Funzioni principali del Matlab

BODE Bode frequency response of LTI systems.

BODE(SYS) draws the Bode plot of the LTI system SYS. The frequency range and number of points are chosen automatically.

BODE(SYS,{WMIN,WMAX}) draws the Bode plot for frequencies between WMIN and WMAX (in radian/second).

BODE(SYS,W) uses the user-supplied vector W of frequencies, in radian/second, at which the Bode response is to be evaluated. See LOGSPACE to generate logarithmically spaced frequency vectors.

BODE(SYS1,SYS2,...,W) plots the Bode response of multiple LTI systems SYS1,SYS2,... on a single plot. The frequency vector W is optional. You can also specify a color, line style, and marker for each system, as in bode(sys1,'r',sys2,'y--',sys3,'gx').

When invoked with left-hand arguments,
[MAG,PHASE,W] = BODE(SYS,...)
returns the frequency vector W and arrays MAG and PHASE of magnitudes (in dB) and phases (in degrees). No plot is drawn on the screen. If SYS has NU inputs and NY outputs and LW=length(W), MAG and PHASE are NY-by-NU-by-LW arrays and the response at the frequency W(k) is given by MAG(:,:,k) and PHASE(:,:,k).

For discrete systems with sample time Ts, BODE uses the transformation Z = exp(j*W*Ts) to map the unit circle to the real frequency axis. The frequency response is only plotted for frequencies smaller than the Nyquist frequency pi/Ts, and the default value 1 (second) is assumed when Ts is unspecified.

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)

DAMP Natural frequency and damping factor of LTI system poles.

[Wn,Z] = DAMP(SYS) returns vectors Wn and Z containing the

natural frequencies and damping factors of the LTI system SYS.
For discrete-time systems, the equivalent s-plane natural
frequency and damping ratio of an eigenvalue lambda are given by:

Wn = abs(log(lamba))/Ts , Z = -cos(angle(log(lamba))) .

Wn and Z are empty vectors if the sample time Ts is undefined.

[Wn,Z,P] = DAMP(SYS) also returns the system poles P.

When invoked without left-hand arguments, DAMP prints the poles
with their natural frequency and damping factor in a tabular format
on the screen. The poles are sorted by increasing frequency.

DCGAIN DC gain of LTI systems.

K = DCGAIN(SYS) computes the steady state (D.C. or low frequency)
gain of the system SYS.

ESORT Sort complex continuous eigenvalues in descending order.

S = ESORT(P) sorts the complex eigenvalues in the vector P in
descending order by real part. The unstable eigenvalues (in
the continuous-time sense) will appear first.

[S,NDX] = ESORT(P) also returns the vector NDX containing the
indexes used in the sort.

IMPULSE Impulse response of LTI systems.

IMPULSE(SYS) plots the impulse response of each input channel of
the LTI system SYS. The time range and number of points are chosen
automatically. The infinite pulse at t = 0 in the response of
continuous systems with direct feedthrough is disregarded.

IMPULSE(SYS,TFINAL) simulates the impulse 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.

IMPULSE(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.

IMPULSE(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 impulse(sys1,'r',sys2,'y--',sys3,'gx').

When invoked with left-hand arguments,
[Y,T] = IMPULSE(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 impulse response of the j-th input channel.

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

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.

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.

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)

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.

REAL Complex real part.
REAL(X) is the real part of X.
See I or J to enter complex numbers.

SS2ZP State-space to zero-pole conversion.
[Z,P,K] = SS2ZP(A,B,C,D,IU) calculates the transfer function in
factored form:

$$H(s) = C(sI-A)^{-1}B + D = k\,\frac{(s-z1)(s-z2)\ldots(s-zn)}{(s-p1)(s-p2)\ldots(s-pn)}$$

of the system:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

from the single input IU. The vector P contains the pole
locations of the denominator of the transfer function. The
numerator zeros are returned in the columns of matrix Z with as
many columns as there are outputs y. The gains for each numerator
transfer function are returned in column vector K.

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.

TF Specify transfer functions or convert LTI model to transfer function.

TFSYS = TF(SYS) converts an arbitrary LTI model SYS to transfer function format. The output TFSYS is a TF object.

TFSYS = TF(SYS,'inv') uses a fast algorithm for state-space SYS, but is typically less accurate for high-order systems.

You can create SISO or MIMO transfer functions (TF) by
SYS = TF(NUM,DEN) Continuous-time system NUM(s)./DEN(s)
SYS = TF(NUM,DEN,T) Discrete-time system NUM(z)./DEN(z) with sampling time T (Set T=-1 if undetermined)
SYS = TF Default empty TF object
SYS = TF(M) Static gain matrix
SYS = TF(NUM,DEN,LTI) Transfer function with LTI properties inherited from 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).
The output SYS is a TF object.

The default TF variables 's' and 'z' can be altered by resetting the 'Variable' property. Alternatives include 'p' for continuous TF and 'z^-1' or 'q' for discrete TF.

For SISO systems, NUM and DEN are row vectors listing the numerator and denominator coefficients in

```
  * descending powers of s or z by default
  * ascending powers of q = z^-1 if 'Variable' is set to 'z^-1' or 'q'
    (DSP convention).

For MIMO systems with NU inputs and NY outputs, NUM and DEN are
NY-by-NU cell arrays of row vectors where NUM{i,j} and DEN{i,j} specify
the transfer function from input j to output i.  For example,
   tf( {-5 ; [1 -5 6]} , {[1 -1] ; [1 1 0]})
specifies the two-output/one-input system

   [     -5 /(s-1)      ]
   [ (s^2-5s+6)/(s^2+s) ]
```

ZPK Specify zero-pole-gain models or convert to zero-pole-gain form.

```
ZSYS = ZPK(SYS)  converts an arbitrary LTI model SYS to
zero-pole-gain form. The output ZSYS is a ZPK object.

ZSYS = ZPK(SYS,'inv')  uses a fast algorithm for state-space SYS,
but is typically less accurate for high-order systems with low
gain at s=0.

You can create SISO or MIMO zero-pole-gain (ZPK) models by:
 SYS = ZPK(Z,P,K)       Continuous-time system.
 SYS = ZPK(Z,P,K,T)     Discrete-time system with sample time T
                   (Set T=-1 if undetermined)
 SYS = ZPK              Default empty zero-pole-gain object
 SYS = ZPK(M)           Static gain matrix
 SYS = ZPK(Z,P,K,LTI)   ZPK model with LTI properties inherited
                   from 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).
The output SYS is a ZPK object.

For SISO systems, Z and P are the vectors of zeros and poles
(set to [] if none) and K is the scalar gain.

For MIMO systems with NU inputs and NY outputs,
  * Z and P are NY-by-NU cellarrays where Z{i,j} and P{i,j} are
    the vectors of zeros and poles of the transfer function
    from input j to output i
  * K is the 2-D matrix of gains for each I/O channel.
For example,
   zpk( {[];[2 3]} , {1;[0 -1]} , [-5;1] )
specifies the one-input/two-output system
   [    -5 /(s-1)      ]
   [ (s-2)(s-3)/s(s+1) ]
```

NOTE: