Principali comandi MATLAB utili per il corso di CONTROLLI AUTOMATICI I

RANK: Number of linearly independent rows or columns.

- **K = RANK(X)** is the number of singular values of X that are larger than a default value, equal to MAX(SIZE(X)) * NORM(X) * EPS.
- **K = RANK(X,tol)** is the number of singular values of X that are larger than tol.

Diagrammi di Bode

BODE: Bode frequency response for continuous-time linear systems.

- **BODE(A,B,C,D,IU)** produces a Bode plot from the single input IU to all the outputs of the continuous state-space system (A,B,C,D). IU is an index into the inputs of the system and specifies which input to use for the Bode response. The frequency range and number of points are chosen automatically.
- **BODE(NUM,DEN)** produces the Bode plot for the polynomial transfer function G(s) = NUM(s)/ DEN(s) where NUM and DEN contain the polynomial coefficients in descending powers of s.
- **BODE(A,B,C,D,IU,W)** or **BODE(NUM,DEN,W)** uses the user-supplied frequency vector W which must contain the frequencies, in radians/sec, at which the Bode response is to be evaluated. See LOGSPACE to generate logarithmically spaced frequency vectors.
- **[MAG,PHASE,W] = BODE(A,B,C,D,...)** or **[MAG,PHASE,W] = BODE(NUM,DEN,...)** returns the frequency vector W and matrices MAG and PHASE (in degrees) with as many columns as outputs and length(W) rows. No plot is drawn on the screen.

Controllabilità e raggiungibilità

CTRB: Form controllability matrix.

- **CTRB(A,B)** returns the controllability matrix Kr = [B AB A^2B . . .]

CTRBF: Controllability staircase form.

- **[Abar,Bbar,Cbar,T,k] = CTRBF(A,B,C)** returns a decomposition into the controllable/uncontrollable subspaces, as described below. `T` is the similarity transformation and `k` is a row vector such that `sum(k)` is the dimension of the controllable portion of `A`.
- **[Abar,Bbar,Cbar,T,k] = CTRBF(A,B,C,TOL)** uses tolerance `TOL`.

If Kr=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

$$\texttt{Abar} = \begin{bmatrix} \texttt{Anc} & \texttt{0} \\ \texttt{A21} & \texttt{Ac} \end{bmatrix}, \ \texttt{Bbar} = \begin{bmatrix} \texttt{0} \\ \texttt{Bc} \end{bmatrix}, \ \texttt{Cbar} = \begin{bmatrix} \texttt{Cnc} & \texttt{Cc} \end{bmatrix}$$

where (Ac,Bc) is controllable, and Cc(sI-Ac)$^{-1}$Bc = C(sI-A)$^{-1}$B.

Osservabilità

OBSV: Form observability matrix.

- **OBSV(A,C)** returns the observability matrix Ko = [C; CA; CA^2 . . .]

OBSVF: Observability staircase form.

- **[Abar,Bbar,Cbar,T,k] = OBSVF(A,B,C)** returns a decomposition into the observable/ unobservable subspaces, as described below. `T` is the similarity transformation and `k` is a row vector such that `sum(k)` is the dimension of the observable portion of `A`.
- **[Abar,Bbar,Cbar,T,k] = OBSVF(A,B,C,TOL)** uses tolerance `TOL`.

If Ko=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

$$\texttt{Abar} = \begin{bmatrix} \texttt{Ano} & \texttt{A12} \\ \texttt{0} & \texttt{Ao} \end{bmatrix}, \texttt{Bbar} = \begin{bmatrix} \texttt{Bno} \\ \texttt{Bo} \end{bmatrix}, \texttt{Cbar} = \begin{bmatrix} \texttt{0} & \texttt{Co} \end{bmatrix}$$

where (Ao,Co) is observable, and Co(sI-Ao)$^{-1}$Bo = C(sI-A)$^{-1}$B.

Sistemi equivalenti e forme canoniche

SS2TF: State-space to transfer function conversion.

- **[NUM,DEN] = SS2TF(A,B,C,D,iu)** calculates the transfer function: H(s) = NUM(s)/DEN(s) = C(sI-A)$^{-1}$B + D of the system: $\dot{x}$ = 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.

SS2SS: Similarity transform.

- **[At,Bt,Ct,Dt] = SS2SS(A,B,C,D,T)** performs the similarity transform z = Tx. The resulting state space system is: $\dot{z}$ = [TAT^{-1}] z + [TB] u, y = [CT^{-1}] z + D u

CANON: State-space to canonical form transformation.

- **[Ab,Bb,Cb,Db] = CANON(A,B,C,D,'type')** transforms the continuous state-space system (A, B, C, D) into the canonical form specified by 'type':
  - 'modal' transforms the state-space system into modal form where the system eigenvalues appear on the diagonal. The system must be diagonalizable.
  - 'companion' transforms the state-space system into companion canonical form where the characteristic polynomial appears in the right column.
- **[Ab,Bb,Cb,Db,T] = CANON(A,B,C,D,'type')** returns also the transformation matrix T, such that z = Tx

The modal form is useful for determining the relative controllability of the system modes.
The companion form is ill-conditioned and should be avoided if possible.

Posizionamento dei poli

PLACE

- **K = PLACE(A,B,P)** computes the state feedback matrix K such that the eigenvalues of A-B*K are those specified in vector P.

The complex eigenvalues in the vector P must appear in consecutive complex conjugate pairs. No eigenvalue may be placed with multiplicity greater than the number of inputs.
The displayed "ndigits" is an estimate of how well the eigenvalues were placed. The value seems to give an estimate of how many decimal digits in the eigenvalues of A-B*K match the specified numbers given in the array P. A warning message is printed if the nonzero closed loop poles are greater than 10% from the desired locations specified in P.

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 $\dot{x}$ = 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.

Simulazione di sistemi dinamici lineari a tempo continuo

LSIM: Simulation of continuous-time linear systems to arbitrary inputs.

- **LSIM(A,B,C,D,U,T)** plots the time response of the linear system: $\dot{x}$ = Ax+Bu, y = Cx+Du, to the input time history U. Matrix U must have as many columns as there are inputs, U. Each row of U corresponds to a new time point, and U must have LENGTH(T) rows. The time vector T must be regularly spaced.
- **LSIM(A,B,C,D,U,T,X0)** can be used if initial conditions exist.
- **LSIM(NUM,DEN,U,T)** plots the time response of the polynomial transfer function G(s)=NUM(s)/ DEN(s) where NUM and DEN contain the polynomial coefficients in descending powers of s.
- **[Y,X] = LSIM(A,B,C,D,U,T)** or **[Y,X] = LSIM(NUM,DEN,U,T)** returns the output and state time history in the matrices Y and X. No plot is drawn on the screen. Y has as many columns as there are outputs, y, and with LENGTH(T) rows. X has as many columns as there are states.

STEP: Step response of continuous-time linear systems.

- **STEP(A,B,C,D,IU)** plots the time response of the linear system: $\dot{x}$ = Ax+Bu, y = Cx+Du, to a step applied to the input IU. The time vector is auto- matically determined.
- **STEP(A,B,C,D,IU,T)** allows the specification of a regularly spaced time vector T.
- **[Y,X] = STEP(A,B,C,D,IU,T)** or **[Y.X,T] = STEP(A,B,C,D,IU)** returns the output and state time response in the matrices Y and X respectively. No plot is drawn on the screen. The matrix Y has as many columns as there are outputs, and LENGTH(T) rows. The matrix X has as many columns as there are states. If the time vector is not specified, then the automatically determined time vector is returned in T.
- **[Y,X] = STEP(NUM,DEN,T)** or **[Y,X,T] = STEP(NUM,DEN)** calculates the step response from the transfer function description G(s) = NUM(s)/DEN(s) where NUM and DEN contain the polynomial coefficients in descending powers of s.

IMPULSE: Impulse response of continuous-time linear systems.

- **IMPULSE(A,B,C,D,IU)** plots the time response of the linear system: $\dot{x}$ = Ax+Bu, y = Cx+Du, to an impulse applied to the single input IU. The time vector is automatically determined.

- **IMPULSE(NUM,DEN)** plots the impulse response of the polynomial transfer function G(s) = NUM(s)/DEN(s) where NUM and DEN contain the polynomial coefficients in descending powers of s.
- **IMPULSE(A,B,C,D,IU,T)** or **IMPULSE(NUM,DEN,T)** uses the user-supplied time vector T which must be regularly spaced.
- **[Y,X,T] = IMPULSE(A,B,C,D,...)** or **[Y,X,T] = IMPULSE(NUM,DEN,...)** returns the output and state time history in the matrices Y and X. No plot is drawn on the screen. Y has as many columns as there are outputs and length(T) rows. X has as many columns as there are states.

Simulazione di sistemi dinamici lineari a tempo discreto

DLSIM: Simulation of discrete-time linear systems.

- **DLSIM(A,B,C,D,U)** plots the time response of the discrete system: x[n+1] = Ax[n]+Bu[n], y[n] = Cx[n]+Du[n], to input sequence U. Matrix U must have as many columns as there are inputs, u. Each row of U corresponds to a new time point.
- **DLSIM(A,B,C,D,U,X0)** can be used if initial conditions exist.
- **DLSIM(NUM,DEN,U)** plots the time response of the transfer function description G(z) = NUM(z)/ DEN(z) where NUM and DEN contain the polynomial coefficients in descending powers of z. If LENGTH(NUM)=LENGTH(DEN) then DLSIM(NUM,DEN,U) is equivalent to FILTER(NUM,DEN,U).
- **[Y,X] = DLSIM(A,B,C,D,U)** or **[Y,X] = DLSIM(NUM,DEN,U)** returns the output and state time history in the matrices Y and X. No plot is drawn on the screen. Y has as many columns as there are outputs and LENGTH(U) rows. X has as many columns as there are states and LENGTH(U) rows.

DSTEP: Step response of discrete-time linear systems.

- **DSTEP(A,B,C,D,IU)** plots the response of the discrete system: x[n+1] = Ax[n]+Bu[n], y[n] = Cx[n]+Du[n], to a step applied to the single input IU. The number of points is determined automatically.
- **DSTEP(NUM,DEN)** plots the step response of the polynomial transfer function G(z) = NUM(z)/ DEN(z) where NUM and DEN contain the polynomial coefficients in descending powers of z.
- **DSTEP(A,B,C,D,IU,N)** or **DSTEP(NUM,DEN,N)** uses the user-supplied number of points, N.
- **[Y,X] = DSTEP(A,B,C,D,...)** or **[Y,X] = DSTEP(NUM,DEN,...)** returns the output and state time history in the matrices Y and X. No plot is drawn on the screen. Y has as many columns as there are outputs and X has as many columns as there are states.

DIMPULSE: Impulse response of discrete-time linear systems.

- **DIMPULSE(A,B,C,D,IU)** plots the response of the discrete system: x[n+1] = Ax[n]+Bu[n], y[n] = Cx[n]+Du[n], to an unit sample applied to the single input IU. The number of points is determined automatically.
- **DIMPULSE(NUM,DEN)** plots the impulse response of the polynomial transfer function G(z) = NUM(z)/DEN(z) where NUM and DEN contain the polynomial coefficients in descending powers of z.
- **DIMPULSE(A,B,C,D,IU,N)** or **DIMPULSE(NUM,DEN,N)** uses the user- supplied number of points, N.
- **[Y,X] = DIMPULSE(A,B,C,D,...)** or **[Y,X] = DIMPULSE(NUM,DEN,...)** returns the output and state time history in the matrices Y and X. No plot is drawn on the screen. Y has as many columns as there are outputs and X has as many columns as there are states.