System Definition:

State Space Model

sys_ss = ss(A,B,C,D)	creates a SS-model
sys_ss = ss(*A,B,C,D,Ts*)	creates the discrete-time SS-model with sample time Ts (in seconds)
sys_ss = ss(sys,'minimal')	creates a SS-realization with no uncontrollable or unobservable states
sys_ss = minreal(ss(sys))	creates a SS-realization with no uncontrollable or unobservable states
sys_ss = ss(sys_tf)	converts a TF-model to a SS-form
sys_ss = ss(sys_zpk)	converts a ZPK-model to a SS-form

Transfer Function

s = tf('s')	creates a Laplace variable 's' that you can use to specify a TF
z = tf('z',ts)	creates a discrete-time variable 'z' that you can use to specify a TF
sys_tf = tf(num,den)	creates a continuous-time TF
sys_tf = tf(num,den, 'InputDelay',T)	creates a continuous-time TF with time delay T
sys_tf = tf(num,den,ts)	creates a discrete-time TF with sample time ts (in seconds)
sys_tf = tf(sys_ss)	converts a SS-model to a TF-form
sys_tf = tf(sys_zpk)	converts a ZPK-model to a TF-form

Zero-Pole-Gain model

sys_zpk = zpk(z,p,k)	creates a ZPK-model
sys_zpk = zpk(Z,p,k,Ts)	creates a discrete-time ZPK-model with sample time Ts (in seconds)
sys_zpk = zpk(sys_ss)	converts a SS-model to a ZPK-form
sys_zpk = zpk(sys_tf)	converts a TF-model to a ZPK-form
s = zpk('s')	creates a Laplace variable 's' that you can use to specify a ZPK-model
z = zpk('z',Ts)	creates a discrete-time variable 'z' that you can use to specify a ZPK-model

Sysstem Circuits

sys_serie = series(sys1,sys2)	Series circuit // L = series(C,P)
sys_feed = feedback(sys1,sys2)	Parallel circuit // T = feedback(L,1) ≙sys = sys_1 / (sys1 + sys2)

System Analysis:

Controllability

R = ctrb(A,B)	calculates the controlability matrix of the SS-model
R = ctrb(sys)	calculates the controlability matrix of the SS-model

Observability

O = obsv(A,C)	calculates the observability matrix of the SS-model
O = obsv(sys)	calculates the observability matrix of the SS-model

Properties

p = pole(sys)	computes the poles p (column vector) of a system model *[For MIMO TF, the poles are returned as the union of the poles for each SISO entry]*
z = zero(sys)	computes zeros z (column vector) of SISO system
z = tzero(sys)	computes the invariant zeros of a MIMO system *[minimal realization: the invariant zeros coincide with the transmission zeros]*
[z,p,k,Ts] = zpkdata(sys)	returns the zeros z, poles p, gain(s) k and sample time Ts of the ZPK-model

Analysis

frsp =evalfr(sys,i*w)	evaluates the TF of the TF, SS, or ZPK model at the complex number i*w
[mag,phase,wout] = bode(H)	returns the magnitude and phase for each frequency in the vector wout
ydb = mag2db(y)	convert magnitude to decibels (dB)
y = db2mag(ydb)	convert decibels (dB) to magnitude
S = stepinfo(sys)	computes the step-response characteristics for a dynamic system model sys

CS Plots

pzmap(sys)	plots the poles and zeros into the complex plane
nyquist(sys)	plots the Nyquist plot of LTI models
step(sys)	plots the step response of LTI systems
impulse(sys)	plots the impulse response of LTI systems
margin(sys)	plots the Bode response of sys and indicates the gain and phase margins
bode(sys)	pots Bode diagram of frequency response
bode(sys,w)	plots system responses for frequencies specified by w
bodemag(sys)	plots the magnitude of the frequency response
step/impulse/bode(sys1,sys2)	plots response of several models
step/impulse(sys,Tfinal)	simulates the response from t = 0 to the final time t = Tfinal
step/impulse(sys,t)	uses the user-supplied time vector t for simulation
[y,t] = step/impulse(sys)	returns the output response y, the time vector t used for simulation

PID

[Gm,Pm,Wcg,Wcp] = margin(sys)	% Gm: Gain-margin, Pm: Phase-margin, Wcg: frequency @ gain-margin, Wcp: frequency @ phase-margin/ cut-off-frequency
k = dcgain(sys)	% computes the DC gain k of the LTI model sys, P(0)
k_crit = Gm;	% gain margin [-]
T_crit = 2*pi/Wcg;	% period of critical oscillation [s]

Discretization

sys_d = c2d(sys_c,Ts, 'tustin')	discretizes the continuous-time dynamic system model using zero-order hold on the inputs and a sample time of Ts

SVD

s = svd(A)	returns the singular values of matrix A in descending order
[U,S,V] = svd(sys)	performs a singular value decomposition of matrix A
x max = V(:,1)	
x min = V(:,n)	
sigma(sys)	plots the singular values of the frequency response of a model sys

LQR

[K,Phi,P] = lqr(sys,Q,R)	K: optimal gain matrix, Phi: solution Phi of the Riccati equation,
[K,Phi,P] = lqr(A,B,Q,R)	P: closed-loop poles of the resulting system

Plotting

figure	creates a new figure window using default property values
figure(N_1,V_1,...,N_N,_N)	modifies properties of the figure
('Color','w')	sets the background color
('Name','Titel of the figure')	specify the Name (the resulting title includes by default the figure number)
('NumberTitle','off')	the resulting title does not include the figure number
grid on / off	displays the major grid lines for the current axes
hold on / off	retains plots in the current axes
box on / off	displays the box outline around the current axes
title('Title')	add a title to the chart
xlabel('xlabel') / ylabel('ylabel')	add axis labels to the chart
legend(label1,...,labelN)	add a legend to the graph
('Location','southwest')	best, noth, south, east, west, northeast, nothwest, southeast, southwest
('NumColumns',2)	
('Orientation','horizontal')	vertical
axis([xmin xmax ymin ymax])	sets the x-axis & y-axis limits for the current axes
xlim([xmin xmax])	sets the x-axis limits for the current axes
plot(x,y1,x,y2,'--b*',x,y3,':')	Line Style, Color, Marker
subplot(m,n,p)	divides the current figure into an m-by-n grid, p current position
subplot(2,2,[3,4]);	a third subplot that spans the lower half of the figure
x = linspace(x1,x2,n)	generates n points (n default = 100)
semilogx()	plot data as logarithmic scales for the x-axis

Style

Color	r, g (green), b (blue), c (cyan), m (magenta), y (yellow), k (black), w (white)
Line Style	'-' (Solid line), '--' (Dashed line), ':' (Dotted line), '-.' (Dash-dot line)
Marker	o, +, *, ., x, s (square), d (Diamond), ^ (Upward-pointing triangle), v (Downward-pointing triangle)

Simulink

sim('Dateiname', Variable)	starts Simulink-Simulation "Dateiname.slx"

Examples:

```
% clean up
    clear all;      % removes all variables from the current workspace
    close all;      % deletes all figures
    clc;            % Clear Command Window

% plant definition
    s   = tf('s');
    num = [1, 2];
    den = [1, 7, 26, 24, 2];
    P   = tf(num,den);

% PI controler
    C = kp*(1+1/Ti/s);

% PID controler
    C = kp*((Td*Ti*s^2+Ti*s+1)/(Ti*s));

% minimal return difference: Variante 1
    D = 1 + L;
    [mag,phs] = bode(D);
    mu_min = min(mag);

% minimal return difference: Variante 2
    mu_min = min(abs(bode(1 + L)))

% define an angular vector
    anglevector = linspace(0,2*pi,101);

    % Werte y=M*x für jedes x aus
    for k = 1:length(anglevector)
        x(:,k) = [cos(anglevector(k));sin(anglevector(k))];
        y(:,k) = M * x(:,k);
    end

% plot a circle
    r = 2;
    xc = 4;
    yc = 3;
    theta = linspace(0,2*pi);
    x = r*cos(theta) + xc;
    y = r*sin(theta) + yc;
    plot(x,y)
    axis equal
```

```
% plot a system analysis
    figure('Name','Analysis')
    grid on; hold on; box off;

    subplot(2,2,1);
    step(L1, L2)
    title('Step Response');
    xlabel('Time [s]');
    ylabel('Amplitude [-]');
    xlim([0 20]);
    ylim([0 5]);
    legend('PI_controler','P_controler');

    subplot(2,2,2);
    impulse(L1,L2)
    title('Impulse Response');
    xlabel('Time [s]');
    ylabel('Amplitude [-]');
    xlim([0 20]);
    ylim([0 5]);
    legend('PI_controler','P_controler');

    subplot(2,2,[3,4]);
    title('Bode Diagram')
    margin(P)

% Open-Loop-Gain
    L_LQR = ss(A,B,K,0);

% Define LQRI matrices
    A_tilde = [A zeros(n,m); -C zeros(m,m)];
    B_tilde = [B; zeros(m,m)];
    C_tilde = [C zeros(m,m); zeros(m,n), gamma*eye(m,m)];
    Q_tilde = C_tilde'*C_tilde;
    K_tilde = lqr(A_tilde,B_tilde,Q_tilde,r_tilde);

    K_lqri = K_tilde(:,1:n);
    Ki_lqri = -K_tilde(:,n+1:end);

% Beobachterverstärkung berechnen
    L = lqr(A',C',B*B',q*eye(ny))';
```

Shortcuts:

ctr r	comments the selected lines
ctr t	uncomments the selected lines
ctr i	correct indentation
f5	run code
ctr + tab	switch between panels

Command Window

disp(X)	displays the value of variable X without printing the variable name