1. **系统建模**（System Modeling）

DC Motor Speed一节的输入和输出分别为电压和转轴转速，现要求系统的输入仍为电压，但输出变为转轴转角。

1.1> 传递函数

由上节所得传递函数

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js+b)(Ls+R)+K^2} \qquad [\frac{rad/sec}{V}]$$

函数两边同除s，得到

$$\frac{\Theta(s)}{V(s)} = \frac{K}{s((Js+b)(Ls+R)+K^2)} \qquad [\frac{rad}{V}]$$

确定系统参数及Matlab建模

```
>> J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2))
```

1.2> 状态空间方程

选择转轴转角、转速、电枢电流为状态变量，输出为转轴转角，可得状态空间方程

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & -\frac{b}{J} & \frac{K}{J}\\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{1}{L}\end{bmatrix} V$$

$$y = \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}$$

Matlab

```
>> motor_ss = ss(P_motor)
```

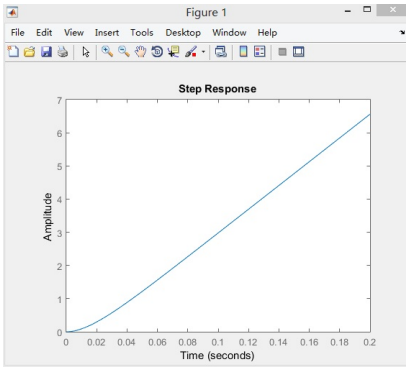
最后确定一下控制目标，对于系统的单位阶跃响应，输出应满足：

- 调节时间 Ts < 40ms
- 超调量 %OS < 16%
- 无稳态误差，即使有单位干扰输入

2. **系统分析**（System Analysis）

2.1> 开环响应

```
>> t = 0:0.001:0.2;
step(P_motor,t)
```



从图线中可以看出，当1v电压作用于系统时，转角值是无界的。开环响应是不稳定的

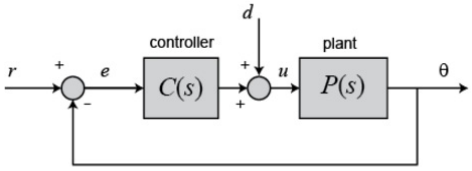
```
>> pole(P_motor)

ans =

   1.0e+06 *

         0
   -1.4545
   -0.0001
```

除了 pole 函数得到函数极点判断系统是否稳定外，Matlab提供了专门的函数 **issatable** 来判断系统的稳定性

```
>> isstable(P_motor)

ans =

     0
```

返回值为0表示系统不稳定，1则系统稳定。

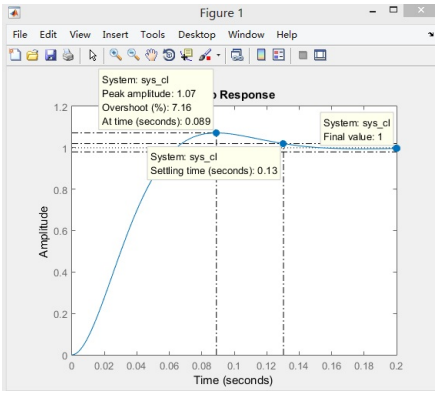
2.1> 闭环响应



先仅令C(s) = 1，来观察原开环系统闭环后的响应

方法一

```
>> sys_cl = feedback(P_motor,1);
>> step(sys_cl,t)
```

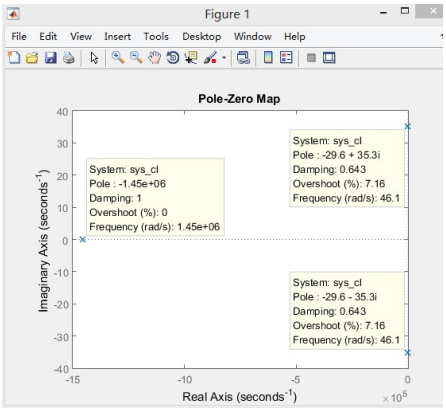


观察系统响应，反馈使得系统变得稳定，且响应满足要求。

方法二

传递函数的极点位置不仅可以反映系统的稳定性，还可以得到阶跃响应的特征。因此系统的响应特征除了阶跃响应图，也可以由极点得到，观察系统极点

```
>> pzmap(sys_cl)
```



如图，点击极点就可以获得相应响应特征。

方法三

这些特征也可以由Matlab提供的函数 **damp()** 来得到

```
>> damp(sys_cl)

        Pole              Damping       Frequency      Time Constant
                                      (rad/seconds)      (seconds)

 -2.96e+01 + 3.53e+01i    6.43e-01      4.61e+01         3.38e-02
 -2.96e+01 - 3.53e+01i    6.43e-01      4.61e+01         3.38e-02
 -1.45e+06                1.00e+00      1.45e+06         6.88e-07
```

负实根比共轭复根实数部分数值大得多，因此闭环系统的响应特征基本由共轭复根来决定，即zeta = 0.643, Wn = 46.1。然后就可计算得到超调量及2%的调节时间

```
>> [Wn,zeta,poles] = damp(sys_cl);
OS = exp((-zeta(1)*pi)/sqrt(1-zeta(1)^2))
Ts = 4/(zeta(1)*Wn(1))

OS =

    0.0716


Ts =

    0.1351
```

求得的数值与阶跃响应图像所得的相同，也证明了主极点对于系统响应的主导地位。

3. **PID控制器设计**（PID controller Design）

3.1> P控制器

分别观察Kp = 1,11,21时系统的响应

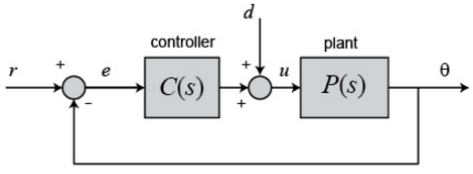
p.m

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2));
Kp = 1;
for i = 1:3
    C(:,:,i) = pid(Kp);
    Kp = Kp + 10;
end
sys_cl = feedback(C*P_motor,1);
```

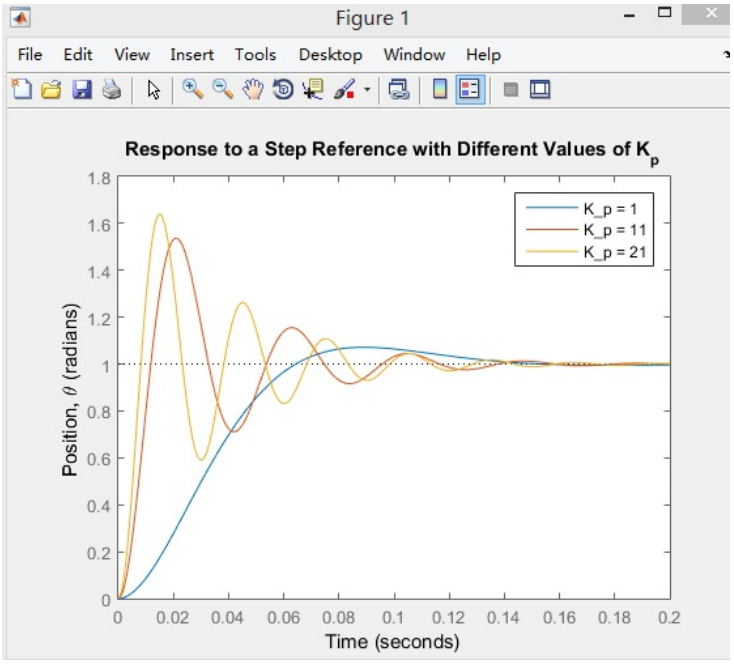
Command Window

```
>> p
>> t = 0:0.001:0.2;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with Different Values of K_p')
legend('K_p = 1', 'K_p = 11', 'K_p = 21')
```

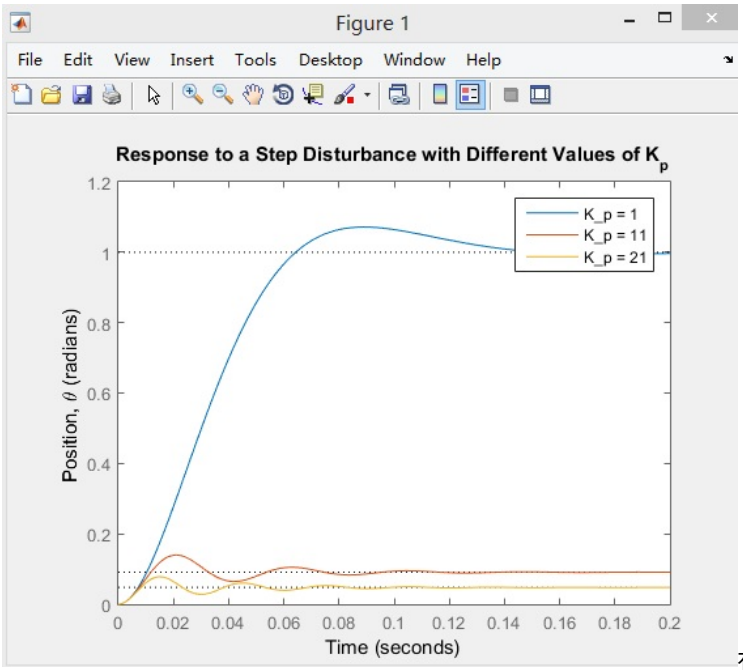
先假设输入为0，观察存在单位阶跃干扰时，系统的响应。有下面的框图看出，此时前向通道只有系统对象P(s)，而控制器C(s)到了反馈通道，且为负反馈。




```
>> dist_cl = feedback(P_motor,C);
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with Different Values of K_p')
legend('K_p = 1', 'K_p = 11','K_p = 21')
```



无干扰



有干扰

由响应可以看出，无干扰时稳态误差为0，因为系统为type 1，而当存在干扰时，系统开始出现稳态误差。当Kp越大时稳态误差越小，但不会减小到0，且随着Kp的增大，超调和调节时间会变大。我们需要积分环节来消除稳态温差，微分环节来减小超调和调节时间。

3.1> PI控制器

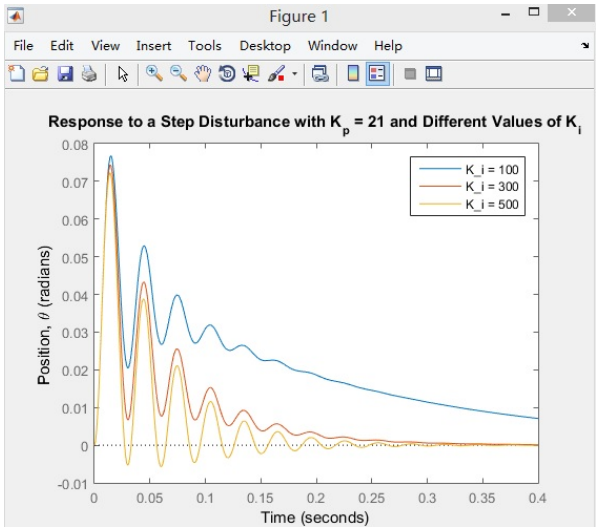
让Kp = 21, Ki = 100,300,500。 观察只存在单位阶跃干扰时的系统响应

pi.m

```
Kp = 21;
Ki = 100;
for i = 1:5
    C(:,:,i) = pid(Kp,Ki);
    Ki = Ki + 200;
end
dist_cl = feedback(P_motor,C);
t = 0:0.001:0.4;
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with K_p = 21 and Different Values of K_i')
legend('K_i = 100', 'K_i = 300', 'K_i = 500')
```

nmand Window

```
> pi
```



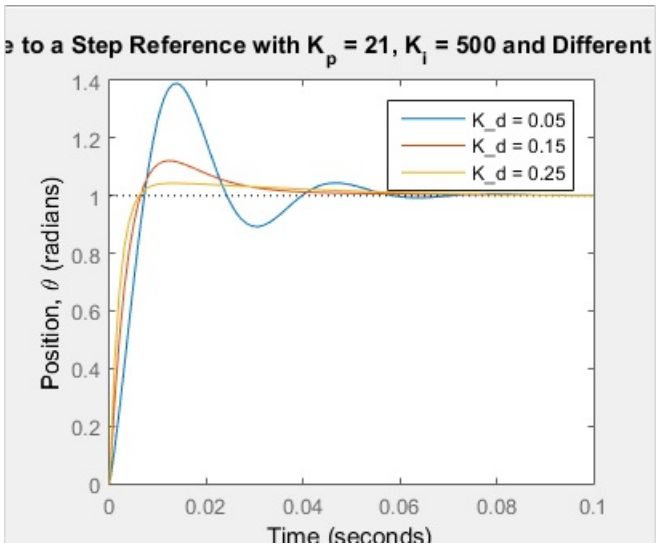
随着Ki的增大，系统越快速地消除稳态误差，取Ki = 500，然后使用微分环节来减小超调和调节时间

3.1> PID控制器

取Kd = 0.05,0.15,0.25

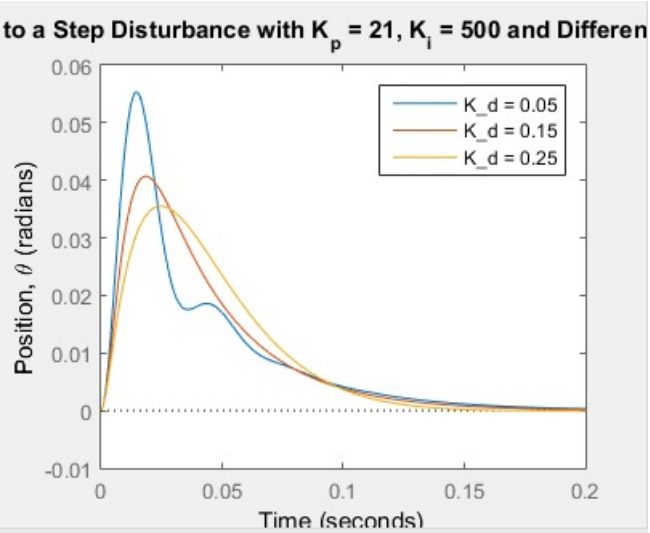
观察此时的系统响应

```
>> Kp = 21;
Ki = 500;
Kd = 0.05;

for i = 1:3
    C(:,:,i) = pid(Kp,Ki,Kd);
    Kd = Kd + 0.1;
end

sys_cl = feedback(C*P_motor,1);
t = 0:0.001:0.1;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with K_p = 21, K_i = 500 and Different Values of K_d')
legend('K_d = 0.05', 'K_d = 0.15', 'K_d = 0.25')
```



观察只存在单位阶跃干扰时的系统响应

```
>> Kp = 21;
Ki = 500;
Kd = 0.05;

for i = 1:3
    C(:,:,i) = pid(Kp,Ki,Kd);
    Kd = Kd + 0.1;
end
>> dist_cl = feedback(P_motor,C);
t = 0:0.001:0.2;
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with K_p = 21, K_i = 500 and Different values of K_d')
legend('K_d = 0.05', 'K_d = 0.15', 'K_d = 0.25')
```



Kd = 0.15近乎满足要求，可以右击图像显示特征，或使用 **stepinfo** 命令，得到所有特征数值

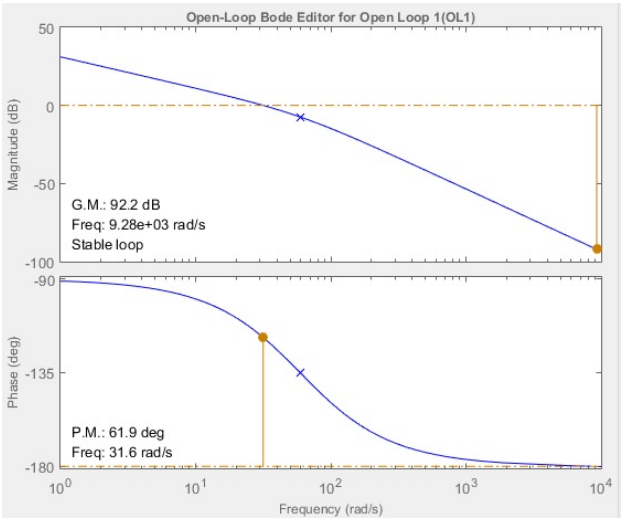
```
>> stepinfo(sys_cl(:,:,2))

ans =

        RiseTime: 0.0046
    SettlingTime: 0.0338
     SettlingMin: 0.9103
     SettlingMax: 1.1212
       Overshoot: 12.1175
      Undershoot: 0
            Peak: 1.1212
        PeakTime: 0.0122
```

Ts = 34ms < 40ms, OS% = 12% < 16%, 无稳态误差，即使有单位干扰输入。满足所有要求，所以确定PID控制器，Kp = 21,Ki = 500,Kp = 0.15。

4. **控制器设计中的频率法**（Frequency Methods for Controller Design）

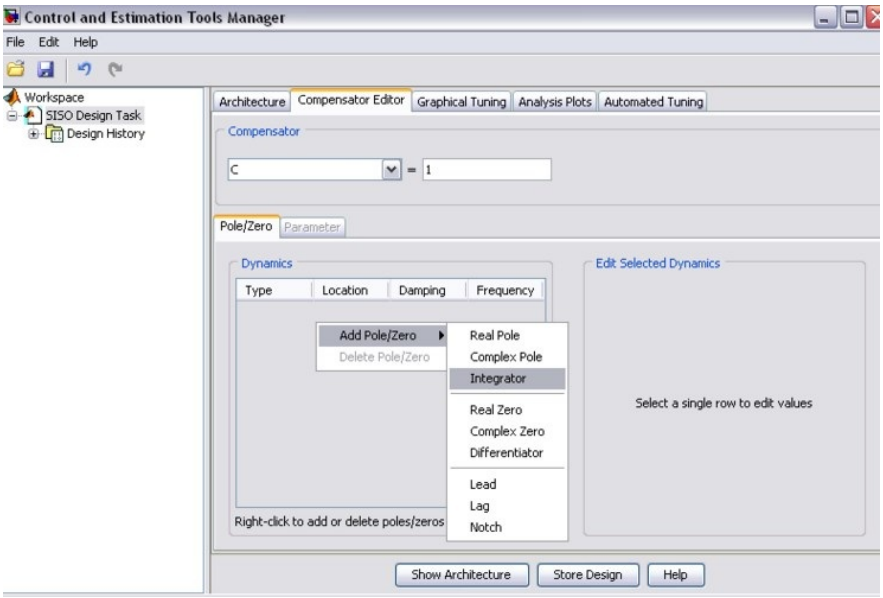
频率法的主要思想是根据开环传递函数的bode如来估计闭环时的响应。使用SISO Design Controller来设计控制器

```
1.  sisotool('bode',P_motor)
```

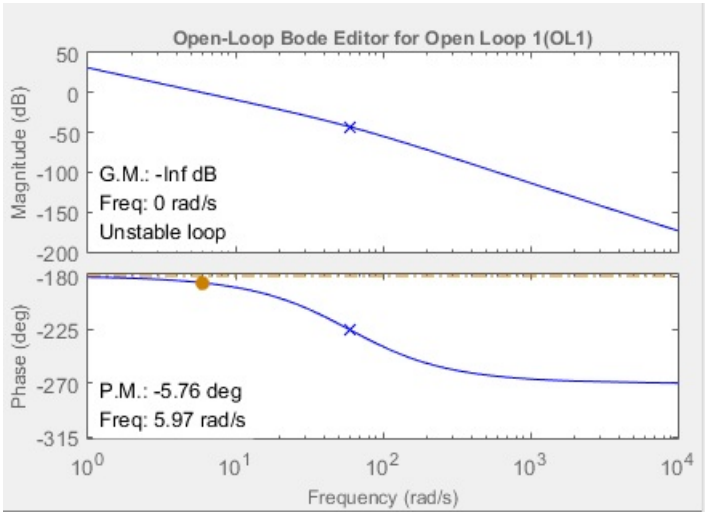


系统稳定，但对应的单位闭环响应并不能满足系统要求（怎么看出？？），因此需要添加补偿器来获得理想的响应

4.1> 添加积分器



对应bode图已发生改变

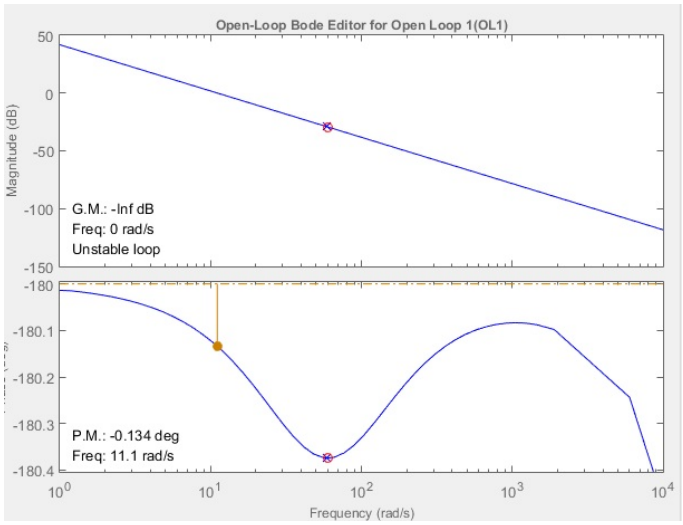


4.2> 规定增益裕度与相位裕度

（接上）要想得到16%的超调和40ms的调节时间，使用PM = 100*zeta的近似估计以及二阶系统时的关系式来估计此系统达到目标所需的增益裕度与相位裕度

```
>> zeta = -log(.16) / sqrt( pi^2 + (log(.16))^2 );
PM = 100*zeta
Wbw = (4/(0.04*zeta))*sqrt((1 - 2*zeta^2) + sqrt(4*zeta^4 - 4*zeta^2+2))

PM =

   50.3868


Wbw =

  251.5743
```

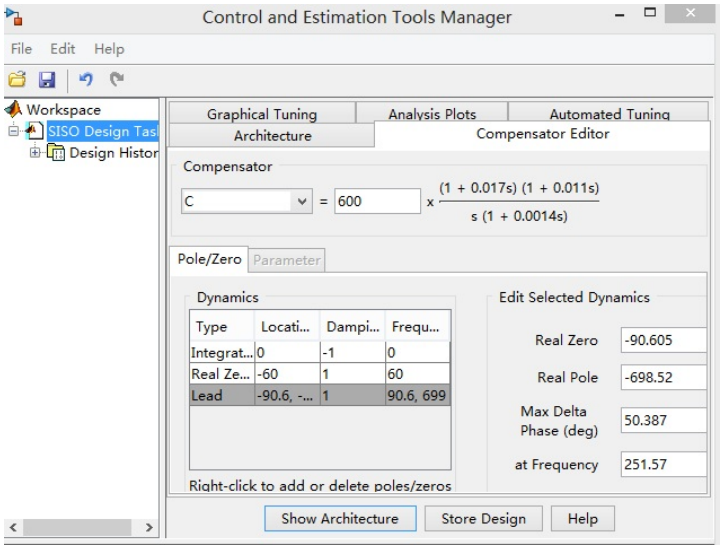
由结果可知我们需要有50deg左右的相位裕度以及250 rad/s左右的闭环频带宽，可以通过开环bode图上的-6dB~-7.5dB幅值对应的频率来估计闭环频带宽。同时在bode图上可以看到在60rad/s左右有一个极点，因此我们添加一个零点使得相位曲线变得平坦，添加零点的方法同上。

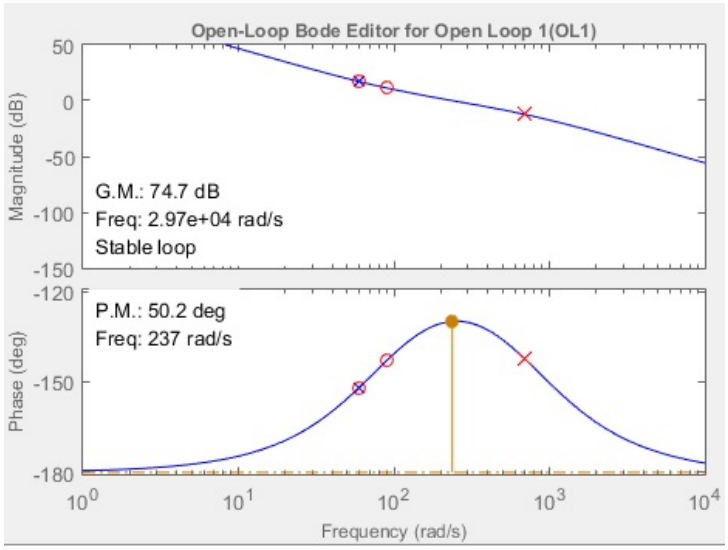


此时需要增加50deg相位来使得250rad/s对应于-6dB~-7.5dB。增加一个前向补偿器（Lead Compensator）来实现目的，前向补偿器的零极点对应有下列方程求出

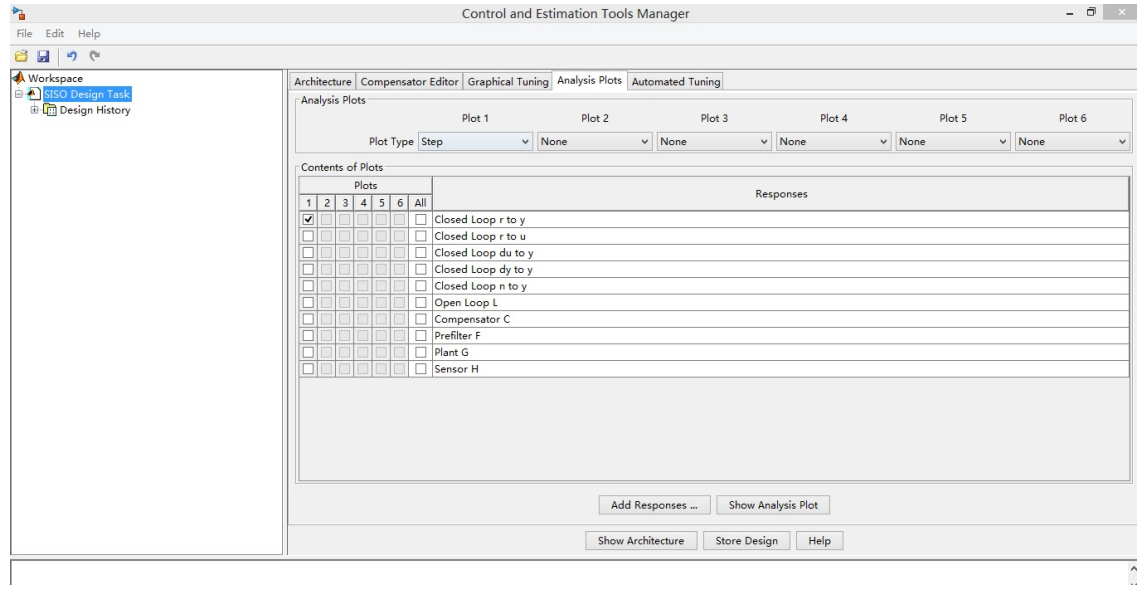
```
>> a = (1 - sin(PM*pi/180))/(1 + sin(PM*pi/180));
I = 1/(Wbw*sqrt(a));
zero = -1/I
pole = -1/(a*I)

zero =

  -90.6050


pole =

 -698.5222
```

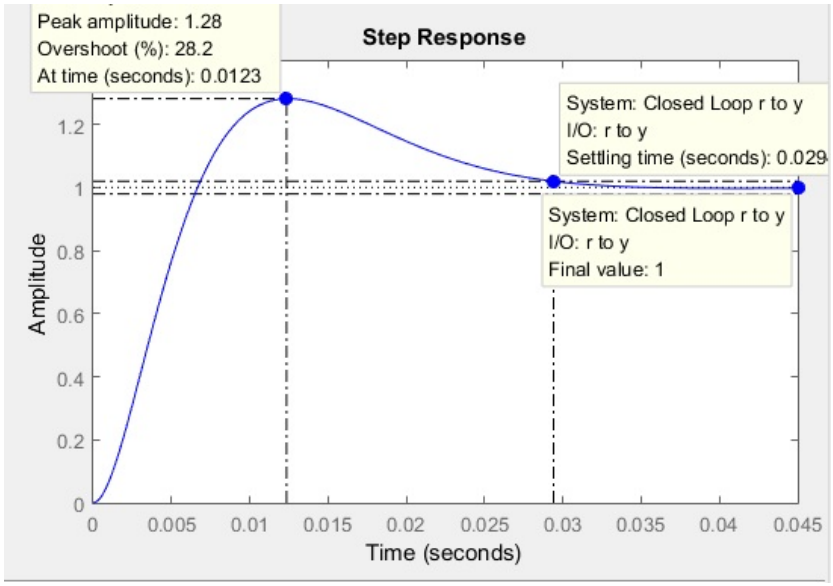
同样的方法添加零极点，我们还需增大传递函数增益为600



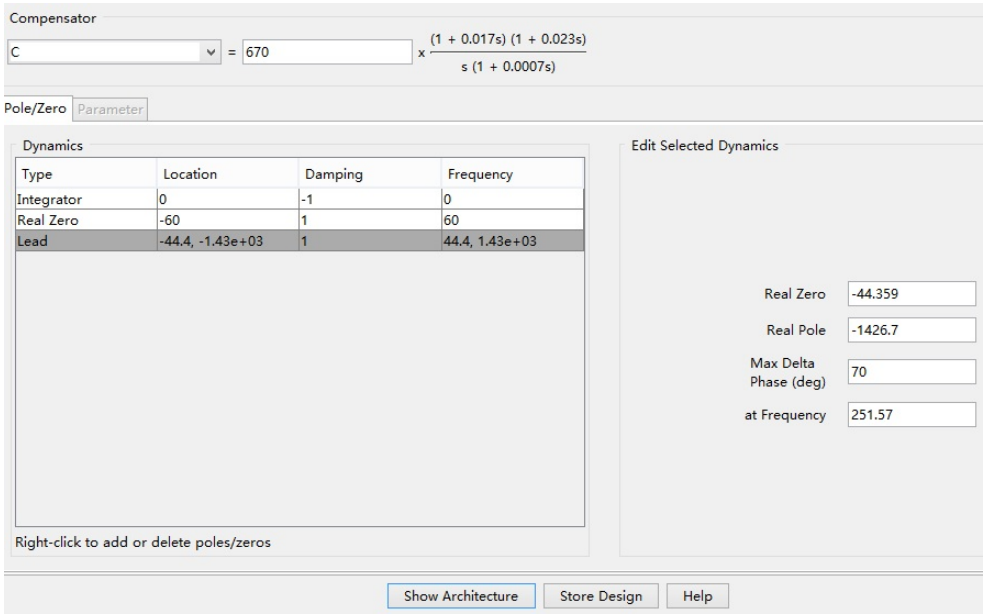


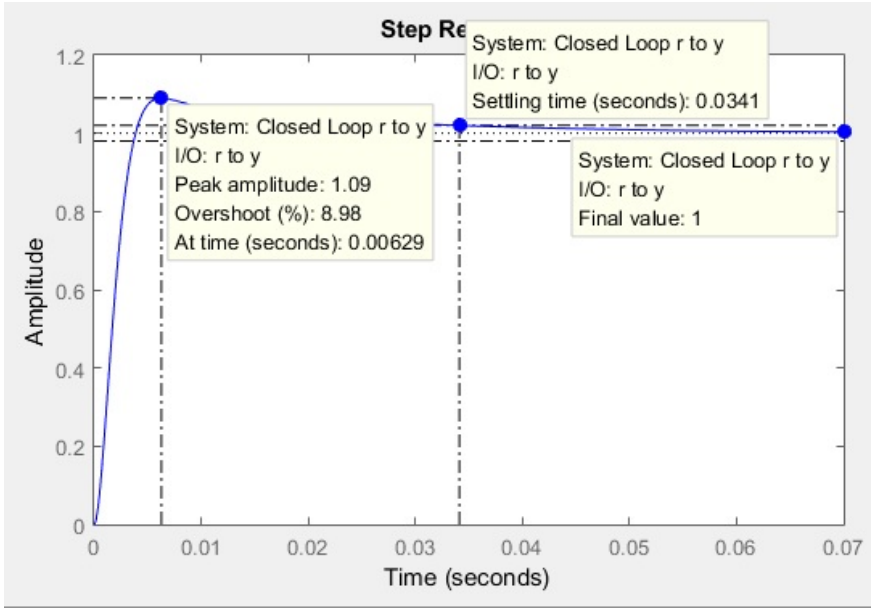
现在来检验闭环系统的阶跃响应





可以看出系统的超调太大了，之所以这样是因为系统并非二阶系统，但已经提供了很好的依据在此基础上调整控制器以满足要求。





改变前向补偿器Max Delta Phase = 70（增大此值，相位曲线会凸起（bump），超调减小），C = 670时发现系统响应满足要求。

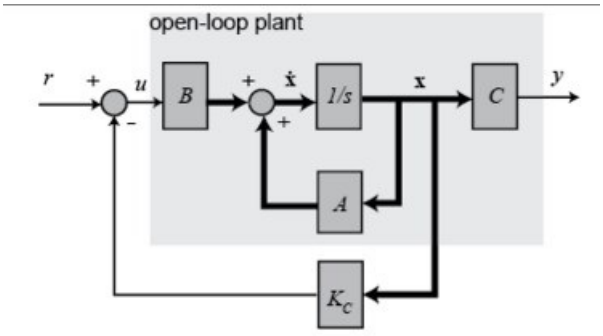
5. **控制器设计中的状态空间法**（State-Space Methods for Controller Design）

创建模型

```
>> J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;

A = [0 1 0
    0 -b/J K/J
    0 -K/L -R/L];
B = [0 ; 0 ; 1/L];
C = [1  0  0];
D = 0;
motor_ss = ss(A,B,C,D);
```

控制框图

状态参数都好测量，使用不含观察器的控制器



$$\dot{\mathbf{x}} = (A - BK_c)\mathbf{x} + Br$$

$$y = C\mathbf{x}$$

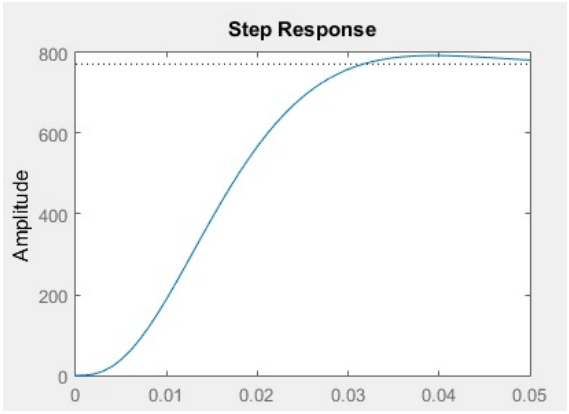
检查系统极点及是否可控

```
>> sys_order = order(motor_ss)
determinant = det(ctrb(A,B))

sys_order =

     3


determinant =

  -3.4636e+24
```

系统有三个极点，行列式不为0，表示可控。

由此我们可以任意设置系统的闭环极点，取 -200，-100+100i，-100-100i，因为此时可以忽略掉第一个极点，而后两个极点对应于二阶系统的zeta = 0.5（超调为16%），sigma = 100（调整时间为0.04s）。

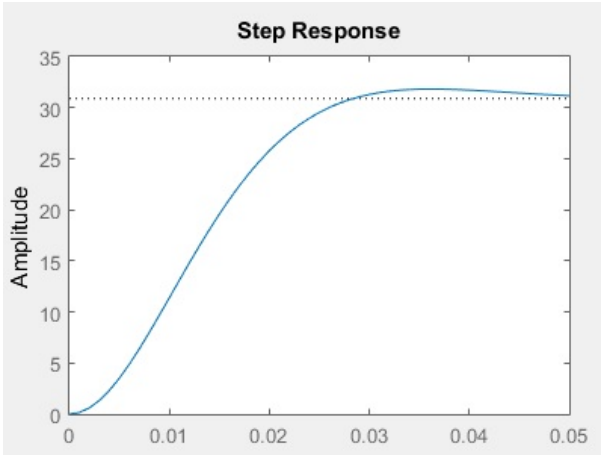
```
>> p1 = -100+100i;
p2 = -100-100i;
p3 = -200;
Kc = place(A,B,[p1, p2, p3])

Kc =

    0.0013   -0.0274   -3.9989

>> t = 0:0.001:0.05;
sys_cl = ss(A-B*Kc,B,C,D);
step(sys_cl,t)
```



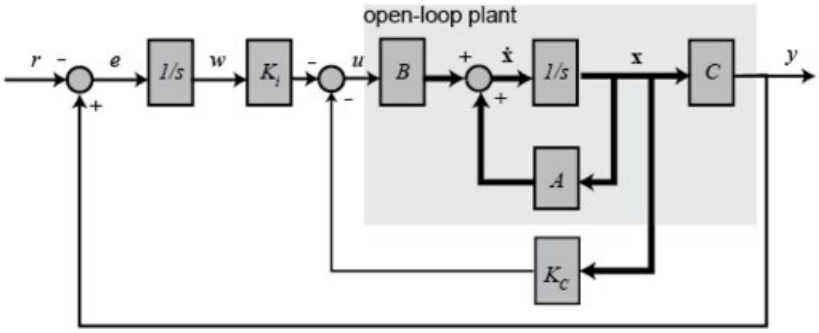
可见不满足要求，稳态误差太大了。

再观察干扰影响，假设输入只有阶跃干扰。在状态空间方程中必须提供合适的输入来表征干扰，在物理上干扰是一个作用于点击枢纽上的负载，在B的第二项添加1/J来表征。

```
>> dist_cl = ss(A-B*Kc,[0; 1/J ; 0], C, D);
step(dist_cl,t)
```



存在很大的误差，需要添加补偿器。



如图，添加了积分环节来消除稳态误差。由于增加积分环节，多了一个系统变量，用w来表示，dw/dt = e = r - y，重写原状态空间方程

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\\ w\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -\frac{b}{J} & \frac{K}{J} & 0\\ 0 & -\frac{K}{L} & -\frac{R}{L} & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\\ w\end{bmatrix}$$

$$+\begin{bmatrix}0\\ 0\\ \frac{1}{L}\\ 0\end{bmatrix}V+\begin{bmatrix}0\\ 0\\ 0\\ -1\end{bmatrix}r$$

$$y=\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\\ w\end{bmatrix}$$

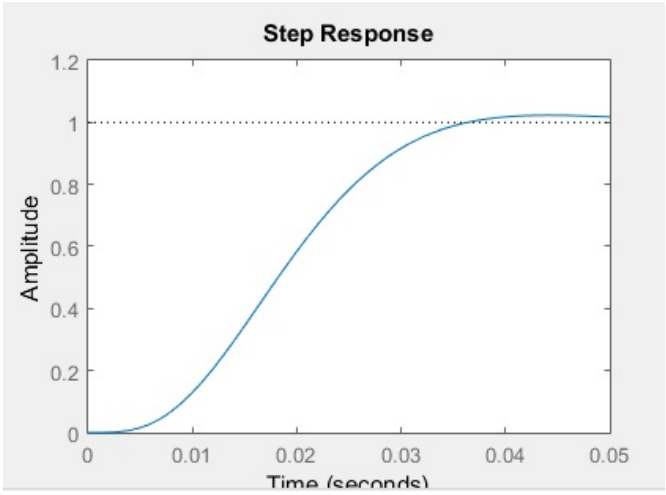
如上框图代表的状态空间方程变为

$$\dot{\mathbf{x}}_a = (A_a - B_a K_a)\mathbf{x}_a + B_r r$$

$$y = C_a \mathbf{x}_a$$

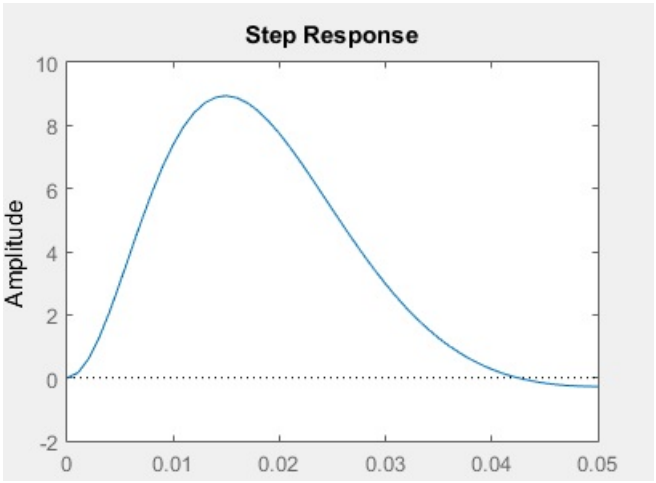
系统增加了因积分产生极点，令其等于-300以使其快速衰减而不影响系统响应

```
>> Aa = [0 1 0 0
      0 -b/J K/J 0
      0 -K/L -R/L 0
      1 0 0 0];
Ba = [0 ; 0 ; 1/L ; 0 ];
Br = [0 ; 0 ; 0; -1];
Ca = [1 0 0 0];
Da = [0];

p4 = -300;
Ka = place(Aa,Ba,[p1,p2,p3,p4]);

t = 0:0.001:.05;
sys_cl = ss(Aa-Ba*Ka,Br,Ca,Da);
step(sys_cl,t)
```



稳态误差为0，同样观察干扰响应

```
>> dist_cl = ss(Aa-Ba*Ka,[0 ; 1/J ; 0; 0],Ca,Da);
step(dist_cl,t)
```



稳态误差为0。

6.**Simulink建模**（Simulink Modeling）

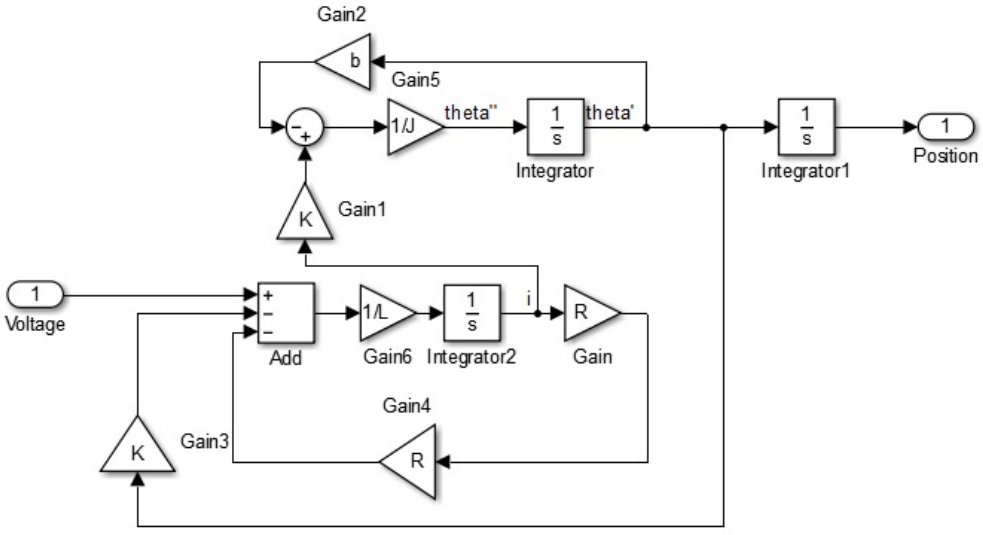
$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

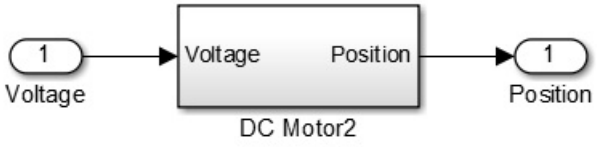
Simulink模型仍由积分环节开始建起

$$\int \int \frac{d^2\theta}{dt^2}\, dt = \int \frac{d\theta}{dt}\, dt = \theta$$

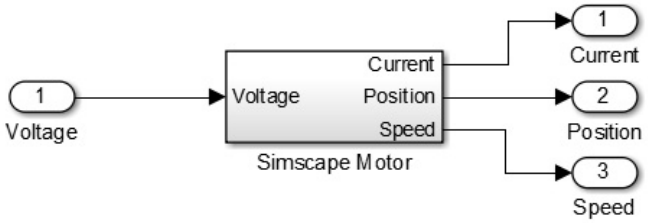
$$\int \frac{di}{dt}\, dt = i$$



封装起来



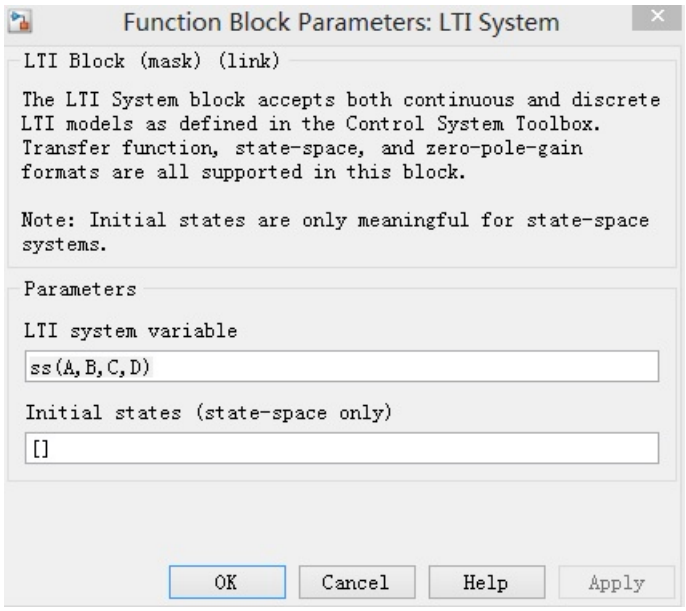
与DC Motor Speed中一样，也可以建立Simscape模型，封装后为

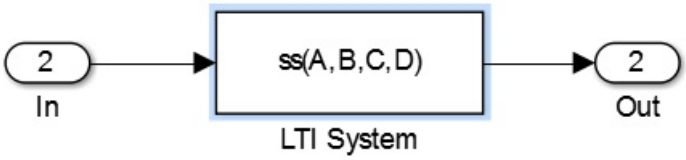


还可以将建立在Matlab中的模型导入到Simulink里，

```
>> A = [0 1 0
     0 -b/J K/J
     0 -K/L -R/L];
B = [0 ; 0 ; 1/L];
C = [1 0 0];
D = 0;
```

打开Simulink，在Control System Toolbox中找到LTI system，添加到新建Simulink模型窗口，双击建模，然后封装



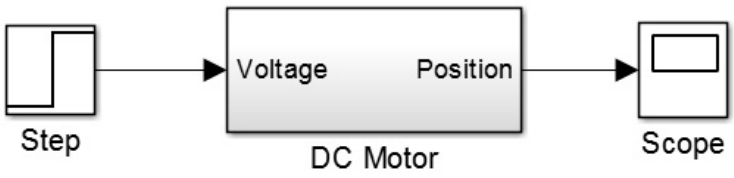


所建的三种模型功能是一样的，即同样的输入对应于同样的输出，不同的是其与用户的交互界面以及所呈现的细节。

7.**Simulink控制**（Simulink Control）

7.1> 开环响应

先考察开环阶跃响应

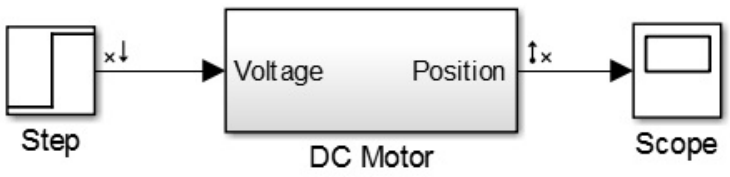


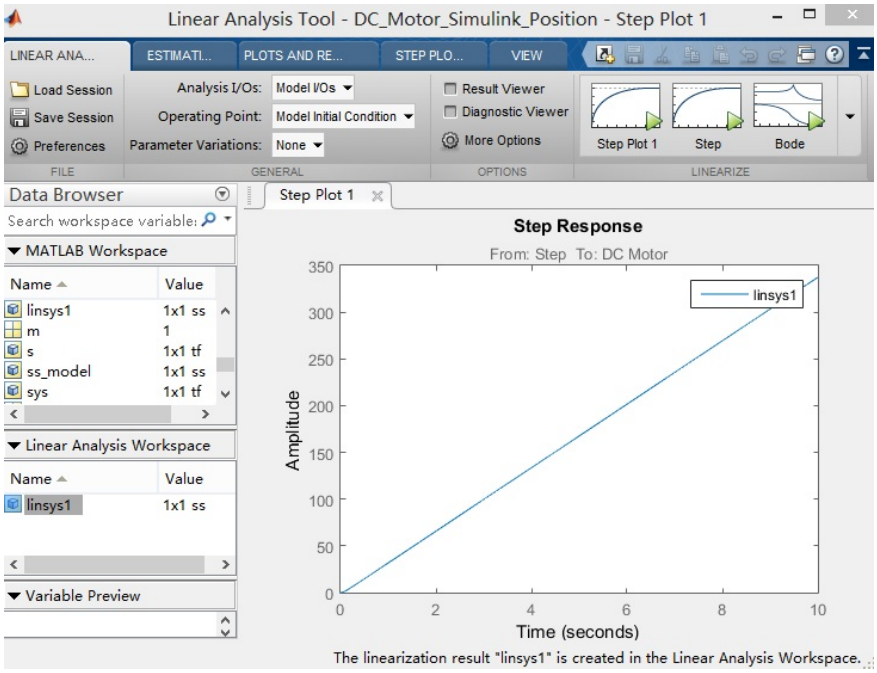
Step Block的Step time改为0，模型运行时间改为（Simulation->Modeling Configuration Parameters）0.2s

运行发现开环不稳定且明显不满足要求。

抽取线性模型到Matlab

步骤如 Introduction 所述，

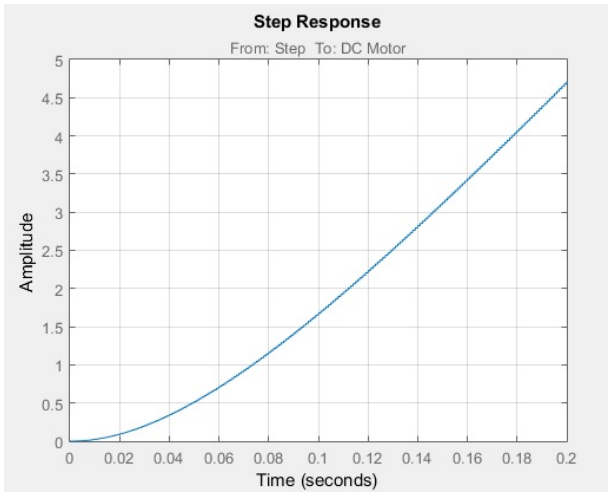




将 linsys1拖入Matlab Workspace后，在Matlab命令空间输入

```
>>   t = 0:0.001:0.2;
step(linsys1,t);
grid
```

得到此开环系统的闭环阶跃响应



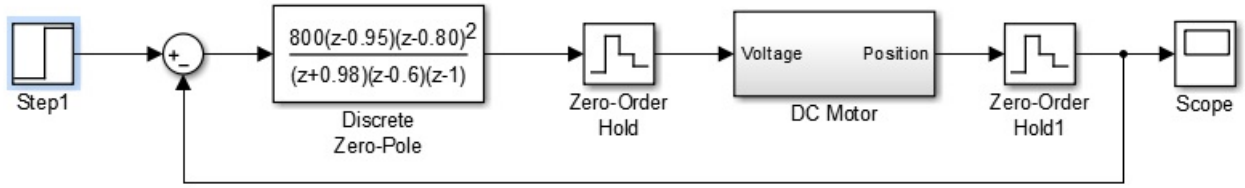
到Matlab空间后若前边内容可以利用不同的分析和控制器设计方法来满足要求。下边我们直接来利用得到的Simulink来进行数字化控制器（digital controller）设计。

7.2> Simulink中的数字化控制

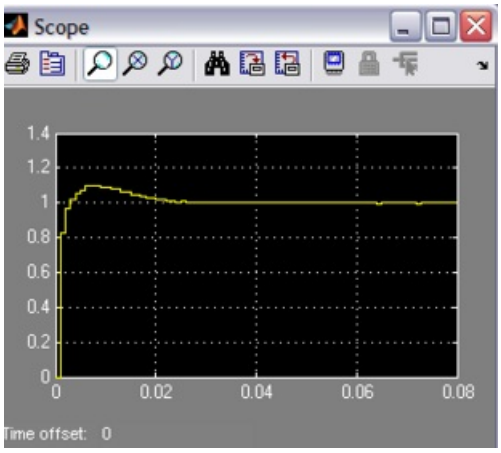
已知数字化控制器的传递函数为

$$C(z) = 800\frac{(z-0.95)(z-0.80)^2}{(z+0.98)(z-0.6)(z-1)}$$

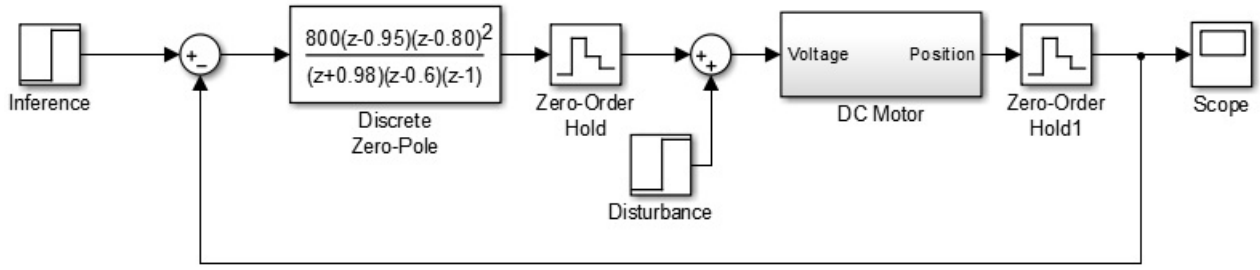
在Discrete Library中取Zero-Order Hold（将离散信号转化为阶跃形式的连续信号）与Discrete Zero-Pole（建立zpk形式的离散传递函数），设计如下Simulink模型



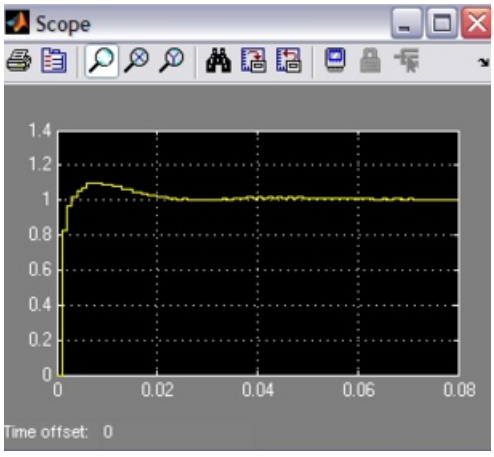
Step Block的Step time改为0，Discrete Zero-Pole及两个Zero-Order Hold的Sample time改为0.001（默认值为1，远远大于仿真时间，导致无法观察响应），仿真时间改为0.08。运行，



观察响应满足要求。再添加阶跃干扰



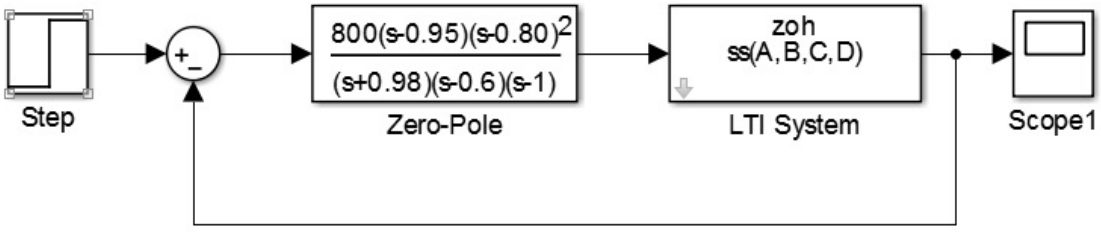
Step time改为0.03，运行后



观察响应，在0.03s有一点波动，但很快，系统又恢复稳定，稳态误差为0。

7.3> 利用Simulink离散连续函数模型

利用之前用Control System Toolbox建立的模型分析，先将模型线性化，移除Input Block与Output Block在Simulink编辑界面选择 Analysis -> Control Design -> Model Discretizer打开模型离散器，设置采样时间为0.001s，Transform method采取默认设置 Zero-order hold，选择Discretize进行模型离散，离散后建立闭环控制



点击运行，所得响应曲线与7.2>相同，阶跃干扰加上后也与上相同。