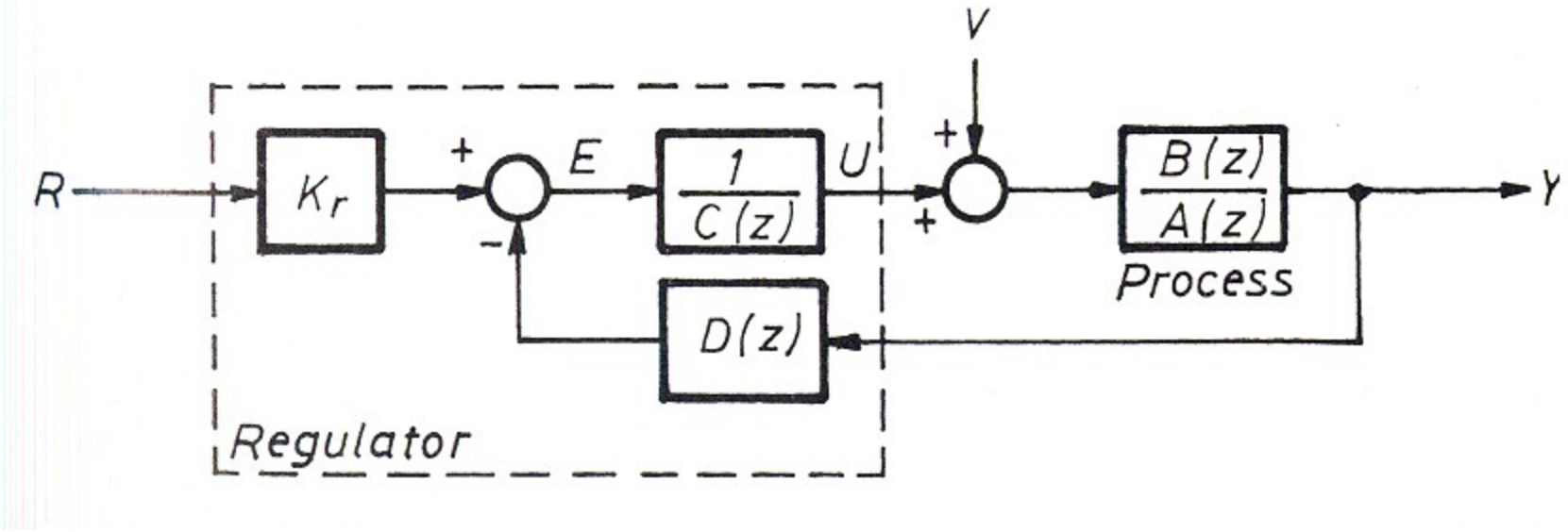
How to Create Matlab Script and Simulink Model for Designing a Pole Placement Controller

William Sandqvist (william@kth.se), Leif Lindbäck (leifl@kth.se)

Pole Placement Controller Design



$$H_{tot} = \frac{Y(z)}{R(z)} = \frac{K_r \times B(z)}{A(z)C(z) + B(z)D(z)}$$

$$P(z) = A(z)C(z) + B(z)D(z)$$

Pole Placement Equation

$$A(z)C(z)+B(z)D(z)=P(z)$$

Choose degree: $n_p = n_a + n_b - 1 \ [degree]$

$$n_c = n_b - 1 \quad n_d = n_a - 1$$

Set: $C(z) = 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{nc} z^{-n}$

$$D(z) = d_0 + d_1 z^{-1} + d_2 z^{-2} + \ldots + d_{nd} z^{-n}$$

$$K_r = \frac{P(z=1)}{B(z=1)}$$

Pole Placement Example

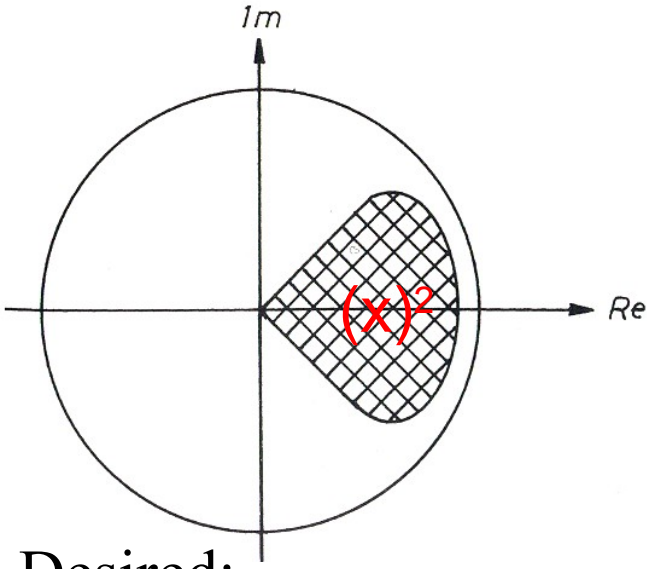
$$y(k) = 0{,}82\, y_{k-1} + 0{,}5\, u_{k-1} + 0{,}5\, u_{k-2} \quad \{Z:\}$$

$$Y = 0{,}82\, Yz^{-1} + 0{,}5\, Uz^{-1} + 0{,}5\, Uz^{-2}$$

$$H(z) = \frac{U}{Y} = \frac{0{,}5\, z^{-1} + 0{,}5\, z^{-2}}{1 - 0{,}82\, z^{-1}}$$

$$B(z) = 0{,}5\, z^{-1} + 0{,}5\, z^{-2}$$

$$A(z) = 1 - 0{,}82\, z^{-1}$$



Desired:
dubble pole in $z = 0{,}5$

Pole Placement Example

$$B(z) = 0{,}5\,z^{-1} + 0{,}5\,z^{-2}$$
$$A(z) = 1 - 0{,}82\,z^{-1}$$

Desired:
dubble pole in $z = 0{,}5$

$$P(z) = (1 - 0{,}5\,z^{-1})^2 = 1 - z^{-1} + 0{,}5\,z^{-2} \;\Rightarrow\; n_p = 2$$

$$n_c = n_b - 1 = 2 - 1 = 1 \;\Rightarrow\; C(z) = 1 + c_1 z^{-1}$$
$$n_d = n_a - 1 = 1 - 1 = 0 \;\Rightarrow\; D(z) = d_0$$

$AC + BD = P$:

$$(1 - 0{,}82\,z^{-1})(1 + c_1 z^{-1}) + (0{,}5\,z^{-1} + 0{,}5\,z^{-2})(d_0) = 1 - z^{-1} + 0{,}25\,z^{-2}$$

Pole Placement Example

$$\left(1-0{,}82\,z^{-1}\right)\left(1+c_1 z^{-1}\right)+\left(0{,}5\,z^{-1}+0{,}5\,z^{-2}\right)\left(d_0\right)=$$

$$=1+c_1 z^{-1}-0{,}82\,z^{-1}-0{,}82\,c_1 z^{-2}+0{,}5\,d_0 z^{-1}+0{,}5\,d_0 z^{-2}=$$

$$=1+\left(c_1-0{,}82+0{,}85\,d_0\right)z^{-1}+\left(0{,}5\,d_0-0{,}82\right)z^{-2}$$

$$=1-z^{-1}+0{,}25\,z^{-2}$$

AC+BD = P

$$c_1-0{,}82+0{,}5\,d_0=-1 \;\Leftrightarrow\; c_1+0{,}5\,d_0=-0{,}18$$

$$0{,}5\,d_0-0{,}82\,c_1=0{,}25 \;\Leftrightarrow\; -0{,}82\,c_1+0{,}5\,d_0=0{,}25$$

Pole Placement Example

$$c_1 - 0{,}82 + 0{,}5\,d_0 = -1 \;\Leftrightarrow\; c_1 + 0{,}5\,d_0 = -0{,}18$$

$$0{,}5\,d_0 - 0{,}82\,c_1 = 0{,}25 \;\Leftrightarrow\; -0{,}82\,c_1 + 0{,}5\,d_0 = 0{,}25$$

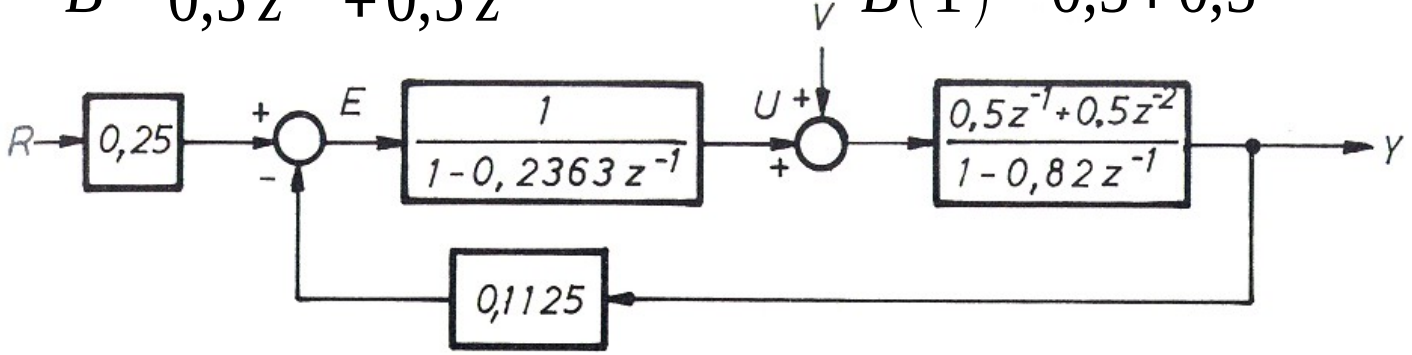
$$\begin{pmatrix} 1 & 0{,}5 \\ -0{,}82 & 0{,}5 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ d_0 \end{pmatrix} = \begin{pmatrix} -0{,}18 \\ 0{,}25 \end{pmatrix}$$

$$c_1 = -0{,}236 \quad d_0 = 0{,}113$$

```
» X=[1, 0.5; -0.82, 0.5]\[-0.18, 0.25]'
X = -0.2363   c1
     0.1125   d0
»
```

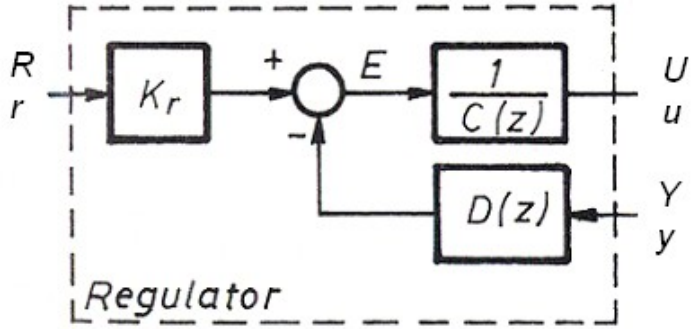
Pole Placement Example

$$\frac{P}{B}=\frac{1-z^{-1}+0{,}25\,z^{-2}}{0{,}5\,z^{-1}+0{,}5\,z^{-2}} \;\Rightarrow\; K_r=\frac{P(1)}{B(1)}=\frac{1-1+0{,}25}{0{,}5+0{,}5}=0{,}25$$



$$E=R\cdot K_r-Y\cdot D$$

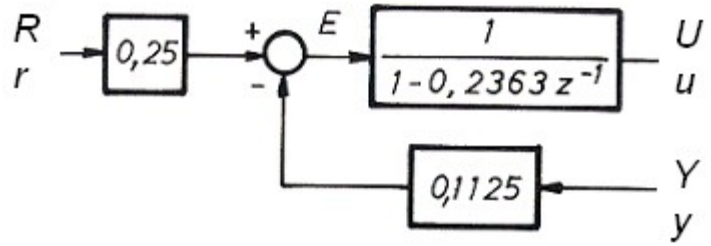
$$U=\frac{E}{C}$$



Pole Placement Example

$$E = R \cdot K_r - Y \cdot D$$

$$U = \frac{E}{C}$$



$$E = 0{,}25\ R - 0{,}1125\ Y$$

$$U = \frac{E}{1 - 0{,}2363\, z^{-1}} \quad \Rightarrow \quad U\left(1 - 0{,}2363\, z^{-1}\right) = E$$

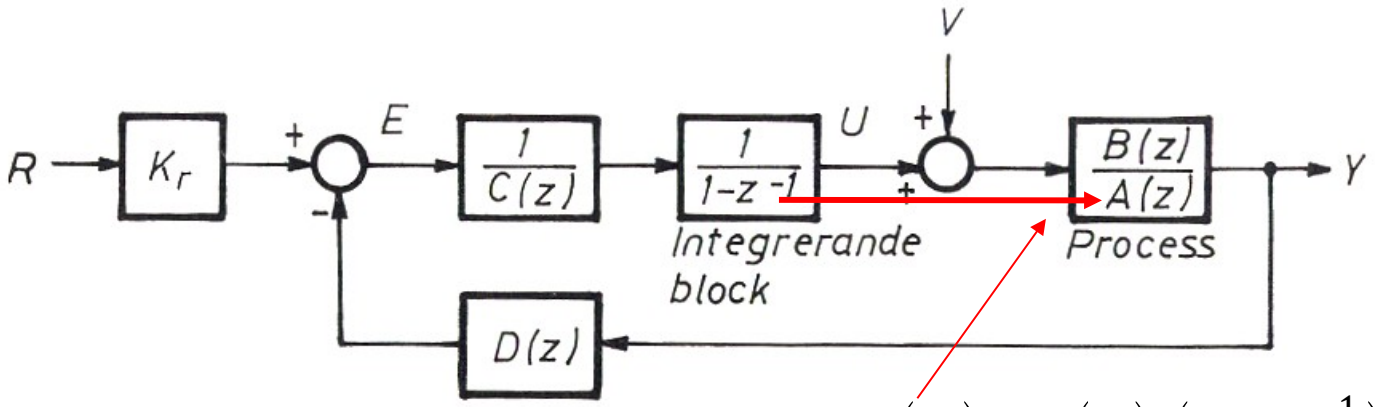
$$\Rightarrow \quad U = E - 0{,}2363\ U\ z^{-1}$$

- Difference equations:

$$e_k = 0{,}25\, r_k - 0{,}1125\, y_k \qquad u_k = e_k - 0{,}2363\ u_{k-1}$$

Integrating Block

To avoid residual error we need an **integrating controller**.



$$A(z) = A(z) \cdot (1 - z^{-1})$$

During calculation the controller is placed in $A(z)$.
(= easiest)

With Integrating Block

Searched $P(z)$ has dubble pole in $z = 0{,}5$ and *extra* pole in origo.

$$B(z) = 0{,}5\,z^{-1} + 0{,}5\,z^{-2}$$

$$A(z) = 1 - 0{,}82\,z^{-1} \Rightarrow A(z) = \left(1 - 0{,}82\,z^{-1}\right)\left(1 - z^{-1}\right)$$

$$P(z) = \left(1 - 0{,}5\,z^{-1}\right)^2 \cdot \left(1 - 0 \cdot z^{-1}\right) = 1 - z^{-1} + 0{,}5\,z^{-2} \;\Rightarrow\; n_p = 2$$

$$n_c = n_b - 1 = 2 - 1 = 1 \;\Rightarrow\; C(z) = 1 + c_1 z^{-1}$$

$$n_d = n_a - 1 = 2 - 1 = 1 \;\Rightarrow\; D(z) = d_0 + d_1 z^{-1}$$

$AC + BD = P$:

$$\left(1 - 0{,}82z^{-1}\right)\left(1 - z^{-1}\right)\left(1 + c_1 z^{-1}\right) + \left(0{,}5\,z^{-1} + 0{,}5\,z^{-2}\right)\left(d_0\right) = 1 - z^{-1} + 0{,}25\,z^{-2}$$

With Integrating Block

$$\left(1-0{,}82\,z^{-1}\right)\left(1-z^{-1}\right)\left(1+c_1\,z^{-1}\right)+\left(0{,}5\,z^{-1}+0{,}5\,z^{-2}\right)\left(d_0\right)=$$

$$=1+\left(c_1-1{,}82+0{,}5\,d_0\right)\,z^{-1}+\left(0{,}82-1{,}82\,c_1+0{,}5\,d_1+0{,}5\,d_0\right)\,z^{-2}+$$

$$+\left(0{,}82\,c_1+0{,}5\,d_1\right)z^{-3}=\;1-z^{-1}+0{,}25\;z^{-2}$$

$$\left(c_1-1{,}82+0{,}5\,d_0\right)=-1\quad c_1+0{,}5\,d_0+0\,d_1=0{,}82$$

$$\left(0{,}82-1{,}82\,c_1+0{,}5\,d_1+0{,}5\,d_0\right)=0{,}25\quad -1{,}82\,c_1+0{,}5\,d_0+0{,}5\,d_1=-0{,}57$$

$$\left(0{,}82\,c_1+0{,}5\,d_1\right)=0\quad 0{,}82\,c_1+0\,d_0+0{,}5\,d_1=0$$

```
» X=[1,0.5,0;-1.82,0.5,0.5;0.82,0,0.5]\[0.82,-0.57,0]'
X =  0.382    c1
     0.876    d0
    -0.626    d1
»
```

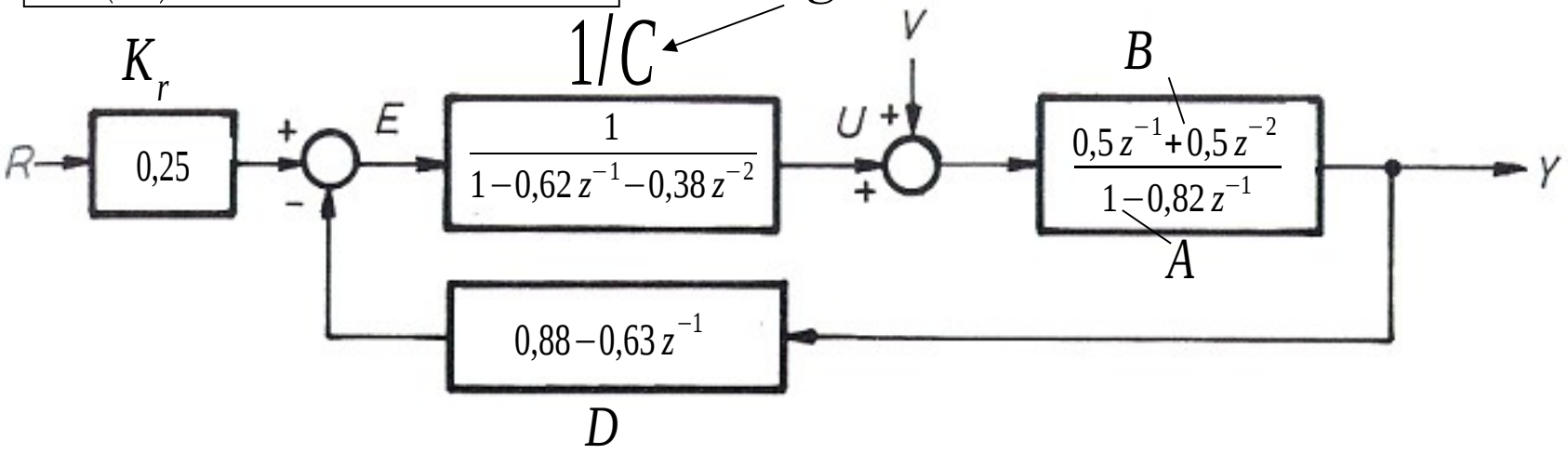
With Integrating Block

$$\frac{P}{B}=\frac{1-z^{-1}+0{,}25\,z^{-2}}{0{,}5\,z^{-1}+0{,}5\,z^{-2}} \;\Rightarrow\; K_r=\frac{P(1)}{B(1)}=\frac{1-1+0{,}25}{0{,}5+0{,}5}=0{,}25$$

$$C=C(z)\cdot(1-z^{-1})=(1-z^{-1})(1+0{,}382\,z^{-1})=1-0{,}618\,z^{-1}-0{,}382\,z^{-2}$$

$$D(z)=0{,}876-0{,}626\,z^{-1}$$

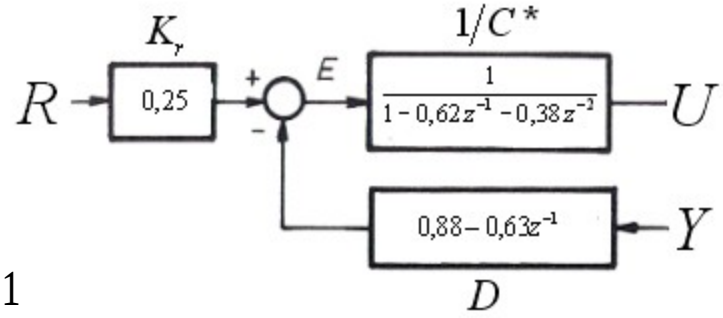
Integrator now in C block!



Controller Difference Equations

$$E = R \cdot K_r - Y \cdot D$$

$$U = \frac{E}{C}$$



$$E = 0{,}25\ R - 0{,}88\ Y + 0{,}63\ Y\ z^{-1}$$

$$U = \frac{E}{1 - 0{,}62\ z^{-1} - 0{,}38\ z^{-2}} \quad \Leftrightarrow \quad U - 0{,}62\ U\ z^{-1} - 0{,}38\ U\ z^{-2} = E$$

$$U = E + 0{,}62\ U\ z^{-1} + 0{,}38\ U\ z^{-2}$$

- Difference Equations:

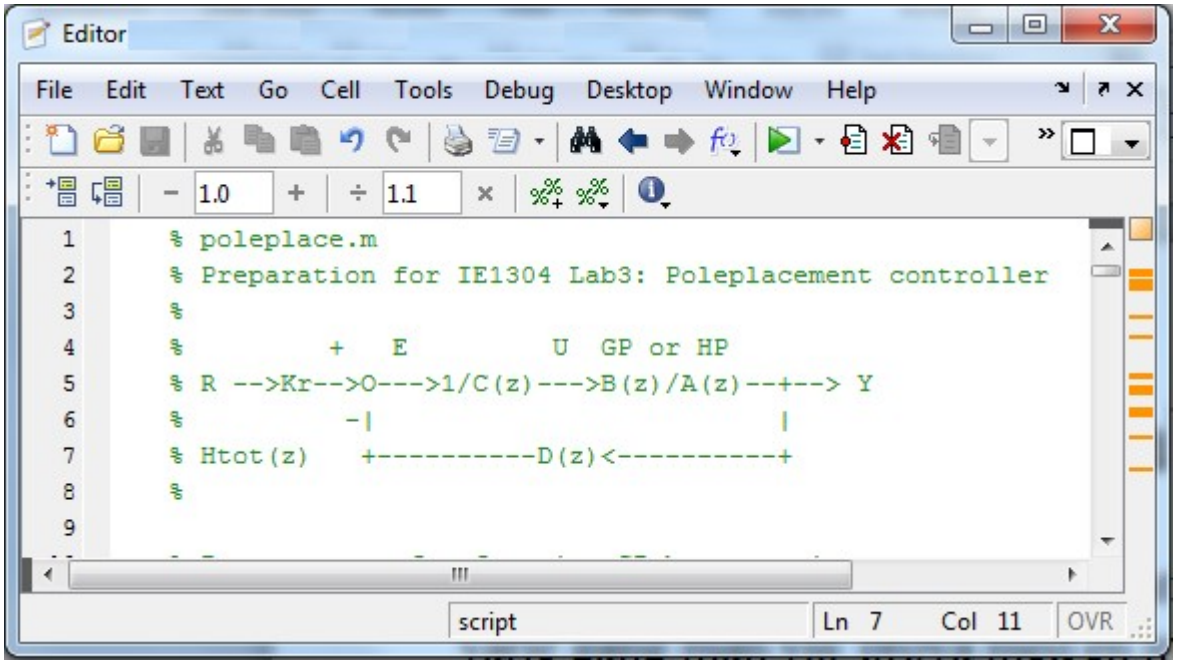
$$e_k = 0{,}25\ r_k - 0{,}88\ y_k + 0{,}63\ y_{k-1} \qquad u_k = e_k + 0{,}62\ u_{k-1} + 0{,}38\ u_{k-2}$$

Matlab Script for Lab 3

Pole placement requires extensive calculations every time you want to try new poles. This is something you do not have time to do during the lab - you have to automate calculations using a Matlab script.

When solving the preparation tasks, develop a MATLAB script to be able to quickly redo the calculations during the lab with the values of the real process.

Matlab Script for Lab 3



Start the script with a comment (**%**). That comment is printed when writing **`help poleplace`** in the Matlab command window - good practice to show what the script does.

Matlab Script for Lab 3

Prior to the lab, we assume that process gain and time constants have the following values:

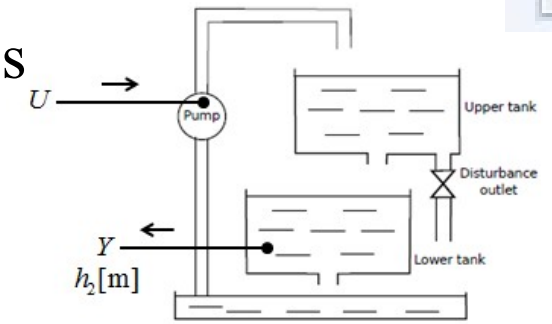


Figure 1: The controlled process

```
% Process transfer function GP has two time constants
Kp = 3   % Change process gain if needed
T1 = 6    % Change first time constant if needed
T2 = 21   % Change second time constant if needed

s = tf('s');
GP = Kp / (1 + T1*s) / (1 + T2*s)
[GpNum,GpDen]=tfdata(GP,'v')
```

These values must be changed to actual process values at the lab

Matlab Script for Lab 3

It is common practice to insert results from the command window as comments in the script!

```
s = tf('s');
GP = Kp / (1 + T1*s) / (1 + T2*s)

% Transfer function:
%          3
% -------------------
% 126 s^2 + 27 s + 1
```

Matlab Script for Lab 3

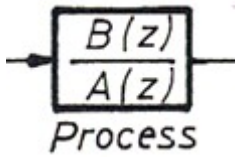
For pole placement calculations, we need the process' discrete-time model. Matlab can print the model in positive (z^n) *or* negative (z^{-n}) form.

```
% Discretize the process
disp('Now discretizing')
h = 0.6 % Change if needed

HP=c2d(GP,h);
fprintf('\nHP Negative notation')
HP.variable = 'z^-1'
fprintf('\nHP Positive notation')
HP.variable = 'z'
```

```
% Default discrete transfer function:
% HP Negative notation
% Transfer function:
% 0.003422 z^-1 + 0.003279 z^-2
% ----------------------------
% 1 - 1.877 z^-1 + 0.8794 z^-2
% Sampling time (seconds): 0.6
%
% HP Positive notation
% Transfer function:
% 0.003422 z + 0.003279
% ---------------------
% z^2 - 1.877 z + 0.8794
% Sampling time (seconds): 0.6
```

Matlab Script for Lab 3

We need the process' transfer function in negative form, split into denominator (A) and numerator (B).




```
% HP Negative notation
% Transfer function:
% 0.003422 z^-1 + 0.003279 z^-2
% ----------------------------
% 1 - 1.877 z^-1 + 0.8794 z^-2
```

```
% HP=B/A
[B,A]=tfdata(HP,'v')

% B = [ 0 0.0034 0.0033]
% A = [ 1.0000 -1.8767  0.8794]
```

Matlab Script for Lab 3

A and B are both of second degree and for now we use the following preliminary poles to create the pole placement polynomial: `z = 0.7 ± 0.2i` and `z = 0`.

$$P(z) = A(z)C(z) + B(z)D(z)$$

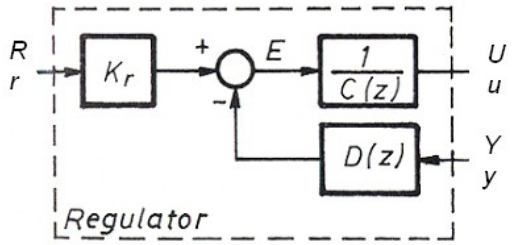
```
% 3 poles suggestion:
p1=0.7+0.2i;  % Change if needed
p2=0.7-0.2i;  % Change if needed
p3=0;

% Prompt user for pole values
% p1 = input('First pole (a+bi): ');
% p2 = input('Second pole (a-bi): ');
% display('Third pole at origo 0+0i');

% Poleplacement polynomial
P = poly([p1,p2,p3])

% P = [1.0000 -1.4000 0.5300 0]
```

When trying many different pole values it might be convenient to have the script prompt the user!

Matlab Script for Lab 3

Controller polynomials C and D:

```
% Controller polynomials C and D
% C = 1 + c_1*z^-1
% D = d_0 + d_1*z^-1
```

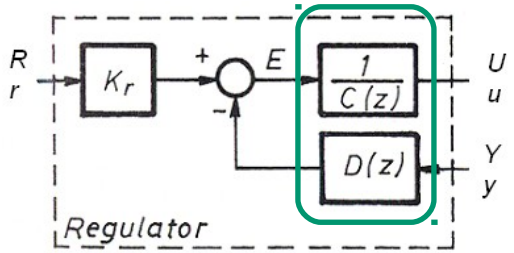


$$A(z)C(z)+B(z)D(z)=P(z)$$

Parameters **c_1**, **d_0** and **d_1** are calculated by identification of the controller polynomial's equivalent terms. This algebra can not be performed with Matlab (if the Maple module is missing) but may be done with "paper and pencil" or with Mathematica. When the algebra is made, Matlab solves the equation system numerically. ***You must present the algebraic calculations at the lab!***

Matlab Script for Lab 3

Describe your algebraic calculations leading to this equation system at the lab!

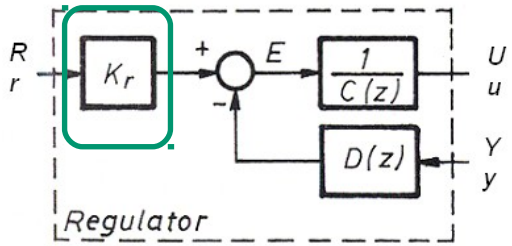



```
left = [1,   B(2) 0
        A(2) B(3) B(2)
        A(3) 0    B(3)];
right = [P(2)-A(2)
         P(3)-A(3)
         P(4)];
solution = left \ right;
c_1 = solution(1)
d_0 = solution(2)
d_1 = solution(3)

%  c_1 = 0.2148
%  d_0 = 76.5198
%  d_1 = -57.6060
```

Now we have the parameters **c_1, d_0** and **d_1!**

Matlab Script for Lab 3

$$K_r = \frac{P(1)}{B(1)}$$




```
% Kr=P(1)/B(1)

Kr = polyval(P,1) / polyval(B,1)

% Kr=19.3997
```

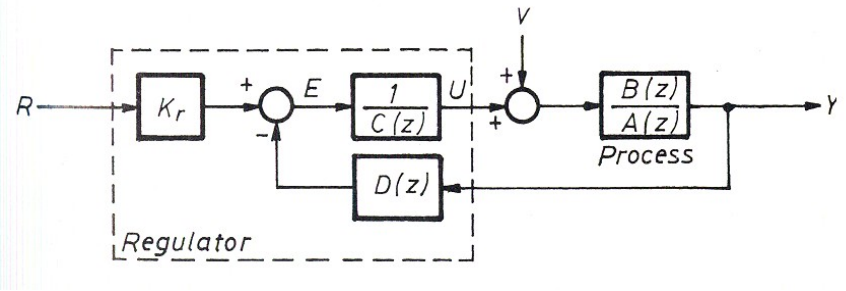
Now we have **all** the parameters!

Matlab Script for Lab 3

Print transfer function for feedback loop

$$H_{tot} = \frac{K_r \times B(z)}{A(z)C(z) + B(z)D(z)}$$

```
% Total transfer function
%           Kr*B
% Htot = ---------
%        A*C + B*D
% Transfer function of controlled system:

fprintf('\nTotal system, positive notation')
Htot = tf( Kr*B, conv(A,C) + conv(B,D),h );
Htot.variable = 'z'


% Transfer function:
%    0.06639 z + 0.06361
% -----------------------
%  z^3 - 1.4 z^2 + 0.53 z
```

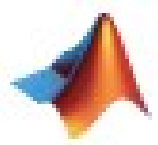


Matlab Script for Lab 3

The following part of the script plots system information. Remove if you do not want the plots to pop up each time the script is executed.

```
% Print system info.

% Process' step response.
figure('name','Step Response of Process','numbertitle','off')
step(GP)

% Step response of feedback loop.
opt = stepDataOptions('StepAmplitude', 1.5, 'InputOffset', 1.5);
figure('name','Step Response of Feedback Loop','numbertitle','off')
step(Htot, opt)

% Gain and phase margins of feedback loop
figure('name','Margins of Feedback Loop','numbertitle','off')
margin(Htot)

% Step response data of feedback loop.
[Y,T]=step(Htot, opt)
stepinfo(Y,T,'SettlingTimeThreshold', 0.05)
```

Matlab Script for Lab 3



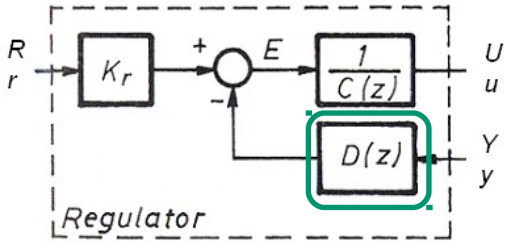
The `pzmap` function is used to check that poles are correctly placed.

```
[P,Z] = pzmap(Htot)
```

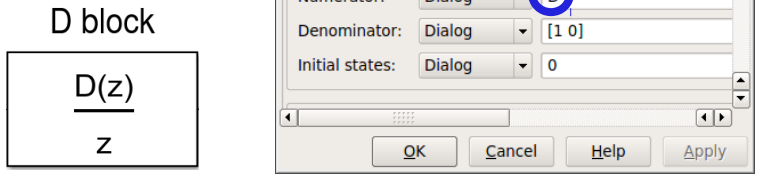
```
% P =  0
%      0.7000+ 0.2000i
%      0.7000- 0.2000i
% Z = -0.9580
```

Poles are as desired, the script is correct!

Export Transfer Functions to Simulink




```
% Controller blocks
D = [d_0  d_1 ]
D_block = tf( D,[1 0],h);
fprintf('\nBlock D positive notation')
D_block.variable = 'z'

 % D =   76.5198   -57.6060
 %
 % Block D positive notation
 % Transfer function:
 % 76.52 z - 57.61
 % ---------------
 %         z
 % Sampling time (seconds): 0.6
```

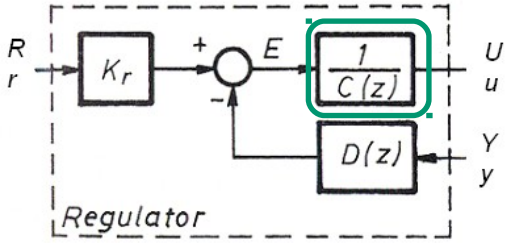


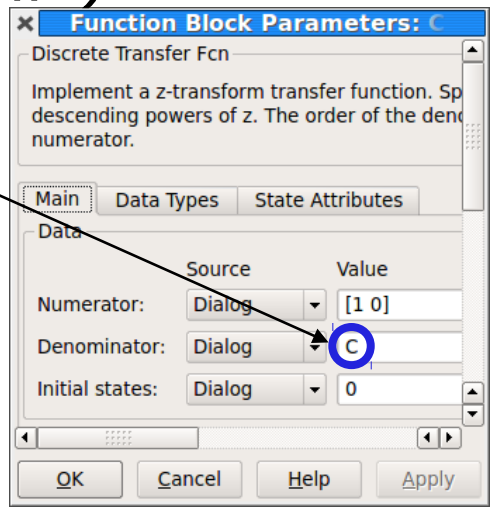
D polynomial is imported in Simulink model.

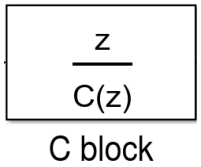
Export Transfer Functions to Simulink (Cont'd)

```
C = [1 c_1 ]
C_block = tf([1 0], C, h);
fprintf('\nBlock C positive notation')
C_block.variable = 'z'
```

```
% C = [1.0000    0.2148]
% Block D positive notation
% Transfer function:
%        z
% -------------
%   z + 0.2148
% Sampling time (seconds): 0.6
```





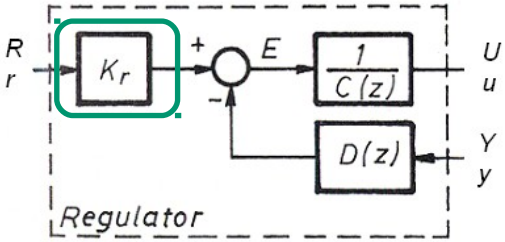


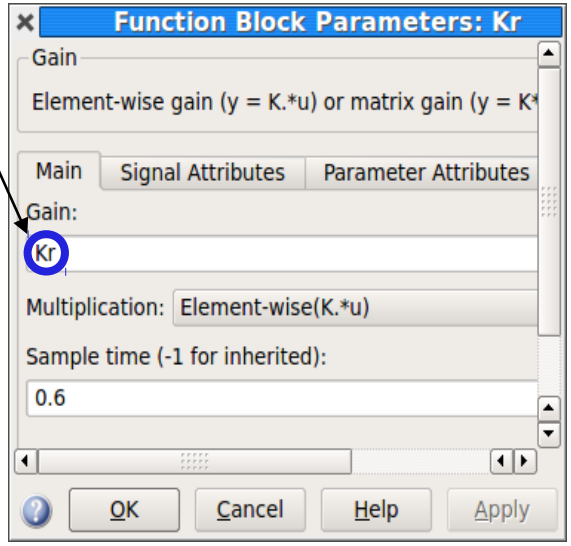
C block

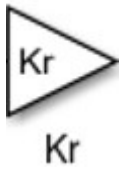
C polynomial is imported in Simulink model.

Export Transfer Functions to Simulink (Cont'd)

```
% Kr = P(1)/B(1)
Kr = polyval(P,1) / polyval(B,1)
% Kr = 19.3997
```







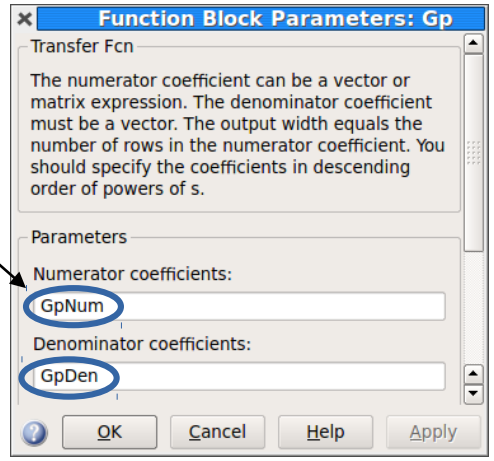
`Kr` value is imported in Simulink model.

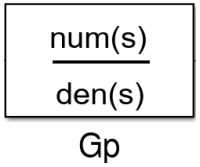
Export Transfer Functions to Simulink (Cont'd)

Simulink can simulate continuous-time and discrete-time systems together. For the process it is convenient to use the continuous-time model.

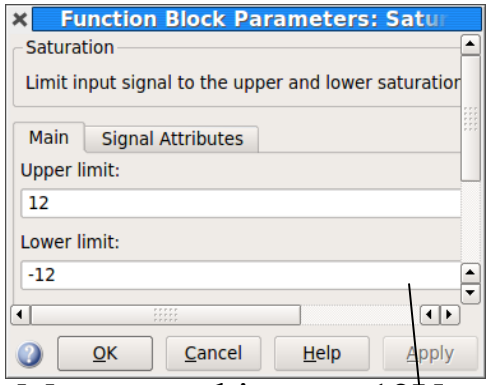
```
[GpNum,GpDen]=tfdata(GP,'v')
```

GpNum and **GpDen** polynomials are imported in Simulink model.

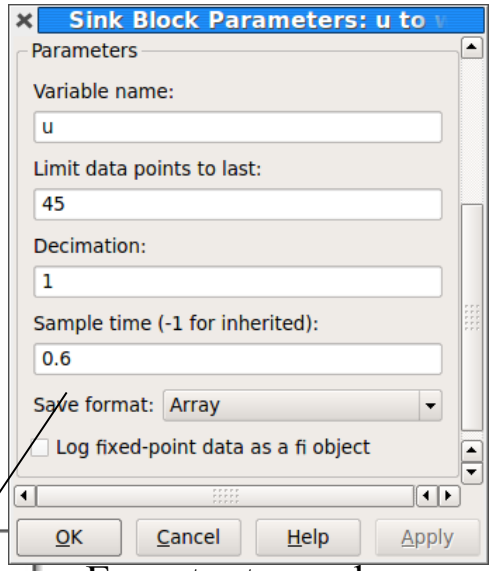




Simulink Model

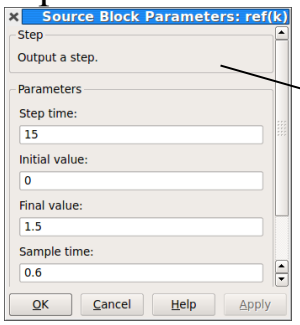


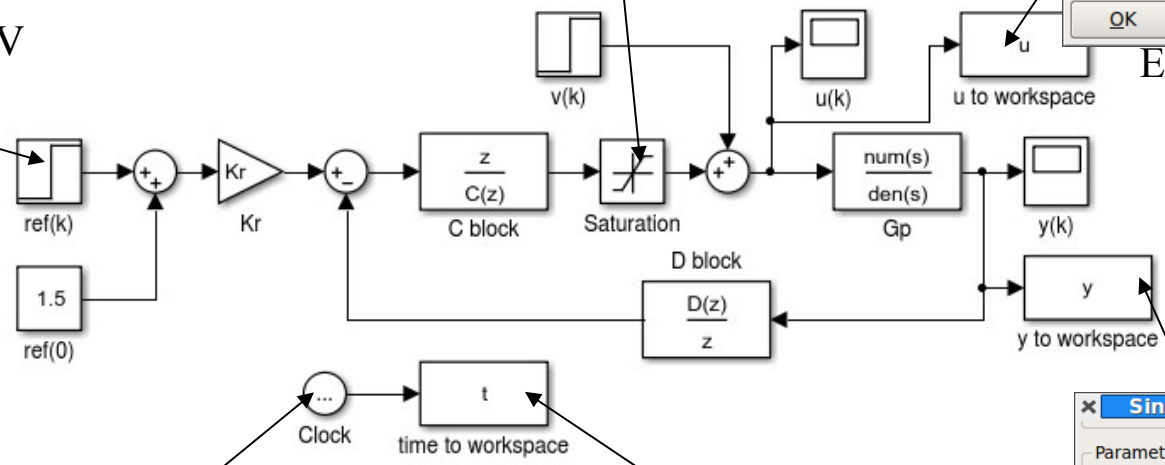
Max control input ± 12V



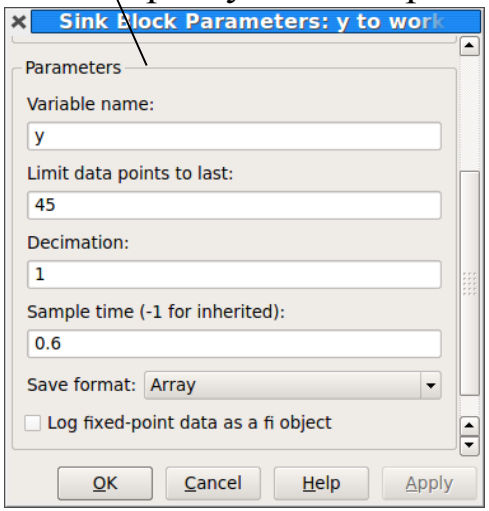
Export u to workspace

Step from 1.5 to 3 V

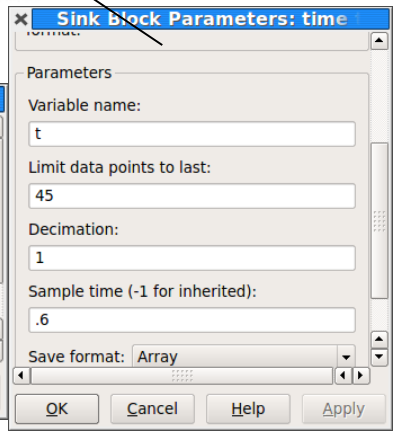


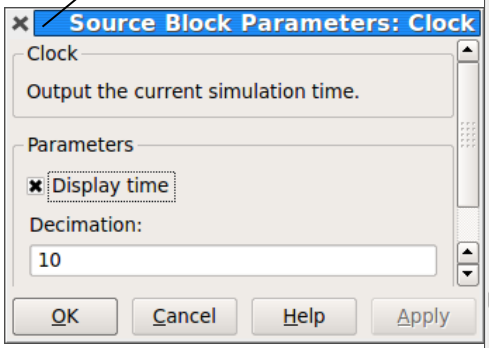


Export y to workspace



Time for workspace plots





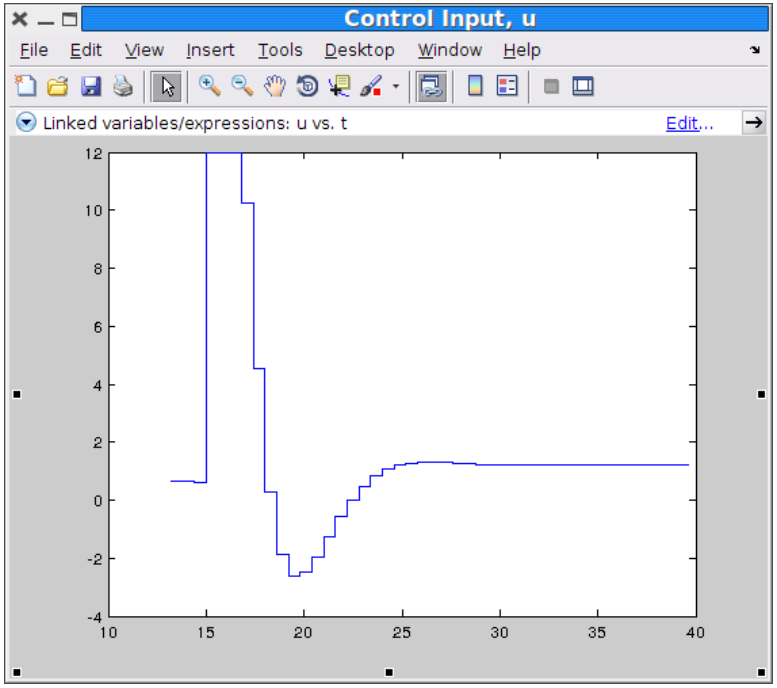
Plot Simulation Result in Matlab

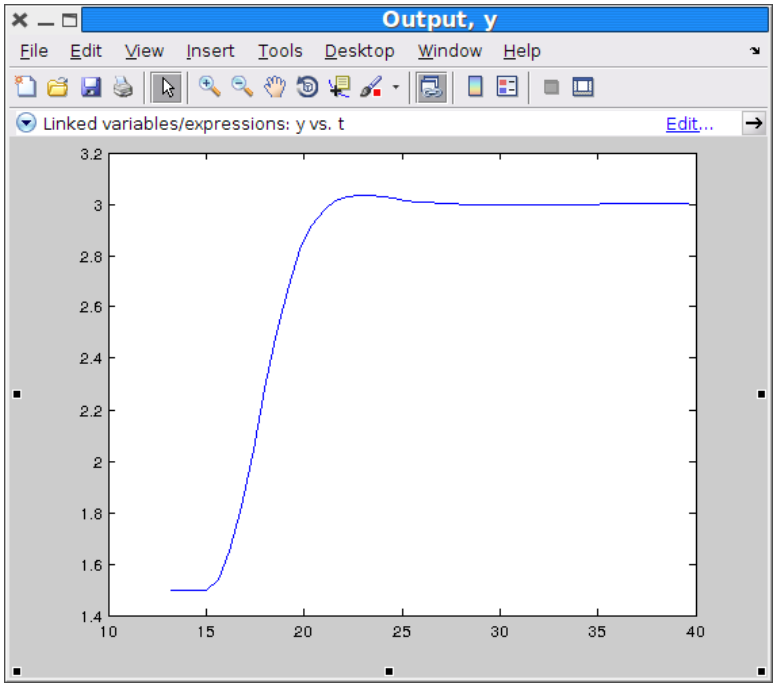
The following Matlab script creates one figure for control input and another for process output. The plots are updated when a new simulation is made.

```
% stepplot.m
%
% Plots output (y) and control input (u).
%
figure('name','Output, y','numbertitle','off')
plot(t,y)
linkdata on
figure('name','Control Input, u','numbertitle','off')
stairs(t,u)
linkdata on
```

Simulation Plots

Simulation result when reference value is changed from 1.5V to 3V after 15s. Process output $y(k)$ is continous, while control input $u(k)$ is discrete.





Matlab Script for C code (Not Part of the Course)

Matlab can write to a file! For example, the pole placement controller can be printed as a C function, which can be included when compiling a C program.

```
fid=fopen('ppc.c','w');
fprintf(fid,'float PolePlaceController(float r, float y)\n');
fprintf(fid,'{\n');
fprintf(fid,'   static float pre_u = 0;\n');
fprintf(fid,'   static float pre_y = 0;\n');
fprintf(fid,'   float e;\n');
fprintf(fid,'   float u;\n\n');
fprintf(fid,'   e = %6.3f*r - %6.3f*y - %6.3f*pre_y;\n',Kr,d_0,d_1);
fprintf(fid,'   u = e -  %6.3f*pre_u;\n\n',c_1);
fprintf(fid,'   pre_u = u;\n');
fprintf(fid,'   pre_y = y;\n\n');
fprintf(fid,'   return u;\n}\n');
fclose(fid);
```

C Function (Not Part of the Course)