# Element Control Theory - Working Task Variant 1

Author: Muhammad Abiyyu Mufti Hanif

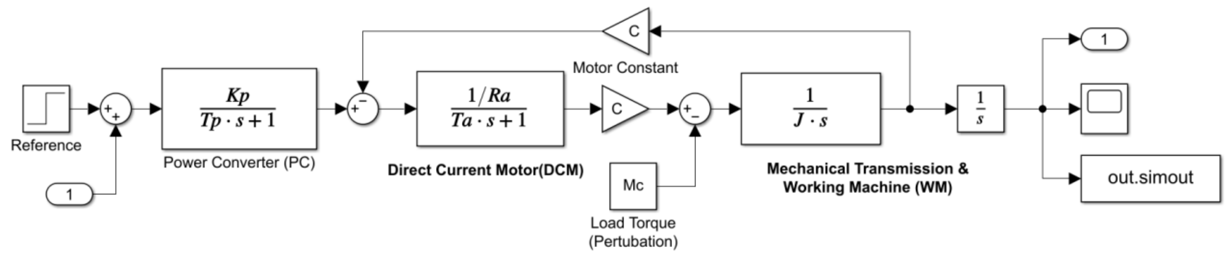
Email: mamuftihanif@gmail.com

This document is created using MATLAB live script.

## Contents

## Task Descriptions

Synthesize the system of angular position control with specific parameter for mechatronic object (Variant1.slx) consisting of Power Converter (PC), Direct Current Motor (DCM), rigid Mechanical Transmission and Working Machine (WM). There is several sensors for voltage, current, velocity and angular position measurement in the system.



## Object Parameters

```
close all; clear; clc;
J = 0.25;    % kg/m^3     : Total inertia moment
Mc = 5;      % Nm         : Maximal load torque
Un = 110;    % V          : Nominal voltage of Motor
Ia = 8;      % A          : Nominal current of motor anchor circuit
Ra = 3.15;   % Ohm        : Resistance of motor anchor circuit
Ta = 0.11;   % s          : Time constant
C = 0.16;    % Wb         : Motor constant
Uc = 10;     % V          : Maximal control voltage of power converter
Tp = 0.001;  % s          : Time constant of power converter
phi = 1;     % rad        : Angular position
delta_phi = 0.1; % Maximum allow error of angular position stabilization < 0.1%
tc = 0.1;                 % Time of step response tc < 0.1 s
tr = 1.0;                 % Time of restoring after loud torque step tr < 1.0 s
sigma = 0.5;              % Maximum overshoot for angular position
Kp = Un/Uc; %             : Gain factor of the power converter
STime = 1;
TPert = 0.5;
```

# Task - 1: Object Description in Mathematical Model

Construct mathematical model of controlled object in the form of State Space Model (matrices A, B and C) and Transfer Functions:

## Calculation of Object Transfer Function with Mason's Rules

### Transfer function of the object

Transfer function of the model with input of Voltage $U(s)$ and output angular position $\varphi(s)$. Here ignoring the perturbation $M_C$:

$$H_{\mathrm{oU}}(s) = \frac{\varphi(s)}{U(s)}$$

$$H_{\mathrm{oU}}(s) = \left(\frac{K_p}{T_p\,s+1}\right)\cdot\left(\frac{\left(\frac{1}{R_a\ (T_a\,s+1)}\right)\cdot C\cdot\left(\frac{1}{J\ s}\right)}{1+\left(\frac{1}{R_a\ (T_a\,s+1)}\right)\cdot C^2\cdot\left(\frac{1}{J\ s}\right)}\right)\cdot\left(\frac{1}{s}\right)$$

Because $T_p = 0.001s$ and it is noticeably smaller than the system response time $t_s = 0.1s$, we will ignore the value of $T_p$. So $T_p = 0$. Therefore, the transfer function is calculated:

$$H_{\mathrm{oU}}(s) = K_P\cdot\left(\frac{\left(\frac{1}{R_a\ (T_a\,s+1)}\right)\cdot C\cdot\left(\frac{1}{J\ s}\right)}{1+\left(\frac{1}{R_a\ (T_a\,s+1)}\right)\cdot C^2\cdot\left(\frac{1}{J\ s}\right)}\right)\cdot\left(\frac{1}{s}\right)$$

$$H_{\mathrm{oU}}(s) = \frac{C\ K_p}{(J\ R_a\,T_a)\ s^3+(J\ R_a)\ s^2+C^2\ s}$$

Let's make the coefficient of highest degree of s as one. The transfer function of the object $H_{\mathrm{oU}}(s)$ can be written as follow:

$$H_{\mathrm{oU}}(s) = \frac{\left(\frac{C\ K_p}{J\ R_a\,T_a}\right)}{s^3+\frac{1}{T_a}s^2+\frac{C^2}{J\ R_a T_a}\ s}$$

The transfer function $H_{\mathrm{oU}}(s)$ has numerator $B_o(s) = \left(\frac{C\ K_p}{J\ R_a\,T_a}\right)$ and denominator $A_o(s) = \left(s^3+\frac{1}{T_a}s^2+\frac{C^2}{J\ R_a T_a}\ s+0\right)$.

Numerically the numerator and the denominator are as follow:

```
num = (C*Kp)/(J*Ra*Ta)
```

```
num = 20.3175
```

```
denum = [1, 1/Ta, C^2/(J*Ra*Ta), 0]
```

```
denum = 1×4
    1.0000    9.0909    0.2955         0
```

Let's also calculate the transfer function with the help of MATLAB function *linmod.*

```
Tp = 0.0;    % s          : Time constant of power converter will be ignored
[num_, denum_] = linmod('Variant1')
```

```
num_ = 1×4
         0         0         0   20.3175
denum_ = 1×4
    1.0000    9.0909    0.2955         0
```

We can see that both numerator and denominator of the transfer function calculated analitically and calculated using linmod function are the same, therefore the analytical calculation is correct.

From the numerator and denominator, here is the transfer function of the object in numeric form:

```
Ho = tf(num, denum)
```

```
Ho =

            20.32
  --------------------------
  s^3 + 9.091 s^2 + 0.2955 s

Continuous-time transfer function.
```

$$H_{\mathrm{oU}}(s) = \frac{20.3175}{s^3 + 9.0909 s^2 + 0.2955\, s}$$

**Perturbation Transfer Function**

On the other hand, if we take the perturbation Load Torque $M_C(s)$ in regard to the output angular position $\varphi(s)$ we can also calculate the transfer function $H_{\mathrm{oM}}(s)$. In this case, we need to ignore the input $U(s)$.

$$H_{\mathrm{oM}}(s) = \frac{\varphi(s)}{M_C(s)}$$

$$H_{\mathrm{oM}}(s) = \left( \frac{\left(\frac{1}{J \cdot s}\right)}{1 + \left(\frac{1}{R_a\ (T_a\, s + 1)}\right) * C^2 * \left(\frac{1}{J\ s}\right)} \right) \cdot \left(\frac{1}{s}\right)$$

$$H_{\mathrm{oM}}(s) = \left( \frac{\left(\frac{1}{J \cdot s}\right)}{1 + \left(\frac{1}{R_a\ (T_a\, s + 1)}\right) * C^2 * \left(\frac{1}{J\ s}\right)} \right) \cdot \left(\frac{1}{s}\right)$$

$$H_{\mathrm{oM}}(s) = \frac{1}{J\ s^2\ \left(\frac{C^2}{J\ R_a\, s\ (T_a\, s + 1)} + 1\right)}$$

$$H_{\mathrm{oM}}(s) = \frac{(R_a\, T_a)\ s + R_a}{(J\ R_a\, T_a)\ s^3 + (J\ R_a)\ s^2 + C^2\ s}$$

$$H_{\mathrm{oM}}(s) = \frac{\frac{1}{J}\ s + \frac{1}{J\ T_a}}{s^3 + \frac{1}{T_a} s^{\ 2} + \frac{C^2}{(J\ R_a\, T_a)}\ s}$$

From this calculation using Mason's rules the transfer function $H_{\mathrm{oM}}(s)$ has numerator $B_M(s) = \left(\frac{1}{J}\ s + \frac{1}{J\ T_a}\right)$ and denominator $A_M(s) = \left(s^3 + \frac{1}{T_a} s^2 + \frac{C^2}{J\ R_a T_a}\ s + 0\right)$.

```
num_pert = [1/J, 1/(J*Ta)]
```

```
num_pert = 1×2
    4.0000   36.3636
```

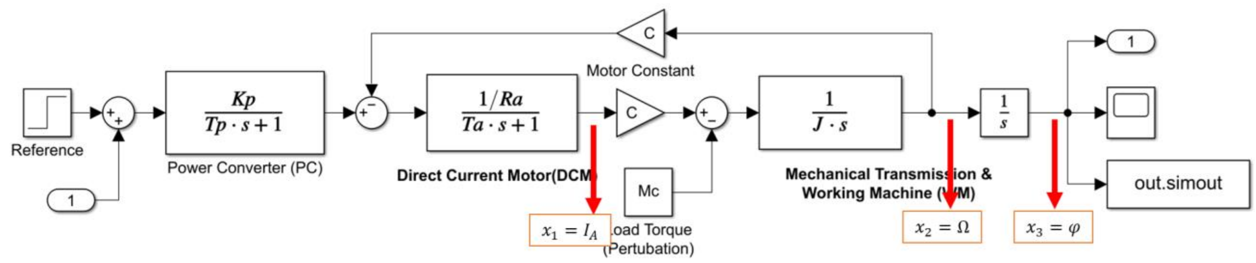
```
denum_pert = denum
```

```
denum_pert = 1×4
    1.0000    9.0909    0.2955         0
```

$$H_{\mathrm{oM}}(s) = \frac{4\, s + 36.3636}{s^3 + 9.0909 s^2 + 0.2955\, s}$$

## Constructing State Space Model

Let's construct the state space model of the object by choosing the state coordinates from the object.


Reference
1
Power Converter (PC)
$\frac{Kp}{Tp \cdot s+1}$
C
Motor Constant
$\frac{1/Ra}{Ta \cdot s+1}$
Direct Current Motor(DCM)
C
Mc
Load Torque (Pertubation)
$\frac{1}{J \cdot s}$
Mechanical Transmission & Working Machine (WM)
$\frac{1}{s}$
1
out.simout
$x_1 = I_A$
$x_2 = \Omega$
$x_3 = \varphi$

From the image we can see that we have 3 state coordinates $x(s) = [I_A,\ \Omega, \varphi]$, one output $y(s) = [\varphi]$ and one input $u(s) \ = U_C$.

We will calculate the state coordinates equations:

For $x_1(s) = I_A$

$$x_1(s) = I_A = \frac{1}{R_a\ (T_a s + 1)} \cdot [K_P U_C(s) - C x_2(s)]$$

$$x_1 T_a s + x_1 = \frac{K_P}{R_a} \cdot U_C - \frac{C}{R_a} \cdot x_2$$

$$s \cdot x_1 = -\frac{1}{T_a} \cdot x_1 - \frac{C}{R_a T_a} \cdot x_2 + \frac{K_P}{R_a T_a} \cdot U_C \text{ ..... (1)}$$

For $x_2\,(s) = \Omega$

$$x_2\,(s) = \Omega = \frac{C}{J\ s} \cdot x_1(s)$$

$$s \cdot x_2 = \frac{C}{J} \cdot x_1 \text{ ..... (2)}$$

For $x_3(s) = \ \varphi$

$$x_3\,(s) = \varphi\ = \frac{1}{s} \cdot x_2(s)$$

$$s \cdot x_3 = x_2 \text{ ..... (3)}$$

Let's put the equation (1), (2), and (3) together, that are depends on state space coordinates $x_1, x_2, x_3$ and input $U$ :

$$s \cdot x_1 = -\frac{1}{T_a} * x_1 - \frac{C}{R_a T_a} * x_2 \ + 0 \cdot x_3 + \ \frac{K_P}{R_a T_a} * U_C$$

$$s \cdot x_2 = \frac{C}{J} * x_1 \ + 0 \cdot x_2 \ + 0 \cdot x_3$$

$$s \cdot x_3 = 0 \cdot x_1 + \ x_2 + 0 \cdot x_3$$

From these 3 linear equations, we can put them in to matrix form

$$s \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \frac{K_P}{R_a T_a} \\ 0 \\ 0 \end{bmatrix} \cdot U_C$$

Putting them as the state space equation $s \cdot x(s) = A \cdot x(s) + B \cdot u(s)$, we got the matrix A and B.

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} ; B = \begin{bmatrix} \frac{K_P}{R_a T_a} \\ 0 \\ 0 \end{bmatrix}$$

Next, let's consider the equation of the output $y(s)$ depending on the state space coordinate $x_1, x_2, x_3$.

$$y(s) = 0 \cdot x_1 \ + 0 \cdot x_2 + x_3$$

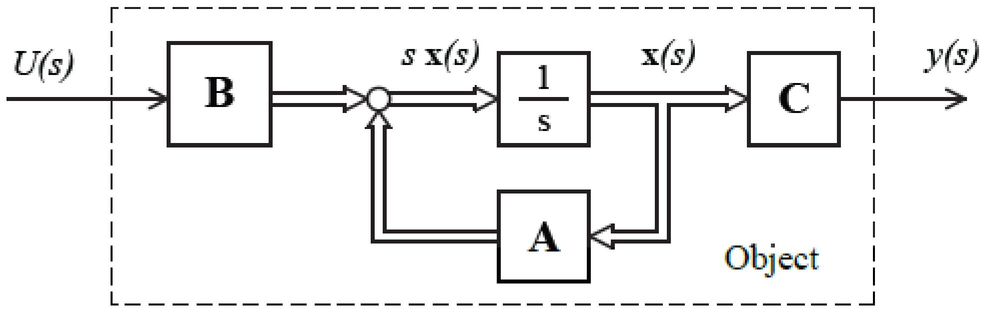
Then, in matrix form:

$$y(s) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Putting that as the state space equation $y(s) = C \cdot x(s)$, we got the matrix C.

$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

With the matrix A, B, and C, we are able to create state space model that structures as described in the following figure.



Let's compute the state space matrices from the analytical formula.

```
A_ = [-1/Ta -C/(Ra*Ta) 0; C/J 0 0; 0 1 0]
```

```
A_ = 3×3
   -9.0909   -0.4618         0
    0.6400         0         0
         0    1.0000         0
```

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -9.0909 & -0.4618 & 0 \\ 0.6400 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

```
B_ = [Kp/(Ra*Ta); 0; 0]
```

```
B_ = 3×1
   31.7460
         0
         0
```

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -9.0909 & -0.4618 & 0 \\ 0.6400 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

```
C_ = [0 0 1]
```

```
C_ = 1×3
     0     0     1
```

$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

From this state space model, we can also compute the transfer function of the object.

$$H_{\mathrm{oU}}(s) = C \cdot \frac{\operatorname{adj}(s \cdot \mathrm{I} - A)}{\det(s \cdot \mathrm{I} - A)} \,.\, B$$

$$(s \cdot \mathrm{I} - A) = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix} - \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{pmatrix} s + \frac{1}{T_a} & \frac{C}{R_a T_a} & 0 \\ -\frac{C}{J} & s & 0 \\ 0 & -1 & s \end{pmatrix}$$

Calculating determinant of $(s \cdot \mathrm{I} - A)$ using the last column as basis.

$$\det(s \cdot \mathrm{I} - A) = 0 \cdot [\quad] - 0 \cdot [\quad] + s \cdot \left[ \left( s + \frac{1}{T_a} \right) \cdot s + \frac{C^2}{R_a \; T_a \, J} \right]$$

$$\det(s \cdot \mathrm{I} - A) = s^3 + \frac{1}{T_a} \cdot s^2 + \frac{C^2}{R_a T_a \, J} \cdot s$$

Arrange the adjoint matrix of $(s \cdot \mathrm{I} - A)$

$$\operatorname{adj}(s \cdot \mathrm{I} - A) = \begin{pmatrix} s^2 & -\frac{C\, s}{R_a\, T_a} & 0 \\ \frac{C\, s}{J} & \frac{s}{T_a} + s^2 & 0 \\ \frac{C}{J} & s + \frac{1}{T_a} & \frac{s}{T_a} + s^2 + \frac{C^2}{J\, R_a\, T_a} \end{pmatrix}$$

And lastly, we put them together to the equation, we got the transfer function.

$$H_{\mathrm{oU}}(s) = C \cdot \frac{\operatorname{adj}(s \cdot \mathrm{I} - A)}{\det(s \cdot \mathrm{I} - A)} \,.\, B = \frac{C\, K_p}{(J\, R_a\, T_a)\, s^3 + (J\, R_a)\, s^2 + C^2\, s}$$

$$H_{\mathrm{oU}}(s) = \frac{\left( \frac{C\, K_p}{J\, R_a\, T_a} \right)}{s^3 + \frac{1}{T_a} s^2 + \frac{C^2}{J\, R_a T_a}\, s}$$

which is the exact formula of the transfer function that we calculated previously using Masson's rules.

Let's create the state space model using the matrices of A, B, and C using MATLAB function *ss* and then from that we extract the transfer function of the object using function *tfdata.*

```
Hss = ss(A_, B_, C_, 0);
[num_s, denum_s] = tfdata(Hss, 'v');
Ho_s = tf(num_s, denum_s)
```

```
Ho_s =

            20.32
  --------------------------
  s^3 + 9.091 s^2 + 0.2955 s

Continuous-time transfer function.
```

As we can see, the result is also the same as the transfer function that extracted using linmode or calculated analytically using Masson's rules.

Therefore, we are successfully constructing the mathematical model of controlled object in form of state space model and also extracting the transfer function using mason's rules and also deriving it from the state space model.

Here is the Matrices and the transfer function once again in symbolic form and numerical form:

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -9.0909 & -0.4618 & 0 \\ 0.6400 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix};\ B = \begin{bmatrix} \frac{K_P}{R_a T_a} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 31.7460 \\ 0 \\ 0 \end{bmatrix};\ C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$H_{\mathrm{oU}}(s) = \frac{\left(\frac{C\,K_p}{J\,R_a\,T_a}\right)}{s^3 + \frac{1}{T_a}s^2 + \frac{C^2}{J\,R_a T_a}\ s} = \frac{20.3175}{s^3 + 9.0909 s^2 + 0.2955\ s}$$

## Task - 2: State and Dynamic Characteristics of the Object

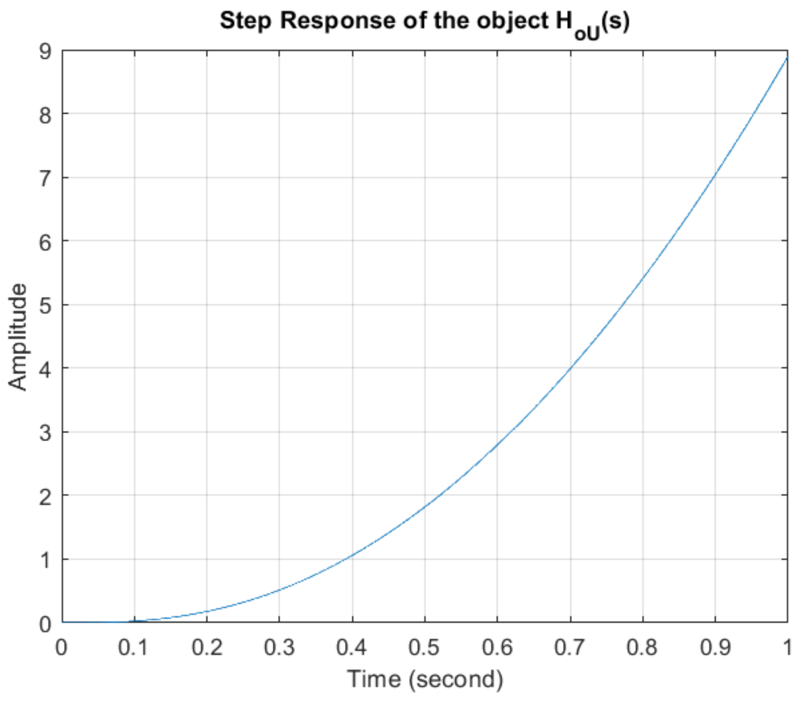
Calculate the state and dynamic characteristics of the object (step response, bode diagram, pole-zero diagrams).

Make analysis of Controllability and Observability for State Space Model of controlled object

### Dynamic Characteristic of the Object

Using the matlab function *step, stepinfo*, *bode* and *pzmap*, we are able to get calculate the state and dynamic characteristic of the object.

```
var1 = sim('Variant1');
plot(var1.tout, var1.simout); grid on;
title("Step Response of the object H_o_U(s)")
xlabel('Time (second)')
ylabel('Amplitude')
grid on;
```



```
disp("Step Information using Cursor")
```

```
Step Information using Cursor
```

```
stepinfo(var1.simout,var1.tout)
```

```
ans = struct with fields:
        RiseTime: 0.5807
    SettlingTime: 0.9909
     SettlingMin: 8.0231
     SettlingMax: 8.9035
```

```
        Overshoot: 0
       Undershoot: 0
             Peak: 8.9035
         PeakTime: 1
```

```
disp("Step Information using Transfer Function")
```

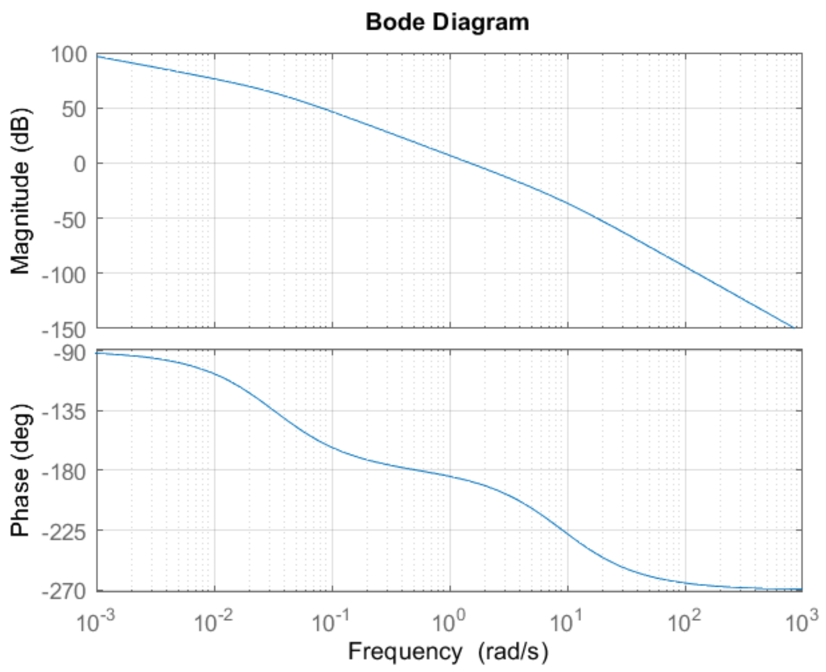
```
Step Information using Transfer Function
```

```
stepinfo(Ho)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

From the step response we can already recognise that the object doesn't have the static characteristic, and it increases its amplitude over time. If we see the object structure, we can also notice that it ended by an integrator, that's why this behaviour happened.

```
bode(Ho); grid on;
```



```
roots(num)
```

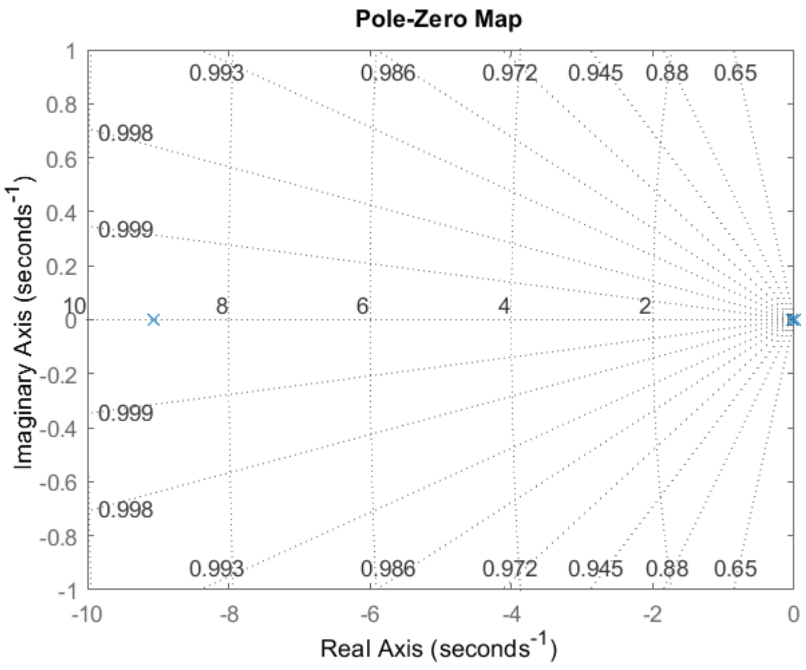
```
ans =

  0×1 empty double column vector
```

```
roots(denum)
```

```
ans = 3×1
          0
    -9.0583
    -0.0326
```

```
pzmap(Ho); grid on;
```



And from the transfer function representation, we notice that the object doesn't have any zeros and has 3 poles.

## Controllability and Observability

To check if an object in a state space model is controllable, we need first to calculate the controllability matrix U. If the determinant of the controllability matrix is not zero ($\det(U) \neq 0$), then an object is controllable.

The controllable matrix for this object is define as follow:

$$U = [B \quad A \cdot B \quad A^2 \cdot B]$$

$$U = \begin{pmatrix} \frac{K_p}{R_a T_a} & -\frac{K_p}{R_a T_a^2} & \frac{K_p \left(\frac{1}{T_a^2} - \frac{C^2}{J R_a T_a}\right)}{R_a T_a} \\ 0 & \frac{C K_p}{J R_a T_a} & -\frac{C K_p}{J R_a T_a^2} \\ 0 & 0 & \frac{C K_p}{J R_a T_a} \end{pmatrix} = \begin{bmatrix} 31.7460 & -288.6003 & 2.6143e+03 \\ 0 & 20.3175 & -184.7042 \\ 0 & 0 & 20.3175 \end{bmatrix}$$

and its determinant is $\det(U) = \frac{C^2 K_p^3}{J^2 R_a^3 T_a^3} \neq 0$. Therefore, the object is controllable.

Let's calculate this controllability matrix in MATLAB.

```
U = [B_ A_*B_ A_^2*B_]
```

```
U = 3×3
10³ ×
    0.0317   -0.2886    2.6143
         0    0.0203   -0.1847
         0         0    0.0203
```

```
if  det(U) ~= 0
    fprintf("det(U) = %f; Object is controllable", det(U));
else
    fprintf("det(U) = %f; Object is not controllable", det(U));
end
```

```
det(U) = 13104.736310; Object is controllable
```

To check if an object in a state space model is observable, we need first to calculate the observability matrix V. If the determinant of the observability matrix is not zero ($\det(V) \neq 0$), then an object is observable.

The observability matrix for this object is define as follow:

$$V = \left[C^T,\ A^T \cdot C^T, \left(A^T\right)^2 \cdot C^T\right]$$

$$V = \begin{pmatrix} 0 & 0 & \frac{C}{J} \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0.6400 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

and its determinant is $\det(V) = -\frac{C}{J} \neq 0$. Therefore, the object is observable.

Let's also calculate this observability matrix in MATLAB.

```
V = [C_.' (A_.')*C_' (A_.')^2*C_.']
```

```
V = 3×3
         0         0    0.6400
         0    1.0000         0
    1.0000         0         0
```

```
if det(V) ~= 0
    fprintf("det(U) = %f; Object is observable", det(V));
else
    fprintf("det(U) = %f; Object is not observable", det(V));
end
```

```
det(U) = -0.640000; Object is observable
```

## Task - 3: Synthesis of Polynomial Controller

Make the analytical synthesis of input-output polynomial controller with the modal control method for Newton desired polynomial.

Calculate the static and dynamic characteristics of the closed loop system with polynomial controller.

### Analytical Synthesis of Polynomial Controller

To synthesize the polynomial controller the algorithm can be followed:

**1. Determine transfer function and polynomials $A(s)$ and $B(s)$ of controlled object.**

From the previous calculation, we know already the transfer function of object $H_{\mathrm{oU}}(s)$.

$$H_{\mathrm{oU}}(s) = \frac{\left(\frac{C\,K_p}{J\,R_a\,T_a}\right)}{s^3 + \frac{1}{T_a}s^2 + \frac{C^2}{J\,R_a T_a}\,s} = \frac{20.3175}{s^3 + 9.0909s^2 + 0.2955\,s}$$

This transfer function $H_{\mathrm{oU}}(s)$ has numerator $B(s) = \left(\frac{C\,K_p}{J\,R_a\,T_a}\right) = 20.3175$ and denominator $A(s) = \left(s^3 + \frac{1}{T_a}s^2 + \frac{C^2}{J\,R_a T_a}\,s + 0\right) = s^3 + 9.0909s^2 + 0.2955\,s$.

The degree of $A(s)$ is 3, while the degree of $B(s)$ is zero.

Let's extract the coefficient of numerator and denominator of the transfer function in MATLAB.

```
b0 = num(end);
a0 = denum(end);
a1 = denum(end-1);
a2 = denum(end-2);
```

**2. Determine the degrees of controller polynomials $R(s)$ and $C(s)$ and degree of desired polynomial of closed loop system.**

$$\deg(R(s)) = \deg(A(s)) - 1 = 3 - 1 = 2;$$
$$\deg(C(s)) \geq \deg(R(s)) = 2;$$
$$\deg(D(s)) = \deg(A(s)) + \deg(C(s)) = 3 + 2 = 5;$$

After the degrees of the polynomials are determined, we can define the polynomials $R(s)$ and $C(s)$ are defined as follow:

$$R(s) = r_2 \cdot s^2 + r_1 \cdot s + r_0$$

$$C(s) = s^2 + c_1 \cdot s + c_0$$

**3. Set the desired polynomial of the closed loop system using Newton polynomial**

Desired polynomial $D(s)$, that has the degree of 5, can be defined using Newton polynomial with $\Omega_0$ as geometric mean root.

$$D(s) = (s+\Omega_0)^5 = s^5 + 5\cdot\Omega_0\cdot s^4 + 10\cdot\Omega_0^2\cdot s^3 + 10\cdot\Omega_0^3\cdot s^2 + 5\cdot\Omega_0^4\cdot s + \Omega_0^5$$

To simplify we can replace the coefficient of the polynomial $D(s)$.

$$D(s) = (s+\Omega_0)^5 = s^5 + d_4\cdot s^4 + d_3\cdot s^3 + d_2\cdot s^2 + d_1\cdot s + d_0$$

with $d_4 = 5\cdot\Omega_0;\ d_3 = 10\cdot\Omega_0^2;\ d_2 = 10\cdot\Omega_0^3;\ d_1 = 5\cdot\Omega_0^4;\ d_0 = \Omega_0^5;$

On the other hand the geometric mean root is calculated by

$$\Omega_0 = \frac{n + 2 * \sqrt{n-1}}{T_C} = 90\,\frac{\text{rad}}{s}$$

with $n = 5$ as the degree of polynomial and $T_C = 0.1s$ as time of the step response.

Let's build the desired polynomial $D(s)$ in MATLAB.

```
n = 5;
mean_root_0 = (n + 2*sqrt(n-1))/tc;

d4 = 5*mean_root_0 ;
d3 = 10*mean_root_0^2 ;
d2 = 10*mean_root_0^3 ;
d1 = 5*mean_root_0^4 ;
d0 = mean_root_0^5 ;
```

**4. Set and solve the synthesis equation**

The synthesis equation can be described as follow.

$$A(s)\cdot C(s) + B(s)\cdot R(s) = D(s)$$

To simplify the calculation of unknown variable $r_2, r_1, r_0, c_1, \text{and}\ c_0$, let also simplify the equation of polynomial $A(s)$ and $B(s)$.

$A(s) = s^3 + a_2\,s^2 + a_1\,s + a_0$; with $a_2 = 9.0909$, $a_1 = 0.2955$, and $a_0 = 0$, and

$B(s) = b_0$, with $b_0 = 20.3175$.

$$A(s)\cdot C(s) + B(s)\cdot R(s) = D(s)$$

$$s^5 + (a_2 + c_1)\,s^4 + (a_1 + c_0 + a_2\,c_1)\,s^3 + (a_0 + a_1\,c_1 + a_2\,c_0 + b_0\,r_2)\,s^2 + (a_0\,c_1 + a_1\,c_0 + b_0\,r_1)\,s$$
$$+ a_0\,c_0 + b_0\,r_0 = s^5 + d_4\,s^4 + d_3\,s^3 + d_2\,s^2 + d_1\,s + d_0$$

From these equations we can do coefficient comparison:

For $s^4$: $a_2 + c_1 = d_4$

For $s^3$: $a_1 + c_0 + a_2\, c_1 = d_3$

For $s^2$: $a_0 + a_1\, c_1 + a_2\, c_0 + b_0\, r_2 = d_2$

For $s^1$: $a_0\, c_1 + a_1\, c_0 + b_0\, r_1 = d_1$

For $s^0$: $a_0\, c_0 + b_0\, r_0 = d_0$

and finally solve the unknown variables $r_2, r_1, r_0, c_1, \text{and } c_0$.

$$r_0 \;=\; \frac{d_0 - a_0\, c_0}{b_0} = \frac{J\, R_a\, T_a\, d_0}{C\, K_p} \;=\; 2.9063e+08;$$

$$r_1 \;=\; \frac{d_1 - \; a_0\, c_1 - \; a_1\, c_0}{b_0} \;=\; \left(d_1 - \frac{C^2\, c_0}{J\, R_a\, T_a}\right) \cdot \left(\frac{J\, R_a\, T_a}{C\, K_p}\right) \;=\; 1.6146e+07;$$

$$r_2 \;=\; \frac{d_2 - a_0 - a_1\, c_1 - \; a_2\, c_0}{b_0} \;=\; \left(d_2 - \;\frac{c_0}{T_a}\; - \;\frac{C^2\, c_1}{J\, R_a\, T_a}\right) \cdot \left(\frac{J\, R_a\, T_a}{C\, K_p}\right) \;=\; 3.2435e+05;$$

$$c_0 = \; d_3 - a_1 - a_2\, c_1 \;=\; d_3 - \frac{C^2}{J\, R_a\, T_a} - \frac{c_1}{T_a} \;= 440.9091;$$

$$c_1 = \; d_4 - a_2 \;=\; d_4 - \frac{1}{T_a} \;= 7.6991e+04;$$

Let's solve the polynomial $R(s)$ and $C(s)$ in MATLAB.

```
c1 = d4 - a2;
c0 = d3 - a1 - a2*c1;
r2 = (d2 - a0 - a1*c1 - a2*c0)/b0;
r1 = (d1 - a0*c1 -a1*c1)/b0;
r0 = (d0 - a0*c1)/b0;
```

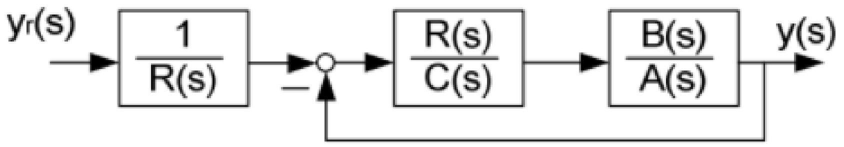
With the following matrices we will define the polynomial $R(s)$ and $C(s)$ and will later be put into the Simulink model.

```
R_poly = [r2 r1 r0];
C_poly = [1 c1 c0];
```

## Running the Simulation of Polynomial Controller on MATLAB

With that calculation, we are able to synthesize and model the polynomial controller in Simulink.

The polynomial controller can be modelled by following this figure. Note that $\frac{R(s)}{C(s)}$ is our controller and $\frac{1}{R(s)}$ is the prefilter.

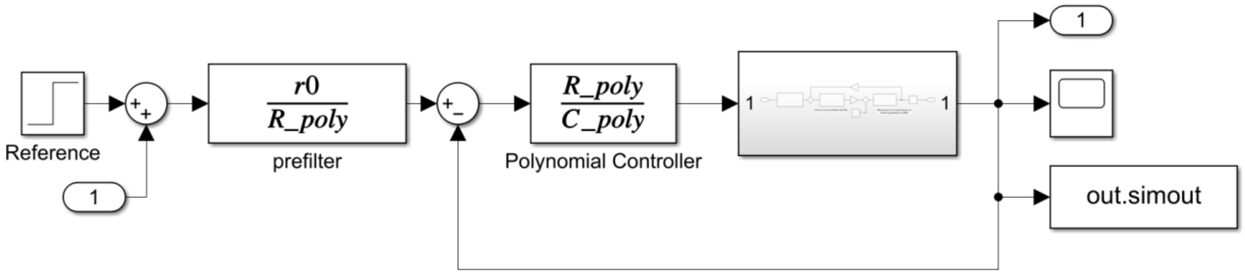


To eliminate the static error that the model will deliver, we need to add the static constant at the numerator of the prefilter.

This static constant can be calculated by compensating the transmission coefficient. Transmission coefficient is value of a transfer function when the s-Laplace operator is equal to zero. After adding the controller, our model will have a desired transfer function of $\frac{B(s)}{D(s)}$. Therefore, the transmission coefficient of our model is $\frac{b_0}{d_0}$, and to compensate that, our static constants is defined as follow:

$$K_{\text{KSP}} = \frac{d_0}{b_0} = r_0$$

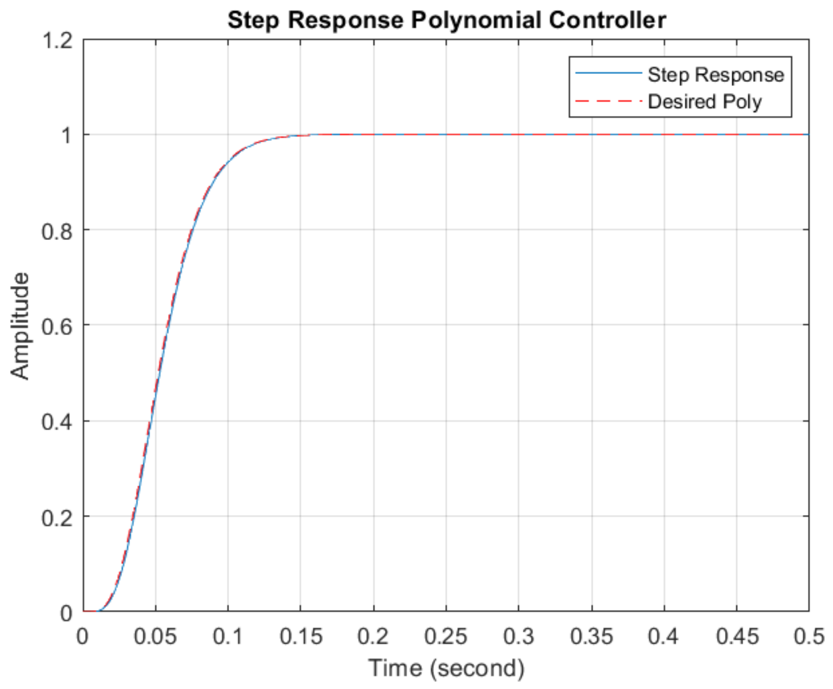
The Simulink model Polynomial.slx is created to model the polynomial controller and its structure diagram is shown in the following figure.



Let's run the simulation without any perturbation to see the step response and this signal will then be compared to the desired polynomial $D(s)$.

```
STime = 0.5;
des_poly = step(tf(d0,[1 d4 d3 d2 d1 d0]),0:1e-3:STime);
out_poly = sim('Polynomial');
plot(out_poly.tout, out_poly.simout); hold on;
plot(out_poly.tout, des_poly, 'r--'); hold off; grid on;
title('Step Response Polynomial Controller');
```

```
xlabel('Time (second)');
ylabel('Amplitude');
ylim([0 1.2]);
legend('Step Response','Desired Poly')
```



Let's also calculate the static and dynamic characteristic of this object with controller using stepinfo function.

```
[num_P, denum_P] = linmod('Polynomial');
Hp = tf(num_P, denum_P);
stepinfo(Hp)
```

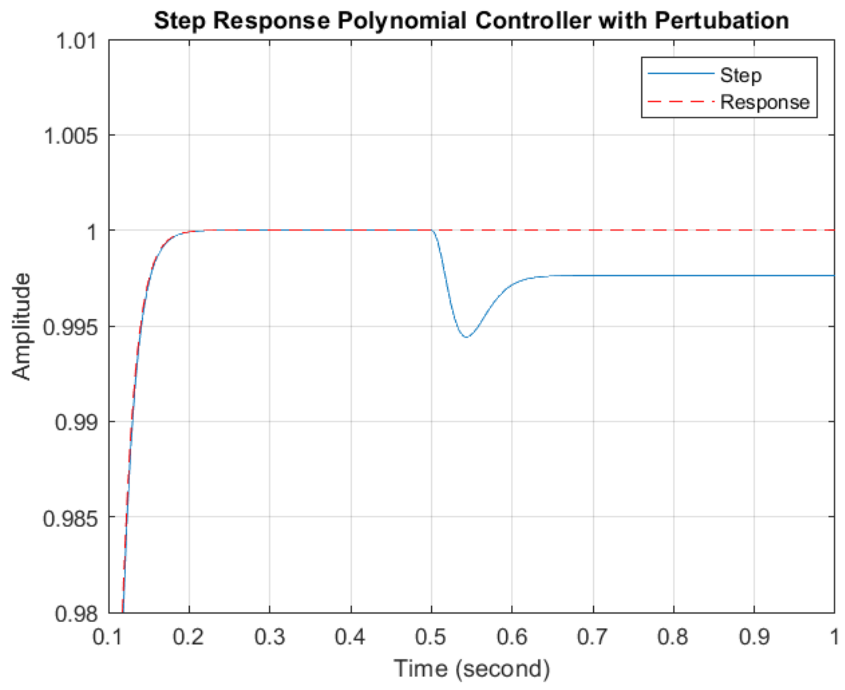
```
ans = struct with fields:
        RiseTime: 0.0618
    SettlingTime: 0.1176
     SettlingMin: 0.9005
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.2153
```

As we can see the step response of the object after adding the controller behave like the desired polynomial.

The step response is fast and under 0.1 s with no overshoot or undershoot, just the technical condition required.

Now let's add the load torque perturbation $M_C = 5$ after 0.5 second.

```
STime = 1;
des_poly = step(tf(d0,[1 d4 d3 d2 d1 d0]),0:1e-3:STime);
out_poly = sim('Polynomial');
plot(out_poly.tout, out_poly.simout); hold on;
plot(out_poly.tout, des_poly, 'r--'); grid on; hold off;
title('Step Response Polynomial Controller with Pertubation')
xlabel('Time (second)')
ylabel('Amplitude')
xlim([0.10 1.000])
ylim([0.98 1.01])
legend('Step', 'Response');
```



After perturbation being added, we notice there is a slight difference of the static value that caused by the perturbation, but the controller still able to stabilize and as long as the load torque perturbation $M_C = 5\,\mathrm{Nm}$, the difference is still ignorable.

But the larger the disturbance, the larger the effect or difference. Therefore, a solution with a non-static polynomial controller is proposed.

## Non-Static Polynomial Controller

To make the polynomial controller as a non-static, we can add an integrator before the object, this integrator will provide non-static characteristic and will eliminate the static error that caused by the perturbation.

To do this, we need to recalculate the polynomials $R(s)$ and $C(s)$, because the additional integrator will add additional degrees to the controller.

**1. Determine the degrees of controller polynomials $R(s)$ and $C(s)$ and degree of desired polynomial of closed loop system.**

$$\deg(R(s)) = \deg(A(s)) = 3;$$
$$\deg(C(s)) = \deg(R(s)) - 1 = 3\ - 1\ =\ 2;$$
$$\deg(D(s)) = \deg(A(s)) + \deg(C(s))\ +\ 1 = 3 + 2 + 1 = 6;$$

After the degrees of the polynomials are determined, we can define the polynomials $R(s)$ and $C(s)$ are defined as follow, note that the additional integrator will be added to the polynomial $C(s)$:

$$R(s) = r_3 \cdot s^3 + r_2 \cdot s^2 + r_1 \cdot s + r_0$$

$$C(s) = s^2 + c_1 \cdot s + c_0$$

**2. Set the desired polynomial of the closed loop system using Newton polynomial**

Desired polynomial $D(s)$, that has the degree of 6, can be defined using Newton polynomial with $\Omega_0$ as geometric mean root.

$$D(s) = (s + \Omega_0)^6 = s^6 + 6 \cdot \Omega_0 \cdot s^5 + 15 \cdot \Omega_0^2 \cdot s^4 + 20 \cdot \Omega_0^3 \cdot s^3 + 15 \cdot \Omega_0^4 \cdot s^2 + 6 \cdot \Omega_0^5 \cdot s + \Omega_0^6$$

To simplify we can replace the coefficient of the polynomial $D(s)$.

$$D(s) = (s + \Omega_0)^6 = s^6 + d_5 \cdot s^5 +\ d_4 \cdot s^4 + d_3 \cdot s^3 + d_2 \cdot s^2 + d_1 \cdot s + d_0$$

with $d_5 =\ 6 \cdot \Omega_0;\ d_4 = 15 \cdot \Omega_0^2;\ d_3 = 20 \cdot \Omega_0^3;\ d_2 = 15 \cdot \Omega_0^4;\ d_1 = 6 \cdot \Omega_0^5;\ d_0 = \Omega_0^6;$

On the other hand the geometric mean root is calculated by

$$\Omega_0 = \frac{n + 2 * \sqrt{n-1}}{T_C} = 104.7214\ \frac{\text{rad}}{s}$$

with $n = 6$ as the degree of polynomial and $T_C = 0.1s$ as time of the step response.

Let's build the desired polynomial $D(s)$ in MATLAB.

```
n = 6;
mean_root_0 = (n + 2*sqrt(n-1))/tc;
dma = poly(-mean_root_0*ones(1,n));
d5a = dma(end-5);
```

```
d4a = dma(end-4);
d3a = dma(end-3);
d2a = dma(end-2);
d1a = dma(end-1);
d0a = dma(end);
```

**3. Solving the Polynomial $R(s)$ and $C(s)$**

Let's solve the polynomial $R(s)$ and $C(s)$ with this new configuration.

$$A(s) \cdot C(s) \cdot \ s + B(s) \cdot R(s) = D(s)$$

$$s^6 + (a_2 + c_1)\, s^5 + (a_1 + c_0 + a_2\, c_1)\, s^4 + (a_0 + a_1\, c_1 + a_2\, c_0 + b_0\, r_3)\, s^3 + (a_0\, c_1 + a_1\, c_0 + b_0\, r_2)\, s^2 + (a_0\, c_0 + b_0\, r_1)\, s + b_0\, r_0 = s^6 + d_5\, s^5 + d_4\, s^4 + d_3\, s^3 + d_2\, s^2 + d_1\, s + d_0$$

Let’s group it using coefficient comparison.

For $s^0$: $b_0\, r_0 = d_0$

For $s^1$: $a_0\, c_0 + b_0\, r_1 = d_1$

For $s^2$: $a_0\, c_1 + a_1\, c_0 + b_0\, r_2 = d_2$

For $s^3$: $a_0 + a_1\, c_1 + a_2\, c_0 + b_0\, r_3 = d_3$

For $s^4$: $a_1 + c_0 + a_2\, c_1 = d_4$

For $s^5$: $a_2 + c_1 = d_5$

Solve each variable of the polynomial $R(s)$ and $C(s)$

$$c_1 \ = d_5 - a_2 = 619.2372;$$

$$c_0 \ = \ d_5 - a_2 = 1.5887e + 05;$$

$$r_3 \ = \ \frac{d_3 - a_0 - a_1\, c_1 - a_2\, c_0}{b_0} = 1.0594e + 06;$$

$$r_2 \ = \ \frac{d_2 - a_0\, c_1 - a_1\, c_0}{b_0} = 8.8787e + 07;$$

$$r_1 \ = \ \frac{d_1 - a_0\, c_0}{b_0} = 3.7193e + 09;$$

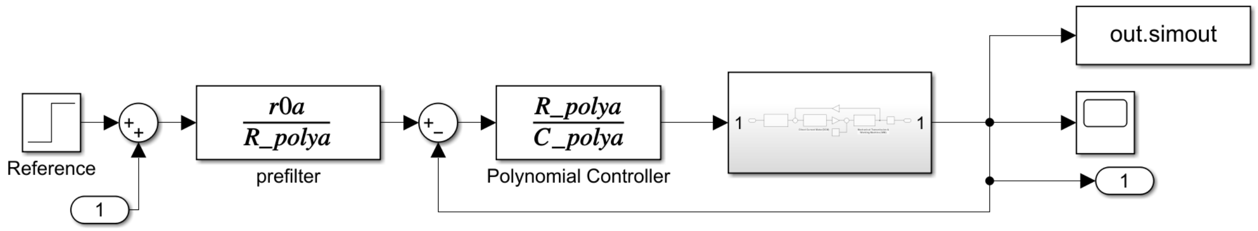
$$r_0 = \ \frac{d_0}{b_0} = 6.4915e + 10$$

```
c1a = d5a - a2;
c0a = d4a - a1 -a2*c1a;
r0a = d0a/b0;
r1a = (d1a - a0*c0a)/b0;
r2a = (d2a - a0*c1a - a1*c0a)/b0;
r3a = (d3a - a0 - a1*c1a - a2*c0a)/b0;
```

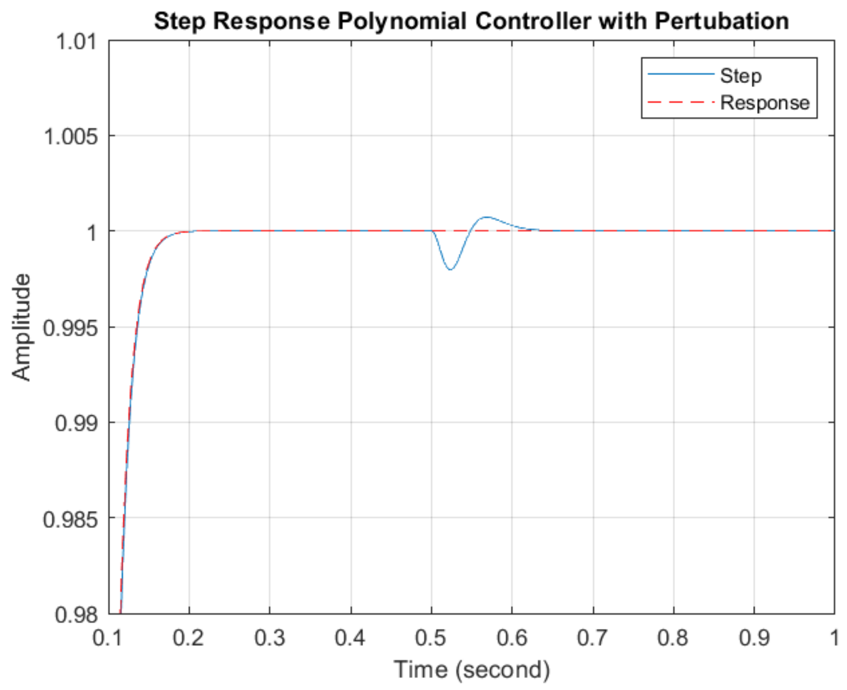
With that calculated, we can make the following matrices as the polynomial $R(s)$ and $C(s)$ that later will be used in the Simulink model. Note that here the additional integrator is added to the polynomial $C(s)$, so it can be simulated directly to the simulink model.

```
R_polya = [r3a r2a r1a r0a];
C_polya = [1 c1a c0a 0];
```

Let's simulate the Simulink model Astatic_Polynomial.slx that can be seen in the following figure.



```
STime = 1;
des_poly = step(tf(d0a,dma),0:1e-3:STime);
out_poly = sim('Astatic_Polynomial');
plot(out_poly.tout, out_poly.simout); hold on;
plot(out_poly.tout, des_poly, 'r--'); grid on; hold off;
title('Step Response Polynomial Controller with Pertubation')
xlabel('Time (second)')
ylabel('Amplitude')
xlim([0.10 1.000])
ylim([0.98 1.01])
legend('Step', 'Response');
```



With this non-static polynomial controller, the signal get back to the desired output after the pertubation added.

Let's also calculate the static and dynamic characteristic of this object with controller.

Using function *linmod*, the entire transfer function of the object with the controller will be extracted. Then the information of the transfer function can be calculated and extracted using *stepinfo* function.

```
[num_P, denum_P] = linmod('Astatic_Polynomial');
Hp = tf(num_P, denum_P);
stepinfo(Hp)
```

```
ans = struct with fields:
        RiseTime: 0.0585
    SettlingTime: 0.1149
     SettlingMin: 0.9039
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.2004
```

The step response is fast and under 0.1 s and the overshoot is lower than 0.5%, just the technical condition required.

We can conclude that the controller works, it gives stable and fast response without any overshoot, undershoot or oscillation, and behave like technical condition required, even after the perturbation added.

## Task - 4: Synthesis State Space Controller

Make the analytical synthesis of state-space controller with the modal control method for Newton desired polynomial.

Calculate the static and dynamic characteristics of the closed loop system with polynomial controller.

### Analytical Synthesis of State Space Controller

To synthesize the polynomial controller the algorithm can be followed:

**1. Determine $A$, $B$ and $C$ matrices of controlled object and polynomial $A(s)$ of its transfer function:**

From the previous calculation, we know already the matrices $A$, $B$ and $C$:

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -9.0909 & -0.4618 & 0 \\ 0.6400 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; B = \begin{bmatrix} \frac{K_P}{R_a T_a} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 31.7460 \\ 0 \\ 0 \end{bmatrix}; C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

and also, the transfer function of object $H_{\text{oU}}(s)$ that has the denominator $A(s) = s^3 + 9.0909s^2 + 0.2955\,s$.

**2. Set the desired polynomial of the closed loop system in standard form:**

For the state space controller, we can set the degree of desired polynomial $D(s)$ equal to the degree of denominator $A(s)$. Therefore, the desired polynomial $D(s)$ with degree of 3 is defined with $\Omega_0$ as geometric mean root:

$$D(s) = (s + \Omega_0)^3 = s^3 + 3 \cdot \Omega_0 \cdot s^2 + 3 \cdot \Omega_0^2 \cdot s + \Omega_0^3 \cdot$$

To simplify we can replace the coefficient of the polynomial $D(s)$.

$$D(s) = (s + \Omega_0)^5 = s^3 + d_2 \cdot s^2 + d_1 \cdot s + d_0$$

with $d_2 = 3 \cdot \Omega_0;\ d_1 = 3 \cdot \Omega_0^2;\ d_0 = \Omega_0^3;$

On the other hand, the geometric mean root is calculated by

$$\Omega_0 = \frac{n + 2 * \sqrt{n-1}}{T_C} = 58.2843\ \frac{\text{rad}}{s}$$

with $n = 3$ as the degree of polynomial and $T_C = 0.1s$ as time for the step response.

Let's build the desired polynomial $D(s)$ in MATLAB.

```
n = 3;
mean_root_0 = (n + 2*sqrt(n-1))/tc;

dss2 = 3*mean_root_0;
dss1 = 3*mean_root_0^2;
dss0 = mean_root_0^3;
```

**3. Calculate the vector of state controller $\overline{K}$ in Canonic Controllable Form:**

$$\overline{K} = [d_0 - a_0,\ d_1 - a_1, d_2 - a_2] = [1.9799e+5 \quad 1.0191e+4 \quad 165.7619]$$

```
K_canctr = [dss0 - a0, dss1 - a1, dss2 - a2];
```

**4. Calculate the state coordinates transformation matrix:**

To calculate the state coordinate transformation matrix $P$, we need the controllability matrices in canonic controllable form and in form of object. From the previous calculation we already calculated the controllability matrix in form of object $U$.

$$U = [B \quad A \cdot B \quad A^2 \cdot B]$$

$$U = \begin{pmatrix} \frac{K_p}{R_a T_a} & -\frac{K_p}{R_a {T_a}^2} & \frac{K_p \left(\frac{1}{{T_a}^2} - \frac{C^2}{J R_a T_a}\right)}{R_a T_a} \\ 0 & \frac{C K_p}{J R_a T_a} & -\frac{C K_p}{J R_a {T_a}^2} \\ 0 & 0 & \frac{C K_p}{J R_a T_a} \end{pmatrix} = \begin{bmatrix} 31.7460 & -288.6003 & 2.6143e+03 \\ 0 & 20.3175 & -184.7042 \\ 0 & 0 & 20.3175 \end{bmatrix}$$

Now, we need the controllability matrix in canonic controllable form.

$$\overline{U} = [\overline{B} \quad \overline{A} \cdot \overline{B} \quad \overline{A}^2 \cdot \overline{B}]$$

For that, we need the matrix $\overline{A}$ and $\overline{B}$ in canonic controllable form.

$$\overline{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\frac{C^2}{J R_a T_a} & -\frac{1}{T_a} \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -0.2955 & -9.0909 \end{bmatrix};$$

$$\overline{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

which gives

$$\overline{U} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & -\frac{1}{T_a} \\ 1 & -\frac{1}{T_a} & \frac{1}{{T_a}^2} - \frac{C^2}{J\,R_a\,T_a} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -9.0909 \\ 1 & -9.0909 & 82.3491 \end{bmatrix}.$$

With that we can calculate the state coordinate transformation matrix $P$ to transform from canonic controllable form into the object form.

$$P = \overline{U} \cdot U^{-1}$$

$$P = \begin{pmatrix} 0 & 0 & \frac{J\,R_a\,T_a}{C\,K_p} \\ 0 & \frac{J\,R_a\,T_a}{C\,K_p} & 0 \\ \frac{R_a\,T_a}{K_p} & 0 & 0 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0.0492 \\ 0 & 0.0492 & 0 \\ 0.0315 & 0 & 0 \end{bmatrix}$$

```
A_canctr = [0 1 0; 0 0 1; -a0 -a1 -a2];
B_canctr = [0;0;1];
U_canctr = [B_canctr A_canctr*B_canctr A_canctr^2*B_canctr];
P_UtoSS = U_canctr/U;
```

**5. Transform the $\overline{K}$ vector from Canonic Controllable Form in to form of the real object $K$**

$$K = \overline{K} \cdot P$$

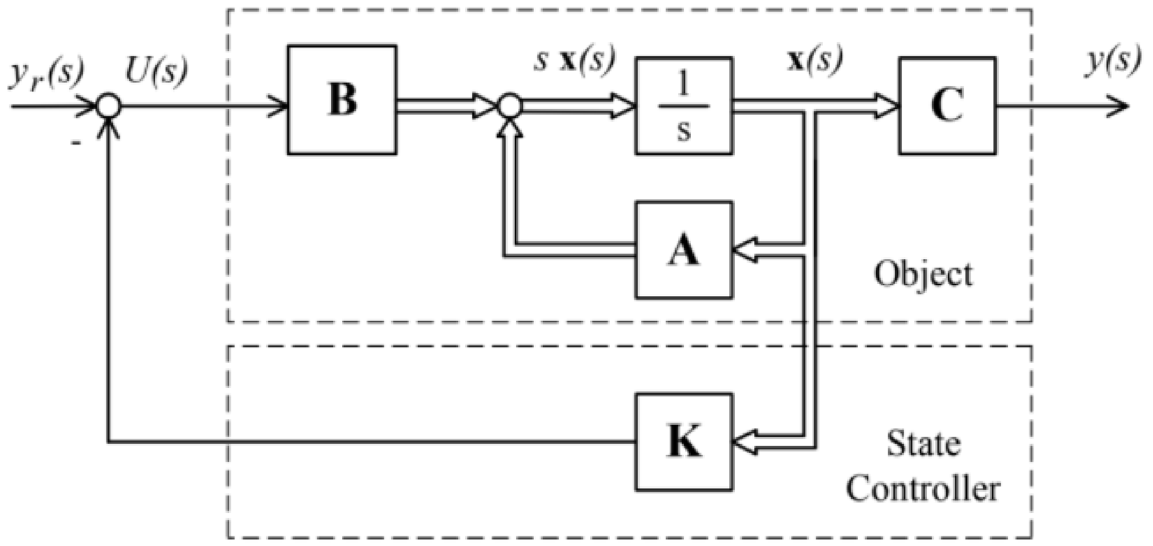
$$K = \left[ \left( \frac{R_a\,T_a\,\left(d_2 - \frac{1}{T_a}\right)}{K_p} \quad \frac{J\,R_a\,T_a\,\left(d_1 - \frac{C^2}{J\,R_a\,T_a}\right)}{C\,K_p} \quad \frac{J\,R_a\,T_a\,d_0}{C\,K_p} \right) \right] = \begin{bmatrix} 5.2215 & 501.5820 & 9.7451e+3 \end{bmatrix}$$

```
K_ = K_canctr*P_UtoSS;
```

## Running the Simulation of State Space Controller on MATLAB

Now that we got the coefficient of state controller feedbacks $K$, we are able to build the state space controller.

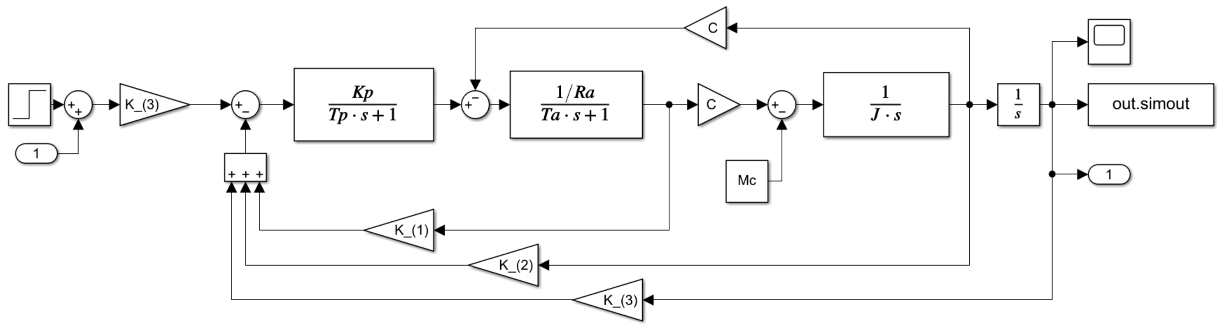
By default, the design of this controller is shown as in the following figure. In practice, each feedback value of the state coordinates will be multiplied by the vector $K$.



To eliminate the static error that the model will deliver, we need to add the static constants as a gain after the reference signal. This static constant can be calculated by compensating the transmission coefficient. Transmission coefficient is value of a transfer function when the s-Laplace operator is equal to zero. After adding the controller, our model will have a desired transfer function of $\frac{B(s)}{D(s)}$. Therefore, the transmission coefficient of our model is $\frac{b_0}{d_0}$, and to compensate that, our static constants is defined as follow:

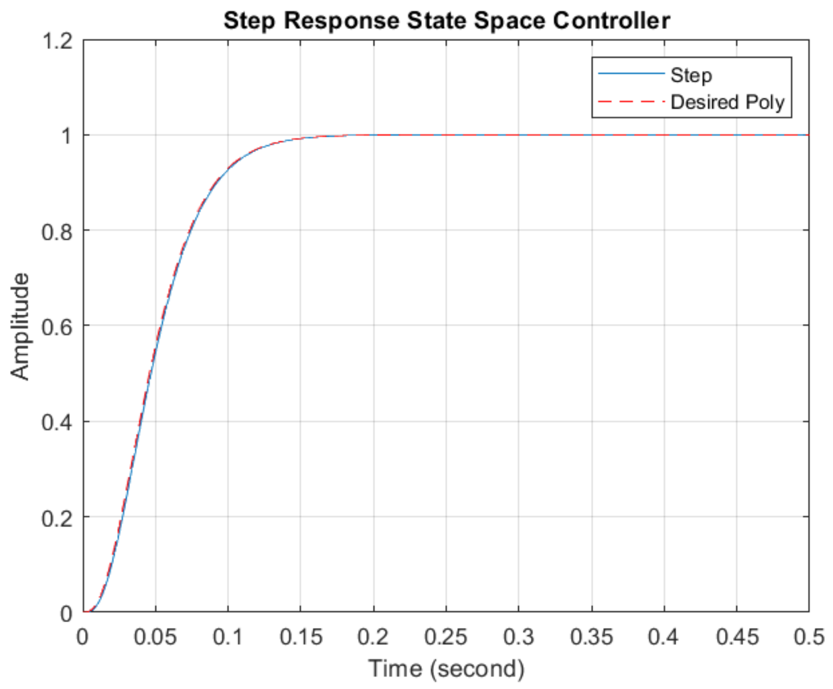
$$K_{\text{KSP}} = \frac{d_0}{b_0} = K(3)$$

The Simulink file StateSpace.slx is created to model the state space controller and its structure diagram can be shown in the following figure.



Let's run the simulation without any perturbation to see the step response.

```
STime = 0.5;
out_ss = sim('StateSpace');
des_ss = step(tf(dss0,[1 dss2 dss1 dss0]),0:1e-3:STime);
plot(out_ss.tout, out_ss.simout); grid on; hold on;
plot(out_ss.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space Controller')
xlabel('Time (second)')
ylabel('Amplitude')
ylim([0 1.2])
legend('Step', 'Desired Poly');
```



Let's also calculate the static and dynamic characteristic of this object with controller.

Using function *linmod*, the entire transfer function of the object with the controller will be extracted. Then the information of the transfer function can be calculated and extracted using *stepinfo* function.

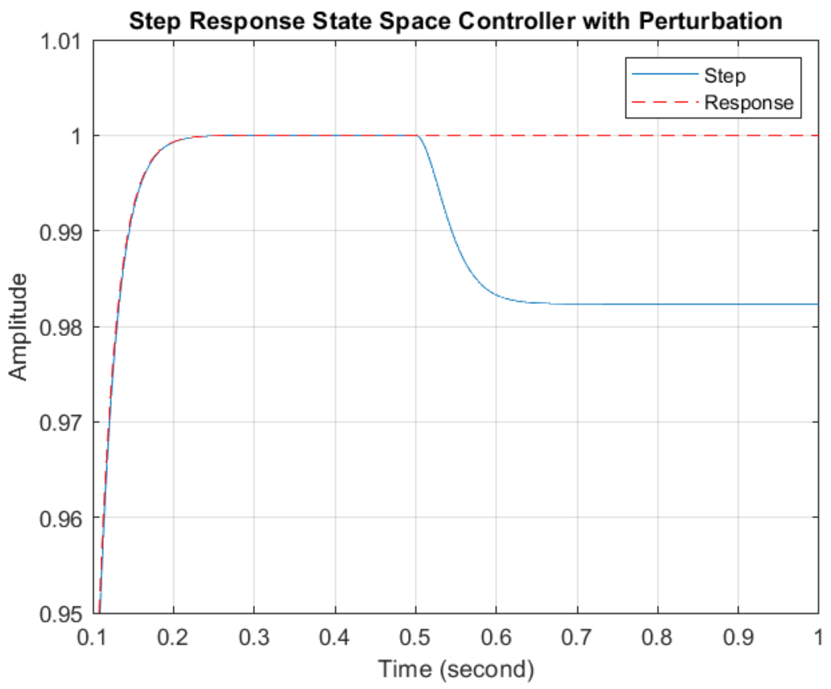
```
[num_SS, denum_SS] = linmod('StateSpace');
Hss = tf(num_SS, denum_SS);
stepinfo(Hss)
```

```
ans = struct with fields:
        RiseTime: 0.0730
    SettlingTime: 0.1292
     SettlingMin: 0.9299
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
```

```
        Peak: 1.0000
    PeakTime: 0.6900
```

Let's add the perturbation at time of 0.5 second after the first step response.

```
STime = 1;
out_ss = sim('StateSpace');
des_ss = step(tf(dss0,[1 dss2 dss1 dss0]),0:1e-3:STime);
plot(out_ss.tout, out_ss.simout); grid on; hold on;
plot(out_ss.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space Controller with Perturbation')
xlabel('Time (second)')
ylabel('Amplitude')
xlim([0.10 1.000])
ylim([0.95 1.01])
legend('Step', 'Response');
```



After perturbation being added, we notice there is a slight difference of the static value that caused by the perturbation, but the controller still able to stabilize and as long as the load torque perturbation $M_C = 5\,\mathrm{Nm}$, the difference is still ignorable.

But the larger the disturbance, the larger the effect or difference. Therefore, a solution with a non-static State Space Controller is proposed.

## Non-static State Space Controller

The basic idea of synthesizing the non-static state-space controller is by adding new integrator to provide the static characteristic and eliminate the static error caused by perturbation.

### 1. Extending the State-Space Matrices

To synthesize the non-static state-space controller, the matrices of the object and the matrices in canonic controllable form are needed to be extended.

```
% Extended State Space Matrices
Aa_(1,:) = [A_(1,1:end), 0];
Aa_(2,:) = [A_(2,1:end), 0];
Aa_(3,:) = [A_(3,1:end), 0];
Aa_(4,:) = [C_, 0]
```

```
Aa_ = 4×4
   -9.0909   -0.4618         0         0
    0.6400         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
```

```
Ba_ = [B_; 0]
```

```
Ba_ = 4×1
   31.7460
         0
         0
         0
```

```
Ca_ = [C_, 0]
```

```
Ca_ = 1×4
     0     0     1     0
```

```
Aa_canctr(1,:) = [0 1 0 0];
Aa_canctr(2,:) = [0, A_canctr(1,1:end)];
Aa_canctr(3,:) = [0, A_canctr(2,1:end)];
Aa_canctr(4,:) = [0, A_canctr(3,1:end)]
```

```
Aa_canctr = 4×4
         0    1.0000         0         0
         0         0    1.0000         0
         0         0         0    1.0000
         0         0   -0.2955   -9.0909
```

```
Ba_canctr = [0; B_canctr]
```

```
Ba_canctr = 4×1
     0
     0
     0
     1
```

**2. Set the desired polynomial of the closed loop system in standard form**

We also need to recalculate the desired polynomial, because the degree of the system is increased since an integrator is added.

```
n = 4;
mean_root_0 = (n + 2*sqrt(n-1))/tc;
dssa = poly(-mean_root_0*ones(1,n));
dss3a = dssa(end-3);
dss2a = dssa(end-2);
dss1a = dssa(end-1);
dss0a = dssa(end);
```

**3. Calculate the vector of state controller $\overline{K}$ in Canonic Controllable Form:**

```
K_canctr_a = [dss0a, dss1a-a0, dss2a-a1, dss3a-a2];
```

**4. Transform the $\overline{K}$ vector from Canonic Controllable Form in to form of the real object $K$**

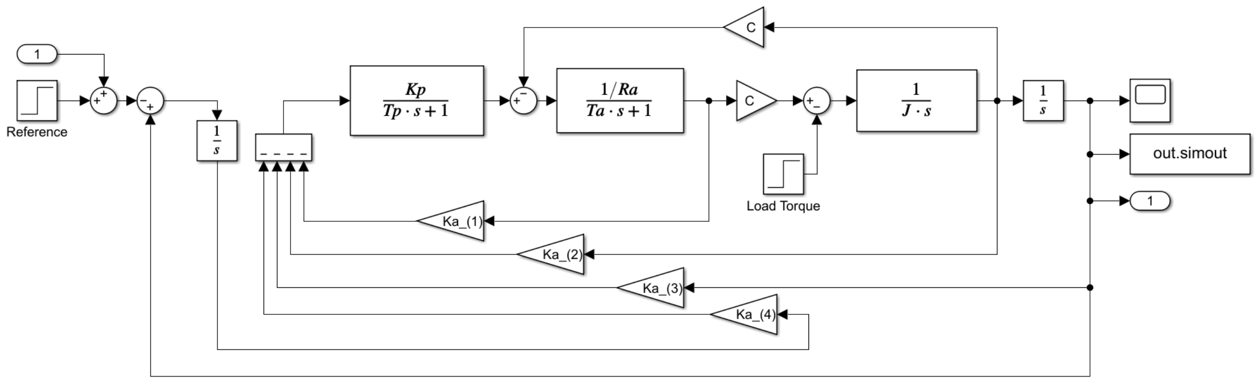
```
Ua_ = [Ba_, Aa_*Ba_, Aa_^2*Ba_, Aa_^3*Ba_]
```

```
Ua_ = 4×4
10^4 ×
    0.0032   -0.0289    0.2614   -2.3681
         0    0.0020   -0.0185    0.1673
         0         0    0.0020   -0.0185
         0         0         0    0.0020
```

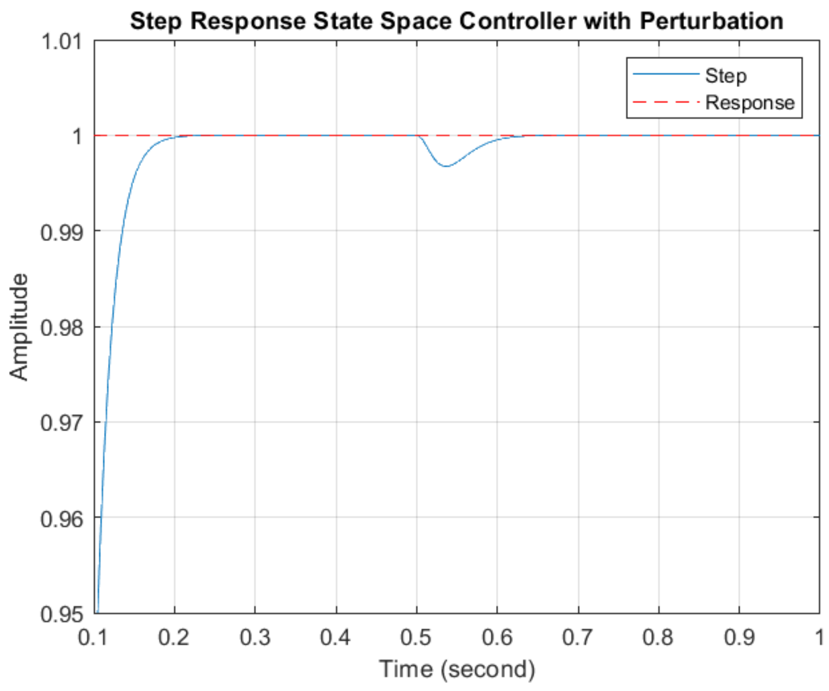
```
Ua_canctr = [Ba_canctr, Aa_canctr*Ba_canctr, Aa_canctr^2*Ba_canctr,
Aa_canctr^3*Ba_canctr];
P_UtoSS_a = Ua_canctr/Ua_;
Ka_ = K_canctr_a*P_UtoSS_a
```

```
Ka_ = 1×4
10^6 ×
    0.0000    0.0016    0.0819    1.5277
```

Let's simulate the state space controller with this new configuration. A new Simulink model called Astatic_StateSpace.slx is created for this purpose. Another advantage using this configuration is we have no need to add an addtional gain to eliminate the static error, because the additional integrator and the new calculated controller vector are able to do that.



```
STime = 1;
out_ss = sim('Astatic_StateSpace');
des_ss = step(tf(dssa,dssa),0:1e-3:STime);
plot(out_ss.tout, out_ss.simout); grid on; hold on;
plot(out_ss.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space Controller with Perturbation')
xlabel('Time (second)')
ylabel('Amplitude')
xlim([0.10 1.000])
ylim([0.95 1.01])
legend('Step', 'Response');
```



```
[num_SS, denum_SS] = linmod('Astatic_StateSpace');
Hss = tf(num_SS, denum_SS);
stepinfo(Hss)
```

```
ans = struct with fields:
        RiseTime: 0.0667
    SettlingTime: 0.1221
     SettlingMin: 0.9023
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.6200
```

We can conclude that the controller works, it gives stable and fast response without any overshoot, undershoot or oscillation, and behave like technical condition required, even after the perturbation being added.

## Task - 5: Synthesis State-Space Controller with Observer in Canonic Observable Form

Make the analytical synthesis of state-space controller with observer in Canonic Observable Form controller with the modal control method for Newton desired polynomial.

Calculate the static and dynamic characteristics of the closed loop system with polynomial controller.

### Analytical Synthesis of State Space Controller with State Observer

To synthesize the state space controller with state observer in canonic observable form, the algorithm can be followed:

**1. Determine A, B and C matrices of controlled object and polynomial A(s) of it`s transfer function:**

From the previous calculation, we know already the matrices $A$, $B$ and $C$:

$$A = \begin{bmatrix} -\frac{1}{T_a} & -\frac{C}{R_a T_a} & 0 \\ \frac{C}{J} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -9.0909 & -0.4618 & 0 \\ 0.6400 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix};\ B = \begin{bmatrix} \frac{K_P}{R_a T_a} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 31.7460 \\ 0 \\ 0 \end{bmatrix};\ C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

and also, the transfer function of object $H_{\mathrm{oU}}(s)$ that has the denominator $A(s) = s^3 + 9.0909s^2 + 0.2955\,s$.

**2. Write the matrices of state observer in canonic observable form:**

$$\widetilde{A} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -\frac{C^2}{J\,R_a\,T_a} \\ 0 & 1 & -\frac{1}{T_a} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -0.2955 \\ 0 & 1 & -9.0909 \end{bmatrix}$$

$$\widetilde{B} = \begin{pmatrix} \frac{C\,K_p}{J\,R_a\,T_a} \\ 0 \\ 0 \end{pmatrix} = \begin{bmatrix} 20.3175 \\ 0 \\ 0 \end{bmatrix}$$

$$\widetilde{C} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Let's calculate these matrices in MATLAB

```
A_canobs = [0 0 -a0; 1 0 -a1; 0 1 -a2];
B_canobs = [b0; 0; 0];
C_canobs = [0 0 1];
```

**3. Set the desired polynomial of the closed loop system in standard form:**

For the state space controller with the observer, we can set the degree of desired polynomial $D(s)$ equal to the degree of denominator $A(s)$. Therefore, the desired polynomial $D(s)$ with degree of 3 is defined with $\Omega_0$ as geometric mean root:

$$D(s) = (s + \Omega_0)^3 = s^3 + 3 \cdot \Omega_0 \cdot s^2 + 3 \cdot \Omega_0^2 \cdot s + \Omega_0^3 \cdot$$

To simplify we can replace the coefficient of the polynomial $D(s)$.

$$D(s) = (s + \Omega_0)^5 = s^3 + d_2 \cdot s^2 + d_1 \cdot s + d_0$$

with $d_2 = 3 \cdot \Omega_0;\ d_1 = 3 \cdot \Omega_0^2;\ d_0 = \Omega_0^3;$

On the other hand the geometric mean root is calculated by

$$\Omega_0 = \frac{n + 2 * \sqrt{n-1}}{T_C} = 58.2843\ \frac{\text{rad}}{s}$$

with $n = 3$ as the degree of polynomial and $T_C = 0.1s$ as time of the step response.

The previous value that used in the synthesis of state space controller can also be used for this.

```
[1, dss2, dss1, dss0]
```

```
ans = 1×4
10⁵ ×
    0.0000    0.0017    0.1019    1.9799
```

**4. Calculate the vector of state controller $\overline{K}$ in canonic controllable form:**

Because in the previous chapter we already calculate the vector state controller $\overline{K}$ in canonic controllable form, that value still can be used again.

$$\overline{K} = [d_0 - a_0,\ d_1 - a_1, d_2 - a_2] = [1.9799e+5 \quad 1.0191e+4 \quad 165.7619]$$

```
K_canctr
```

```
K_canctr = 1×3
10⁵ ×
    1.9799    0.1019    0.0017
```

**5. Transform the $\widetilde{K}$ vector in form of state observer in canonic observer form**

To do the transformation, we need to calculate the state coordinate transformation matrix $P$, but this time the transformation is from the canonic controllable form into the canonic observer form.

In the previous chapter, we already calculate the controllability matric in canonic controllable form.

$$\overline{U} = [\overline{B} \quad \overline{A} \cdot \overline{B} \quad \overline{A}^2 \cdot \overline{B}]$$

$$\overline{U} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & -\frac{1}{T_a} \\ 1 & -\frac{1}{T_a} & \frac{1}{{T_a}^2} - \frac{C^2}{J\,R_a\,T_a} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -9.0909 \\ 1 & -9.0909 & 82.3491 \end{bmatrix}$$

Now, we need to calculate the controllability matric in canonic observer form.

$$\widetilde{U} = \left[\widetilde{B} \quad \widetilde{A}\cdot\widetilde{B} \quad \widetilde{A}^2\cdot\widetilde{B}\right]$$

$$\widetilde{U} = \begin{pmatrix} \frac{C\,K_p}{J\,R_a\,T_a} & 0 & 0 \\ 0 & \frac{C\,K_p}{J\,R_a\,T_a} & 0 \\ 0 & 0 & \frac{C\,K_p}{J\,R_a\,T_a} \end{pmatrix} = \begin{bmatrix} 20.3175 & 0 & 0 \\ 0 & 20.3175 & 0 \\ 0 & 0 & 20.3175 \end{bmatrix}$$

```
U_canobs = [B_canobs A_canobs*B_canobs A_canobs^2*B_canobs];
```

And then we can calculate the state coordinate transformation matrix $P$.

$$P = \overline{U}\cdot\widetilde{U}^{-1}$$

$$P = \begin{pmatrix} 0 & 0 & \frac{J\,R_a\,T_a}{C\,K_p} \\ 0 & \frac{J\,R_a\,T_a}{C\,K_p} & -\frac{J\,R_a}{C\,K_p} \\ \frac{J\,R_a\,T_a}{C\,K_p} & -\frac{J\,R_a}{C\,K_p} & -\frac{C^2\,T_a - J\,R_a}{C\,K_p\,T_a} \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0.0492 \\ 0 & 0.0492 & -0.4474 \\ 0.0492 & -0.4474 & 4.0531 \end{bmatrix}$$

```
P_UtoCanobs = U_canctr/U_canobs;
```

Finally, we can transform the $\widetilde{K}$ vector in canonic observable form.

$$\widetilde{K} = \overline{K}\cdot P$$

$$\widetilde{K} = \left[8.1586 \quad 427.4130 \quad 5.8571e+03\right]$$

```
K_canobs = K_canctr*P_UtoCanobs
```

```
K_canobs = 1×3
10^3 ×
    0.0082    0.4274    5.8571
```

**6. Calculate the vector $\widetilde{L}$ of observer tuning loop in canonic observable form:**

Lastly, before we model in Simulink, we need to calculate the vector L of observer tuning loop in canonic observable form.

To do that, first we need to determine the desired polynomial of the observer tuning loop system $D^*(s)$. The degree of $D^*(s)$ is equal the degree of $D(s)$, but this time we use $\Omega_t$ as the geometric mean root.

$$D^*(s) = (s+\Omega_t)^3 = s^3 + 3\cdot\Omega_t\cdot s^2 + 3\cdot\Omega_t^2\cdot s + \Omega_t^3\cdot$$

To simplify we can replace the coefficient of the polynomial $D^*(s)$.

$$D^*(s) = (s+\Omega_0)^5 = s^3 + d_2^*\cdot s^2 + d_1^*\cdot s + d_0^*$$

with $d_2^* = 3\cdot\Omega_t;\ d_1^* = 3\cdot\Omega_t^2;\ d_0^* = \Omega_t^3;$

The geometric mean root $\Omega_t$ must be twice or trice bigger than the $\Omega_0$.

$$\Omega_t = (2 \div 3)\Omega_0$$

```
omega_t = mean_root_0*2;

dlo2 = 3*omega_t;
dlo1 = 3*omega_t^2;
dlo0 = omega_t ^3;
```

Then the $\widetilde{L}$ vector of observer tuning loop can be calculated

$$\widetilde{L} = [d_0^* - a_0,\ d_1^* - a_1, d_2^* - a_2] = [1.5840e+06 \quad 4.0764e+04 \quad 340.6147]$$

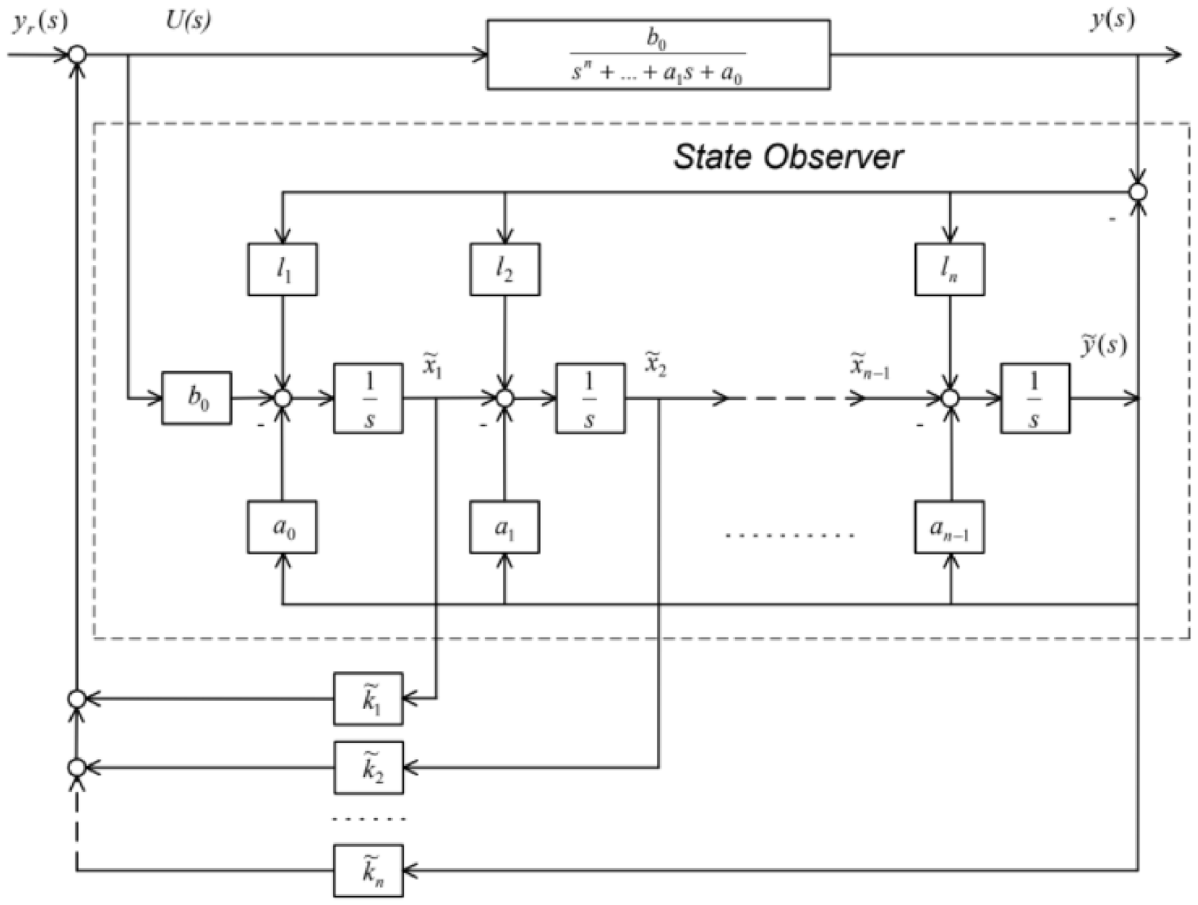
```
L_obs = [dlo0 - a0, dlo1 - a1, dlo2 - a2]
```

```
L_obs = 1×3
10^6 ×
    3.3268    0.0669    0.0004
```

## Running the Simulation of State Space Controller with Observer in Canonic Observable Form on MATLAB

Now that we got the coefficient of state controller feedbacks $\widetilde{K}$ and the vector of observer tuning loop $\widetilde{L}$, we are able to build the state space controller with observer in canonic observable form.

By default, the design of this controller is shown as in the following figure.



Instead of using multiple sensors in each state coordinates, we use an observer and then get the value of the state coordinates from this observer. Then, each feedback value of the state coordinates will be multiplied by the vector $\tilde{K}$ just like previous chapter. The $L$ vector of observer tuning loop is used to tune the state observer, in case any different of the output signal, for example caused by the perturbation, this different will be compensated by this tuning vector $\tilde{L}$.

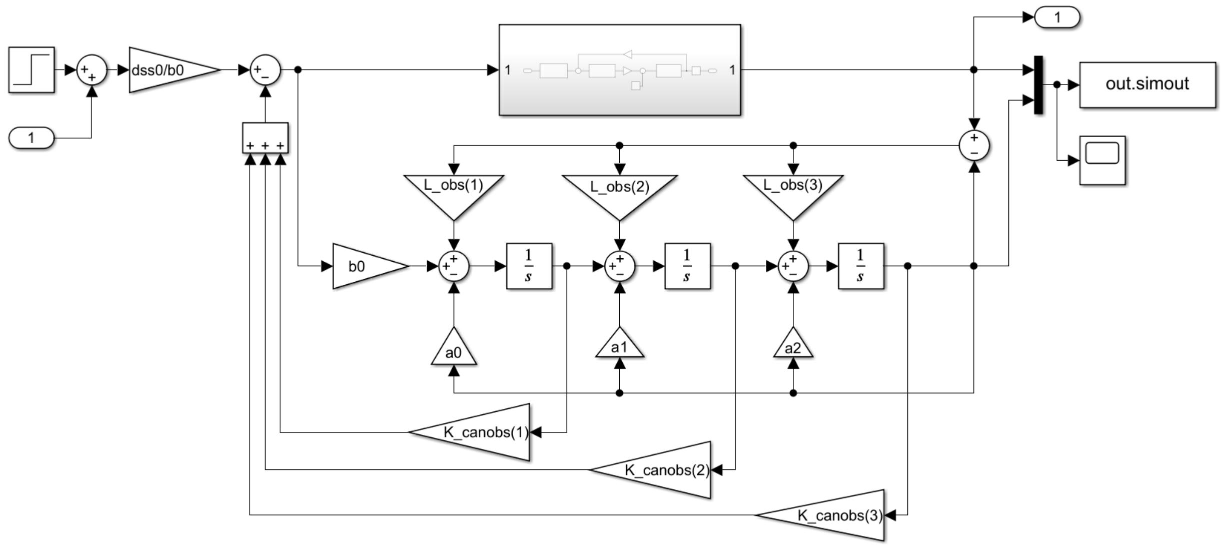
To eliminate the static error that the model will deliver, we need to add the static constant as a gain after the reference signal.

This static constant can be calculated by compensating the transmission coefficient. Transmission coefficient is value of a transfer function when the s-Laplace operator is equal to zero. After adding the controller, our model will have a desired transfer function of $\frac{B(s)}{D(s)}$.

Therefore, the transmission coefficient of our model is $\frac{b_0}{d_0}$, and to compensate that, our static constants is defined as follow:

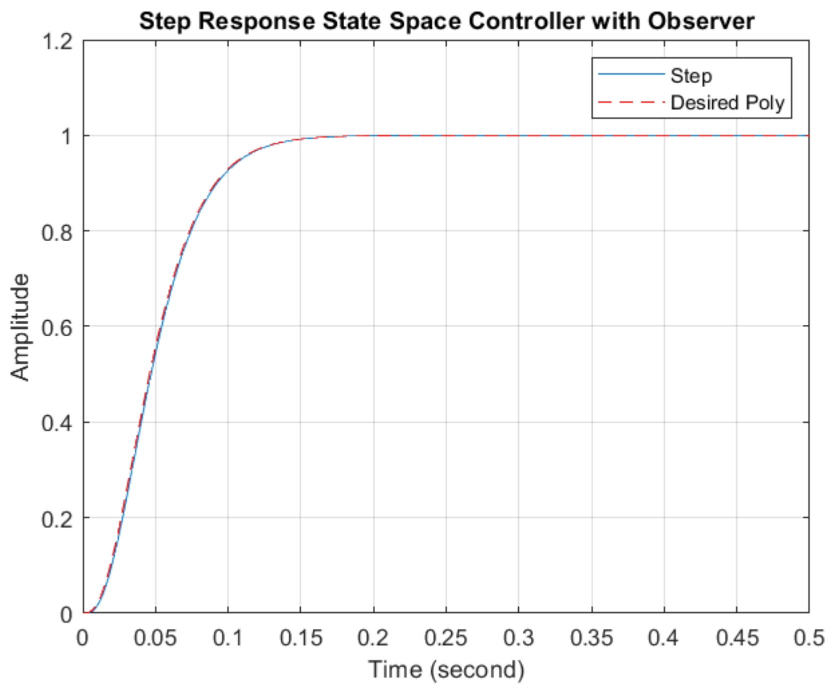
$$K_{\text{KSP}} = \frac{d_0}{b_0};$$

The Simulink file StateSpaceObserver.slx is created to model the state space controller with observer in canonic observable form and its structure diagram can be shown in the following figure.



Let's run the simulation without any perturbation to see the step response.

```
STime = 0.5;
out_sso = sim('StateSpaceObserver');
des_ss = step(tf(dss0,[1 dss2 dss1 dss0]),0:1e-3:STime);
plot(out_sso.tout, out_sso.simout); grid on; hold on;
plot(out_sso.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space Controller with Observer')
xlabel('Time (second)')
ylabel('Amplitude')
ylim([0 1.2])
legend('Step', 'Desired Poly');
```



Let's also calculate the static and dynamic characteristic of this object with controller.

Using function *linmod*, the entire transfer function of the object with the controller will be extracted. Then the information of the transfer function can be calculated and extracted using *stepinfo* function.

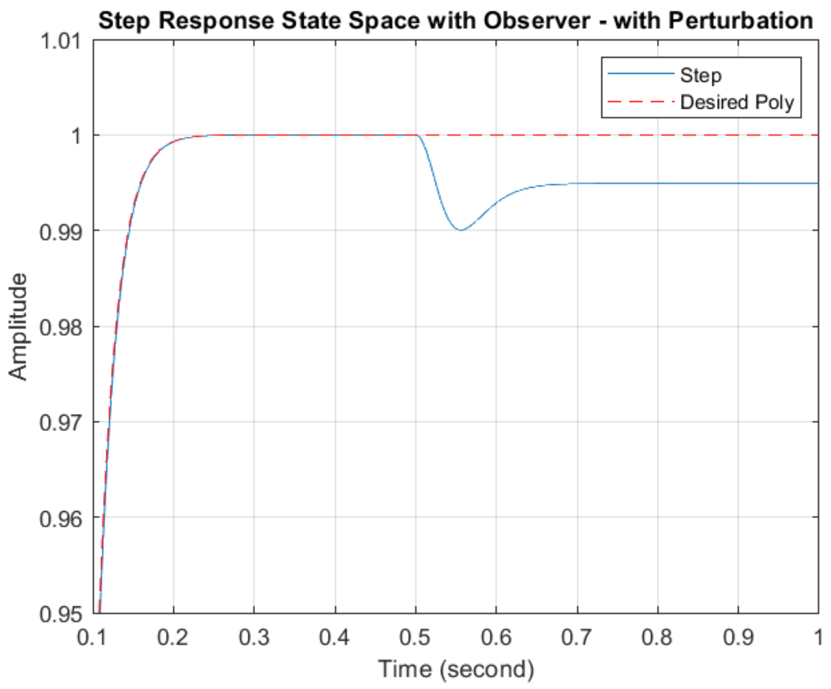
```
[num_SO, denum_SO] = linmod('StateSpaceObserver');
Hso = tf(num_SO, denum_SO);
stepinfo(Hso)
```

```
ans = struct with fields:
        RiseTime: 0.0730
    SettlingTime: 0.1292
     SettlingMin: 0.9299
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.6900
```

Now let's add the perturbation at time of 0.5 second after the first step response.

```
STime = 1;
out_sso = sim('StateSpaceObserver');
des_ss = step(tf(dss0,[1 dss2 dss1 dss0]),0:1e-3:STime);
plot(out_sso.tout, out_sso.simout); grid on; hold on;
plot(out_sso.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space with Observer - with Perturbation')
xlabel('Time (second)')
ylabel('Amplitude')
```

```
xlim([0.10 1.000])
ylim([0.95 1.01])
legend('Step', 'Desired Poly');
```



After perturbation being added, we notice there is a slight difference of the static value that caused by the perturbation, but the controller still able to stabilize and as long as the load torque perturbation $M_C = 5\ \mathrm{Nm}$, the difference is still ignorable.

But the larger the disturbance, the larger the effect or difference. Therefore, a solution with a non-static State Space Controller with Observer is proposed.

## Non-Static State Space Controller with Observer

The basic idea of synthesizing the non-static state-space controller is by adding new integrator to provide the static characteristic and eliminate the static error caused by perturbation.

### 1. Extending the State-Space Matrices

To synthesize the non-static state-space controller, the matrices of the object, the matrices in canonic controllable and observable form are needed to be extended. As we already do the extension in the previous chapter for the object form and the canonic controllable form, we need to do the transformation for the canonic observable form.

```
Aa_canobs(1,:) = [A_canobs(1,1:end), 0];
Aa_canobs(2,:) = [A_canobs(2,1:end), 0];
Aa_canobs(3,:) = [A_canobs(3,1:end), 0];
Aa_canobs(4,:) = [C_canobs, 0]
```

```
Aa_canobs = 4×4
         0         0         0         0
    1.0000         0   -0.2955         0
         0    1.0000   -9.0909         0
         0         0    1.0000         0
```

```
Ba_canobs = [B_canobs; 0]
```

```
Ba_canobs = 4×1
   20.3175
         0
         0
         0
```

```
Ca_canobs = [C_canobs, 0]
```

```
Ca_canobs = 1×4
     0     0     1     0
```

### 5. Transform the $\widetilde{K}$ vector in form of state observer in canonic observer form

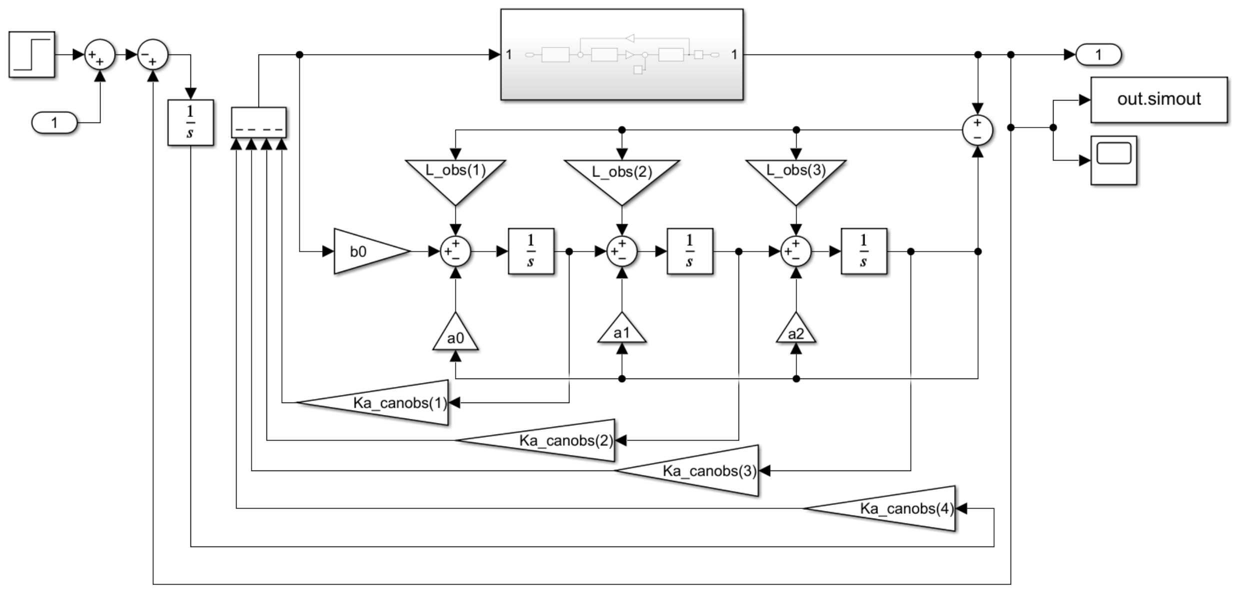
```
Ua_canobs = [Ba_canobs, Aa_canobs*Ba_canobs, Aa_canobs^2*Ba_canobs, Aa_canobs^3*Ba_canobs];
P_UtoCanobs_a = Ua_canctr/Ua_canobs;
Ka_canobs = K_canctr_a*P_UtoCanobs_a
```

```
Ka_canobs = 1×4
10^6 ×
    0.0000    0.0015    0.0681    1.5277
```

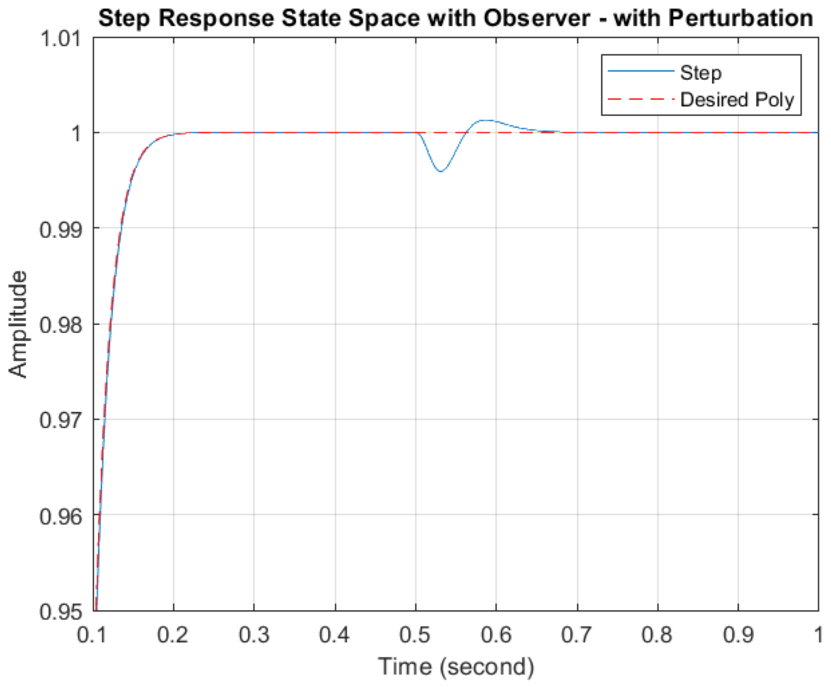
Let's simulate the state space controller with this new configuration. A new Simulink model called Astatic_StateSpace.slx is created for this purpose.

Another advantage using this configuration is we have no need to add an addtional gain to eliminate the static error, because the additional integrator and the new calculated controller vector are able to do that.

Moreover, we don't need to recalculate the $\widetilde{L}$ matrices, because we didn't change the observer, we only add an extra integrator to the object.



```
STime = 1;
out_sso = sim('Astatic_StateSpaceObserver');
des_ss = step(tf(dss0a, dssa),0:1e-3:STime);
plot(out_sso.tout, out_sso.simout); grid on; hold on;
plot(out_sso.tout, des_ss,'r--'); grid on; hold off;
title('Step Response State Space with Observer - with Perturbation')
xlabel('Time (second)')
ylabel('Amplitude')
xlim([0.10 1.000])
ylim([0.95 1.01])
legend('Step', 'Desired Poly');
```



```
[num_so, den_so]= linmod('Astatic_StateSpaceObserver');
Hsso = tf(num_so,den_so);
stepinfo(Hsso)
```

```
ans = struct with fields:
        RiseTime: 0.0667
    SettlingTime: 0.1221
     SettlingMin: 0.9023
     SettlingMax: 1.0000
       Overshoot: 2.2204e-14
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.6200
```

We can conclude that the controller work, it gives stable and fast response without any overshoot, undershoot or oscillation, and behave like technical condition required, even after the perturbation is added.

## Task - 6: Synthesis of Digital Polynomial Controller

Synthesize the input-output polynomial controller for Newton desired polynomial in digital form.

Calculate the static and dynamic characteristics of the closed loop system with digital controller

### Analytical Synthesis of Polynomial Controller in Digital Form

To synthesize the input-output polynomial controller for Newton desired polynomial in digital form, there are multiple ways, the easiest is to use the Z-transformation to the existing polynomial controller for example using Tustin or Euler approximation. This approach is assuming that we have relatively low sample time $T_q$. In this task, we won't do that approach, instead the approach of synthesis using modal control method for system in digital form. Therefore, the following algorithm can be followed:

**1. Determine discret transfer function and polynomials A(z) and B(z) of controller object.**

Because the original object is continuous, we will used Z-transformation and transform to discrete object equivalent. For this transformation, the zero-order holder approximation will be used with sample time of $T_q = 0.01$. This value is chosen after considering Shannon’s theorem for discretization under condition $\omega_q \geq 2\omega_C$. The sample time is in this case one tenth smaller than the time of step response $t_C$ of the required technical condition.

Here is the continuous transfer function

$$H_{\mathrm{oU}}(s) = \frac{20.3175}{s^3 + 9.0909s^2 + 0.2955\,s}$$

and this is transfer function after Z-transformation using zero order holder approximation.

$$H_{\mathrm{oU}}(z) = \frac{B(z)}{A(z)} = \frac{3.3107e-06\,z^2 + 1.2947e-05\,z + 3.1635e-06}{z^3 - 2.9131z^2 + 2.8262z - 0.9131}$$

```
Tq = 1e-2;
Hd = c2d(Ho, Tq, 'zoh')
```

```
Hd =
 
  3.311e-06 z^2 + 1.295e-05 z + 3.164e-06
  ---------------------------------------
    z^3 - 2.913 z^2 + 2.826 z - 0.9131
 
Sample time: 0.01 seconds
Discrete-time transfer function.
```

```
[num_d, denum_d] = tfdata(Hd);
num_d = num_d{:};
denum_d = denum_d{:};
roots(num_d)
```

```
ans = 2×1
   -3.6488
   -0.2619
```

```
roots(denum_d)
```

```
ans = 3×1
    1.0000
    0.9997
    0.9134
```

We notice that after the transformation we have additional zeros (2 zeros) and 3 poles.

To simplify the calculation the numerator and denominator of the digital transfer function of the object will be written as follows:

$A(z) = z^3 + a_2 z^2 + a_1 z + a_0$; with $a_2 = -2.9131$, $a_1 = 2.8262$, and $a_0 = -0.9131$, and

$B(z) = b_2 z^2 + b_1 z + b_0$, with $b_0 = 3.3107e - 06$, $b_0 = 1.2947e - 05$, and $b_0 = 3.1635e - 06$.

```
ad0 = denum_d(end);
ad1 = denum_d(end-1);
ad2 = denum_d(end-2);

bd0 = num_d(end);
bd1 = num_d(end-1);
bd2 = num_d(end-2);
```

**2. Determine the degrees of controller polynomials $R(z)$ and $C(z)$ and degree of desired polynomial of closed loop system $D(z)$:**

$$\deg(R(z)) = \deg(A(z)) - 1 = 3 - 1 = 2;$$
$$\deg(C(z)) \geq \deg(R(z)) = 2;$$
$$\deg(D(z)) \geq \deg(A(z)) + \deg(C(s)) = 3 + 2 = 5;$$

After the degrees of the polynomials are determined, we can define the polynomials $R(z)$ and $C(z)$ are defined as follow:

$$R(z) = r_2 \cdot z^2 + r_1 \cdot z + r_0$$

$$C(z) = z^2 + c_1 \cdot z + c_0$$

**3. Set the desired polynomial of the closed loop system in discreet form:**

Desired polynomial $D(z)$, that has the degree of 5, can be defined using Newton polynomial with $\Omega_d$ as geometric mean root.

$$D(z) = (z - \Omega_d)^5 = z^5 - 5 \cdot \Omega_d \cdot z^4 + 10 \cdot \Omega_d^2 \cdot z^3 - 10 \cdot \Omega_d^3 \cdot z^2 + 5 \cdot \Omega_d^4 \cdot z - \Omega_d^5$$

To simplify we can replace the coefficient of the polynomial $D(s)$.

$$D(s) = (z - \Omega_d)^5 = z^5 + d_4 \cdot z^4 + d_3 \cdot z^3 + d_2 \cdot z^2 + d_1 \cdot z + d_0$$

with $d_4 = -5 \cdot \Omega_d;\ d_3 = 10 \cdot \Omega_d^2;\ d_2 = -10 \cdot \Omega_d^3;\ d_1 = 5 \cdot \Omega_d^4;\ d_0 = -\Omega_d^5;$

On the other hand, the geometric mean root $\Omega_d$ is calculated by

$$\Omega_d = e^{T_q \cdot \Omega_0} = 0.4066$$

with $T_q = 0.01$ as the discrete sample time and $\Omega_0 = 90\ \frac{\text{rad}}{s}$.

```
n = 5;
mean_root_0 = (n + 2*sqrt(n-1))/tc;
mean_root_d = exp(-Tq*mean_root_0);
Dd = poly(mean_root_d*ones(1,n));
dd0 = Dd(end);
dd1 = Dd(end-1);
dd2 = Dd(end-2);
dd3 = Dd(end-3);
dd4 = Dd(end-4);
dd5 = Dd(end-5);
```

**4. Set and solve synthesis equation in discreet form:**

The synthesis equation can be described as follow.

$$A(z) \cdot C(z) + B(z) \cdot R(z) = D(z)$$

$$z^5 + (a_2 + c_1 + b_2\, r_2)\, z^4 + (a_1 + c_0 + a_2\, c_1 + b_1\, r_2 + b_2\, r_1)\, z^3 + (a_0 + a_1\, c_1 + a_2\, c_0 + b_0\, r_2 + b_1\, r_1 + b_2\, r_0)\, z^2 + (a_0\, c_1 + a_1\, c_0 + b_0\, r_1 + b_1\, r_0)\, z + a_0\, c_0 + b_0\, r_0 = s^5 + d_4 \cdot s^4 + d_3 \cdot s^3 + d_2 \cdot s^2 + d_1 \cdot s + d_0$$

From this equation we can do coefficient comparison:

for $z^4$: $a_2 + c_1 + b_2\, r_2 = d_4$

for $z^3$: $a_1 + c_0 + a_2\, c_1 + b_1\, r_2 + b_2\, r_1 = d_3$

for $z^2$: $a_0 + a_1\, c_1 + a_2\, c_0 + b_0\, r_2 + b_1\, r_1 + b_2\, r_0 = d_2$

for $z^1$: $a_0\, c_1 + a_1\, c_0 + b_0\, r_1 + b_1\, r_0 = d_1$

for $z^0$: $a_0\, c_0 + b_0\, r_0 = d_0$

To solve this, we can arrange the linear equation,

$$
\begin{aligned}
1\cdot c_1 + 0\cdot c_0 + b_2\cdot r_2 + 0\cdot r_1 + 0\cdot r_0 &= d_4 - a_2\\
a_2\cdot c_1 + 1\cdot c_0 + b_1\cdot r_2 + b_2\cdot r_1 + 0\cdot r_0 &= d_3 - a_1\\
a_1\cdot c_1 + a_2\cdot c_0 + b_0\cdot r_2 + b_1\cdot r_1 + b_2\cdot r_0 &= d_2 - a_0\\
a_0\cdot c_1 + a_1\cdot c_0 + 0\cdot r_2 + b_0\cdot r_1 + b_1\cdot r_0 &= d_1\\
0\cdot c_1 + a_0\cdot c_0 + 0\cdot r_2 + 0\cdot r_1 + b_0\cdot r_0 &= d_0
\end{aligned}
$$

and put them into matrix form

$$
\begin{bmatrix} 1 & 0 & b_2 & 0 & 0\\ a_2 & 1 & b_1 & b_2 & 0\\ a_1 & a_2 & b_0 & b_1 & b_2\\ a_0 & a_1 & 0 & b_0 & b_1\\ 0 & a_0 & 0 & 0 & b_0 \end{bmatrix} \cdot \begin{bmatrix} c_1\\ c_0\\ r_2\\ r_1\\ r_0 \end{bmatrix} = \begin{bmatrix} d_4 - a_2\\ d_3 - a_1\\ d_2 - a_0\\ d_1\\ d_0 \end{bmatrix}
$$

The coefficient of the polynomial $R(z)$ and $C(z)$ can then be solved by matrix operation.

$$
\begin{bmatrix} c_1\\ c_0\\ r_2\\ r_1\\ r_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_2 & 0 & 0\\ a_2 & 1 & b_1 & b_2 & 0\\ a_1 & a_2 & b_0 & b_1 & b_2\\ a_0 & a_1 & 0 & b_0 & b_1\\ 0 & a_0 & 0 & 0 & b_0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} d_4 - a_2\\ d_3 - a_1\\ d_2 - a_0\\ d_1\\ d_0 \end{bmatrix}
$$

Let's solve these linear equations in MATLAB.

```
M_D = [dd4-ad2; dd3-ad1; dd2-ad0; dd1; dd0];
             %C1  %C0  %R2  %R1  %R0
M_O(1,:) = [  1,   0, bd2,   0,   0];
M_O(2,:) = [ad2,   1, bd1, bd2,   0];
M_O(3,:) = [ad1, ad2, bd0, bd1, bd2];
M_O(4,:) = [ad0, ad1,   0, bd0, bd1];
M_O(5,:) = [  0, ad0,   0,   0, bd0];
M_C = inv(M_O)*M_D;
cd1 = M_C(1);
cd0 = M_C(2);
rd2 = M_C(3);
rd1 = M_C(4);
rd0 = M_C(5);
```

With the following matrices we will define the polynomial $R(z)$ and $C(z)$ and will later be put into the Simulink model.

```
cd_poly = [1 cd1 cd0]
```

```
cd_poly = 1×3
    1.0000    0.6468    0.1721
```

```
rd_poly = [rd2 rd1 rd0]
```

```
rd_poly = 1×3
10^5 ×
    0.7049   -1.1288    0.4618
```

## Running the Simulation of Digital Polynomial Controller on Matlab

The structure of the digital polynomial controller is similar to the polynomial controller. The Simulink file DiscretePolynomial.slx is created to simulate this controller. This model compares the continuous polynomial controller with the digital polynomial controller. The digital polynomial controller in this model is also placed to control the object in discrete form as well as the object in the continuous form.

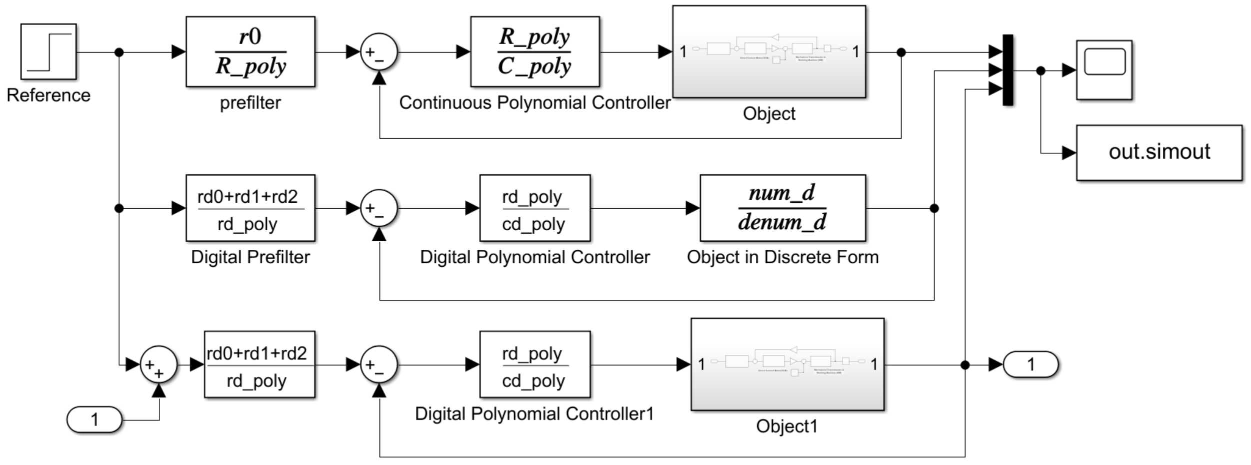
To eliminate the static error that the model will deliver, we need to add the static constants as a gain after the reference signal.

This static constant can be calculated by compensating the transmission coefficient. Transmission coefficient is value of a transfer function when the s-Laplace operator is equal to zero. But because the controller in this case is in digital form, we have to use the theorem of limits, which conclude that if s-Laplace operator is equal zero, then the z is goes to infinity.

Therefore, the static constants can be defined as follow:

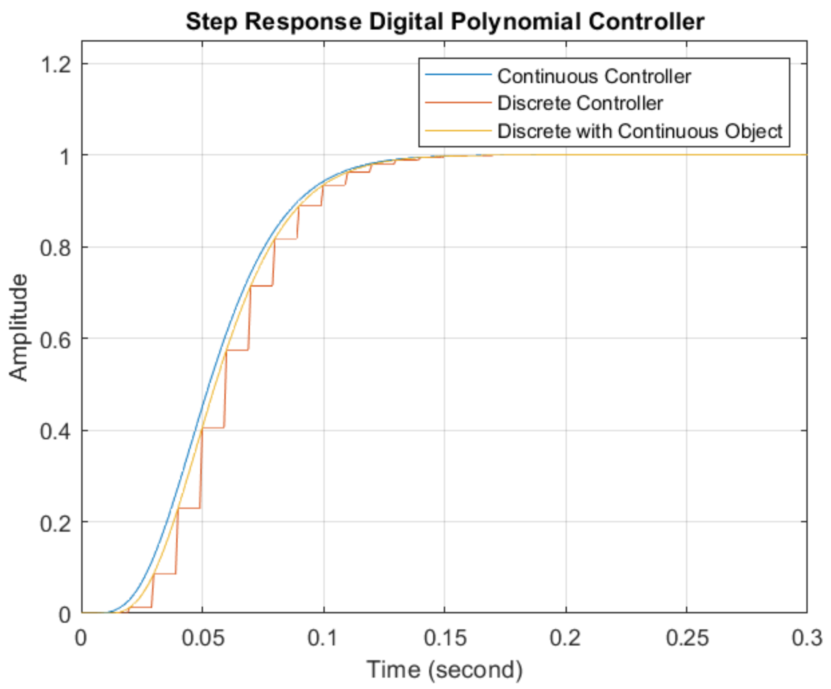
$$K_{\text{KSP}} = r_0 + r_1 + r_2$$

The structure diagram of this model is shown in the following figure.



Let's run the simulation without any perturbation to see the step response.

```
 STime = 0.3;
 out_sdo = sim('DiscretePolynomial');
 plot(out_sdo.tout, out_sdo.simout); grid on;
 title('Step Response Digital Polynomial Controller')
 xlabel('Time (second)')
 ylabel('Amplitude')
 ylim([0,1.25]);
 legend('Continuous Controller', 'Discrete Controller', 'Discrete with Continuous
Object')
```

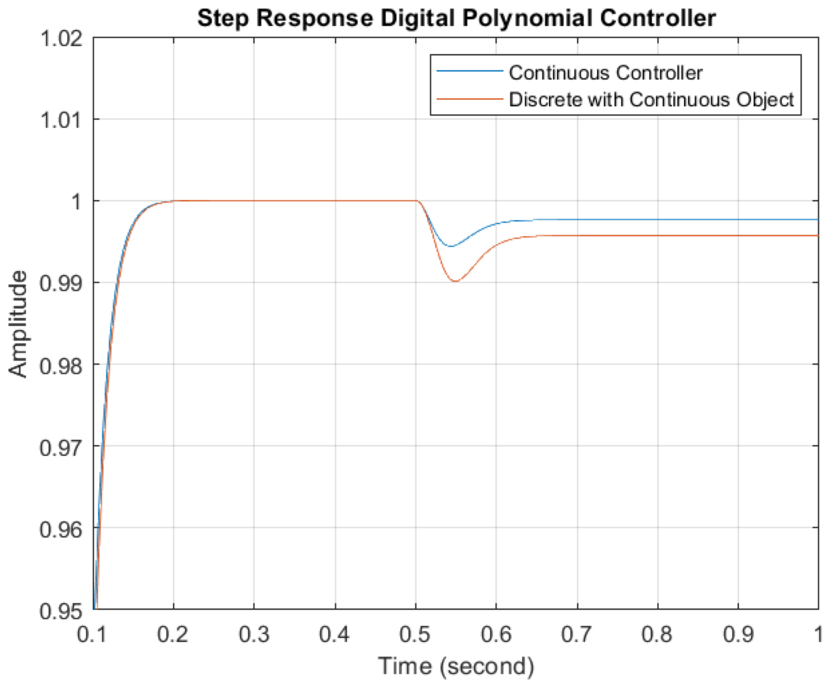


Now if we add the perturbation at time of 0.5 second after the first step response.

We will notice there is a slight difference of the static value, but the controller still able to stabilize. But with the load torque perturbation $M_C = 5\,\mathrm{Nm}$, the difference is still ignorable.

```
 STime = 1;
 out_sdo = sim('DiscretePolynomial');
 plot(out_sdo.tout, out_sdo.simout(1:end,1), out_sdo.tout, out_sdo.simout(1:end,3));
grid on;
 title('Step Response Digital Polynomial Controller')
 xlabel('Time (second)')
 ylabel('Amplitude')
 ylim([0.95,1.02]);
 xlim([0.1,1])
```

```
legend('Continuous Controller', 'Discrete with Continuous Object')
```



Let's also calculate the static and dynamic characteristic of this object with controller.

Using function *dlinmod*, the entire transfer function of the object with the controller in discrete form will be extracted. Then the information of the transfer function can be calculated and extracted using *stepinfo* function.

```
[num_dO, denum_dO] = dlinmod('onlyDiscrete', Tq);
Hdo = tf(num_dO, denum_dO,Tq);
stepinfo(Hdo)
```

```
ans = struct with fields:
        RiseTime: 0.0600
    SettlingTime: 0.1200
     SettlingMin: 0.9343
     SettlingMax: 1.0000
       Overshoot: 3.0864e-12
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.6900
```

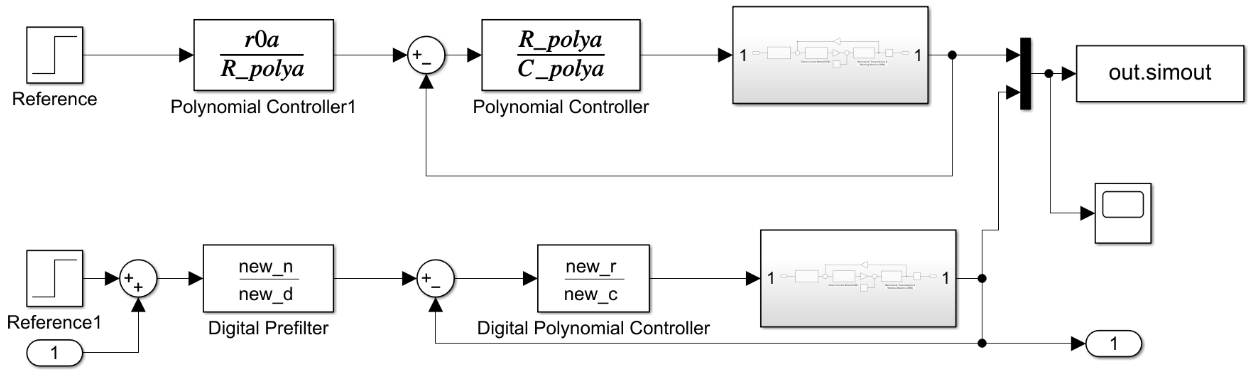
## Non-Static Digital Polynomial Controller

So that the controller can eliminate the static error that caused by the perturbation, we will also synthesize the non-static digital polynomial controller.

For this synthesize, we won't recalculate the polynomial like done previously, but we will do the Z-Transformation to the non-static polynomial controller that already been synthesize in Task-3.

For this transformation we will using Tustin approximation method. Note that for approximation method, a small sample time is needed. Therefore, we will use $T_q = 1e - 3$.

The block that will be change to digital is the polynomial controller $\frac{R(s)}{C(s)}$ and the prefilter block $\frac{\text{r0a}}{R(s)}$.



Let's transform the polynomial controller $\frac{R(s)}{C(s)}$ using Tustin approximation.

```
Tqd = 1e-3;
new_ = c2d(tf(R_polya,C_polya),Tqd,'tustin');
[new_r, new_c] = tfdata(new_);
new_r = new_r{:}
```

```
new_r = 1×4
10^6 ×
    0.8187   -2.3876    2.3218   -0.7529
```

```
new_c = new_c{:}
```

```
new_c = 1×4
    1.0000   -2.4233    1.9644   -0.5411
```

Let's also transform the prefilter using Tustin approximation.

```
pref_new = c2d(tf(r0a,R_polya),Tqd,'tustin');
[new_n, new_d] = tfdata(pref_new);
new_n = new_n{:}
```

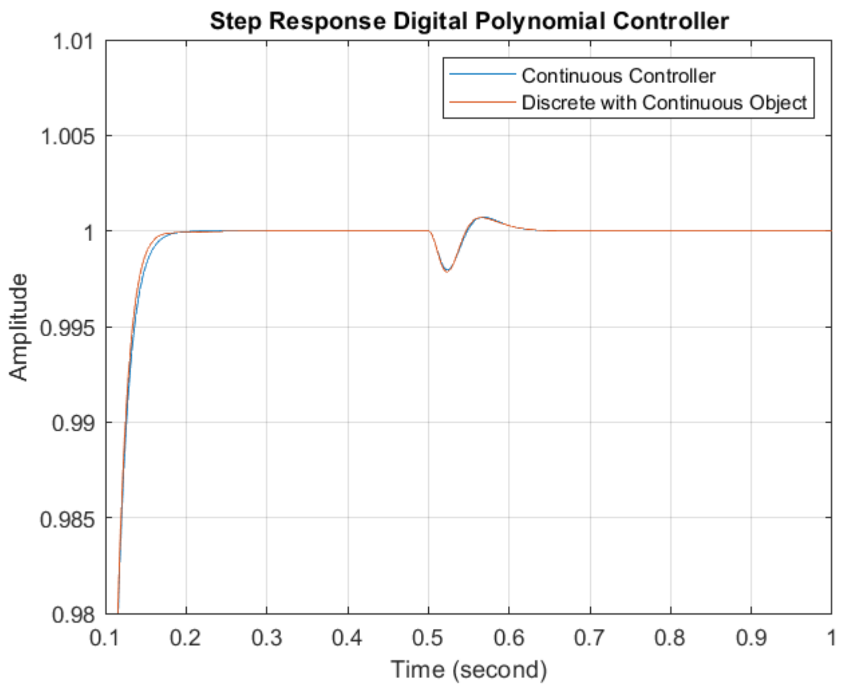
```
new_n = 1×4
10^-4 ×
    0.0735    0.2204    0.2204    0.0735
```

```
new_d = new_d{:}
```

```
new_d = 1×4
     1.0000   -2.9162    2.8359   -0.9196
```

Let's run the simulation

```
STime = 1;
out_sdo = sim('Astatic_DigitalPolynomial');
plot(out_sdo.tout, out_sdo.simout(1:end,1), out_sdo.tout, out_sdo.simout(1:end,2));
grid on;
title('Step Response Digital Polynomial Controller')
xlabel('Time (second)')
ylabel('Amplitude')
ylim([0.98,1.01]);
xlim([0.1,1])
legend('Continuous Controller', 'Discrete with Continuous Object')
```



We can conclude that the controller works, it gives stable and fast response without overshoot or undershoot, and behave like technical condition required, even after we added the perturbation.

## Task - 7: Conclusion

This report shows the synthesize of different type of controller that can be applied to the object. All the controller has been synthesized and able to fulfil the technical requirement. In the case of system with perturbation, we also synthesize the non-static type of each controller, with non-static type, the static error caused by the perturbation can also be eliminated.

The complete Simulink models as well as this document in form of live script can be found in this following repository:

https://github.com/AbiyyuMufti/ElementControlTheory.git