Quick Matlab Reference

Some Basic Commands

Note command syntax is case-sensitive!

help <command>	Display the Matlab help for <command>. Extremely useful! Try "help help".
who	lists all of the variables in your matlab workspace.
whos	list the variables and describes their matrix size.
clear	deletes all matrices from active workspace.
clear x	deletes the matrix x from active workspace.
save	saves all the matrices defined in the current session into the file, matlab.mat.
load	loads contents of matlab.mat into current workspace.
save filename	saves the contents of workspace into filename.mat
save filename x y z	saves the matrices x, y and z into the file titled filename.mat.
load filename	loads the contents of filename into current workspace; the file can be a binary (.mat) file or an ASCII file.
!	the ! preceding any unix command causes the unix command to be executed from matlab.

Matrix commands

[1 2; 3 4]	Create the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.
zeros(n)	creates an nxn matrix whose elements are zero.
zeros(m,n)	creates a m-row, n-column matrix of zeros.
ones(n)	creates a n x n square matrix whose elements are 1's
ones(m,n)	creates a mxn matrix whose elements are 1's.
ones(A)	creates an m x n matrix of 1's, where m and n are based on the size of an existing matrix, A.
zeros(A)	creates an mxn matrix of 0's, where m and n are based on the size of the existing matrix, A.
eye(n)	creates the nxn identity matrix with 1's on the diagonal.
A'	Transpose of A

Plotting commands

plot(x,y)	creates an Cartesian plot of the vectors x & y.
plot(y)	creates a plot of y vs. the numerical values of the elements in the y-vector.
semilogx(x,y)	plots log(x) vs y.
semilogy(x,y)	plots x vs log(y)
loglog(x,y)	plots log(x) vs log(y).
grid	creates a grid on the graphics plot.
title('text')	places a title at top of graphics plot.
xlabel('text')	writes 'text' beneath the x-axis of a plot.
ylabel('text')	writes 'text' beside the y-axis of a plot.
text(x,y,'text')	writes 'text' at the location (x,y) .
text(x,y,'text','sc')	writes 'text' at point x,y assuming lower left corner is (0,0) and upper right corner is (1,1).
gtext('text')	writes text according to placement of mouse
hold on	maintains the current plot in the graphics window while executing subsequent plotting commands.
hold off	turns OFF the 'hold on' option.
polar(theta,r)	creates a polar plot of the vectors r & theta where theta is in radians.
bar(x)	creates a bar graph of the vector x. (Note also the command stairs(y).)
bar(x,y)	creates a bar-graph of the elements of the vector y, locating the bars according to the vector elements of 'x'. (Note also the command stairs(x,y).)
hist(x)	creates a histogram. This differs from the bargraph in that frequency is plotted on the vertical axis.
mesh(z)	creates a surface in xyz space where z is a matrix of the values of the function z(x,y). z can be interpreted to be the height of the surface above some xy reference plane.
surf(z)	similar to mesh(z), only surface elements depict the surface rather than a mesh grid.
contour(z)	draws a contour map in xy space of the function or surface z.
meshc(z)	draws the surface z with a contour plot beneath it.
meshgrid	[X,Y]=meshgrid(x,y) transforms the domain specified by vectors x and y into arrays X and Y that can be used in evaluating functions for 3D mesh/surf plots.
print	sends the contents of graphics window to printer.
print filename -dps	writes the contents of current graphics to 'filename' in postscript format.

Misc. commands

length(x)	returns the number elements in a vector.
size(x)	returns the size m(rows) and n(columns) of matrix x.
rand	returns a random number between 0 and 1.
randn	returns a random number selected from a normal distribution with a mean of 0 and variance of 1.
rand(A)	returns a matrix of size A of random numbers.
fliplr(x)	reverses the order of a vector. If x is a matrix, this reverse the order of the columns in the matrix.
flipud(x)	reverses the order of a matrix in the sense of exchanging or reversing the order of the matrix rows. This will not reverse a row vector!
reshape(A,m,n)	reshapes the matrix A into an mxn matrix from element (1,1) working column-wise.

Some symbolic toolbox commands

syms t	Define the variable t to be symbolic. The value of t is now t.
f = t^3+sin(t)	Let f be $t^3 + \sin(t)$ symbolically.
diff(f)	Differentiate f.
diff(f,t)	Differentiate f with resp. to t.
int(f)	Integrate f.
int(f,t,a,b)	Integrate f with resp. to t from a to b.
inv(A)	Matrix inverse of A.
det(A)	Determinant.
rank(A)	Rank.
eig(A)	Eigenvalues and eigenvectors.
poly(A)	Characteristic polynomial.
expm(A)	Matrix exponential.
help symbolic	Get help on all symbolic toolbox commands.

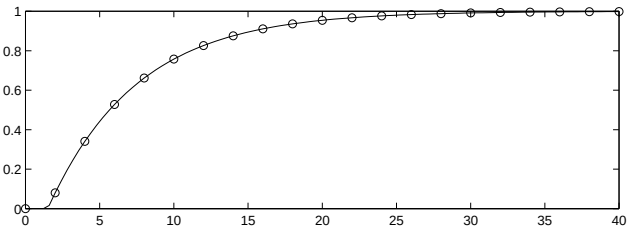
Short guide to Control Systems Toolbox

This section is an introduction on how to use Control Systems Toolbox for control analysis and design, especially of computer controlled systems. The basic data structure is the LTI (linear time-invariant) model. There is a number of ways to create, manipulate and analyze models. Some operations are best done on the LTI system, and others directly on the matrices and polynomials of the model. First, some examples on basic system manipulation:

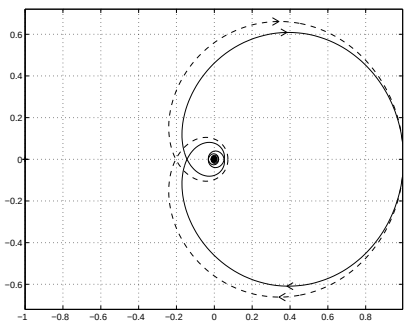
```
>> % Continuous transfer function G1(s)=e^(-1.5s)/(6s+1) :
>> G1 = tf(1,[6 1],'InputDelay',1.5)
      Transfer function:
                    1
   exp(-1.5*s) * -------
                 6 s + 1

      Input delay: 1.5
>> % ZOH sampling of G1 using h=2:
>> H1 = c2d(G1,2)
      Transfer function:
      0.07996 z + 0.2035
      ------------------
        z^2 - 0.7165 z

      Sampling time: 2
>> % Extract zeros, poles and gain from H1 as vectors:
>> [z,p,k] = zpkdata(H1,'v')
z =
   -2.5453
p =
         0
    0.7165
k =
    0.0800
>> % Calculate step responses:
>> [yc,tc] = step(G1,40); [yd,td] = step(H1,40);
>> plot(tc,yc,'-',td,yd,'o')
```




```
>> % Nyquist plots of G1 and H1 (positive and negative frequencies):
>> nyquist(G1,H1);
```



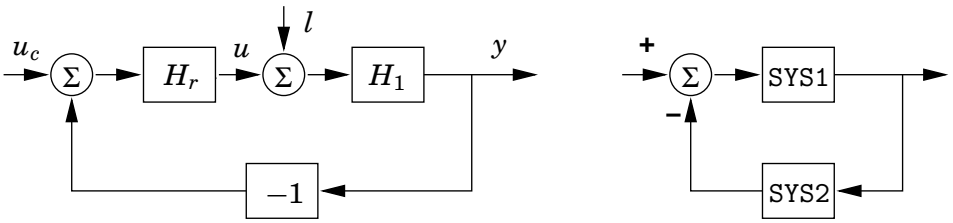
State feedback example

Problem 5a from the exam in August 1998, available on the web.

```
>> % Discrete-time state space model, set for example h=1;
>> Phi = [1.7 -0.72; 1 0]; Gamma = [1 0]'; C = [1 -0.7]; D=0; h=1;
>> sys1 = ss(Phi,Gamma,C,D,h);
>> % Check reachability:
>> Wc = ctrb(sys1)
Wc =
    1.0000    1.7000
         0    1.0000
>> rank(Wc)
ans =
     2
>> % Solve the discrete time characteristic polynomial:
>> desired_poles = roots([1 -1.2 0.5]);
>> % Place the poles:
>> L = place(Phi,Gamma,desired_poles)
L =
    0.5000   -0.2200
>> % Use Lc to set static gain = 1:
>> Lc = 1 / (C/(eye(2)-(Phi-Gamma*L))*Gamma)
Lc =
    1.0000
```

Connecting systems

LTI systems can be interconnected in a number of ways. For example, you may add and multiply systems (or constants) to achieve parallel and series connections, respectively. Assume that H_1 above is controlled by a PI controller with $K = 1$, $Ti = 4$ and $h = 2$, according to the standard block diagram to the left:




```
>> % Discrete-time PI controller with K=1, Ti=4, h=2:
>> K=1; Ti=4; h=2;
>> Hr = K*(1+tf(h/Ti,[1 -1],h))
      Transfer function:
      z - 0.5
      -------
       z - 1

      Sampling time: 2
```

There is a function `feedback` for constructing feedback systems. The block diagram to the right is obtained by the call `feedback(SYS1,SYS2)`. Note the sign conventions. To find the transfer function from the set point u_c to the output y, you identify that `SYS1` is H_1H_r and `SYS2` is 1. You can also use block matrices of systems to describe multiinput multioutput systems.

```
>> Hyuc = feedback(H1*Hr,1);
>> %
>> % You can also derive the matrix of transfer functions,
>> % from inputs uc and l to outputs y and u:
>> %                         +--------+
>> %              uc ------->|        |----> y
>> %                         | CLSYS  |
```

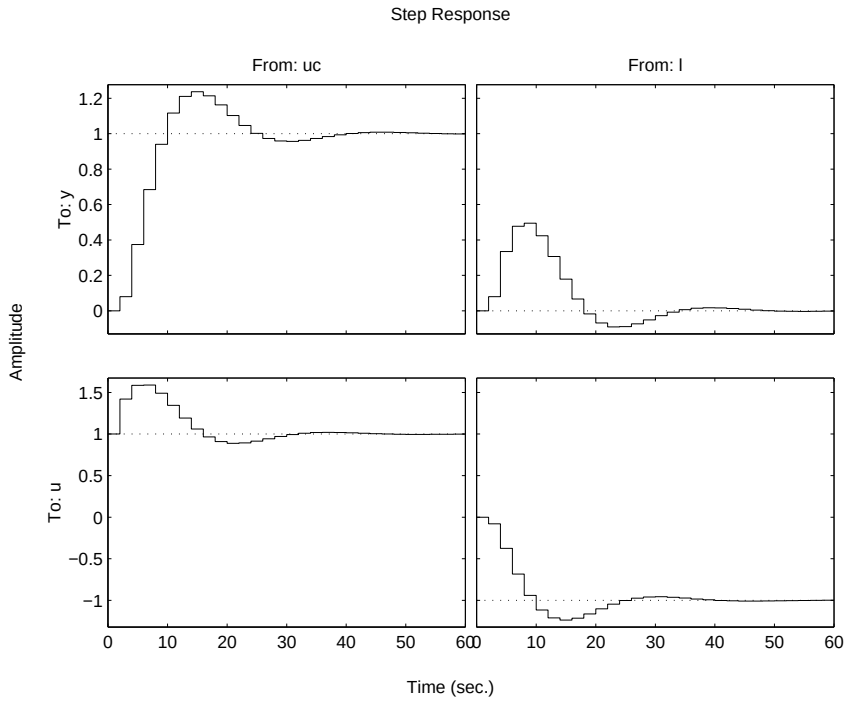
```
>> %               l ----->|       |----> u
>> %                       +-------+
>> % From y=H1(l+u), u=Hr(uc-y) it follows that
>> SYS1=[0,H1;Hr,0]; SYS2=[1,0;0,-1];
>> CLSYS = feedback(SYS1,SYS2);
>> % It is possible to assign names to the signals:
>> set(CLSYS,'InputName',{'uc','l'},'OutputName',{'y','u'})
>> CLSYS
      Transfer function from input "uc" to output...
             0.07996 z^2 + 0.1635 z - 0.1018
       y:  -----------------------------------
           z^3 - 1.637 z^2 + 0.8801 z - 0.1018

              z^3 - 1.217 z^2 + 0.3583 z
       u:  -----------------------------------
           z^3 - 1.637 z^2 + 0.8801 z - 0.1018

      Transfer function from input "l" to output...
             0.07996 z^2 + 0.1236 z - 0.2035
       y:  -----------------------------------
           z^3 - 1.637 z^2 + 0.8801 z - 0.1018

            -0.07996 z^2 - 0.1635 z + 0.1018
       u:  -----------------------------------
           z^3 - 1.637 z^2 + 0.8801 z - 0.1018

      Sampling time: 2
>> % It is often better to do the feedback in state space:
>> CLSYS = feedback(ss(SYS1),SYS2);
>> set(CLSYS,'InputName',{'uc','l'},'OutputName',{'y','u'});tf(CLSYS)
>> % Show step responses from set point and load disturbance to output and control signal:
>> step(CLSYS)
```



In order to mix continuous-time and discrete-time systems, you should simulate in Simulink. The plots above do NOT show the output of the continuous system between the sampling points.

Some useful functions from Control Systems Toolbox

Do `help` <*function*> to find possible input and output arguments.

```
Creation and conversion of continuous or discrete time LTI models.
  ss          - Create/convert to a state-space model.
  tf          - Create/convert to a transfer function model.
  zpk         - Create/convert to a zero/pole/gain model.
  ltiprops    - Detailed help for available LTI properties.
  ssdata etc. - Extract data from a LTI model.
  set         - Set/modify properties of LTI models.
  get         - Access values of LTI model properties.

Sampling of systems.
  c2d         - Continuous to discrete conversion.
  d2c         - Discrete to continuous conversion.

Model dynamics.
  pole, eig   - System poles.
  pzmap       - Pole-zero map.
  covar       - Covariance of response to white noise.

State-space models.
  ss2ss       - State coordinate transformation.
  canon       - State-space canonical forms.
  ctrb, obsv  - Controllability and observability matrices.

Time response.
  step        - Step response.
  impulse     - Impulse response.
  initial     - Response of state-space system with given initial state.
  lsim        - Response to arbitrary inputs.
  ltiview     - Response analysis GUI.
  gensig      - Generate input signal for LSIM.
  stepfun     - Generate unit-step input.

Frequency response.
  bode        - Bode plot of the frequency response.
  nyquist     - Nyquist plot.
  ltiview     - Response analysis GUI.

System interconnections.
  + and -     - Add and subtract systems (parallel connection).
  *           - Multiplication of systems (series connection).
  / and \     - Division of systems (right and left, respectively).
  inv         - Inverse of a system.
  [ ]         - Horizontal/vertical concatenation of systems.
  feedback    - Feedback connection of two systems.

Classical design tools.
  rlocus      - Root locus.
  rlocfind    - Interactive root locus gain determination.
  rltool      - Root locus design GUI
  place       - Pole placement (state feedback or estimator).
  estim       - Form estimator given estimator gain.
  reg         - Form regulator given state-feedback and estimator gains.

LQG design tools. Notation differs from CCS.
  lqr,dlqr    - Linear-quadratic (LQ) state-feedback regulator.
  lqry        - LQ regulator with output weighting.
  lqrd        - Discrete LQ regulator for continuous plant.
  kalman      - Kalman estimator.
  kalmd       - Discrete Kalman estimator for continuous plant.
  lqgreg      - Form LQG regulator given LQ gain and Kalman estimator.

Matrix equation solvers.
  dlyap       - Solve discrete Lyapunov equations.
  dare        - Solve discrete algebraic Riccati equations.
```