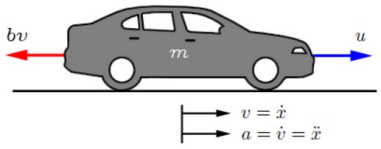
1. **系统建模**（System Modeling）

巡航控制的目的是维持运行中车辆的稳速前进，可以抵抗外在的干扰，如风和路况的改变。通过测量行驶速度与目标速度比较，然后根据控制理论调整车阀以实现稳速。



我们只考虑车辆的简单动力学模型，如上所示。假设我们可以直接控制牵引力u，忽略动力系统、轮胎的动力学响应，同时假设来自滚动摩擦和风的阻力随着车速线性变化。

1.1> 动力学方程

由牛顿第二定律有

$$m\dot{v} + bv = u$$

我们关心的输出为车速v,因此

$$y = v$$

确定系统的参数为

m = 1000kg

b = 50 Ns/m

1.2> 建立数学模型

1.2.1> 状态空间模型

系统中只有一个储能元素，因此只需一个状态变量v，由上式状态空间方程可以写为

$$\dot{\mathbf{x}} = [\dot{v}] = \left[\frac{-b}{m}\right][v] + \left[\frac{1}{m}\right][u]$$

$$y = [1][v]$$

Matlab:

```
>> m = 1000;
b = 50;

A = -b/m;
B = 1/m;
C = 1;
D = 0;

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

1.2.2> 传递函数模型

假设初始条件为0，由拉普拉斯变换得系统的传递函数

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms+b} \qquad [\frac{m/s}{N}]$$

Matlab:

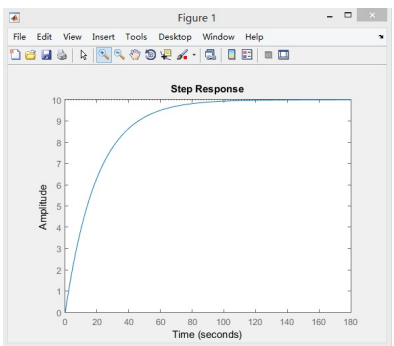
```
>> s = tf('s');
P_cruise = 1/(m*s+b)
```

2. **系统分析**（System Analysis）

2.1> 阶跃响应(Step Response)

假设我们提供500N的牵引力，即u = 500N，观察其阶跃响应

```
>> step(u*P_cruise)
```

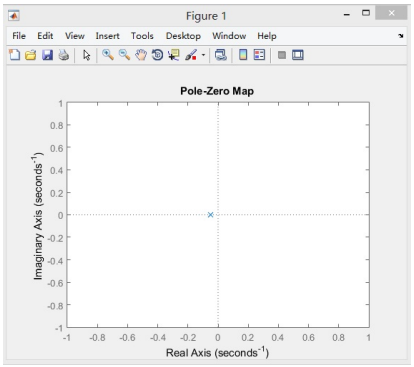


观察到其稳态速度为10m/s，假设满足我们的要求，但是上升时间Ts太慢了，约为60s，因此我们设计一个反馈控制器来提高其响应速度，并且

不会给系统动态响应的带来其他消极影响。

2.2> 开环零极点（Open-loop poles/zeros）

```
>> pzmap(P_cruise)
>> axis([-1 1 -1 1])
```



系统只有一个极点 s = -b/m

1> 极点值在S平面左边，因此系统稳定；

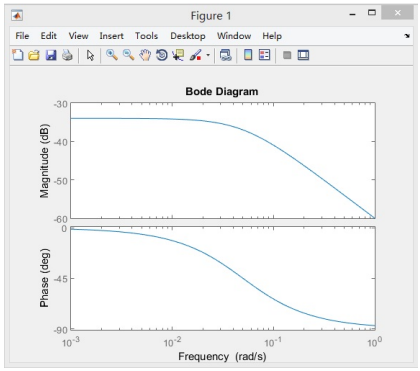
2> 极点值为负实根，因此系统响应不会振荡；

3> 系统的响应速度由极点的幅值决定，幅值越大，系统响应越快。

一般我们不会改变系统的参数值来改变系统的响应，相反，我们设计控制器来改变闭环系统的零极点来得到我们期望的响应。

2.3> 闭环bode图(Closed-loop Bode plot)

使用bode图来分析开环系统的频率响应

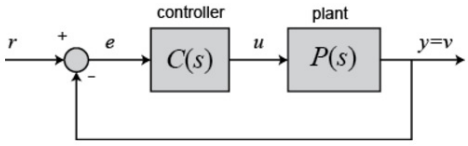


可以看出在w = b/m = 0.05rad/s(corner frequency）时的幅值、相位依次为-34dB、-45deg，并且以20dB/dec的斜率在高频段下降。

为设计控制器，我们确定下面的控制目标：

- 调节时间 Ts < 5s
- 超调量 %OS < 10%
- 稳态误差 < 2%

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



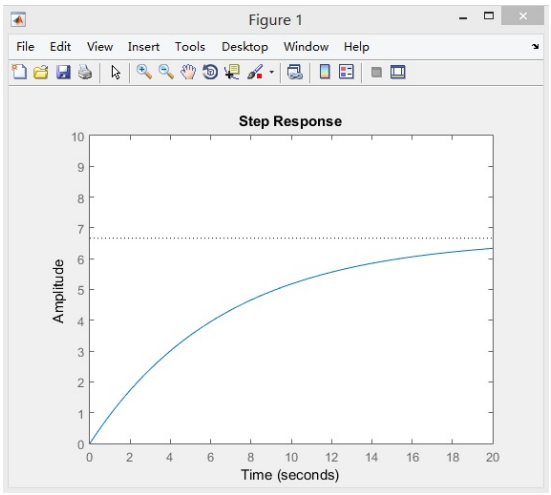
3.1> P控制（Propotional control）

首先利用比例环节来加快系统的响应，取Kp = 100，带P控制的系统闭环传递函数变为：

$$T(s) = \frac{Y(s)}{R(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} = \frac{K_p}{ms + b + K_p}$$

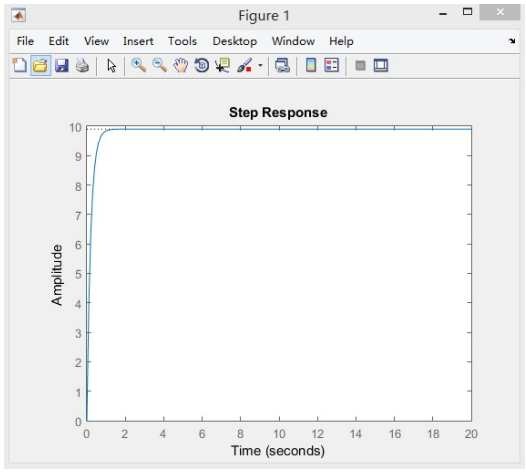
观察此时系统的阶跃响应:

```
>> m = 1000;
b = 50;
r = 10;

s = tf('s');
P_cruise = 1/(m*s + b);

Kp = 100;
C = pid(Kp);

I = feedback(C*P_cruise,1)

t = 0:0.1:20;
step(r*I,t)
axis([0 20 0 10])
```



\

从结果来看，上升时间和稳态误差都还远不满足要求，继续加大Kp来减小上升时间与稳态误差。取Kp = 5000



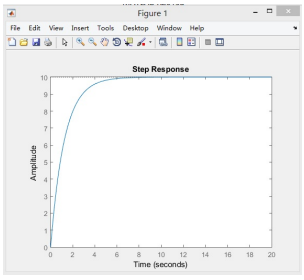
此时，稳态误差基本为0，上升时间约为0.5s，但在实际过程中汽车不可能在0.5s内由静止加速到10m/s，这是由于发动机及动力系统的功率限制。驱动器限制在控制工程中非常常见，设计完控制器后必须考虑实际能不能实现控制动作。
因此我们可以减小Kp选择在满足要求的同时能够实现控制的取值。

3.2> PI控制（PI control）
带有PI控制的闭环系统的传递函数变为

$$T(s) = \frac{Y(s)}{R(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} = \frac{K_p s + K_i}{ms^2 + (b + K_p)s + K_i}$$

带有积分环节可以减小或消除稳态误差（Ki不至过小），但在设定Ki值时不宜过大，否则会使系统趋于不稳定。我们选择Kp = 800, Ki = 40
观察此时系统的阶跃响应:

```
>> Kp = 800;
Ki = 40;
C = pid(Kp,Ki);

I = feedback(C*P_cruise,1);

step(r*I,t)
axis([0 20 0 10])
```



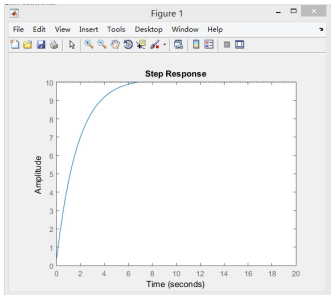
满足响应要求

3.3> PID控制（PID control）
PI控制已经很好地满足要求了，因此实际过程中没有必要增加微分环节，但我们这里还是看看增加微分环节后的响应
带有PID控制的闭环系统的传递函数变为

$$T(s) = \frac{Y(s)}{R(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} = \frac{K_d s^3 + K_p s + K_i}{(m + K_d)s^2 + (b + K_p)s + K_i}$$

取Kp = 600, Ki = 40,Kd = 40

```
>> Kp = 600;
Ki = 40;
Kd = 40;
C = pid(Kp,Ki,Kd);

I = feedback(C*P_cruise,1);
>> step(r*I,t)
axis([0 20 0 10])
```



也基本满足要求

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

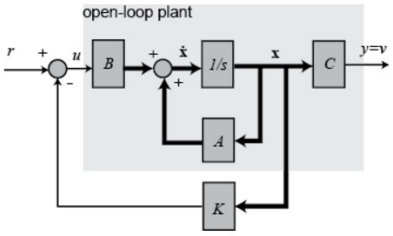
确定的状态空间方程为

$$[\dot{v}] = \left[\frac{-b}{m}\right][v] + \left[\frac{1}{m}\right][u]$$

$$y = [1][v]$$

A = [-b/m],B= [1/m],C = [1],D= [0]

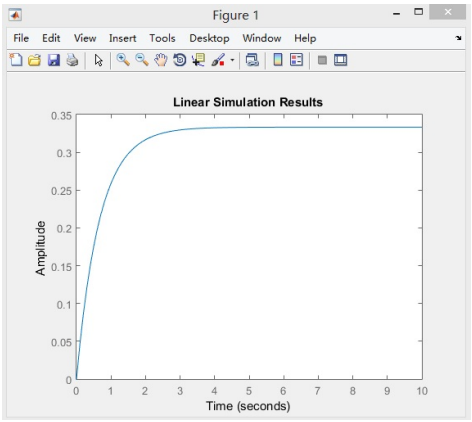
使用极点位置设计控制器



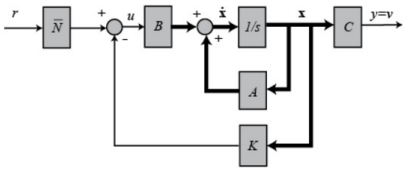
x = [v], u = r - Kx

如前说述，A - BK的特征值为极点位置，可以通过改变矩阵K的取值来改变极点的位置，我们增大极点的幅值来加快系统的动态响应，取极点值为-1.5

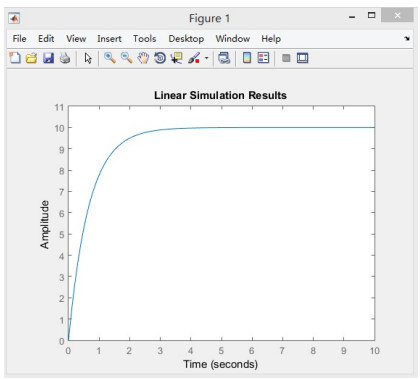
```
>> m = 1000;
b = 50;
t = 0:0.1:10;
u = 500*ones(size(t));

A = [-b/m];
B = [1/m];
C = [1];
D = [0];
sys = ss(A,B,C,D);

x0 = [0];

p1 = -1.5;

K = place(A,B,[p1])

sys_cl = ss(A-B*K,B,C,D);
lsim(sys_cl,u,t,x0);
axis([0 10 0 0.35])
```



由阶跃响应观察系统的动态响应加快了，但是稳态误差太大，我们增加范围因子Nbar来较小稳态误差

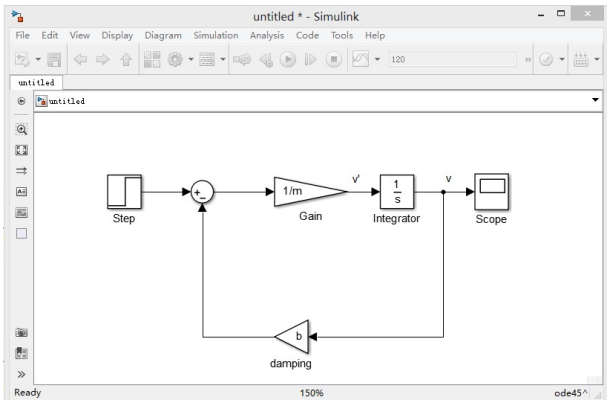


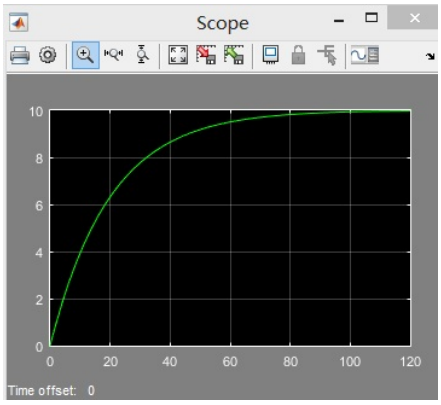

```
>> Nbar = rscale(sys,K)*10/500;
sys_cl = ss(A-B*K,B*Nbar,C,D);

lsim(sys_cl,u,t,x0);
axis([0 10 0 11])
```



此时响应满足我们的要求。

5.**Simulink建模** （Simulink Modeling）





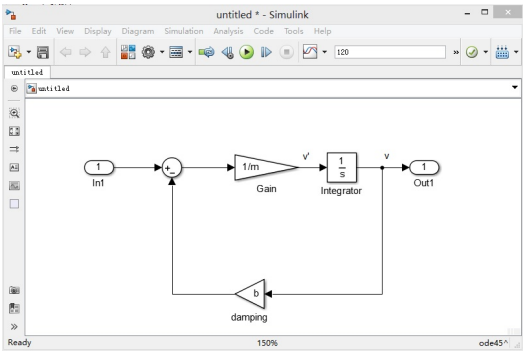
设置Step,Step time = 0,initial value = 0,final value = u;

配置仿真(Configuration Parameters):Stop time = 120 s 后运行模型，观察速度曲线

6.Simulink控制器设计 （Simulink Controller Design）

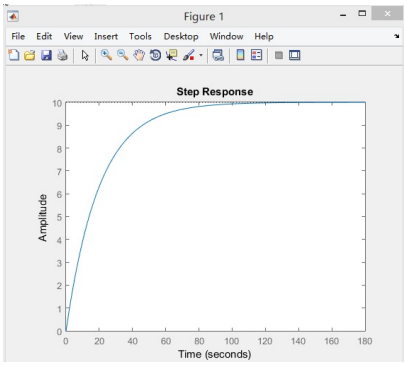
6.1> 抽取线性模型到Matlab（Extracting a linear model into Matlab）

将模型的输入输出换为 Input 与 output模块（Port & Subsystem Library），这定义了抽取模型的输入、输出，如下所示



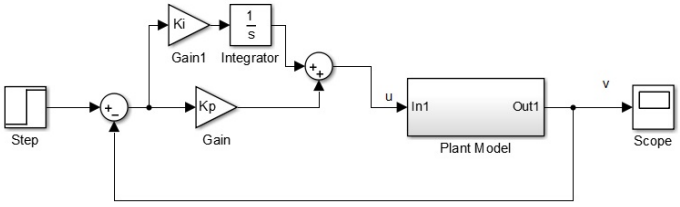
保存为ccmodel.mdl，与Introduction里抽取包含控制的火车模型到Matlab中不同，对于简单的线性模型Matlab提供了 **linmod('ccmodel')** 来在Matlab中使用Somulink中的模型

```
>>   m = 1000;
b = 50;
u = 500;
[A,B,C,D] = linmod('ccmodel');
cruise_ss = ss(A,B,C,D);
>> step(u*cruise_ss)
```



6.2> 进行PI控制（Implementing PI Control）

将之前所建系统的Simulink模型用Subsystem进行封装，然后进行PI控制



更为简单地，PID环节直接可以用一个PID控制器（Continous Library）或传递函数

设置Step，final value = 10，仿真时间设为10s

```
>>  m = 1000;
b = 50;
r = 10;
Kp = 800;
Ki = 40;
```

然后运行模型，得PI控制的系统速度响应