  The initial condition X0
is optional.  You can also specify a color, line style, and marker
for each system, as in  lsim(sys1,'r',sys2,'y--',sys3,'gx',u,t).

When invoked with left hand arguments,
     [Y,T] = LSIM(SYS,U,...)
returns the output history Y and time vector T used for simulation.
No plot is drawn on the screen.  The matrix Y has LENGTH(T) rows
and as many columns as outputs in SYS.

For state-space systems,
     [Y,T,X] = LSIM(SYS,U,...)
also returns the state trajectory X, a matrix with LENGTH(T) rows
and as many columns as states.

See also  GENSIG, STEP, IMPULSE, INITIAL.

LSIM normally linearly interpolates the input (using a first order hold)
which is more accurate for continuous inputs. For discrete inputs such
as square waves LSIM tries to detect these and uses a more accurate
zero-order hold method. LSIM can be confused and for accurate results
a small time interval should be used.

J.N. Little 4-21-85
Revised 7-31-90  Clay M. Thompson
    Revised A.C.W.Grace 8-27-89 (added first order hold)
                1-21-91 (test to see whether to use foh or zoh)
Revised 12-5-95 Andy Potvin
Revised 5-8-96  P. Gahinet
Copyright (c) 1986-96 by The MathWorks, Inc.
$Revision: 1.5 $  $Date: 1996/10/06 12:59:31 $
```

LYAP Solve continuous-time Lyapunov equations.

```
X = LYAP(A,C) solves the special form of the Lyapunov matrix
equation:

        A*X + X*A' = -C

X = LYAP(A,B,C) solves the general form of the Lyapunov matrix
equation (also called Sylvester equation):

        A*X + X*B = -C

See also  DLYAP.

S.N. Bangert 1-10-86
Copyright (c) 1986-96 by The MathWorks, Inc.
$Revision: 1.2 $  $Date: 1996/10/18 15:16:28 $
Last revised JNL 3-24-88, AFP 9-3-95
```

```
NORM   Matrix or vector norm.
   For matrices...
     NORM(X) is the largest singular value of X, max(svd(X)).
```

```
   NORM(X,2) is the same as NORM(X).
   NORM(X,1) is the 1-norm of X, the largest column sum,
                  = max(sum(abs((X)))).
   NORM(X,inf) is the infinity norm of X, the largest row sum,
                  = max(sum(abs((X')))).
   NORM(X,'fro') is the Frobenius norm, sqrt(sum(diag(X'*X))).
   NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.

For vectors...
  NORM(V,P) = sum(abs(V).^P)^(1/P).
  NORM(V) = norm(V,2).
  NORM(V,inf) = max(abs(V)).
  NORM(V,-inf) = min(abs(V)).

See also COND, CONDEST, NORMEST.

Copyright (c) 1984-96 by The MathWorks, Inc.
$Revision: 5.5 $  $Date: 1996/04/05 15:20:20 $
Built-in function.
```

NULL Null space.

```
Z = NULL(A) is an orthonormal basis for the null space of A obtained
from the singular value decomposition.  That is,  A*Z has negligible
elements, size(Z,2) is the nullity of A, and Z'*Z = I.

Z = NULL(A,'r') is a "rational" basis for the null space obtained
from the reduced row echelon form.  A*Z is zero, size(Z,2) is an
estimate for the nullity of A, and, if A is a small matrix with
integer elements, the elements of R are ratios of small integers.

The orthonormal basis is preferable numerically, while the rational
basis may be preferable pedagogically.

Example:

    A =
        1     2     3
        1     2     3
        1     2     3

   null(A) =

      -0.1690   -0.9487
       0.8452    0.0000
      -0.5071    0.3162

   null(A,'r') =

       -2    -3
        1     0
        0     1

See also SVD, ORTH, RANK, RREF.

Copyright (c) 1984-96 by The MathWorks, Inc.
$Revision: 5.6 $  $Date: 1996/10/04 16:12:51 $
```

OBSV Form the observability matrix.

```
   OB = OBSV(A,C)  returns the observability matrix [C; CA; CA^2 ...]

   CO = OBSV(SYS)  returns the observability matrix of the state-space
   system SYS with realization (A,B,C,D).  This is equivalent to
   OBSV(sys.a,sys.c)

   See also  OBSVF.

   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.1 $  $Date: 1996/06/28 18:26:28 $
```

OBSVF Observability staircase form.

```
   [ABAR,BBAR,CBAR,T,K] = OBSVF(A,B,C) returns a decomposition
   into the observable/unobservable subspaces.

   [ABAR,BBAR,CBAR,T,K] = OBSVF(A,B,C,TOL) uses tolerance TOL.

   If Ob=OBSV(A,C) has rank r <= n, then there is a similarity
   transformation T such that

      Abar = T * A * T' ,  Bbar = T * B  ,  Cbar = C * T' .

   and the transformed system has the form

          | Ano   A12|           |Bno|
   Abar =  ----------  ,  Bbar =  ---  ,  Cbar = [ 0 | Co].
          |  0    Ao |           |Bo |
                                          -1           -1
   where (Ao,Bo) is controllable, and Co(sI-Ao) Bo = C(sI-A) B.

   See also  OBSV, CTRBF.

   Author : R.Y. Chiang  3-21-86
   Revised 5-27-86 JNL
   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.1 $  $Date: 1996/08/02 15:11:23 $
```

ODE23 Solve non-stiff differential equations, low order method.

```
  [T,Y] = ODE23('F',TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates the
  system of differential equations y' = F(t,y) from time T0 to TFINAL with
  initial conditions Y0.  'F' is a string containing the name of an ODE
  file.  Function F(T,Y) must return a column vector.  Each row in
  solution array Y corresponds to a time returned in column vector T.  To
  obtain solutions at specific times T0, T1, ..., TFINAL (all increasing
  or all decreasing), use TSPAN = [T0 T1 ... TFINAL].

  [T,Y] = ODE23('F',TSPAN,Y0,OPTIONS) solves as above with default
  integration parameters replaced by values in OPTIONS, an argument
  created with the ODESET function.  See ODESET for details.  Commonly
  used options are scalar relative error tolerance 'RelTol' (1e-3 by
  default) and vector of absolute error tolerances 'AbsTol' (all
  components 1e-6 by default).
```

```
[T,Y] = ODE23('F',TSPAN,Y0,OPTIONS,P1,P2,...) passes the additional
parameters P1,P2,... to the ODE file as F(T,Y,FLAG,P1,P2,...) (see
ODEFILE).  Use OPTIONS = [] as a place holder if no options are set.

It is possible to specify TSPAN, Y0 and OPTIONS in the ODE file (see
ODEFILE).  If TSPAN or Y0 is empty, then ODE23 calls the ODE file
[TSPAN,Y0,OPTIONS] = F([],[],'init') to obtain any values not supplied
in the ODE23 argument list.  Empty arguments at the end of the call list
may be omitted, e.g. ODE23('F').

As an example, the commands

    options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
    ode23('rigidode',[0 12],[0 1 1],options);

solve the system y' = rigidode(t,y) with relative error tolerance 1e-4
and absolute tolerances of 1e-4 for the first two components and 1e-5
for the third.  When called with no output arguments, as in this
example, ODE23 calls the default output function ODEPLOT to plot the
solution as it is computed.

[T,Y,TE,YE,IE] = ODE23('F',TSPAN,Y0,OPTIONS) with the Events property in
OPTIONS set to 'on', solves as above while also locating zero crossings
of an event function defined in the ODE file.  The ODE file must be
coded so that F(T,Y,'events') returns appropriate information.  See
ODEFILE for details.  Output TE is a column vector of times at which
events occur, rows of YE are the corresponding solutions, and indices in
vector IE specify which event occurred.

[T,X,Y] = ODE23('MODEL',TSPAN,Y0,OPTIONS,UT,P1,P2,...) simulates a
SIMULINK model using the 'ODE23' integrator.  The OPTIONS argument is
created with the ODESET function.  See also SIM.

See also ODEFILE and
    other ODE solvers:  ODE45, ODE113, ODE15S, ODE23S
    options handling:   ODESET, ODEGET
    output functions:   ODEPLOT, ODEPHAS2, ODEPHAS3, ODEPRINT
    odefile examples:   ORBITODE, ORBT2ODE, RIGIDODE, VDPODE

ODE23 is an implementation of the explicit Runge-Kutta (2,3) pair of
Bogacki and Shampine called BS23.  It uses a "free" interpolant of order
3.  Local extrapolation is done.

Details are to be found in The MATLAB ODE Suite, L. F. Shampine and
M. W. Reichelt, SIAM Journal on Scientific Computing, 18-1, 1997.

Mark W. Reichelt and Lawrence F. Shampine, 6-14-94
Copyright (c) 1984-96 by The MathWorks, Inc.
$Revision: 5.43 $  $Date: 1996/11/10 17:46:29 $
```

ORTH Orthogonalization.

```
  Q = ORTH(A) is an orthonormal basis for the range of A.
  That is, Q'*Q = I, the columns of Q span the same space as
  the columns of A, and the number of columns of Q is the
  rank of A.
```

```
   See also SVD, RANK, NULL.

   Major revision, 4-13-93, C. Moler.
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.5 $  $Date: 1996/05/10 16:06:37 $

 Beginning with MATLAB 4.1, the algorithms for NULL and ORTH
 use singular value decomposition, SVD, instead of orthogonal
 factorization, QR.  This doubles the computation time, but
 provides more reliable and consistent rank determination.
```

PINV Pseudoinverse.

```
   X = PINV(A) produces a matrix X of the same dimensions
   as A' so that A*X*A = A, X*A*X = X and A*X and X*A
   are Hermitian. The computation is based on SVD(A) and any
   singular values less than a tolerance are treated as zero.
   The default tolerance is MAX(SIZE(A)) * NORM(A) * EPS.

   PINV(A,TOL) uses the tolerance TOL instead of the default.

   See also RANK.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.5 $  $Date: 1996/05/10 16:07:14 $
```

PLACE Pole placement technique

```
   K = PLACE(A,B,P)  computes a state-feedback matrix K such that
   the eigenvalues of  A-B*K  are those specified in vector P.
   No eigenvalue should have a multiplicity greater than the
   number of inputs.

   [K,PREC,MESSAGE] = PLACE(A,B,P)  returns PREC, an estimate of how
   closely the eigenvalues of A-B*K match the specified locations P
   (PREC measures the number of accurate decimal digits in the actual
   closed-loop poles).  If some nonzero closed-loop pole is more than
   10 off from the desired location, MESSAGE contains a warning
   message.

   See also  ACKER.

   M. Wette 10-1-86
   Revised 9-25-87 JNL
   Revised 8-4-92 Wes Wang
   Revised 10-5-93, 6-1-94 Andy Potvin

   Ref:  Kautsky, Nichols, Van Dooren, "Robust Pole Assignment in Linear
         State Feedback," Intl. J. Control, 41(1985)5, pp 1129-1155


   Copyright (c) 1986-96 by The MathWorks, Inc.
   $Revision: 1.2 $  $Date: 1996/09/27 21:43:23 $
```

POLYVALM Evaluate polynomial with matrix argument.
POLYVALM(V,X) where V is a vector whose elements are the coefficients of a polynomial, is the value of the polynomial evaluated with matrix argument X. X must be a square matrix.

See also POLYVAL, POLYFIT.

J.N.Little 4-20-86

$Revision: 5.4 $ $Date: 1996/10/27 14:01:11 $

Polynomial evaluation c(x) using Horner's method

QR Orthogonal-triangular decomposition.
[Q,R] = QR(A) produces an upper triangular matrix R of the same dimension as A and a unitary matrix Q so that A = Q*R.

[Q,R,E] = QR(A) produces a permutation matrix E, an upper triangular R and a unitary Q so that A*E = Q*R. The column permutation E is chosen so that abs(diag(R)) is decreasing.

[Q,R] = QR(A,0) produces the "economy size" decomposition. If A is m-by-n with m > n, then only the first n columns of Q are computed.

[Q,R,E] = QR(A,0) produces an "economy size" decomposition in which E is a permutation vector, so that Q*R = A(:,E). The column permutation E is chosen so that abs(diag(R)) is decreasing.

By itself, QR(A) returns the output of LINPACK'S ZQRDC routine. TRIU(QR(A)) is R.

For sparse matrices, QR can compute a "Q-less QR decomposition", which has the following slightly different behavior.

R = QR(A) returns only R. Note that R = chol(A'*A).
[Q,R] = QR(A) returns both Q and R, but Q is often nearly full.
[C,R] = QR(A,B), where B has as many rows as A, returns C = Q'*B.
R = QR(A,0) and [C,R] = QR(A,B,0) produce economy size results.

The sparse version of QR does not do column permutations.
The full version of QR does not return C.

The least squares approximate solution to A*x = b can be found with the Q-less QR decomposition and one step of iterative refinement:

```
x = R\(R'\(A'*b))
r = b - A*x
e = R\(R'\(A'*r))
x = x + e;
```

See also LU, NULL, ORTH, QRDELETE, QRINSERT.

$Revision: 5.5 $ $Date: 1996/08/15 21:52:13 $
Built-in function.

```
RANDN  Normally distributed random numbers.
  RANDN(N) is an N-by-N matrix with random entries, chosen from
  a normal distribution with mean zero and variance one.
  RANDN(M,N) and RANDN([M,N]) are M-by-N matrices with random entries.
  RANDN(M,N,P,...) or RANDN([M,N,P...]) generate random arrays.
  RANDN with no arguments is a scalar whose value changes each time it
  is referenced.  RANDN(SIZE(A)) is the same size as A.

  S = RANDN('state') is a 2-element vector containing the current state
  of the normal generator.  RANDN('state',S) resets the state to S.
  RANDN('state',0) resets the generator to its initial state.
  RANDN('state',J), for integer J, resets the generator to its J-th state.
  RANDN('state',sum(100*clock)) resets it to a different state each time.

  MATLAB Version 4.x used random number generators with a single seed.
  RANDN('seed',0) and RANDN('seed',J) cause the MATLAB 4 generator
  to be used.
  RANDN('seed') returns the current seed of the MATLAB 4 normal generator.
  RANDN('state',J) and RANDN('state',S) cause the MATLAB 5 generator
  to be used.

  See also RAND, SPRAND, SPRANDN, RANDPERM.

  Copyright (c) 1984-96 by The MathWorks, Inc.
  $Revision: 5.6 $  $Date: 1996/04/19 22:11:11 $
  Built-in function.


RANK   Matrix rank.
   RANK(A) provides an estimate of the number of linearly
   independent rows or columns of a matrix A.
   RANK(A,tol) is the number of singular values of A
   that are larger than tol.
   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.5 $  $Date: 1996/08/29 17:43:14 $


REG  Form regulator given state-feedback and estimator gains.
   RSYS = REG(SYS,K,L) produces an observer-based regulator RSYS
   for the state-space system SYS, assuming all inputs of SYS are
   controls and all outputs are measured.  The matrices K and L
   specify the state-feedback and observer gains.  For
                .
       SYS:   x = Ax + Bu ,   y = Cx + Du

   the resulting regulator is
        .
       x_e = [A-BK-LC+LDK] x_e + Ly
         u = -K x_e

   This regulator should be connected to the plant using positive
   feedback.  REG behaves similarly when applied to discrete-time
   systems.
```

```
RSYS = REG(SYS,K,L,SENSORS,KNOWN,CONTROLS)  handles more
general regulation problems where
  * the plant inputs consist of controls u, known inputs Ud,
    and stochastic inputs w,
  * only a subset y of the plant outputs are measured.
The I/O subsets y, Ud, and u are specified by the index vectors
SENSORS, KNOWN, and CONTROLS.  The resulting regulator RSYS
uses [Ud;y] as input to generate the commands u.

You can use pole placement techniques (see PLACE) to design the
gains K and L, or alternatively use the LQ and Kalman gains
produced by LQR/DLQR and KALMAN.

See also  ESTIM, PLACE, LQR, DLQR, LQGREG, KALMAN.

Clay M. Thompson 6-29-90
Copyright (c) 1986-96 by The MathWorks, Inc.
$Revision: 1.2 $  $Date: 1996/10/18 15:15:58 $
```

RESIDUE Partial-fraction expansion (residues).

```
[R,P,K] = RESIDUE(B,A) finds the residues, poles and direct term of
a partial fraction expansion of the ratio of two polynomials B(s)/A(s).
If there are no multiple roots,
   B(s)       R(1)       R(2)             R(n)
   ----  =  -------- + -------- + ... + -------- + K(s)
   A(s)     s - P(1)   s - P(2)         s - P(n)
Vectors B and A specify the coefficients of the numerator and
denominator polynomials in descending powers of s.  The residues
are returned in the column vector R, the pole locations in column
vector P, and the direct terms in row vector K.  The number of
poles is n = length(A)-1 = length(R) = length(P). The direct term
coefficient vector is empty if length(B) < length(A), otherwise
length(K) = length(B)-length(A)+1.

If P(j) = ... = P(j+m-1) is a pole of multplicity m, then the
expansion includes terms of the form

             R(j)        R(j+1)                R(j+m-1)
           -------- + ------------   + ... + ------------
           s - P(j)   (s - P(j))^2           (s - P(j))^m

[B,A] = RESIDUE(R,P,K), with 3 input arguments and 2 output arguments,
converts the partial fraction expansion back to the polynomials with
coefficients in B and A.

Warning: Numerically, the partial fraction expansion of a ratio of
polynomials represents an ill-posed problem.  If the denominator
polynomial, A(s), is near a polynomial with multiple roots, then
small changes in the data, including roundoff errors, can make
arbitrarily large changes in the resulting poles and residues.
Problem formulations making use of state-space or zero-pole
representations are preferable.

See also POLY, ROOTS, DECONV.
```

```
   Reference:  A.V. Oppenheim and R.W. Schafer, Digital
   Signal Processing, Prentice-Hall, 1975, p. 56-58.

   C.R. Denham and J.N. Little, MathWorks, 1986, 1989.
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.4 $  $Date: 1996/10/27 14:01:22 $
```

ROOTS Find polynomial roots.

```
   ROOTS(C) computes the roots of the polynomial whose coefficients
   are the elements of the vector C. If C has N+1 components,
   the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).

   See also POLY, RESIDUE, FZERO.

   J.N. Little 3-17-86
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.4 $  $Date: 1996/10/05 14:22:34 $

 ROOTS finds the eigenvalues of the associated companion matrix.
```

RREF   Reduced row echelon form.

```
   R = RREF(A) produces the reduced row echelon form of A.

   [R,jb] = RREF(A) also returns a vector, jb, so that:
       r = length(jb) is this algorithm's idea of the rank of A,
       x(jb) are the bound variables in a linear system, Ax = b,
       A(:,jb) is a basis for the range of A,
       R(1:r,jb) is the r-by-r identity matrix.

   [R,jb] = RREF(A,TOL) uses the given tolerance in the rank tests.

   Roundoff errors may cause this algorithm to compute a different
   value for the rank than RANK, ORTH and NULL.

   See also RREFMOVIE, RANK, ORTH, NULL, QR, SVD.

   CBM, 11/24/85, 1/29/90, 7/12/92.
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/03/26 02:31:24 $
```

Slash

```
Matrix division.
 \   Backslash or left division.
     A\B is the matrix division of A into B, which is roughly the
     same as INV(A)*B , except it is computed in a different way.
     If A is an N-by-N matrix and B is a column vector with N
     components, or a matrix with several such columns, then
     X = A\B is the solution to the equation A*X = B computed by
     Gaussian elimination. A warning message is printed if A is
     badly scaled or nearly singular.  A\EYE(SIZE(A)) produces the
```

```
    inverse of A.
    If A is an M-by-N matrix with M < or > N and B is a column
    vector with M components, or a matrix with several such columns,
    then X = A\B is the solution in the least squares sense to the
    under- or overdetermined system of equations A*X = B. The
    effective rank, K, of A is determined from the QR decomposition
    with pivoting. A solution X is computed which has at most K
    nonzero components per column. If K < N this will usually not
    be the same solution as PINV(A)*B.  A\EYE(SIZE(A)) produces a
    generalized inverse of A.

 /  Slash or right division.
    B/A is the matrix division of A into B, which is roughly the
    same as B*INV(A) , except it is computed in a different way.
    More precisely, B/A = (A'\B')'. See \.

 ./ Array right division.
    B./A denotes element-by-element division.  A and B
    must have the same dimensions unless one is a scalar.
    A scalar can be divided with anything.

 .\ Array left division.
    A.\B. denotes element-by-element division.  A and B
    must have the same dimensions unless one is a scalar.
    A scalar can be divided with anything.

  Copyright (c) 1984-96 by The MathWorks, Inc.
  $Revision: 5.2 $  $Date: 1996/04/16 22:02:49 $
```

SS Create state-space models or convert LTI model to state space.

```
  You can create a state-space model by:
   SYS = SS(A,B,C,D)       Continuous-time model
   SYS = SS(A,B,C,D,T)     Discrete-time model with sampling time T
                           (Set T=-1 if undetermined)
   SYS = SS                Default empty SS object
   SYS = SS(D)             Static gain matrix
   SYS = SS(A,B,C,D,LTI)   State-space model with LTI properties
                           inherited from the system LTI (can be
                           SS, TF, or ZPK)
  All the above syntaxes may be followed by Property/Value pairs.
  (Type "help ltiprops" for details on assignable properties).
  Setting D=0 is interpreted as the zero matrix of adequate
  dimensions.  The output SYS is an SS object.

  SYS_SS = SS(SYS)  converts an arbitrary LTI model SYS to state space,
  i.e., computes a state-space realization SYS_SS of SYS.

  See also   SET, SSDATA, DSS, TF, ZPK.

      Author(s): A. Potvin, 3-1-94
      Revised: P. Gahinet, 4-1-96
      Copyright (c) 1986-96 by The MathWorks, Inc.
      $Revision: 1.6 $  $Date: 1996/10/31 21:47:15 $
```

SSDATA Quick access to state-space data.

```
[A,B,C,D] = SSDATA(SYS)  returns the values of the A,B,C,D
matrices.  If SYS is not a state-space model, it is first
converted to state space.
```

```
   [A,B,C,D,TS,TD] = SSDATA(SYS)  also returns the sample time
    TS and input delays TD.  For continuous systems, TD is a vector
    with one entry per input channel.  For discrete systems, TD is
    the empty matrix [].

    Other properties of SYS can be accessed with GET or by direct
    structure-like referencing (e.g., SYS.Ts)

    See also  GET, DSSDATA.

       Author(s): P. Gahinet, 4-1-96
       Copyright (c) 1994 by The MathWorks, Inc.
       $Revision: 1.4 $
```

SS2SS Change of state coordinates for state-space systems.

```
     SYS = SS2SS(SYS,T) performs the similarity transformation z = Tx
     on the state vector x of the system SYS.  The resulting state-space
     system is described by:

                    .       -1
                    z = [TAT  ] z + [TB] u
                           -1
                    y = [CT   ] z + D u

     (Respectively,

                -1  .       -1
           [TET  ] z = [TAT  ] z + [TB] u
                           -1
                    y = [CT   ] z + D u

     in the descriptor case).

     See also   CANON, AUGSTATE, SSBAL, BALREAL.

    Clay M. Thompson  7-3-90
    Copyright (c) 1986-96 by The MathWorks, Inc.
    $Revision: 1.2 $  $Date: 1996/06/28 18:28:36 $
```

SS2TF State-space to transfer function conversion.

```
   [NUM,DEN] = SS2TF(A,B,C,D,iu)  calculates the transfer function:

               NUM(s)          -1
       H(s) = -------- = C(sI-A) B + D
               DEN(s)
   of the system:
       .
       x = Ax + Bu
       y = Cx + Du

   from the iu'th input.  Vector DEN contains the coefficients of the
   denominator in descending powers of s.  The numerator coefficients
   are returned in matrix NUM with as many rows as there are
   outputs y.
```

```
   See also TF2SS.

   J.N. Little 4-21-85
   Revised 7-25-90 Clay M. Thompson, 10-11-90 A.Grace
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 1.16 $  $Date: 1996/03/18 21:06:12 $
```

STEP Step response of LTI systems.

```
   STEP(SYS)  plots the step response of each input channel of
   the LTI system SYS.  The time range and number of points are
   chosen automatically.

   STEP(SYS,TFINAL)  simulates the step response from t = 0 to the
   final time t = TFINAL.  For discrete-time systems with unspecified
   sampling time, TFINAL is interpreted as the number of samples.

   STEP(SYS,T)  uses the user-supplied time vector T for simulation.
   For discrete-time systems, T should be of the form  Ti:Ts:Tf
   where Ts is the sample time of the system.  For continuous systems,
   T should be of the form  Ti:dt:Tf  where dt will become the sample
   time of a discrete approximation to the continuous system.

   STEP(SYS1,SYS2,...,T)  plots the step response of multiple LTI
   systems SYS1,SYS2,... on a single plot.  The time vector T is
   optional.  You can also specify a color, line style, and marker
   for each system, as in  step(sys1,'r',sys2,'y--',sys3,'gx').

   When invoked with left-hand arguments,
        [Y,T] = STEP(SYS,...)
   returns the output response Y and the time vector T used for
   simulation.  No plot is drawn on the screen.  If SYS has
   NU inputs and NY outputs, and LT=length(T), the array Y is
   LT-by-NY-by-NU  and Y(:,:,j) gives the step response of the
   j-th input channel.

   For state-space systems,
        [Y,T,X] = STEP(SYS,...)
   also returns the state trajectory X which is an LT-by-NX-by-NU
   array if SYS has NX states.

   See also  INITIAL, IMPULSE, LSIM.

Extra notes on user-supplied T: For continuous-time systems, the system is
converted to discrete time with a sample time of dt=t(2)-t(1).  The time
vector plotted is then t=t(1):dt:t(end).

    J.N. Little 4-21-85
    Revised A.C.W.Grace 9-7-89, 5-21-92
    Revised A. Potvin 12-1-95
    Copyright (c) 1986-96 by The MathWorks, Inc.
    $Revision: 1.8 $  $Date: 1996/10/06 12:59:17 $
```

SUBSPACE Angle between subspaces.

```
   SUBSPACE(A,B) finds the angle between two subspaces specified
```

```
   by the columns of A and B.  If A and B are vectors of unit
   length, this is the same as ACOS(A'*B).

   If the angle is small, the two spaces are nearly linearly
   dependent.  In a physical experiment described by some
   observations A, and a second realization of the experiment
   described by B, SUBSPACE(A,B) gives a measure of the amount
   of new information afforded by the second experiment not
   associated with statistical errors of fluctuations.

   L. Shure 11-03-88, CBM 5-3-93, 4-18-94.
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.3 $  $Date: 1996/10/28 22:49:14 $
```

SVD Singular value decomposition.

```
   [U,S,V] = SVD(X) produces a diagonal matrix S, of the same
   dimension as X and with nonnegative diagonal elements in
   decreasing order, and unitary matrices U and V so that
   X = U*S*V'.

   S = SVD(X) returns a vector containing the singular values.

   [U,S,V] = SVD(X,0) produces the "economy size"
   decomposition. If X is m-by-n with m > n, then only the
   first n columns of U are computed and S is n-by-n.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.2 $  $Date: 1996/05/10 19:19:43 $
   Built-in function.
```

TF2SS Transfer function to state-space conversion.

```
   [A,B,C,D] = TF2SS(NUM,DEN)  calculates the state-space
   representation:
       .
       x = Ax + Bu
       y = Cx + Du

   of the system:
                NUM(s)
       H(s) = --------
                DEN(s)

   from a single input.  Vector DEN must contain the coefficients of
   the denominator in descending powers of s.  Matrix NUM must
   contain the numerator coefficients with as many rows as there are
   outputs y.  The A,B,C,D matrices are returned in controller
   canonical form.  This calculation also works for discrete systems.
   To avoid confusion when using this function with discrete systems,
   always use a numerator polynomial that has been padded with zeros
   to make it the same length as the denominator.  See the User's
   guide for more details.

   See also SS2TF.

   J.N. Little 3-24-85
```

```
   $Revision: 1.13 $  $Date: 1996/07/29 19:45:19 $
   Latest revision 4-29-89 JNL, 7-29-96 PG


TOEPLITZ Toeplitz matrix.
   TOEPLITZ(C,R) is a non-symmetric Toeplitz matrix having C as its
   first column and R as its first row.

   TOEPLITZ(C) is a symmetric (or Hermitian) Toeplitz matrix.

   See also HANKEL.

   Revised 10-8-92, LS - code from A.K. Booer.
   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.2 $  $Date: 1996/08/08 21:13:31 $


TRACE  Sum of diagonal elements.
   TRACE(A) is the sum of the diagonal elements of A, which is
   also the sum of the eigenvalues of A.

   Copyright (c) 1984-96 by The MathWorks, Inc.
   $Revision: 5.2 $  $Date: 1996/01/01 23:16:22 $
```

References

[B1] Cavallo, Alberto, Roberto Setola, and Francesco Vasca, *Using MATLAB, SIMULINK and Control System Toolbox: A Practical Approach*, Prentice Hall, 1996. (ISBN 0-13-261058-2)

[B2] Chen, Chi-Tsong, *Analog and Digital Control System Design: Transfer-Function, State-Space, and Algebraic Methods*, Oxford University Press, 1993. (ISBN 0-03-094070-2)

[B3] Close, Charles and Dean Frederick, *Modeling and Analysis of Dynamic Systems*, 2nd edition, John Wiley and Sons, Inc., 1993. (ISBN 0-471-12517-2)

[B4] D'Azzo, John J., and Constantine H. Houpis, *Linear Control System Analysis and Design: Conventional and Modern*, 4th edition, McGraw-Hill, 1995. (ISBN 0-07-016321-9)

[B5] Dorato, Peter, Chaouki Abdallah, and Vito Cerone, *Linear-Quadratic Control: An Introduction*, Prentice Hall, 1995. 9ISBN 0-02-329962-2)

[B6] Dorf, Richard C., and Robert H. Bishop, *Modern Control Systems*, 7th edition, Addison-Wesley, 1995. (ISBN 0-201-50174-0)

[B7] Franklin, Gene F., J. David Powell, and Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, 3rd edition, Addison-Wesley, 1994. (ISBN 0-201-52747-2)

[B8] Franklin Gene F., J. David Powell, and Michael L. Workman, *Digital Control of Dynamic Systems*, 2nd edition, Addison-Wesley, 1990. (ISBN 0-201-11938-2)

[B9] Frederick, Dean K., and Joe H. Chow, *Feedback Control Problems Using MATLAB and the Control System Toolbox*, PWS Publishing Company, 1995. (ISBN 0-534-93798-5)

[B10] Hanselman, Duane C., and Benjamin C. Kuo, *MATLAB Tools for Control System Analysis and Design*, 2nd edition, Prentice Hall, 1995. (PC version ISBN 0-13-202293-1 - Macintosh version ISBN 0-13-202574-4)

[B11] Kuo, Benjamin C., *Automatic Control Systems*, 7th edition, Prentice Hall, 1995. (ISBN 0-13-304759-8)

[B12] Leonard, Naomi Ehrich, and William S. Levine, *Using MATLAB to Analyze and Design Control Systems*, 2nd edition, Addison-Wesley, 1995. (ISBN 0-8053-2193-4)

[B13] Moscinski, Jerzy, and Zbigniew Ogonowski, editors, *Advanced Control with MATLAB and SIMULINK*, Prentice Hall, 1996. (ISBN 0-13-309667X)

[B14] Nise, Norman S., *Control Systems Engineering*, 2nd edition, Addison-Wesley, 1995. (ISBN 0-8053-5424-7)

[B15] Ogata, Katsuhiko, *Designing Linear Control Systems with MATLAB*, Prentice Hall, 1994. (ISBN 0-13-293226-1)

[B16] Ogata, Katsuhiko, *Solving Control Engineering Problems with MATLAB*, Prentice Hall, 1994. (ISBN 0-13-045907-0)

[B17] Phillips, Charles L., and Royce D. Harbor, *Feedback Control Systems*, 3rd edition, Prentice Hall, 1996. (ISBN 0-13-371691-0)

[B18] Phillips, Charles L., and H. Troy Nagle, *Digital Control System Analysis and Design*, 3rd edition, Prentice Hall, 1995. (ISBN 0-13-309832-X)

[B19] Rohrs, Charles E., James L. Melsa, and Donald G. Schultz, *Linear Control Systems*, McGraw Hill, 1993. (ISBN 0-07-041525-0)

[B20] Sciavicco, Lorenzo, and Bruno Siciliano, *Modeling and Control of Robot Manipulators* McGraw-Hill, 1996. (ISBN 0-07-057217-8)

[B21] Strum, Robert, and Donald Kirk, *Contemporary Linear Systems Using MATLAB 4.0*, PWS Publishing Company, 1994. (ISBN 0-534-947107)

[B22] Vaccaro, Richard J., D*igital Control: A State-Space Approach*, McGraw-Hill, 1995. (ISBN 0-07-066781-0)