目录

1. **建立数学模型**（System Modeling）

分析、控制动态系统的第一步是由物理定律（如力学中的牛顿定律和电学中的Kirchoff's current law (KCL)、Kirchoff's voltage law (KVL)）或实验数据建立数学方程，大多数的动态系统可以用下式表达：

x' = dx/dt = f(x(t), u(t), t))

其中，x(t)是状态向量，系统的阶数n为状态向量的维数，通常，n等于系统中独立储能元素的个数；u(t)是控制输入向量

为了使其易于处理，通常有两个假设：

（1）**时不变系统**：函数关系f不随着时间t变化，即 x' = f (x,u)。 物理定律不随着时间而变化，即方程中的系数不会变化，这是此假设成立的依据。

（2）**线性系统**：将动态系统进行线性化近似，即 x' = Ax + Bu。 事实上几乎所有的系统都是非线性的，但在小范围内，我们经常能够进行线性化处理。

当系统满足这两个假设时，我们称其为**线性时不变系统**（LTI，linear time invariant)，而这构成了经典控制理论的主要内容。本教程中的例子都由线性常系数微分方程进行建模，因此为LTI，进一步表示为状态空间模型（时域）和传递函数模型（频域）进行分析计算

状态空间方程（State-Space Pepresentation)

x' = Ax + Bu （状态方程）

y = Cx + Du （输出方程）

其中，x是状态向量，y是输出向量，A是系统矩阵，B是输入矩阵，C是输出矩阵，D是反馈矩阵

状态空间方程为系统在时域上的表示形式，能够方便地处理初值非零的多输入多输出（MIMO,multi-input/multi-output)系统，甚至非线性系统，因此广泛用于现代控制理论。

在Matlab中使用 **ss(A,B,C,D)** 建立状态空间方程表示的数学模型

传递函数（Transfer Function Peresentation)

LTI有一个重要的性质就是当输入为正弦信号时，则输出也为同频率的正弦信号，而信号的幅值、相位会发生改变。幅值、相位差可以表示为频率的函数，称为系统的频率响应函数。

而拉普拉斯变换可以直接把时域的函数 f(t) 转换到频域F(s)。频率响应函数常用来处理能用常系数微分方程表示且初始值为0的单输入单输出（SISO，single-input/single-output）系统：

$$a_n\frac{d^n y}{dt^n}+...+a_1\frac{dy}{dt}+a_0 y(t)=b_m\frac{d^m u}{dt^m}+...+b_1\frac{du}{dt}+b_0 u(t)$$

初值为0，拉普拉斯变换后有：

$$a_n s^n Y(s)+...+a_1 sY(s)+a_0 Y(s)=b_m s^m U(s)+...+b_1 sU(s)+b_0 U(s)$$

合并同类项整理后有：

$$G(s)=\frac{Y(s)}{U(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+...+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+...+a_1 s+a_0}$$

表示成零极点形式有：

$$G(s)=\frac{N(s)}{D(s)}=K\frac{(s-z_1)(s-z_2)...(s-z_{m-1})(s-z_m)}{(s-p_1)(s-p_2)...(s-p_{n-1})(s-p_n)}$$

G(s)称为传递函数

在Matlab中申明s为传递函数变量 **s = tf('s')**, 然后直接输入传递函数表达式 或 使用 **tf(num,den)** 命令(num,den 为分子分母多项式的系数）建立传递函数表达的数学模型。

而零极点表示的传递函数申明函数为 **zpk(z,p,k)**

状态空间方程与传递函数之间的转换

$$G(s) = \frac{Y(s)}{U(s)} = \mathrm{C}(s\mathrm{I} - \mathrm{A})^{-1}\mathrm{B} + \mathrm{D}$$

Matlab：

Conversion between different models

Converting From	Converting to	Matlab function
Transfer Function	Zero-pole-gain	`[z,p,k]=tf2zp(num,den)`
Transfer Function	State Space	`[A,B,C,D]=tf2ss(num,den)`
Zero-pole-gain	Transfer Function	`[num,den]=zp2tf(z,p,k)`
Zero-pole-gain	State Space	`[A,B,C,D]=zp2ss(z,p,k)`
State Space	Transfer Function	`[num,den]=ss2tf(A,B,C,D)`
State Space	Zero-pole-gain	`[z,p,k]=ss2zp(A,B,C,D)`

上表均是不同模型形式的参数转化，设传递函数为tf_model，状态空间方程为ss_model，零点-极点-增益为zpk_model。模型间的直接转换有

```
tf_model = tf(sys)      //将其他类型的模型转换为多项式传递函数模型
```

```
ss_model = ss(sys)  //将其他类型的模型转换为状态空间模型
```

```
zpk_model = zpk(sys) //将其他类型的模型转换为零极点增益模型
```

由模型得到参数（G可以为任何一个形式表示的模型）

```
[A,B,C,D] = ssdata(G)
```

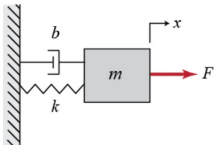
```
[z,p,k] = zpkdata(G,'v')
```

```
[num,den] = tfdata(G,'v')
```

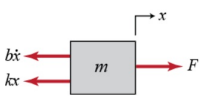
举例分析及Matlab求解：

1> 机械系统

弹簧阻尼质量快系统



受力分析如下



由牛顿定律建立动力学方程： F(t) = mx'' + bx' + kx

1.1> 状态空间方程

选择状态向量为 X = [x x']'，分别对应系统中的弹簧势能和质量块的动能。则有状态方程：

X' = [x' x'']' = [0 1; -k/m -b/m] X + [0, 1/m]' F(t)

如果我们关心的输出是质量块的位置则有输出方程：

Y = [1 0] X

Matlab:

```
m = 1;
k = 1;
```

```
b = 0.2;
F = 1;

A = [0 1; -k/m -b/m];
B = [0 1/m]';
C = [1 0];
D = [0];

sys = ss(A,B,C,D)
```

1.2> 传递函数

在初始值为0（质量块初始时刻处于平衡位置）时，进行拉普拉斯变换由

ms^2 X(s) + bs X(s) + k X(s) = F(s)

G(s) = X(s) / F(s) = 1 / (ms^2 + bs + k)

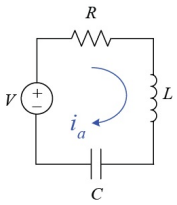
Matlab:

```
s = tf('s');
sys = 1/(m*s^2+b*s+k)
```

或

```
num = [1];
den = [m b k];
sys = tf(num,den)
```

2> 电学系统



由KVL建立电学方程：$V(t) - L\frac{di}{dt} - Ri - \frac{1}{C}\int idt = 0$

2.1> 状态空间方程

选择状态向量为 X = [q i]'，分别对应系统中的储能器件电容和电感。则有状态方程：

$$\dot{\mathbf{x}} = \begin{bmatrix} i \\ \frac{di}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{R}{L} & -\frac{1}{LC} \end{bmatrix} \begin{bmatrix} q \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V(t)$$

(系统矩阵A的二行一列和二列应该交换)

如果我们关心的输出是电路中的电流则有输出方程：

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

2.2> 传递函数

在初始值为0时，也可由所得的状态空间方程得到传递函数：

$$\frac{I(s)}{V(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} = \begin{bmatrix} 0 & 1 \end{bmatrix} \left(s \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}$$

$$\frac{I(s)}{V(s)} = \frac{s}{Ls^2 + Rs + \frac{1}{C}}$$

Matlab建模同上。

2. **分析系统**（System Analysis）

控制系统动态响应的一般要求是：提高稳定性及响应速度，减小、消除稳态误差，预防超调。得到状态空间方程或传递函数形式的数学模型后我们可以在时域和频域上分析系统的响应。时间响应是系统随时间所作出的响应，由暂态响应（取决于初始条件）与动态响应（取决于输入）两部分组成。尽管我们的数学模型是由微分方程得到的并且对微分方程直接积分可以得到一些简单系统的时间响应，但是对于绝大多数的动态系统却是数学上无解，幸运的是，Matlab提供了不同类型输入的动态响应处理方法。正如前边所说，频率响应会使得输入的正弦信号的幅值和相位改变，我们通过取不同的频率得到对应的传递函数取值来表示频率响应，G(jw)（G(s)为开环系统的传递函数，令s = jw）为复数，可以绘出对应的幅值和相位图（the Bode Plot）或直接在复平面上表示出来（the Nyquist Diagram），两种方法所反映的信息完全相同。

1> **稳定性**

BIBO（Bounded Input Bounded Output），即对于所有的有界输入所得的输出也是有界的。

频率响应研究系统的稳定性非常便利，如果传递函数的所有极点都在复平面的左面，则系统是稳定的；反之，有极点在复平面的右边或虚轴上，那么系统就是不稳定的。

我们直接可以通过 **pole(G)** 来求得传递函数G的极点，事实上系统矩阵的特征值极为系统的极点值，因此也可用 **eig(A)** 来求极点

2> **系统的阶数**

2.1> 一阶系统
一阶系统的微分方程形式为:

$y' + ay = bu$ 或 $ty' + y = ku$

传递函数形式为:

$G(s) = b/(s+a) = k/(ts + 1)$

t为时间常数，它代表阶跃响应下系统到达63%稳态值及脉冲响应下系统到达37%初始值经过的时间。当a为正时系统稳定。

2.2> 二阶系统
一阶系统的微分方程形式为:

$my'' + by' + ky = u$ 或 $y'' + 2\zeta w_n y' + w_n^2 y = k w_n^2 u$

传递函数形式为:

$G(s) = 1/(ms^2+bs+k) = kw_n^2/(s^2+2\zeta w_n s+w_n^2)$

超调量 $\%OS = e^{\left(\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}\right)} \cdot 100\%$

上升时间（Tr)、调节时间（Ts,Tolerance Fraction为+/-1%)、自然频率(Wn)、阻尼比（ζ ）、超调量（%OS)之间的关系为（Tr右式的系数2.2与后文给出的不同，可能有误）：

$$T_s \approx \frac{4.6}{\zeta\omega_n}$$

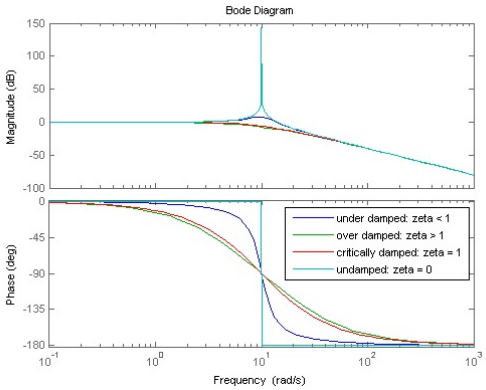
$$T_r \approx \frac{2.2}{\zeta\omega_n}$$

$$\zeta = \frac{-\ln(\%OS/100)}{\sqrt{\pi^2 + \ln(\%OS/100)^2}}$$

ζ < 1, 欠阻尼系统：极点为复数，实部为负，因此系统稳定，但当接近稳定值时会振荡；
ζ > 1,过阻尼系统：极点为两个负实数，因此系统稳定且不会出现振荡，但动态响应较慢；
ζ = 1,临界阻尼系统：极点为重负实数，系统稳定、动态响应最快且无振荡；
ζ = 0,无阻尼系统：极点为纯虚数，系统不稳定，会不断振荡。

3> **Matlab**:
阶跃响应 **step(G)**，Matlab提供了分析线性时不变的GUI（LTI Viewer），输入 **ltiview('step',G)**
伯德图 **bode(G)**， **ltiview('bode',G)**
系统的零极点图 **pzmap(G)**

```
 //G1、G2、G3、G4分别为欠阻尼、过阻尼、临界阻尼、无阻尼传递函数，设已经定义
bode(G1,G2,G3,G4)
legend('under damped: zeta < 1','over damped: zeta > 1','critically damped: zeta = 1','undamped: zeta = 0')
```



3. **PID控制器设计**（PID Controller Design）
PID是一种简单但应用极其广泛的误差补偿器，PID控制器的输出为

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

拉普拉斯变换后得到其传递函数

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

其中k_p为比例增益，k_i为积分增益，k_d为微分增益

在Matlab中可以通过前面构建普通传递函数的方法构建，也可以使用专门的函数 **C = pid(k_p,k_i,k_d)** ,然后 **tf(C)** 就可以得到如上分式表示的传递函数。设系统原传递函数为G，PID作用后的单位反馈的传递函数则为 **T = feedback(G*C,1)**
PID对于系统响应的影响

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

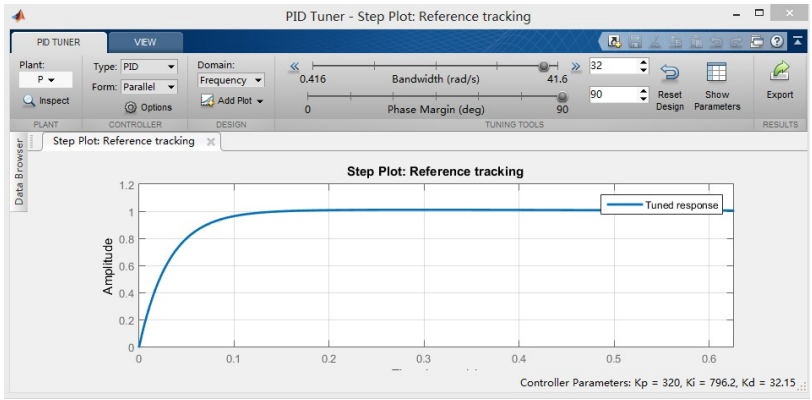
设计PID控制器的几点建议：
1> 先观察系统开环的动态响应，观察需要调节的特征，然后根据2> 3> 4>对应调节
2> 使用比例环节来减小上升时间，加快动态响应
3> 使用微分环节来减小超调
4> 使用积分环节来消除稳态温差
5> 综合调节P、I、D的参数值使得系统得到理想的动态响应
最后，注意不一定PID各环节都要使用，如果PI已经得到很理想的动态响应，那么，完全没有必要再增加微分环节，使系统越简单越好。
Matlab PID调节工具：直接使用函数 **pidtune** 或通过 **pidtool** 调用 GUI
例.现有一系统，其传递函数为1/(s^2 + 10*s + 20)，设计PID控制器使其带宽为32，相角裕度为90

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
opts = pidtuneOptions('CrossoverFrequency',32,'PhaseMargin',90);
[C, info] = pidtune(P, 'pid', opts)
```

```
C =

          1
  Kp + Ki * --- + Kd * s
          s

  with Kp = 320, Ki = 796, Kd = 32.2

Continuous-time PID controller in parallel form.


info =

              Stable: 1
    CrossoverFrequency: 32
          PhaseMargin: 90
```

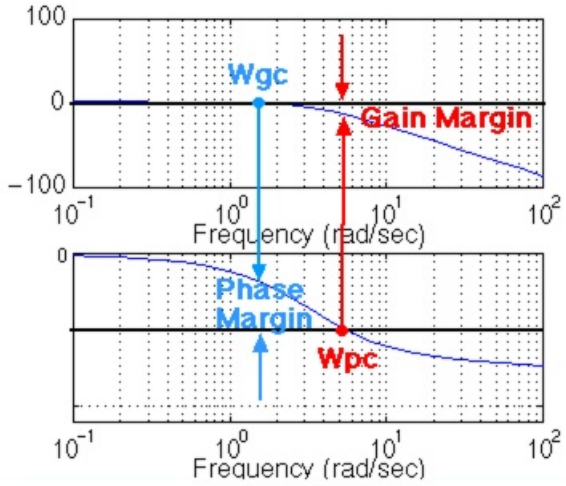
或使用GUI（注意，目标参数为频域里的参数，因此Domain选择Frequency，调节量变为目标量）

```
pidtool(P,'pid')
```



4. **根轨迹控制器设计**（Root Locus Controller Design）

5. **控制器设计中的频率分析法**（Frequency Methods for Controller Design）
尽管频率法并不直观，但其在现实系统如由物理实验数据所建的模型中很实用，如可以通过分析开环传递函数的频率响应可以预测闭环时的响应。
5.1> bode图
5.1.1> 增益裕度（Gain Margin）、相角裕度(Phase Margin)



(开环传递函数）Wpc所在那条线的角度是-180deg

margin(G) 得到标有裕度的伯德图

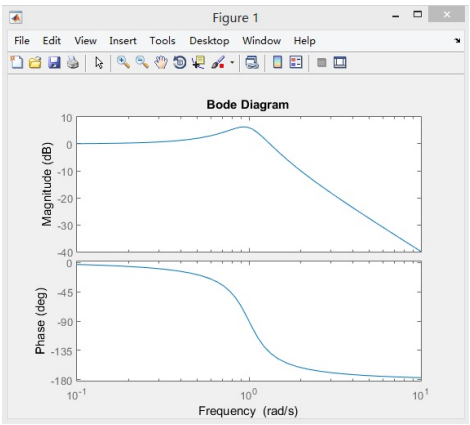
5.1.2> 带宽频（bandwidth frequency）

带宽频（bandwidth frequency)的定义是闭环系统响应为-3dB时对应的频率，在实际中更有用的是通过开环的响应来得到带宽频，理论表明，开环系统的幅值响应在-6dB ~ -7.5dB 对应的频域相位在-135deg ~ -225deg之间，则可用将频率视为带宽频。

带宽频是一个重要的参数，当输入正弦信号的频域小于带宽频时，系统响应会很好地跟踪输入信号；当大于带宽频时，会出现前边所说的频域、相位改变。

分析系统响应一个很有用的函数 **lsim(G,u,t，x0)** 系统对于输入信号u的响应（初始值为x0，缺省为0）

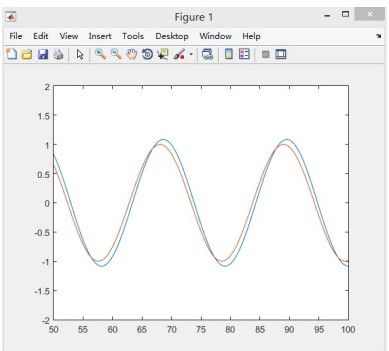
[例] 有闭环传递函数 G(s) = 1 / (s^2+0.5s+1), 观察其伯德图估计输入正弦信号分别为0.3rad/s和3rad/s时系统的响应



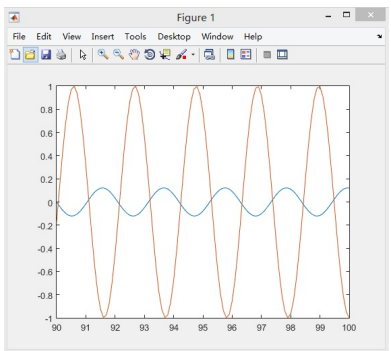
观察其bode图，-3dB对应的频率大概为1.4rad/s,即为带宽频。0.3rad/s < W_{bw}对应的幅值和相位都接近0，即输出幅值、相位与输入基本一致，3rad/s > W_{bw}对应的幅值相位分别接近-20dB（即为输入幅值的1/10）、-180^{o}（与输入信号相位相反），可以观察验证

```
>> s = tf('s');
G = 1/(s^2 + 0.5*s + 1);
w = 0.3;
t = 0:0.1:100;
u = sin(w*t);
[y, t] = lsim(G, u, t);
plot(t,y,t,u)
axis([50 100 -2 2]);
```

```
>>  s = tf('s');
G = 1/(s^2 + 0.5*s + 1);
w = 3;
t = 0:0.1:100;
u = sin(w*t);
[y, t] = lsim(G, u, t);
plot(t,y,t,u)
axis([90 100 -1 1]);
```



w = 0.3rad/s



w = 3rad/s

5.1.3> 由bode图进行闭环预测

由开环传递函数的bode图进行闭环响应预测需要搞清这么几点：

- 使用bode图时系统在开环时必须稳定；
- $W_{gc} < W_{pc}$ 时，系统闭环时将稳定；
- 对于二阶系统，当相位裕度在0~60deg时，闭环系统的阻尼比近似等于相位裕度除以100；

- 对于带宽频一个很粗略的估计是系统的自然频率（the natural frequency），常用于一阶系统，同时一阶系统的响应不会出现超调；
- 对于二阶系统，频带宽W_{bw}可由阻尼比、调节时间（或峰值时间）计算，运行wbw.m文件。

```
>> wbw
Damping Ratio?    0.2
Settling Time(0) or Peak Time(1)?   0
Settling Time?   2

Wbw =

   15.0958
```

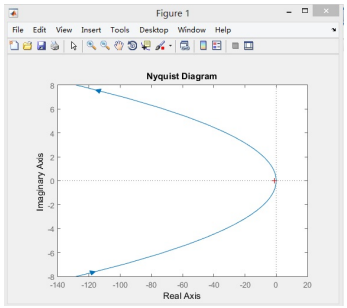
5.2> 奈奎斯特图(Nyquist Diagram)

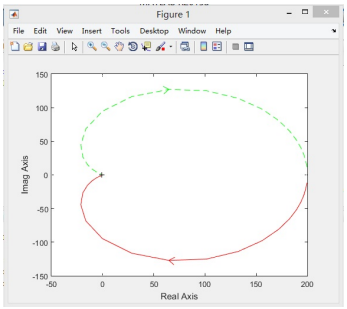
Nyquist图可以通过分析开环传递函数的频率响应可以预测闭环时的响应。Nyquist用来实现设计目标而不论系统开环是否稳定（于此相反，Bode图要求系统开环都是稳定的）。

Matlab中绘制Nyquist图的函数： **nyquist(G)** 当系统开环传递函数有虚数极点时，Matlab中提供的Nyquist图画法不能完全表示，可以执行此文件：

nyquist1.m

```
>> s = tf('s');
>> sys = (s+2)/(s^2);
>> nyquist(sys)
>> nyquist1(sys)
```



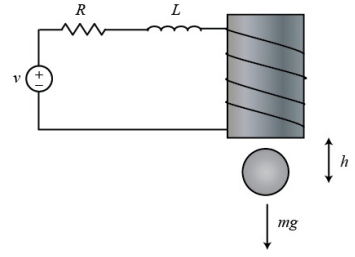


使用自带的nyquist命令，所得图形不正确　　使用nyquist1函数，图形正确

绘制出Nyquist图后可以通过劳斯判据（The Cauchy Criterion） 来判别系统是否稳定。

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

上节是在频域中分析系统的一些特征，状态空间法是在时域中分析、设计系统。下面通过一个例子来学习状态空间法设计控制器



6.1> 建立数学模型

由牛顿定律和KVL建立有

$$M\frac{d^2h}{dt^2} = Mg - \frac{Ki^2}{h}$$

$$V = L\frac{di}{dt} + iR$$

设各参数取值分别为：M=0.05kg,R=1Ω,L=0.01H,K=0.0001,g=9.81m/s^2,取h=0.01m（设此时电流为14A），在其附近线性化系统。

$$\frac{d\vec{x}}{dt} = A\vec{x} + Bu$$

$$y = C\vec{x} + Du$$

其中 x = [h; h'; i]，控制变量u为输入电压V，关心的输出为h，可以写出状态空间方程，从而确定A,B,C,D。

```
>> A = [0 1 0;980 0 -2.8;0 0 -100];
B = [0; 0; 100];
C = [1 0 0];
```

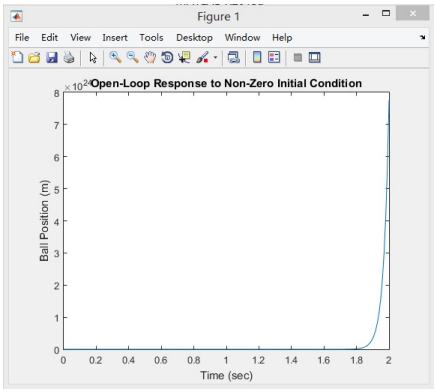
6.2> 分析数学模型

```
>> poles = eig(A)

poles =

   31.3050
  -31.3050
 -100.0000
```

有一正值，开环系统不稳定

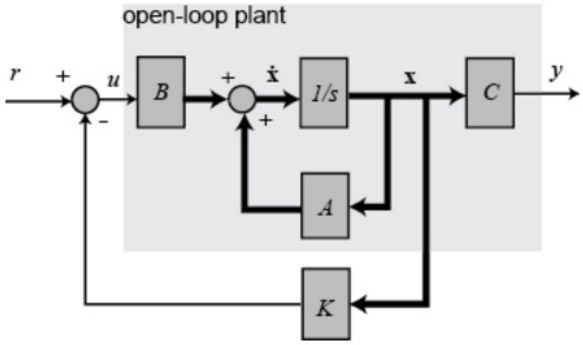
观察非0初值时的响应

```
>> t = 0:0.01:2;
u = zeros(size(t));
x0 = [0.01 0 0];

sys = ss(A,B,C,0);

[y,t,x] = lsim(sys,u,t,x0);
plot(t,y)
title('Open-Loop Response to Non-Zero Initial Condition')
xlabel('Time (sec)')
ylabel('Ball Position (m)')
```



6.3> 使用极点位置设计控制器(Control Design Using Pole Placement)

如果我们能够知道各时刻各状态变量的值，即用传感器实时测量磁质质量块的位置、速度、电路中的电流，称为full-state。简化起见，暂时不考虑R。设计控制器作用于原系统如下所示



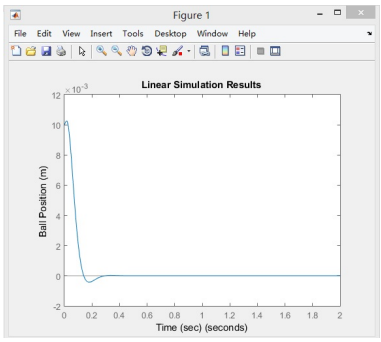
如上图，$u = -Kx$

闭环反馈系统的状态空间方程变为 $\dot{x} = (A - BK)x$

$y = (C - DK)x$

这样系统的极点位置就变为 A-BK 的特征值，我们可以通过改变矩阵B的值来得到目标极点。Matlab提供了 **place（A,B,P)**函数来确定目标极点所对应的K值，其中P为目标极点向量。

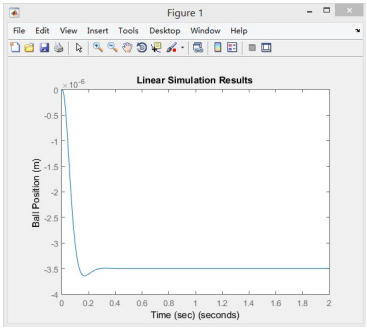
```
>> p1 = -20 + 20i;
p2 = -20 - 20i;
p3 = -100;

K = place(A,B,[p1 p2 p3]);
sys_cl = ss(A-B*K,B,C,0);

lsim(sys_cl,u,t,x0);
xlabel('Time (sec)')
ylabel('Ball Position (m)')
```



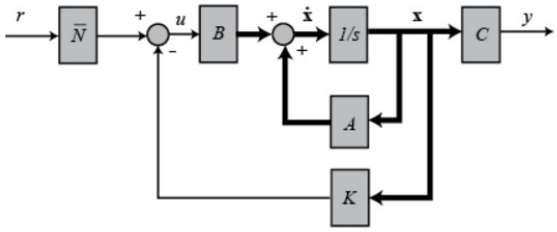
由响应曲线可见，结果相当理想。

当初值为0，输入信号为阶跃信号时的情况，重写输入 u = 0.001*ones(size(t))；再次执行上面的程序发现输出并未跟随阶跃信号。

```
>> t = 0:0.01:2;
u = 0.001*ones(size(t));

sys_cl = ss(A-B*K,B,C,0);

lsim(sys_cl,u,t);
xlabel('Time (sec)')
ylabel('Ball Position (m)')
axis([0 2 -4E-6 0])
```

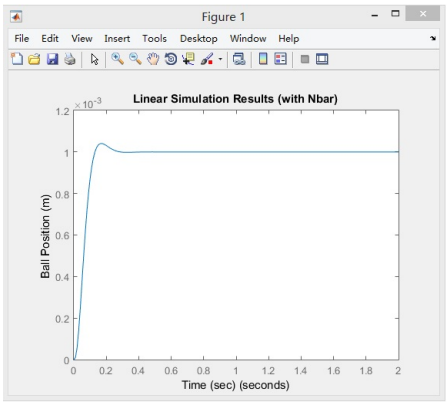


考虑上边框图所示的带控制器的系统模型，我们没有直接比较输出，而是在测量所有的状态量，乘一增益向量K，然后从输入中减去，没有任何理由使得 u 即 Kx 就会等于输出。我们通过前置控制量Nbar来解决这个问题



Matlab中使用函数 rscale.m来确定Nbar的合适值，Nbar = **rscale(sys, K)** （sys为原开环系统状态空间方程），现在

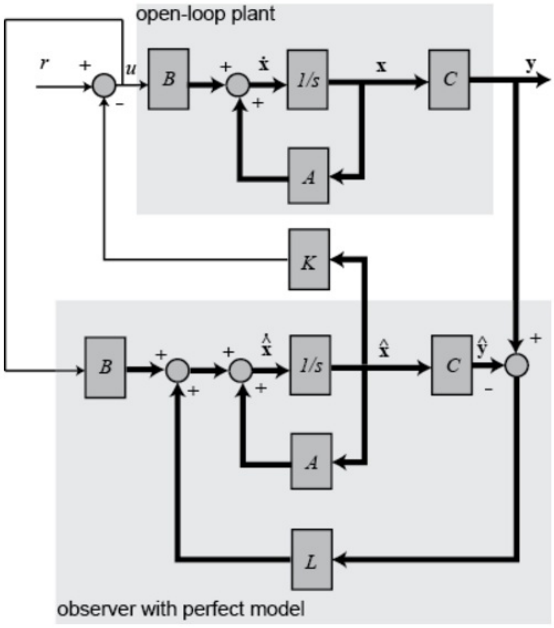
```
>> Nbar = rscale(sys,K)

Nbar =

 -285.7143

>> lsim(sys_cl,Nbar*u,t)
title('Linear Simulation Results (with Nbar)')
xlabel('Time (sec)')
ylabel('Ball Position (m)')
axis([0 2 0 1.2*10^-3])
```



响应稳定在0.001。

6.4> 设计观察器（Observer Design)

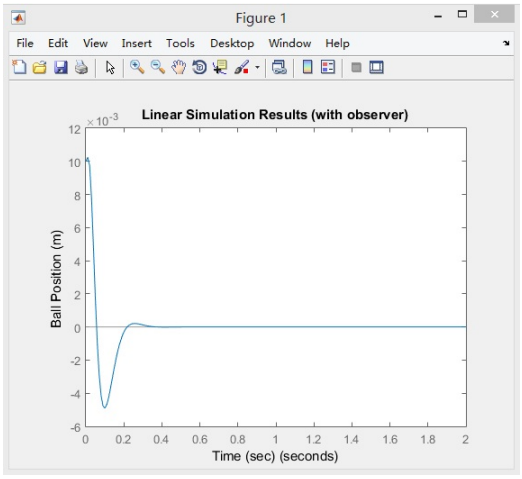
实际情况中我们常常不能获得状态量的当前值，这是就需要设计观察器来估计它们，如下所示



上边观察器只适用于 y = Cx 即D = 0的系统，观察器基本是控制系统的复制，它们有相同的输入，微分方程也基本相同。这里先只考虑非0初始值，输入为0时的响应。

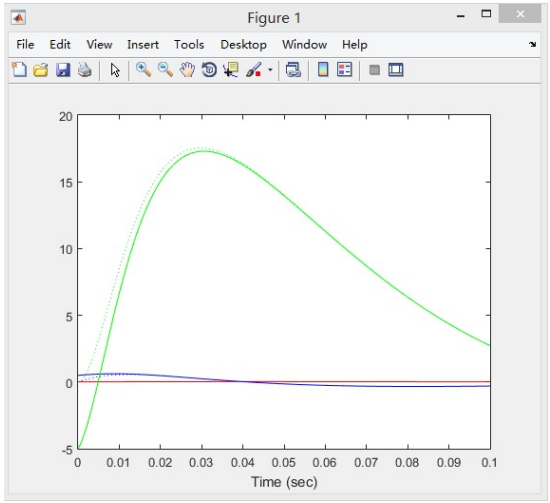
首先分析观察器，由于我们需要观察器有比对象系统快得多的响应，我们将观察器传递函数的极点取的大五倍，使其有比系统快得多的响应。通常观察器的初始状态为0，使得误差初始值与系统相等，即为x0,观察系统响应：

```
>> op1 = -100;
op2 = -101;
op3 = -102;
>> L = place(A',C',[op1 op2 op3])';
>> At = [ A-B*K              B*K
       zeros(size(A))     A-L*C ];

Bt = [    B*Nbar
       zeros(size(B)) ];

Ct = [ C     zeros(size(C)) ];
>> sys = ss(At,Bt,Ct,0);
lsim(sys,zeros(size(t)),t,[x0 x0]);
```



各状态量的变化如下：

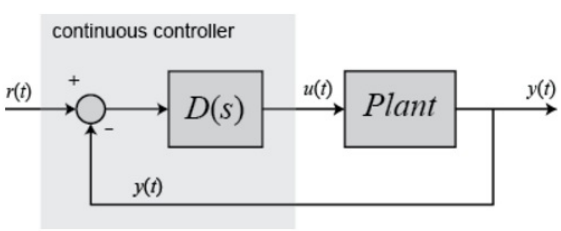
```
>> t = 0:1E-6:0.1;
x0 = [0.01 0.5 -5];
[y,t,x] = lsim(sys,zeros(size(t)),t,[x0 x0]);

n = 3;
e = x(:,n+1:end);
x = x(:,1:n);
x_est = x - e;

% Save state variables explicitly to aid in plotting
h = x(:,1); h_dot = x(:,2); i = x(:,3);
h_est = x_est(:,1); h_dot_est = x_est(:,2); i_est = x_est(:,3);

plot(t,h,'-r',t,h_est,':r',t,h_dot,'-b',t,h_dot_est,':b',t,i,'-g',t,i_est,':g')
xlabel('Time (sec)')
```



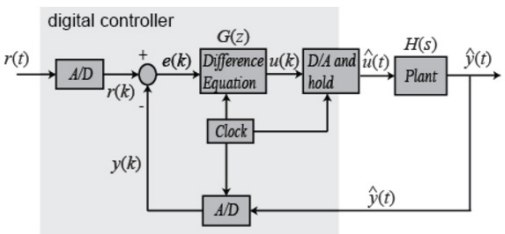
结果比较理想。

7. **数字化控制器设计**（Digital Controller Design）

这节我们的讨论从连续时间模型到离散时间模型，介绍Z变换来分析和设计离散系统的控制器。



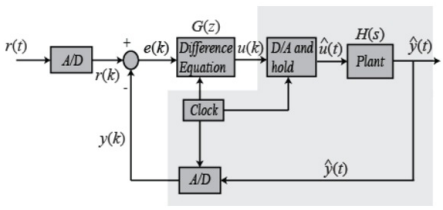
连续控制器

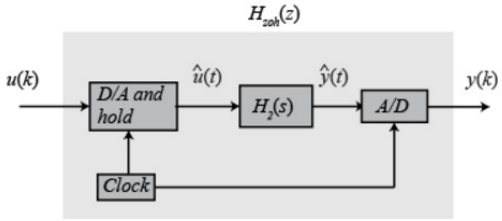


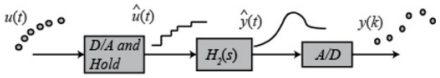
数字控制器

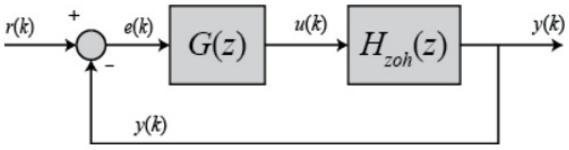
7.1> Zero-Hold Equivalence

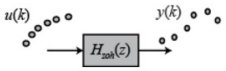
对于数字控制器，我们重新布局下面的部分：











这样，我们只需要设计$H_{zoh}(z)$就可以处理离散函数了。

Matlab提供 sys_d = **c2d(sys,Ts,'zoh')** 函数将给定的连续系统（continuous system，可以是传递函数也可以是状态空间方程)转换为离散系统(discrete system)，其中Ts是采样时间(单位为s)，其值需小于1/30Wbw，Wbw为系统的闭环带宽频

[例] 对于弹簧阻尼质量块系统，设M=1,b=10,k=20,采样时间为1/100s，将其离散化

7.1.1> 传递函数

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

```
>> M = 1;
b = 10;
k = 20;

s = tf('s');
sys = 1/(M*s^2+b*s+k);

Ts = 1/100;
sys_d = c2d(sys,Ts,'zoh')
```

```
sys_d =

  4.837e-05 z + 4.678e-05
  -----------------------
  z^2 - 1.903 z + 0.9048

Sample time: 0.01 seconds
Discrete-time transfer function.
```

也可以通过 sys_d = **tf(num,den,Ts)**直接写出离散系统形式的的传递函数，当采样时间不确定时可以令Ts = -1。

7.1.2> 状态空间方程

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

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

```
>> A = [0        1;
    -k/M   -b/M];

B = [  0;
    1/M];

C = [1 0];

D = [0];

Ts = 1/100;

sys = ss(A,B,C,D);
sys_d = c2d(sys,Ts,'zoh')
```

```
sys_d =

 a =
           x1        x2
  x1     0.999  0.009513
  x2   -0.1903    0.9039

 b =
            u1
  x1  4.837e-05
  x2   0.009513

 c =
      x1  x2
  y1   1   0

 d =
      u1
  y1   0

Sample time: 0.01 seconds
Discrete-time state-space model.
```

即离散后的状态空间方程为

$$\begin{bmatrix} x(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} 0.9990 & 0.0095 \\ -0.1903 & 0.9039 \end{bmatrix} \begin{bmatrix} x(k-1) \\ v(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0095 \end{bmatrix} F(k-1)$$

$$y(k-1) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x(k-1) \\ v(k-1) \end{bmatrix}$$

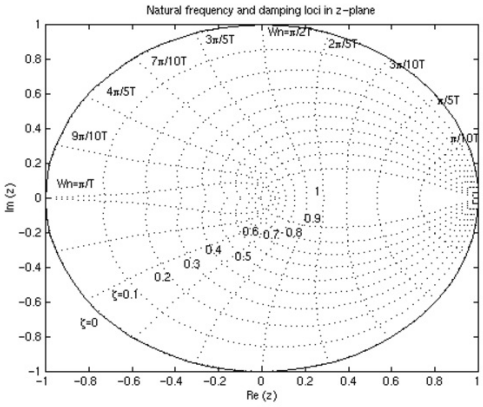
7.2> 稳定性和暂态响应（Stability and Transient Response)

7.2.1> Z平面（Z-plane）

与连续系统在S平面的极点位置可以判断如稳定性等系统的特性相同，可以分析Z平面上的极点位置来分析离散系统。z平面与s平面间的位置关系为：

$$z = e^{sT}$$

其中T是采样时间，s为S平面中的位置，z为对应Z平面中的位置。下图是Z平面上自然频率Wn与阻尼比ζ间的关系



Z平面的稳定性边线不再是虚轴，而是单位圆z = 1，当所有极点都在单位圆内，则系统稳定，否则系统不稳定。
为了由极点分析暂态响应，下面三个式子依然成立（Ts的Tolerance Fraction为+/-1%）：

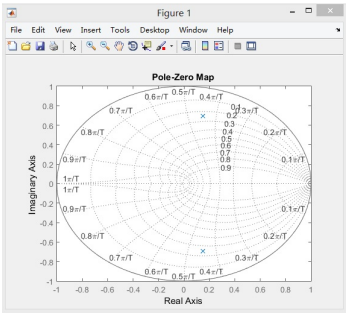
$$\zeta\omega_n \geq \frac{4.6}{Ts}$$

$$\omega_n \geq \frac{1.8}{Tr}$$

$$\zeta = \frac{-\ln(\%OS/100)}{\sqrt{\pi^2 + \ln(\%OS/100)^2}}$$

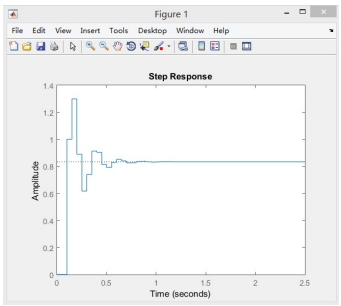
注意：在Z平面中的自然频率单位为rad/sample，要使用上边的式子必须转换为rad/sec
7.2.2> 稳定性和暂态响应分析（Stability and Transient Response Analysis）
在Z平面中绘制零极点

```
>> numDz = 1;
denDz = [1 -0.3 0.5];
sys = tf(numDz, denDz, -1);

pzmap(sys)
axis([-1 1 -1 1])
zgrid
```



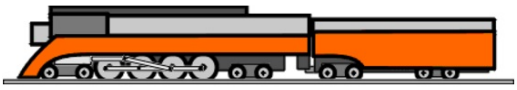
由上图可以看出系统稳定，且近似取值 Wn = 9pi/(20T) rad/sample, ζ = 0.25
假设采样时间为1/20s 则 Wn = 28.2 rad/sec，使用上边的三个式子取等号，得到Tr = 0.06s,Ts = 0.65s,OS = 45，即最大超调为45%，使用阶跃响应来验证我们的预测

```
>> sys = tf(numDz, denDz, 1/20);
step(sys, 2.5);
```

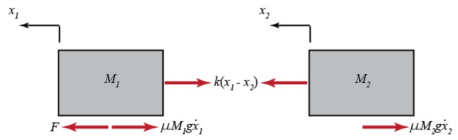


观察所得的响应，暂态响应符合我们的预测
由此，我们能够使用极点的位置以及上面给出的三个式子来预测系统的暂态响应了。

8. **Simulink建模**（Simulink Modeling）
Simulink可以直接呈现、仿真代表物理系统的数学模型，模型以图像化的块表（block diagram）的形式呈现出来，在库（libraries)中可以找到代表不同现象和模型的块。Simulink一个很明显的优势是通过Simulink模型可以快速地得到解析方法难以解决的复杂系统的动态响应、得到系统模型的数值近似解。
火车系统（Train System)



所图所示的火车沿着轨道行驶在一维空间，火车由发动机和车体两部分组成，质量分别为M_1、M_2，对火车模型进行如下简化：

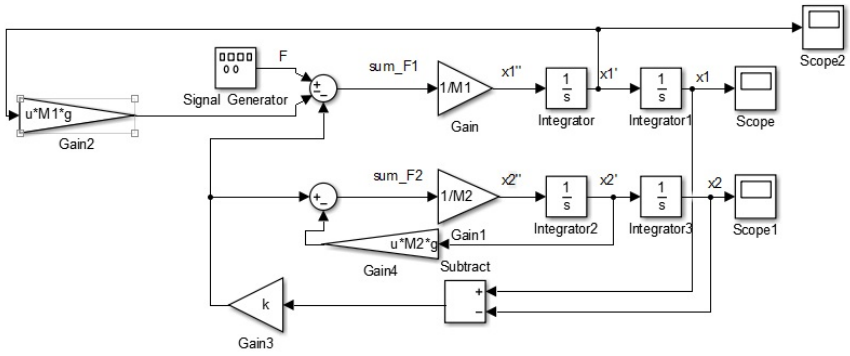


F为发动机提供的牵引力，将发动机和车体之间的连接假设为弹性模量为k的弹簧连接，地面的动摩擦系数为u
由牛顿定律有

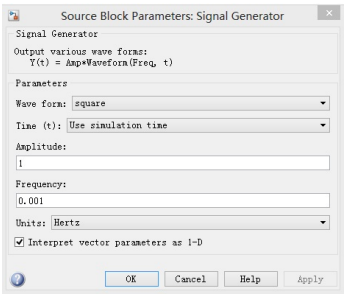
$$\Sigma F_1 = F - k(x_1 - x_2) - \mu M_1 g \dot{x}_1 = M_1 \ddot{x}_1$$

$$\Sigma F_2 = k(x_1 - x_2) - \mu M_2 g \dot{x}_2 = M_2 \ddot{x}_2$$

现在根据上边得到数学模型进行Simulink建模
设置观察量为 x1、x2、x1'



设置Signal Generator参数，选择波形为square，值为-1（在0~T/2，F=1，在T/2~T，F=-1，即先加速后加速）

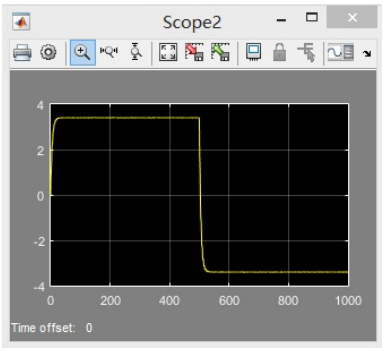


建立一个确定参数的.m文件

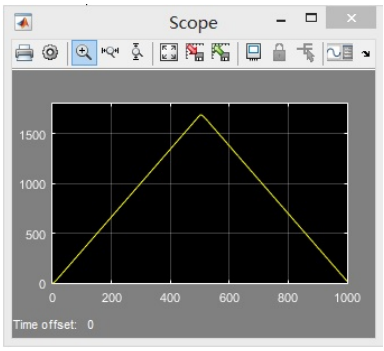
```
train.m
M1 = 1;
M2 = 0.5;
k  = 1;
F  = 1;
u = 0.02;
g  = 9.8;
```

在Matlab命令窗口界面输入 后 Simulink 会识别这些变量

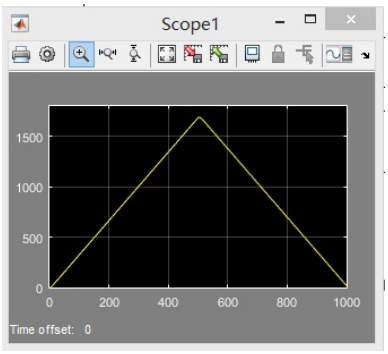
在Simulink中选择Simulation-Model Configration Parameters设置stop time为1000s，然后Run 进行仿真



x1'



x1



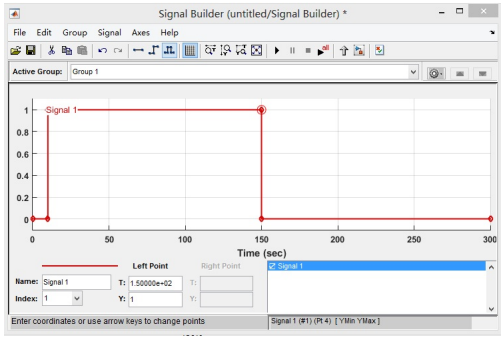
x2

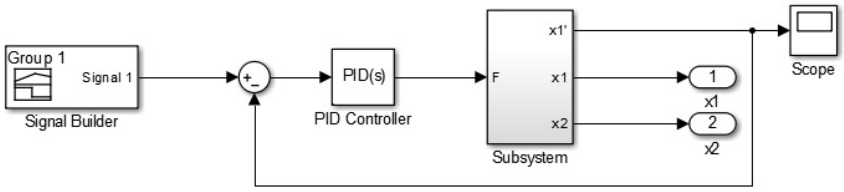
9. **Simulink控制**（Simulink Control）
上节介绍了Matlab可以仿真物理系统，更为普遍地，Matlab也可以仿真整个控制系统，包括除物理对象之外的控制算法。
9.1> PID控制
现在我们来控制这个火车，使得火车平稳地开始和结束行驶，在行驶过程中以恒定的速度前进。

首先我们选择单位反馈的PID控制，为了使模型更容易理解，将火车模型进行封装起来，输入F换为in block(Sources Library),输出x1、x1'、x2换为 out block(Sink Library),然后全选(ctr+A)模型，右击选择 Create Subsystem from selection。

添加PID控制器(Continous Library)、Signal Builder(Source Library)

Signal Builder可以设置想要的折现信号，首先选择时间范围 Axes->Change Time Range->Max time设为300,然后设置各节点位置，具体方法是点击节点所在折线，然后点击节点，然后在下边输入此节点的时间T以及对应的Y值，如图

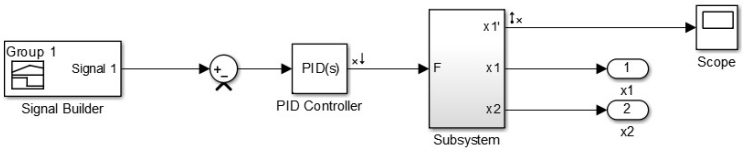




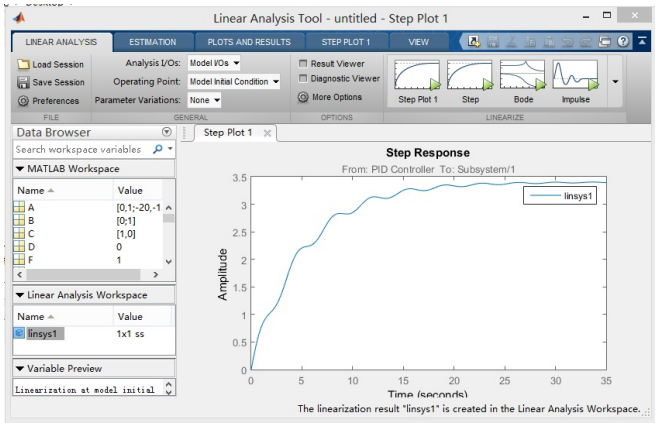
令P = 0、I = 0、D = 0，点击Run，观察运行结果

9.2> 抽取模型到Matlab（Extracting a model into MATLAB）

Simulink的控制工具箱提供将抽取模型到Matlab中的功能，这对于复杂模型及非线性模型非常有用。以无PID控制的原火车模型为例，首先对模型进行简单处理，右击输入F即PID的输出，选择Linear Anysis Points-> Open-loop Input，同样对于输出x1'，右击选择Linear Anysis Points-> Open-loop Output，这里我们只是将模型导入Matlab而不用控制，因此删除负反馈线，现在模型如下所示：



然后通过Linear Analysis Tool来抽取模型，在Simulink编辑界面选择 Analysis -> Control Design -> Linear Analysis，这样就打开了Linear Analysis Tool。点击Step就会出现模型的阶跃响应



于此同时在Linear Analysis Workspace中创建对象 linsys1，将其拖入上边的 Matlab Workspace，这样模型就成功导入到Matlab中了，我们可以通过Matlab中的函数来研究系统

```
>> sys_cl = feedback(linsys1,1);
>> pole(sys_cl)

ans =

  -0.9237 + 0.0000i
  -0.0000 + 0.0000i
  -0.2342 + 1.6574i
  -0.2342 - 1.6574i
```

可见，闭环系统并不稳定

9.3> 在Simulink中设计控制器（Controller Design within Simulink）

可以双击Simulink中的PID控制器选择 Tune 按钮打开PID Tuner GUI进行控制器的设计以达到得到理想动态响应的目的。这里介绍一种更普遍的方法，在Simulink编辑界面选择 Analysis -> Control Design -> Control System Designer 进入Control and Esitimation Tools Manager，打开界面依次选择

Select Blocks -> PID controller -> OK，这样选择了PID控制器，重新将Simulink模型闭环连接，右击信号输出线Linear Anysis Points-> Input

Perturbation(?,教程中为Input Point，新版本中并无这个选项，下同)，右击x1'输出线Linear Anysis Points-> Output Measurment
点击Tune Blocks打开 SISO Design Configuration -> Next -> Plot1 选择 Root Locus进行设计 -> Next -> Plot1 选择 Step 响应，勾选Plot 1-> Finish
出现根轨迹图。
至此为止，系统为0型系统(i = 0),我们通过增加积分环节使其成为1型系统来消除稳态误差，在Control and Estimation Tool Manager界面中选择Compensator Editor或直接在生成的根轨迹图中右击，Add pole/zero -> integrator，Add pole/zero -> zero ,Location -0.15
设计完成后会出现阶跃响应，再点击Compensator 下方的Update Simulink Block Parametres按钮，会使对应模块参数值改变以满足之前步骤的设计，这里是PID的数值。双击PID控制器可以看到里面的参数值已经改变，再次运行模型，观察到理想响应。

基本概念：
The gain margin is defined as the change in open-loop gain required to make the system unstable.
The phase margin is defined as the change in open-loop phase shift required to make a closed-loop unstable.
The bandwidth frequency is defined as the frequency at which the closed-loop magnitude response is equal to -3dB.