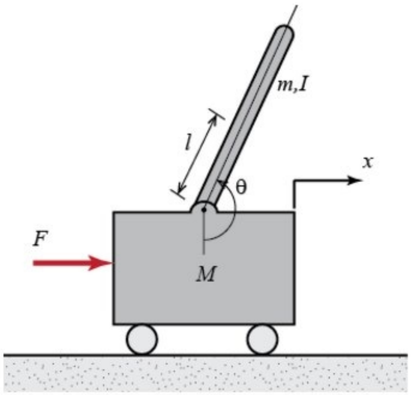
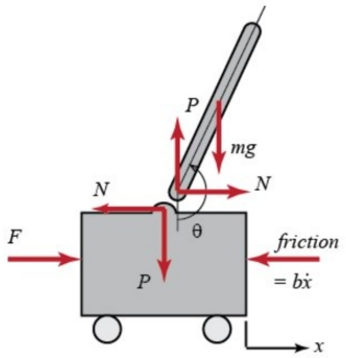
1. **系统建模**（System Modeling）

在控制理论的著作中，倒立摆模型非常常见，一方面是由于如果不加控制的话模型不可能保持稳定，另一方面是由于模型的非线性。同时现实世界中也大量存在倒立摆系统，如独轮车、发射时的火箭、导弹、双足、四足机器人等。

倒立摆的模型如下



1.1> 进行受力分析，建立动力学方程



对于小车：

$$M\ddot{x} + b\dot{x} + N = F \qquad (1)$$

对于倒立摆：

$$m\ddot{x}_p = \sum_{pend} F_x = N$$

$$\Rightarrow N = m\ddot{x}_p \qquad (2)$$

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \qquad (3)$$

其中x_p可由x表示

$$x_p = x + l\sin\theta$$

$$\dot{x}_p = \dot{x} + l\dot{\theta}\cos\theta$$

$$\ddot{x}_p = \ddot{x} - l\dot{\theta}^2\sin\theta + l\ddot{\theta}\cos\theta$$

将上式带入（2）式得

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

再将所得的N的表达式带入（1）式得F与x,theta的关系式：

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \qquad (4)$$

沿垂直于倒立摆的方向列方程

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta$$

将（3）式带入上式得x与theta的关系式：

$$(I+ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \qquad (5)$$

系统的输入为F，输出为theta，x。由此我们得到了系统的非线性驱动方程式（governing equations）（4）和（5），但我们所用的分析与设计方法仅适用于线性系统，因此需要将上边所得的方程进行线性化，首先令 $\theta = \pi$, 且假设倒立摆维持在一个很小的角度范围内摆动，用ϕ表示倒立摆偏离中心位置的角度，即有$\theta = \pi + \phi$. 使用下面的近似值来处理方程中的摆角

$$\cos\theta = \cos(\pi+\phi) \approx -1$$

$$\sin\theta = \sin(\pi+\phi) \approx -\phi$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

带入所得的两个非线性方程（4）和（5）得到运动的线性方程（用u代替力F）：

$$(I+ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u$$

1.2> 传递函数

假设初始条件为0，进行拉普拉斯变换有

$$(I+ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2$$

$$(M+m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s)$$

传递函数只能表示单输入、单输出的关系，首先确定输出$\Phi(s)$和输入$U(s)$的传递函数式，需要消除两式中的$X(s)$，得

$$(M+m)\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s^2 + b\left[\frac{I+ml^2}{ml} - \frac{g}{s^2}\right]\Phi(s)s - ml\Phi(s)s^2 = U(s)$$

写为传递函数的形式：

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I+ml^2)}{q}s^2 - \frac{(M+m)mgl}{q}s - \frac{bmgl}{q}} \qquad [\frac{rad}{N}]$$

其中，$q = [(M+m)(I+ml^2) - (ml)^2]$

进一步得到输出$X(s)$和输入$U(s)$的传递函数式

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad [\frac{m}{N}]$$

Matlab:

```
>> M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
q = (M+m)*(I+m*l^2)-(m*l)^2;
```

```
s = tf('s');

P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I + m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);

P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*l)*s/q - b*m*g*l/q);

sys_tf = [P_cart ; P_pend];

inputs = {'u'};
outputs = {'x'; 'phi'};

set(sys_tf,'InputName',inputs)
set(sys_tf,'OutputName',outputs)

sys_tf

sys_tf =

  From input "u" to output...
                     4.182e-06 s^2 - 0.0001025
   x:  -----------------------------------------------------------
       2.3e-06 s^4 + 4.182e-07 s^3 - 7.172e-05 s^2 - 1.025e-05 s

                              1.045e-05 s
   phi:  -------------------------------------------------------
         2.3e-06 s^3 + 4.182e-07 s^2 - 7.172e-05 s - 1.025e-05

Continuous-time transfer function.
```

1.2> 状态空间方程

将上边所得的线性方程以矩阵形式表示出来：

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

Matlab

```
p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices

A = [0      1              0           0;
     0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p       m*g*l*(M+m)/p  0];
B = [     0;
     (I+m*l^2)/p;
          0;
        m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'u'};
outputs = {'x'; 'phi'};

sys_ss = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs)
```

为了对系统进行控制，设置以下的控制目标

对于作用于小车的单位脉冲力，倒立摆的设计需满足

- θ的调节时间小于5s
- 摆角偏离竖直方向不能超过0.05rad

对于小车位移为0.2m的阶跃响应，倒立摆的设计需满足

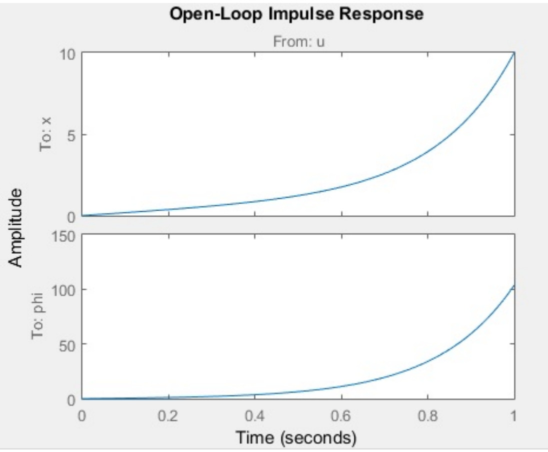
- θ的调节时间小于5s
- x的上升时间小于0.5s
- 摆角偏离竖直方向不超过20deg（即0.35rad）
- x和θ的稳态误差小于2%

2. **系统分析**（System Anaysis）

2.1> 开环脉冲响应

假设给小车作用一瞬时力，观察系统的开环响应

```
>> t=0:0.01:1;
impulse(sys_tf,t);
title('Open-Loop Impulse Response')
```



显然，响应不满足要求，小车及摆角呈无限增大的趋势，系统不稳定。观察两个传递函数的零极点来判断

```
>> [zeros poles] = zpkdata(P_pend,'v')

zeros =

     0


poles =

    5.5651
   -5.6041
   -0.1428
```

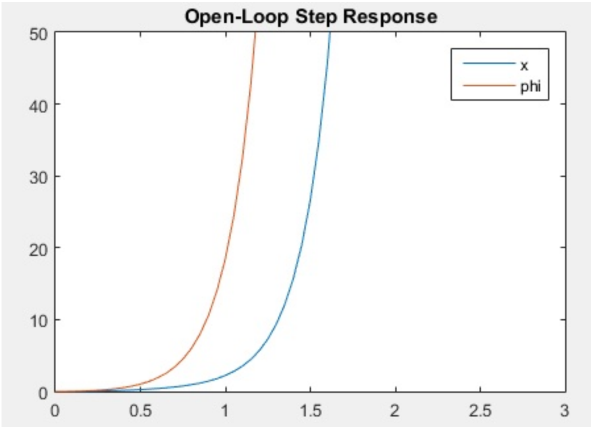
```
>> [zeros poles] = zpkdata(P_cart,'v')

zeros =

    4.9497
   -4.9497


poles =

         0
    5.5651
   -5.6041
   -0.1428
```

和预想的一样，都有落在复平面右半轴的极点，即开环系统不稳定。

2.1> 开环阶跃响应

给小车作用恒力，观察系统阶跃响应

```
>> t = 0:0.05:10;
u = ones(size(t));
[y,t] = lsim(sys_tf,u,t);
plot(t,y)
title('Open-Loop Step Response')
axis([0 3 0 50])
legend('x','phi')
```



同样，可以看到系统不稳定。

可以使用函数 **lisminfo** 得到响应的特征值

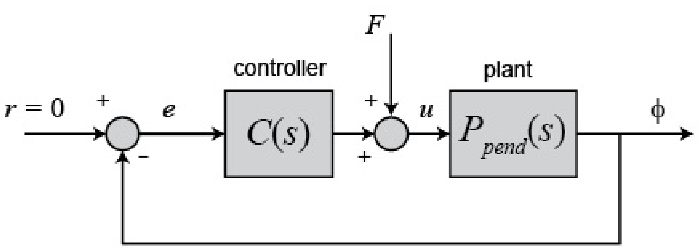
```
>> step_info = lsiminfo(y,t);
cart_info = step_info(1)
pend_info = step_info(2)

cart_info =

    SettlingTime: 9.9959
             Min: 0
         MinTime: 0
             Max: 8.7918e+21
         MaxTime: 10


pend_info =

    SettlingTime: 9.9959
             Min: 0
         MinTime: 0
             Max: 1.0520e+23
         MaxTime: 10
```

对于后边的PID等控制作一说明：PID控制、根轨迹、频率响应的控制器设计方法适用于单输入单输出系统（SISO），因此在用这几种方法设计控制器时，只关心其倒立摆的位置输出theta，忽略小车的位置输出x。

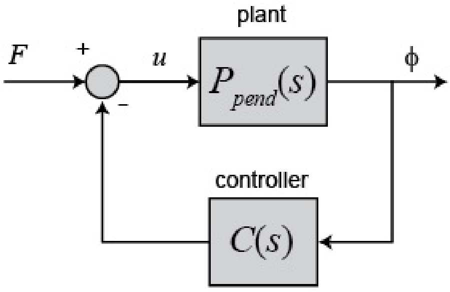
3. **PID控制器设计**（PID controller Design）

3.1> 系统结构

这个系统的控制器结构与标准控制结构有点不同，因为我们需要控制的是倒立摆的位置，使其在初始干扰后能迅速回到竖直方向，我们跟踪的输入信号应该为0，而额为的作用力F作为系统的脉冲干扰来处理，控制器的框图如下：



进行变换以便分析



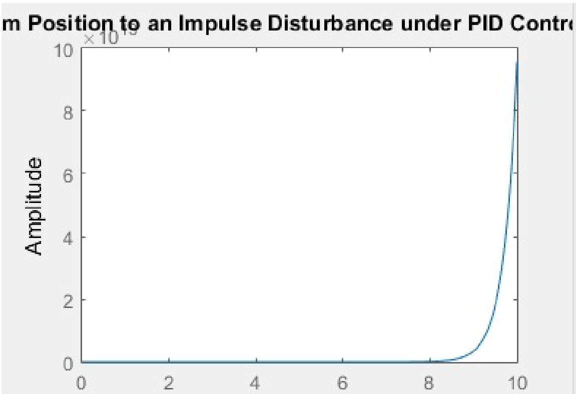
得到闭环系统的传递函数

$$T(s) = \frac{\Phi(s)}{F(s)} = \frac{P_{pend}(s)}{1 + C(s)}$$

3.2> PID控制

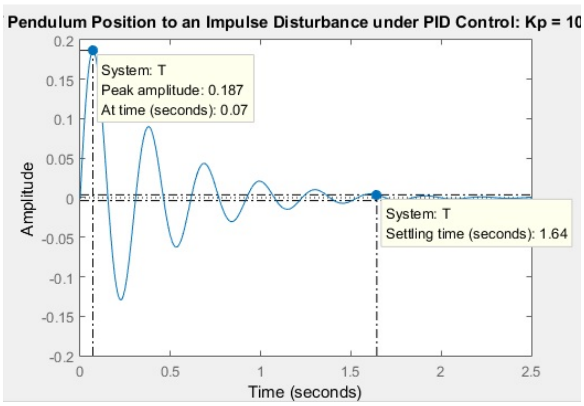
先令Kp = 1,Ki = 1,Kd = 1来观察闭环系统脉冲响应

```
>> Kp = 1;
Ki = 1;
Kd = 1;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
>> t=0:0.01:10;
impulse(T,t)
title('Response of Pendulum Position to an Impulse Disturbance under PID Control: Kp = 1, Ki = 1, Kd = 1');
```



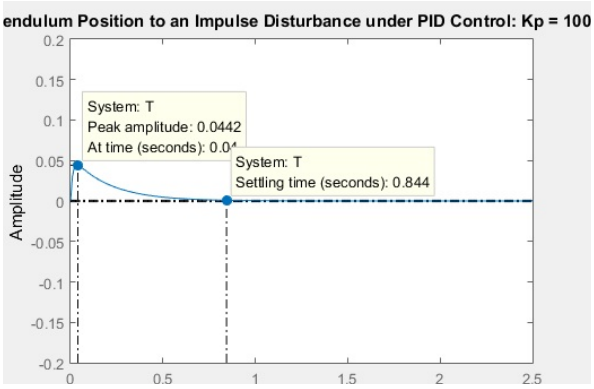
可以看出，系统仍然不稳定，增加增益Kp = 100

```
>> Kp = 100;
Ki = 1;
Kd = 20;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title('Response of Pendulum Position to an Impulse Disturbance under PID Control: Kp = 100, Ki = 1, Kd = 20');
```



此时，系统稳定，调节时间为1.64s，小于5s，稳态误差也接近为0，但是响应峰值大于0.05rad。Kd可以减小超调，因此再重新设置Kd = 20

```
>> Kp = 100;
Ki = 1;
Kd = 20;
C = pid(Kp,Ki,Kd);
T = feedback(P_pend,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 2.5, -0.2, 0.2]);
title('Response of Pendulum Position to an Impulse Disturbance under PID Control: Kp = 100, Ki = 1, Kd = 20');
```

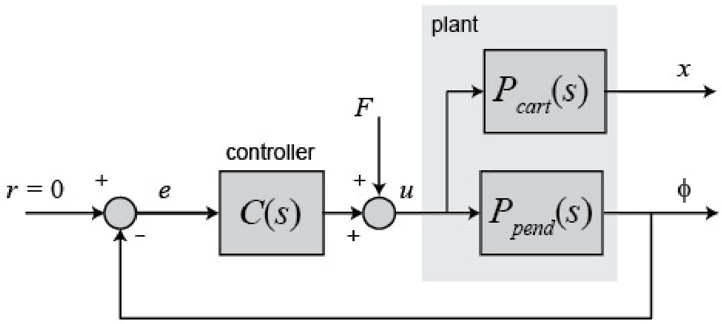


此时，满足响应要求。

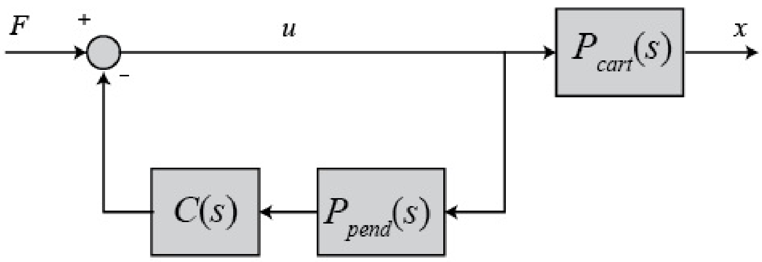
3.3> 小车的位置会怎样

上边所示的框图并不完整，我们没有考虑小车的位移x，在我们控制摆角的时候小车的位移会怎样呢？

补全系统框图



进行变换以便分析



得到闭环的传递函数为

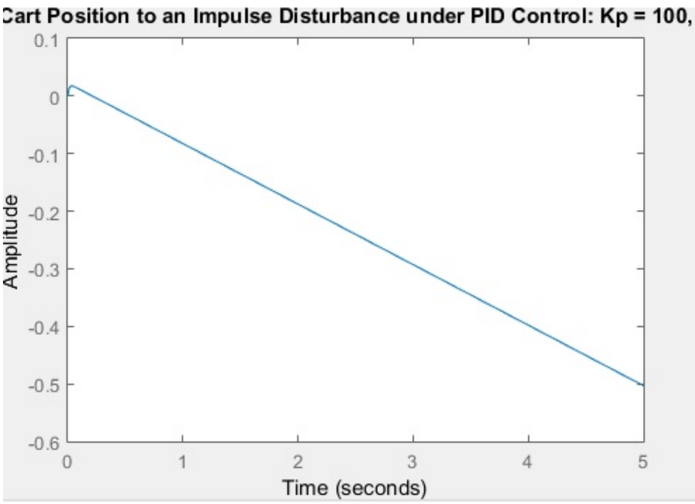
$$T_2(s) = \frac{X(s)}{F(s)} = \frac{P_{cart}(s)}{1 + P_{pend}(s)C(s)}$$

其中，

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(I+ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad \left[\frac{m}{N}\right]$$

$$q = [(M+m)(I+ml^2) - (ml)^2]$$

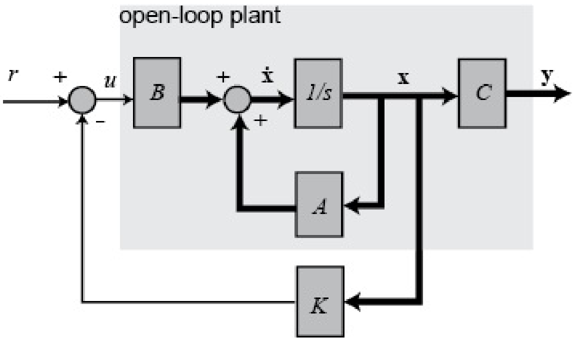
```
>> P_cart = (((I+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 + (b*(I + m*l^2))*s^3/q - ((M + m)*m*g*l)*s^2/q - b*m*g*l*s/q);
T2 = feedback(1,P_pend*C)*P_cart;
t = 0:0.01:5;
impulse(T2, t);
title('Response of Cart Position to an Impulse Disturbance under PID Control: Kp = 100, Ki = 1, Kd = 20');
```



如图所示，小车匀速向x轴负方向运动，因此，尽管PID控制能够稳定摆角，但是并不易用在实际的物理系统上。

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

假设所有状态量均可测，即为全状态反馈（full-state feedback），控制框图如下，注意这种方法反馈的量为状态量而非输出量



4.1> 开环极点

状态方程中的系统矩阵A的特征值为开环系统的极点

```
poles = eig(A)

poles =

         0
   -5.6041
   -0.1428
    5.5651
```

与前边所得一样，系统不稳定。

4.2> 线性二次规划（LQR, Linear Quadratic Regulation）

接下来的目标是找到状态反馈控制增益矩阵K。在此之前需先考察系统的可控性

$$\mathcal{C} = [A|AB|A^2B|\cdots|A^{n-1}B]$$

可控性矩阵矩阵是4X4的，当其秩为4时系统可控

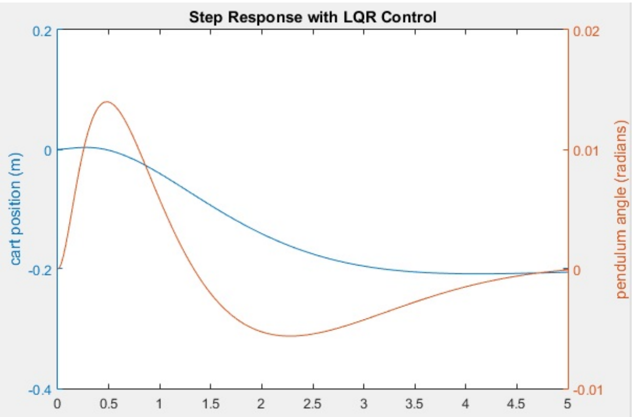
```
>> co = ctrb(sys_ss);
controllability = rank(co)

controllability =

     4
```

由此，系统可控。使用LQR来确定反馈增益K。matlab函数 **lqr** 允许选择两个参数R和Q来平衡控制力u和稳态误差，对应于我们想要优化的成本函数，最简单的情况是假设$R = 1, Q = C'C$.

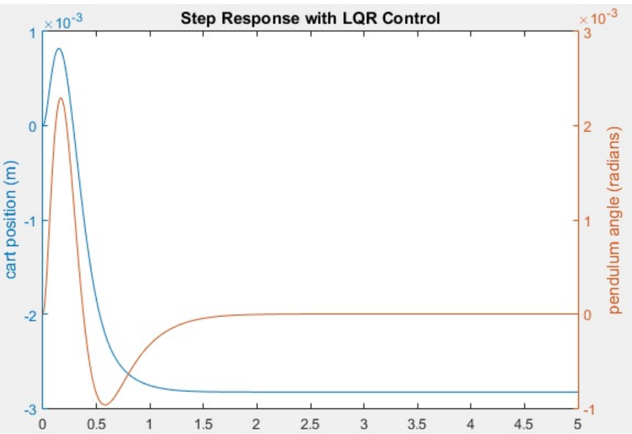
```
>> Q = C'*C

Q =

     1     0     0     0
     0     0     0     0
     0     0     1     0
     0     0     0     0
```

(1,1)元素代表小车位置的权重，(3,3)代表倒立摆角度的权重。执行下边程序，观察0.2m的阶跃输入响应

```
>> Q = C'*C;
R = 1;
K = lqr(A,B,Q,R)

Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];

states = {'x' 'x_dot' 'phi' 'phi_dot'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_cl = ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',outputs);

t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with LQR Control')

K =

   -1.0000   -1.6567   18.6854    3.4594
```



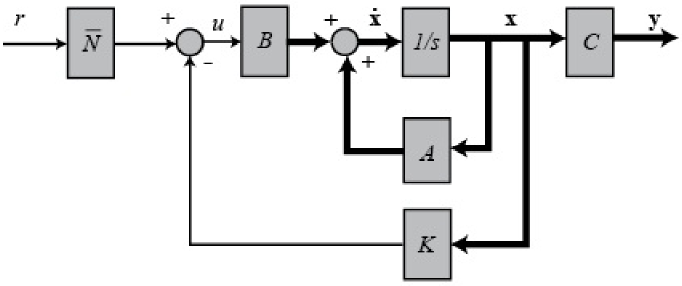
两个输出变量的超调都满足要求，但是上升时间及调节时间需要改善。试验发现增大Q的(1,1)和(3,3)元素值可以减小响应时间，重新设置Q(1,1) = 5000,Q(3,3) = 100，其他程序不变，再次运行得



如上，这时系统的暂态响应满足设计要求，但是小车位移的稳态误差太大。

4.3> 添加前向补偿

为了减小小车的稳态误差，增加前向补偿

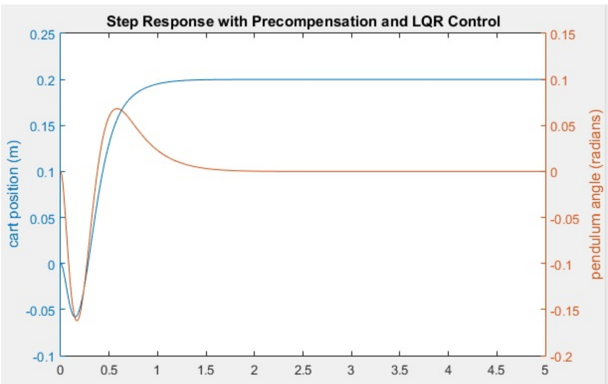


修改C矩阵，使得输入只反映小车位移x

```
>> Cn = [1 0 0 0];
sys_ss = ss(A,B,Cn,0);
Nbar = rscale(sys_ss,K)

Nbar =

  -70.7107
```

此时，观察响应

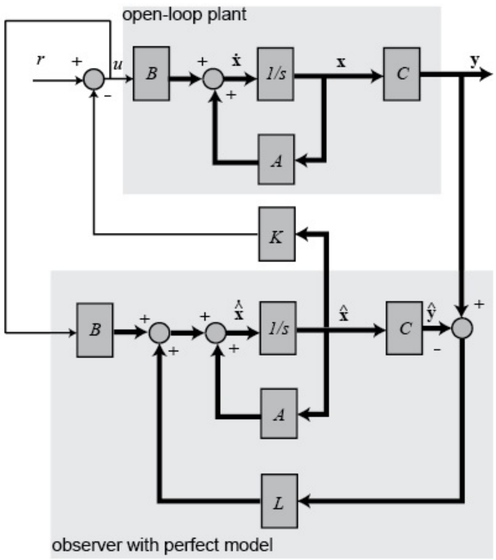
```
>> sys_cl = ss(Ac,Bc*Nbar,Cc,Dc,'statename',states,'inputname',inputs,'outputname',outputs);

t = 0:0.01:5;
r =0.2*ones(size(t));
[y,t,x]=lsim(sys_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Precompensation and LQR Control')
```



如图，稳态误差减小到了要求范围之内。

4.4> 基于观察器的系统控制

虽然上边的控制器已经能很好地满足响应要求，但这是基于所有系统变量可测的条件下，在实际模型中有些变量并不好测，因此我们需要基于观察器来设计控制器，框图如下：



先考察系统的可观察性（Observable），可观察性决定系统是否能通过输出估计状态变量。与可控性相似，如果系统的观察矩阵（observability matrix）为全秩，则系统具有可观察性，观察矩阵由下式定义：

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

matlab中的 **obsv** 函数来构造观察矩阵

```
>> ob = obsv(sys_ss);
observability = rank(ob)

observability =

     4
```

观察矩阵是8X4的，因为我们有两个输出变量，得到矩阵的秩为4，矩阵是满秩的，但是注意当构造 obsv(A,C(2,:)) 观察矩阵时，并非满秩，因此如果关心的输出只有摆角theta，那么系统是不具可观察性的。

由框图可得：

$$\dot{\hat{\mathbf{x}}} = A\hat{\mathbf{x}} + Bu + L(\mathbf{y} - \hat{\mathbf{y}})$$

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = (A\mathbf{x} + Bu) - (A\hat{\mathbf{x}} + Bu + L(C\mathbf{x} - C\hat{\mathbf{x}}))$$

$$\dot{\mathbf{e}} = (A - LC)\mathbf{e}$$

当矩阵 A-LC 稳定（特征值为负）时，稳态误差会降为0，而衰减速度取决于估计器 A-LC 的极点，因为要使用状态量的估计值作为控制器的输入，因此期望估计值的获得要比闭环响应更快，也就是让观察器的极点比控制器的对应相应快。常见的设置是让估计器的极点比最慢的控制器极点快4-10倍。同时不能使估计器极点值不能过大，否则如果有噪音干扰或传感器的测量误差会出现问题。

先确定闭环系统的极点

```
>> poles = eig(Ac)

poles =

  -8.4910 + 7.9283i
  -8.4910 - 7.9283i
  -4.7592 + 0.8309i
  -4.7592 - 0.8309i
```

最慢的极点实数部分为-4.7592，由此设置观察器的极点为[-40，-41，-42，-43]，求L

```
>> P = [-40 -41 -42 -43];
L = place(A',C',P)'

L =

   1.0e+03 *

    0.0826   -0.0010
    1.6992   -0.0402
   -0.0014    0.0832
   -0.0762    1.7604
```

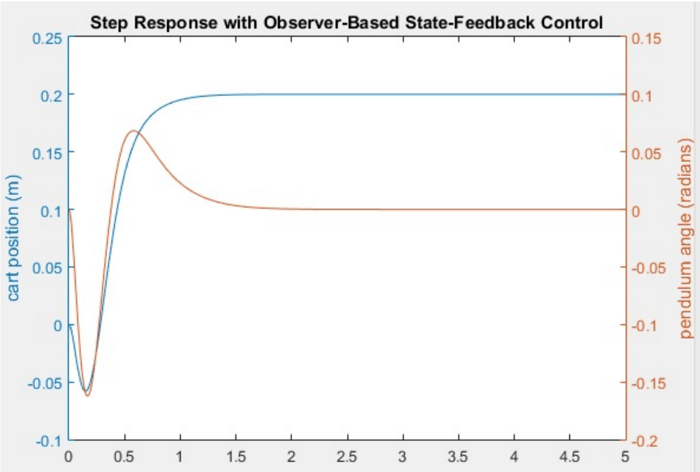
现在得到带有观察器的闭环系统的方程

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{e} \end{bmatrix} + \begin{bmatrix} B\bar{N} \\ 0 \end{bmatrix} r$$

$$\mathbf{y} = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{e} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} r$$

观察此时的系统响应

```
>> Ace = [(A-B*K) (B*K);
       zeros(size(A)) (A-L*C)];
Bce = [B*Nbar;
       zeros(size(B))];
Cce = [Cc zeros(size(Cc))];
Dce = [0;0];

states = {'x' 'x_dot' 'phi' 'phi_dot' 'e1' 'e2' 'e3' 'e4'};
inputs = {'r'};
outputs = {'x'; 'phi'};

sys_est_cl = ss(Ace,Bce,Cce,Dce,'statename',states,'inputname',inputs,'outputname',outputs);

t = 0:0.01:5;
r = 0.2*ones(size(t));
[y,t,x]=lsim(sys_est_cl,r,t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
set(get(AX(1),'Ylabel'),'String','cart position (m)')
set(get(AX(2),'Ylabel'),'String','pendulum angle (radians)')
title('Step Response with Observer-Based State-Feedback Control')
```



如上，所得的响应与全状态量反馈控制基本一致，满足系统要求。

由上，对于多输入多输出系统（MIMO），利用状态空间方程设置控制器比其他方法更为简便、有效。

5. **Simulink建模**（Simulink Modeling）

5.1> 用Simulink建立非线性模型

Simulink模型可以直接处理非线性方程，因此不需要对方程进行非线性处理。仍然由积分环节开始建立Simulink模型，

$$\ddot{x} = \frac{1}{M}\sum_{cart} F_x = \frac{1}{M}(F - N - b\dot{x})$$

$$\ddot{\theta} = \frac{1}{I}\sum_{pend} \tau = \frac{1}{I}(-Nl\cos\theta - Pl\sin\theta)$$

其中，N、P

$$m\ddot{x}_p = \sum_{pend} F_x = N$$

$$\Rightarrow N = m\ddot{x}_p$$

$$m\ddot{y}_p = \sum_{pend} F_y = P - mg$$

$$\Rightarrow P = m(\ddot{y}_p + g)$$

$$x_p = x + l\sin\theta$$

$$\dot{x}_p = \dot{x} + l\dot{\theta}\cos\theta$$

$$\ddot{x}_p = \ddot{x} - l\dot{\theta}^2\sin\theta + l\ddot{\theta}\cos\theta$$

$$y_p = -l\cos\theta$$

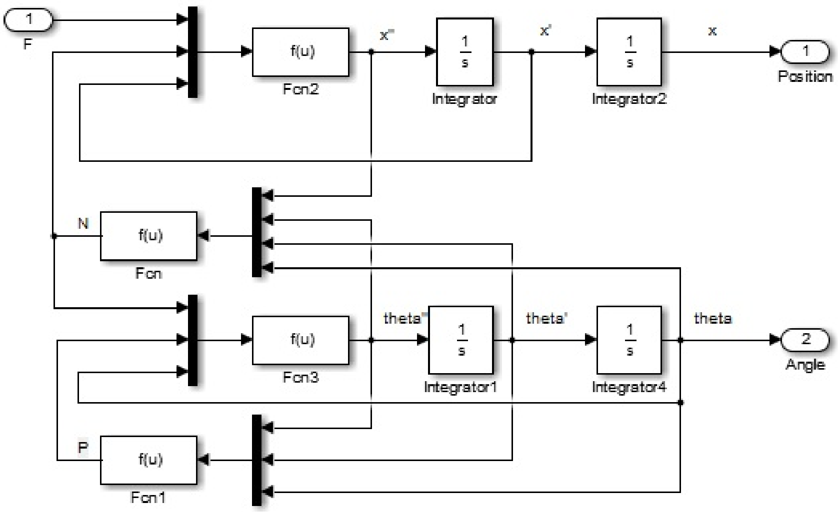
$$\dot{y}_p = l\dot{\theta}\sin\theta$$

$$\ddot{y}_p = l\dot{\theta}^2\cos\theta + l\ddot{\theta}\sin\theta$$

$$N = m(\ddot{x} - l\dot{\theta}^2\sin\theta + l\ddot{\theta}\cos\theta)$$

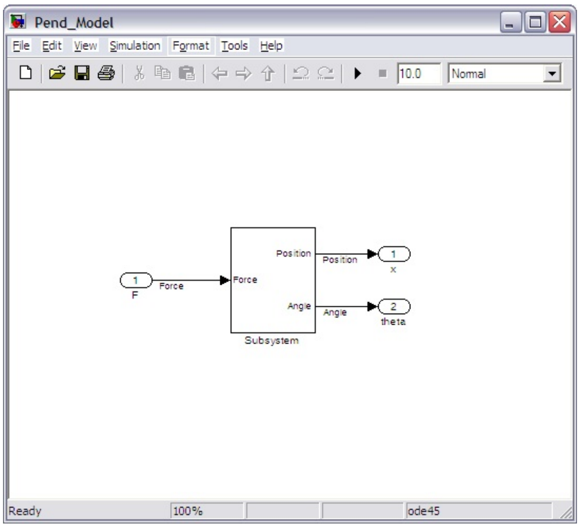
$$P = m(l\dot{\theta}^2\cos\theta + l\ddot{\theta}\sin\theta) + g$$

建立Simulink模型如下



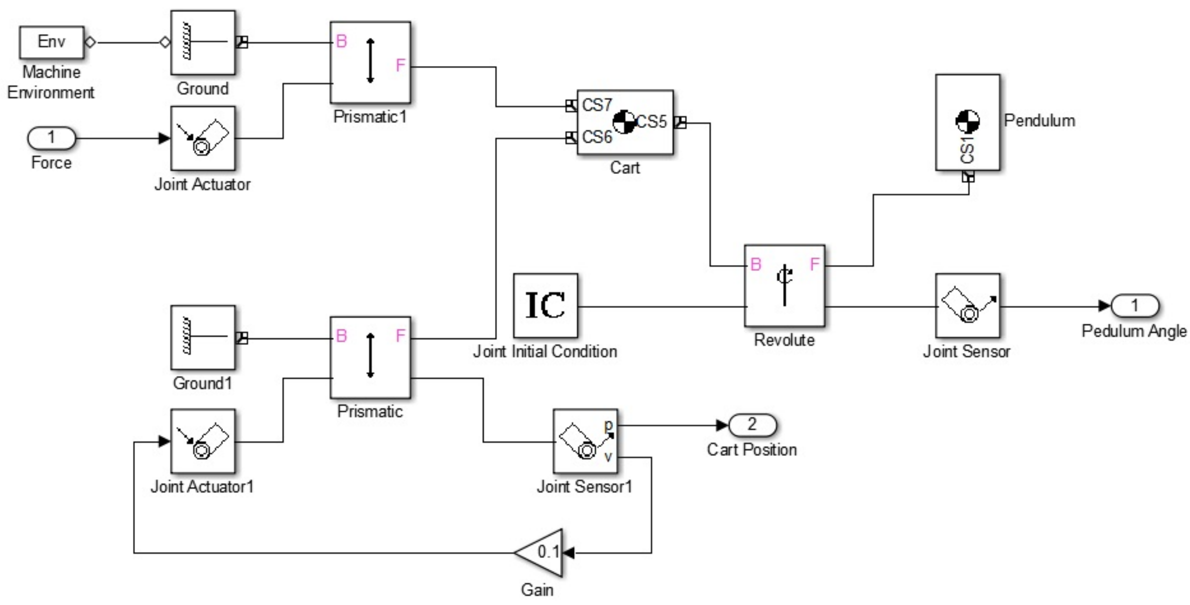
由theta'积分得到theta的integrator block，设置参数 initinal condition 为 pi

封装后有



5.2> 用Simscape建立非线性模型

Simsacpe建立实际物理对象，不需要建立数学方程即可获得其响应，倒立摆的Simscape模型如下



其中，

Body block from the Simscape/SimMechanics/Mechanical/Bodies library

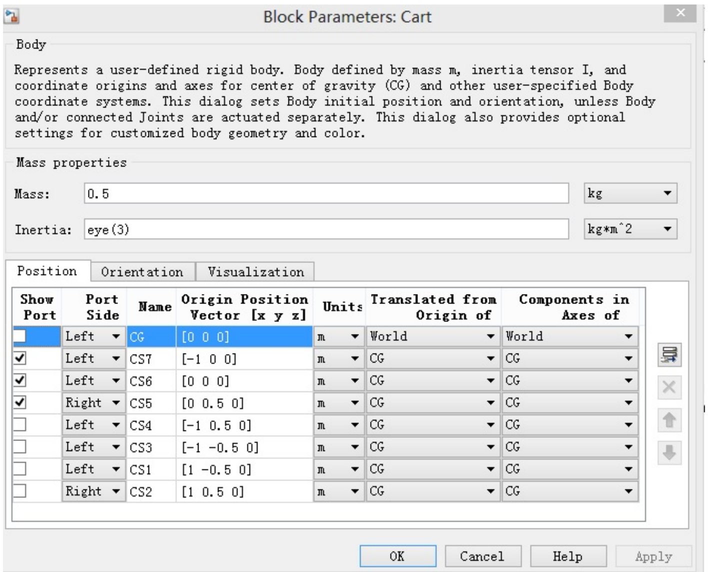
Revolute block from the Simscape/SimMechanics/Joints library to define the joint connecting the pendulum to the cart.

Joint Initial Condition block and a Joint Sensor block from the Simscape/SimMechanics/Sensors & Actuators library

Prismatic blocks from the Simscape/SimMechanics/Joints library

Ground blocks from the Simscape/SimMechanics/Bodies library

Joint Actuator blocks and one Joint Sensor block from Simscape/SimMechanics/Sensors & Actuators library



Block Parameters: Cart

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass: 0.5 kg

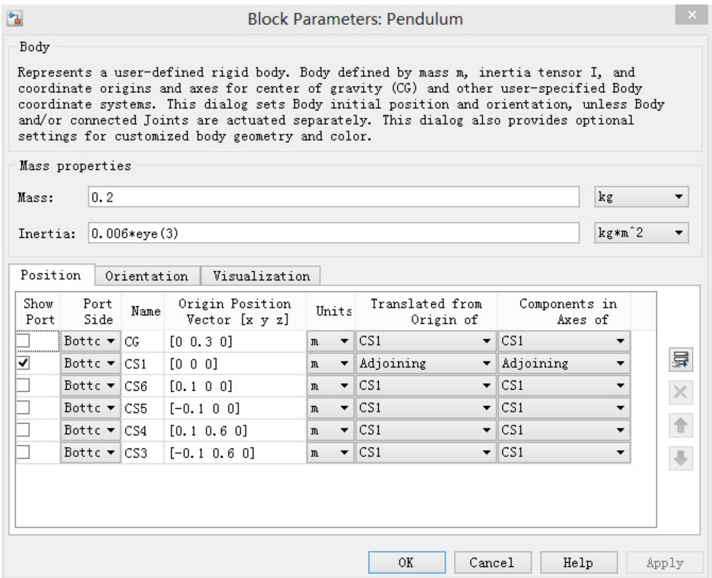
Inertia: eye(3) kg*m^2

Position | Orientation | Visualization

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Left	CG	[0 0 0]	m	World	World
☑	Left	CS7	[-1 0 0]	m	CG	CG
☑	Left	CS6	[0 0 0]	m	CG	CG
☑	Right	CS5	[0 0.5 0]	m	CG	CG
☐	Left	CS4	[-1 0.5 0]	m	CG	CG
☐	Left	CS3	[-1 -0.5 0]	m	CG	CG
☐	Left	CS1	[1 -0.5 0]	m	CG	CG
☐	Right	CS2	[1 0.5 0]	m	CG	CG

OK Cancel Help Apply

Cart Body Block



Block Parameters: Pendulum

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass: 0.2 kg

Inertia: 0.006*eye(3) kg*m^2

Position | Orientation | Visualization

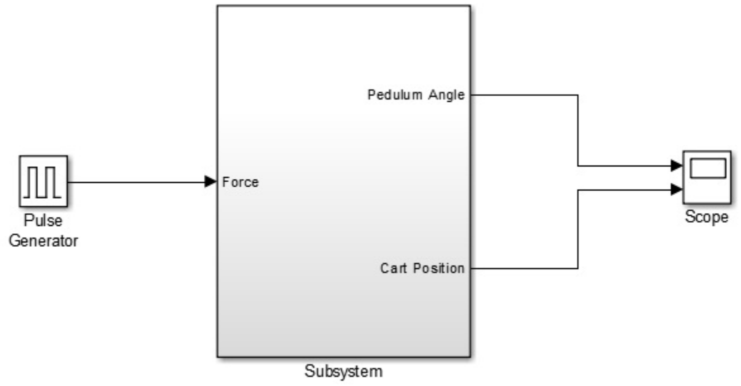
Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Bottc	CG	[0 0.3 0]	m	CS1	CS1
☑	Bottc	CS1	[0 0 0]	m	Adjoining	Adjoining
☐	Bottc	CS6	[0.1 0 0]	m	CS1	CS1
☐	Bottc	CS5	[-0.1 0 0]	m	CS1	CS1
☐	Bottc	CS4	[0.1 0.6 0]	m	CS1	CS1
☐	Bottc	CS3	[-0.1 0.6 0]	m	CS1	CS1

OK Cancel Help Apply

Pebdulum Body Block

Joint Sensor勾选 Position与Velocity，勾掉最下方的Output selected parameters as one signal

封装后，观察开环脉冲响应

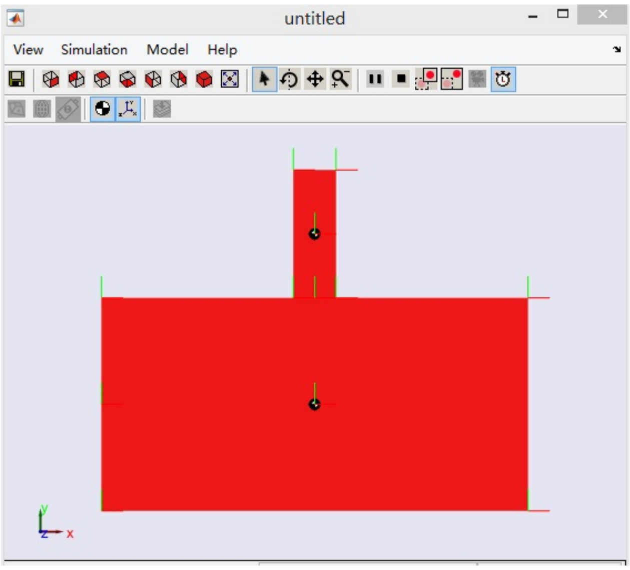


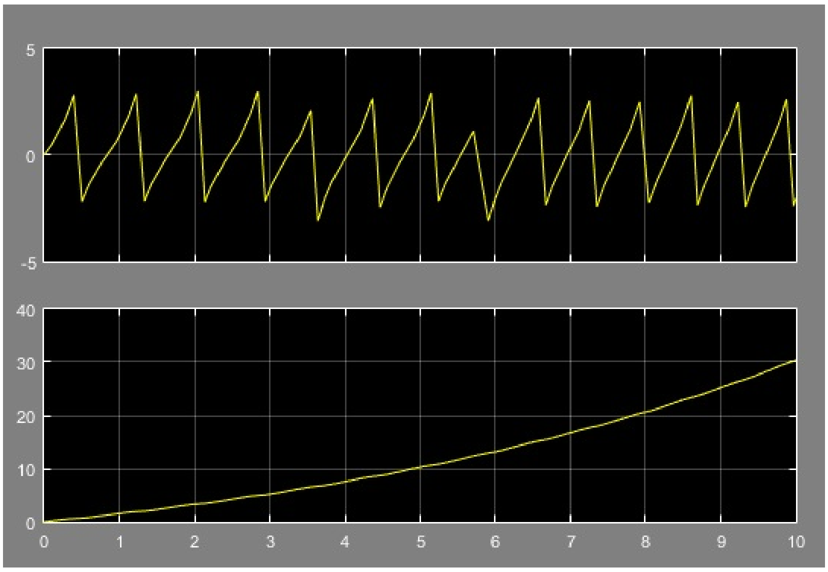
[Inverted_Pendulum_Simscape.slx]

Pulse Generator设置： Amplitude 1000; Period(secs) 10; Pulse Width(% of period) 0.01 (脉冲持续时 间为0.001s的单位脉冲：0.01%*10=0.001， 1000*0.001=1)

Scope设置： Parameters-> number of axes:2

仿真设置： Simulation-> Configration Parameters->Simscape-> SimMechanics ->勾选Show animation during simulation





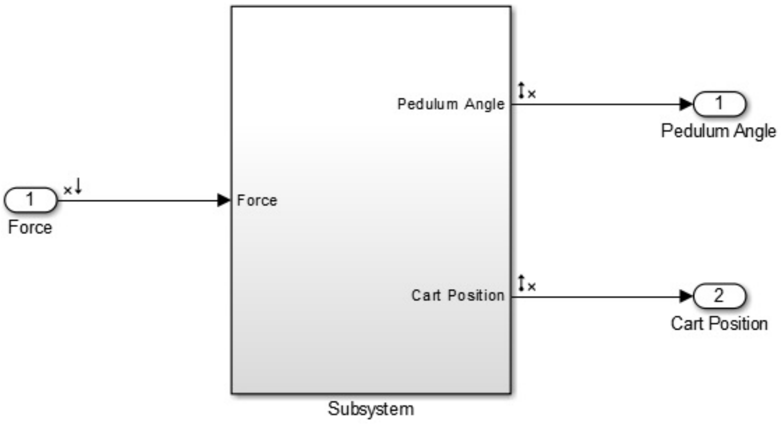
为了更好地与之前的结果进行比较，抽取线性模型

5.2> 从Simulation中抽取线性模型

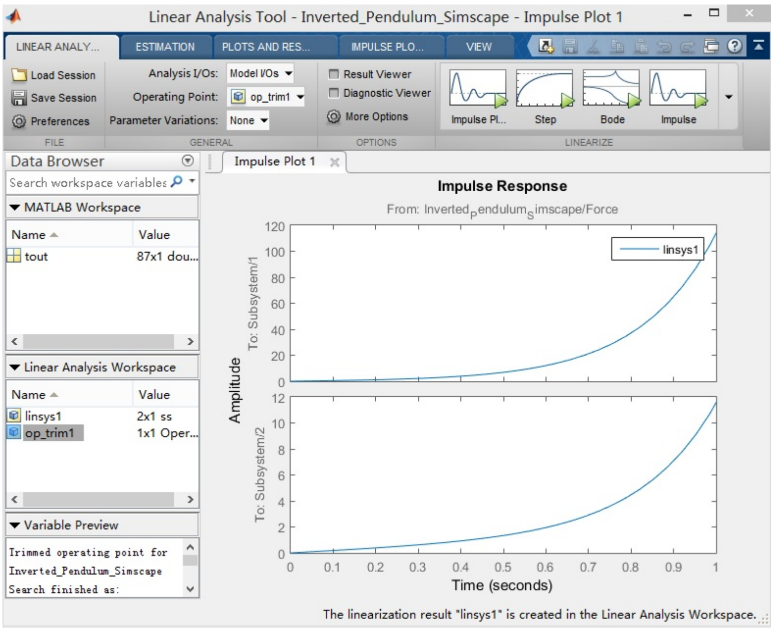
从Simlink或Simscape中封装模型操作，Simulink模型需在Matlab中定义参数值：

```
M = 0.5;
m = 0.2;
b = 0.1;
I = 0.006;
g = 9.8;
l = 0.3;
```

步骤如 Introduction 所述



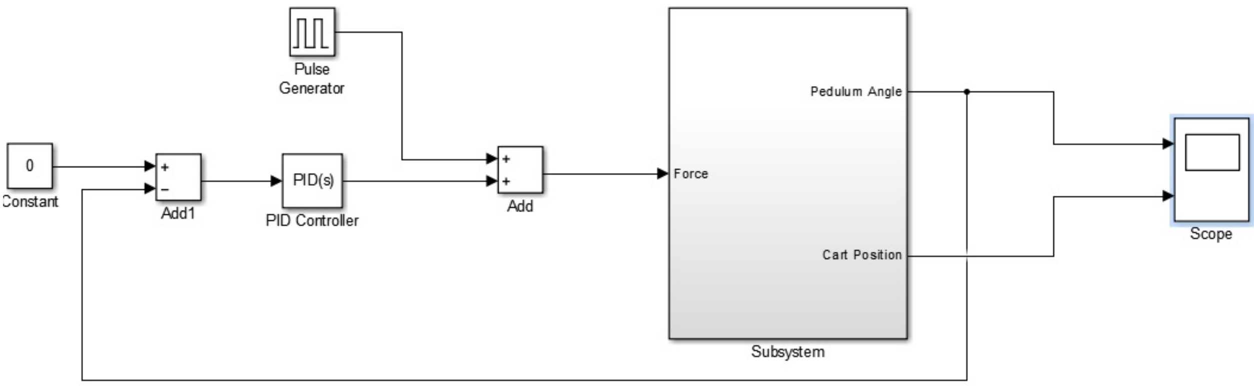
线性化模型：Operating Point-> Trim model，打开Trim Model tab，选择 Start Trimming，在Linear Analysis Workspace中出现完成后的模型op_trim1，此时Operating Point选择op_trim1，然后点击 Impulse Response观察出现的脉冲响应。



与System Analysis所得开环脉冲响应比较发现基本一致。

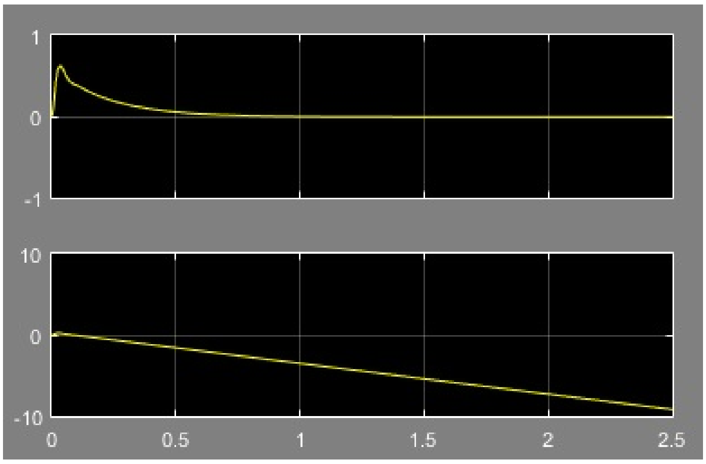
5. **Simulink控制器设计**（Simulink Controller Design）

在上边开环系统的基础上添加PID控制器，Kp = 100,Ki = 1,Kd = 20，设置仿真时间为2.5s进行仿真



Inverted_Pendulum_Control.slx

所得响应图线为



与PID控制器设计所得响应一致。