**同 济 大 学**

**实验报告**

模型预测控制



**课程名称**   最优化原理

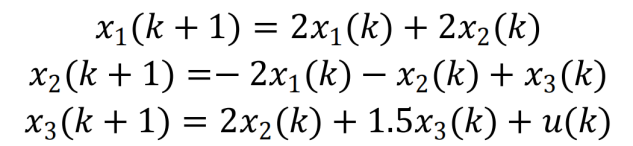
**学生姓名**  姚 天 亮

**学 号**  2150248

**学 院** 电子与信息工程学院

1. **作业要求**

假设天问一号降速过程中的部分所需控制模型如下：



状态及输入的约束条件：



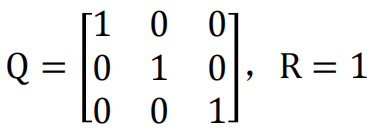
预测步数：



初值：



代价函数系数矩阵：



问题：

结合课堂内容所学，请将其整理成状态空间的形式并采用模型预测控制方法求解最优解。请使用 Matlab 编程实现，其中 QP 求解器用加速梯度投影法。

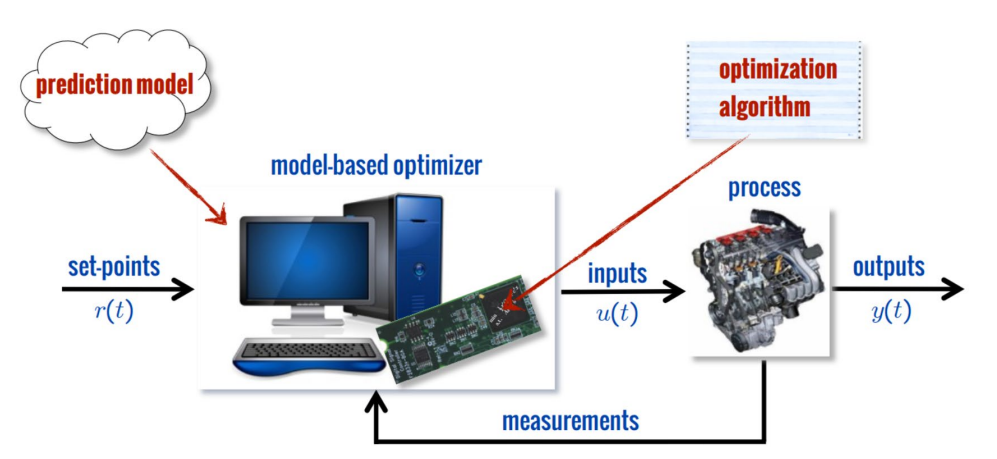
1. **理论推导**

本作业题的目的将控制模型整理为状态空间的形式，并采用模型预测控制方法进行最优解的求解。

模型整理成的状态空间如下：



要求使用的模型控制，使用过程的动态模型预测其未来的发展并选择“最佳”控制措施。本质上模型预测控制是求解一个开环最优的控制问题。



具有：预测模型 滚动优化 反馈校正的三要素。

预测模型是根据系统的现时刻的控制输入以及过程的历史信息，预测

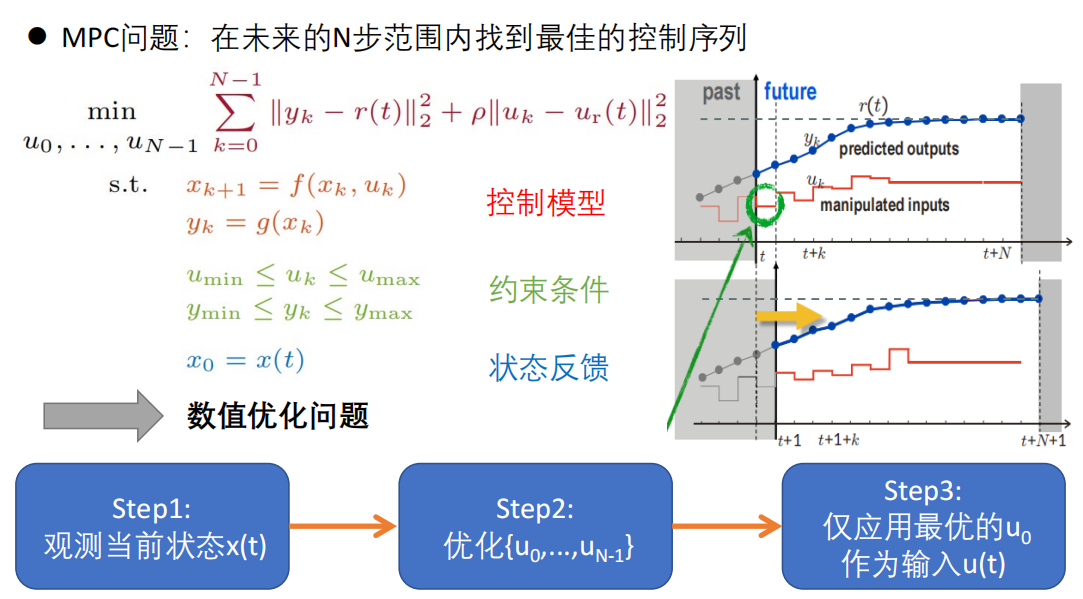
过程输出的未来值。因此，需要一个描述系统动态行为的模型作为预测模型。

模型预测控制采用滚动式的有限时域优化策略。即优化过程不是一次离线完成的，而是反复在线进行的。某一时刻计算得到的控制序列只实施第一个控制作用，在下一个采样时刻重新求取最优控制率。

在预测控制中，通过输出测量值与模型预估值进行比较，得出模型预测误差，再利用模型预测误差来对模型的预测值进行修正。

对模型施加了反馈校正的过程，使预测控制具有很强的抗扰动和克服系统不确定性的能力。 预测控制中利用了反馈信息，因此这是一种闭环优化控制算法。

MPC问题的数学描述：

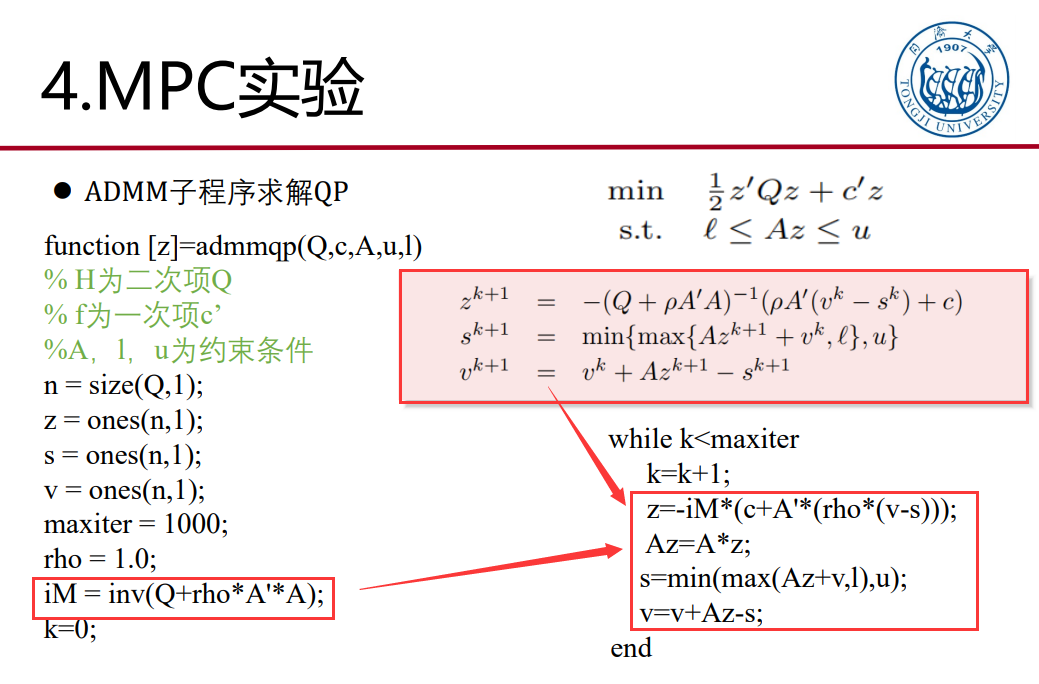


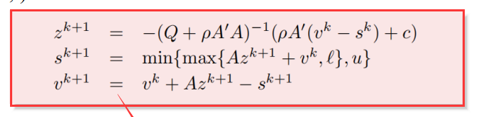
使用加速梯度投影法的QP求解器是一种基于牛顿法的学习算法，它可以近似损失函数为一个简单的抛物线，并通过计算抛物线的最小值来更新权重。具体的实现原理与步骤如下：

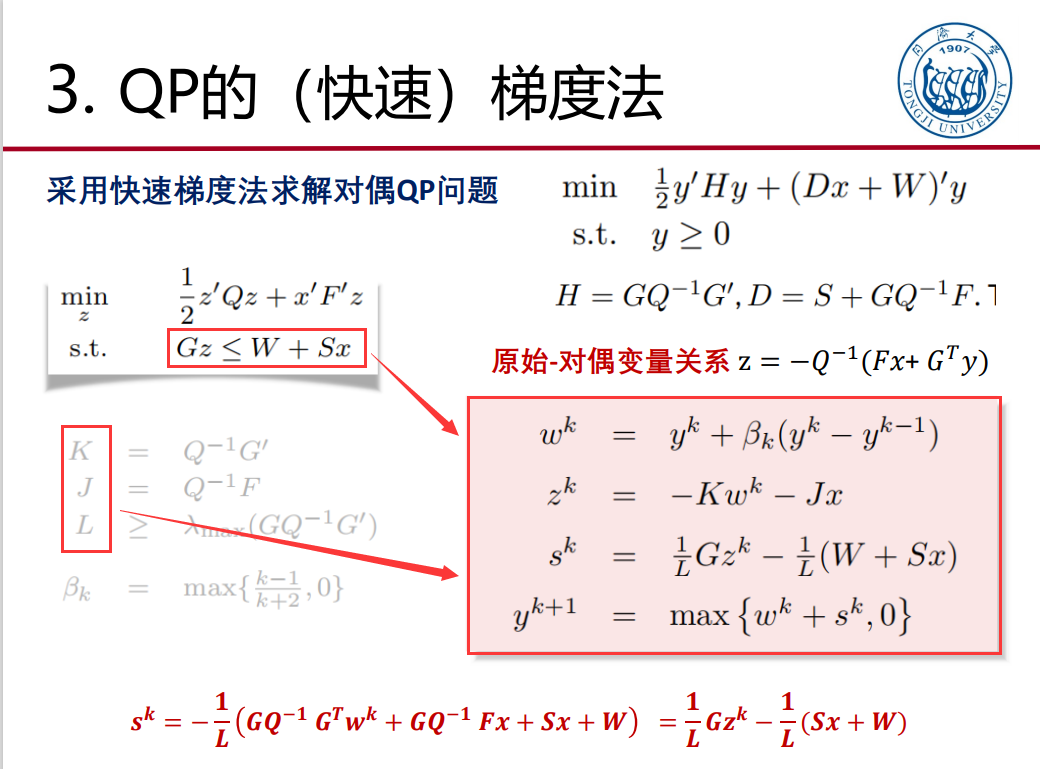
首先，对损失函数L进行二阶泰勒展开，得到一个抛物线的近似。

然后，根据权重差定义权重的更新规则，并用差商线性近似L的二阶导数[1](https://zhuanlan.zhihu.com/p/427712761" \t "https://www.bing.com/_blank)。

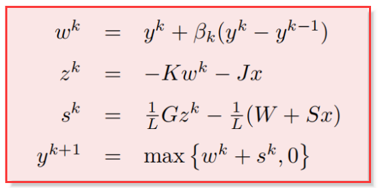
最后，利用当前和之前的权重、权重差和损失斜率，计算新的权重，并重复这个过程直到收敛。

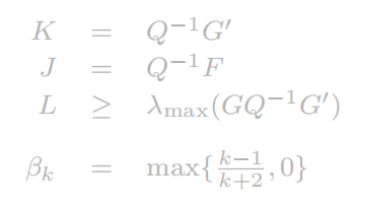




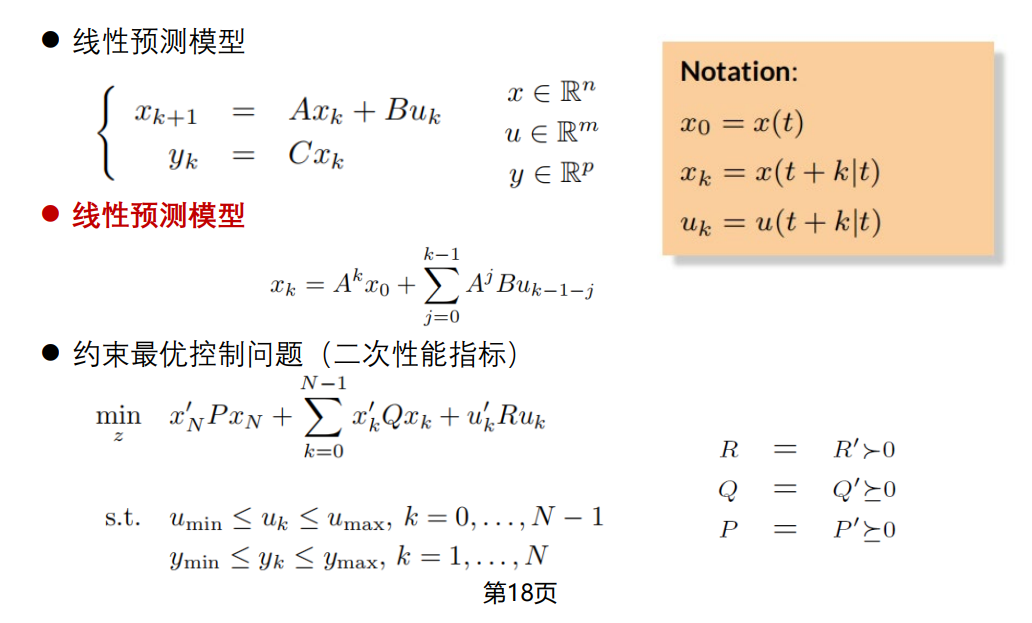


对上述的更新，主要是对在w,z,s,y更新。

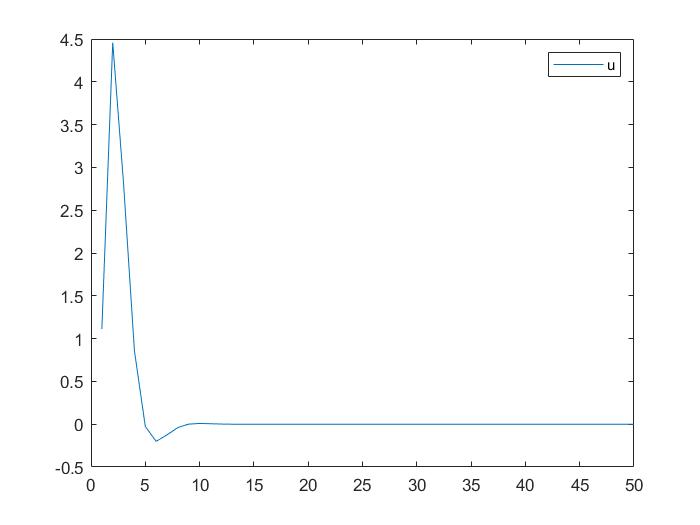
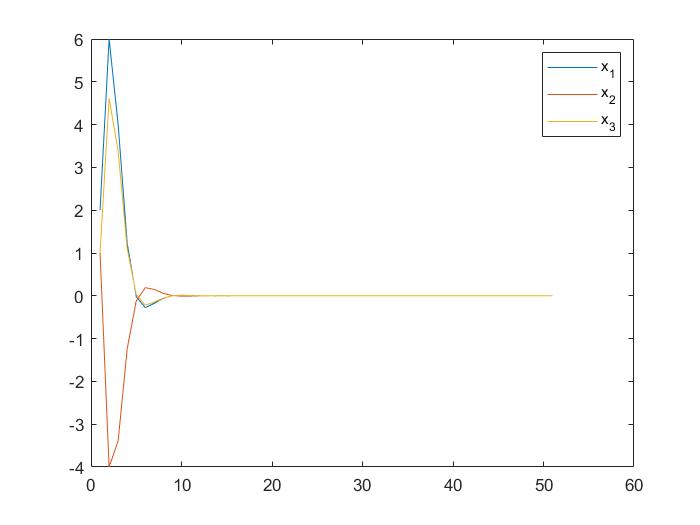








1. **实验结果**



**附录**

1. **main.m**

%2150248 姚天亮 自动化

%MPC：QP加速梯度投影法

% 线性系统系数矩阵

A=[2 2 0; -2 -1 1;0 2 1.5];

B=[0;0;1];

% 初始状态量-如果不能在下一步回到约束范围内，则会造成无解

x0=[2;1;1];

% 预测步长

Np=10;

% 优化目标参数，加权矩阵

Q=eye(3); R=1;

% 转化为用控制量ut表示的，关于状态量的推导方程的矩阵

At=[]; Bt=[]; temp=[];

% 转换后的加权矩阵

Qt=[]; Rt=[];

% 加权矩阵的计算过程，以及推导方程矩阵的叠加过程

for i=1:Np

At=[At; A^i];

Bt=[Bt zeros(size(Bt,1), size(B,2));A^(i-1)\*B temp];

temp=[A^(i-1)\*B temp];

Qt=[Qt zeros(size(Qt,1),size(Q,1));zeros(size(Q,1),size(Qt,1)) Q];

Rt=[Rt zeros(size(Rt,1),size(R,1));zeros(size(R,1),size(Rt,1)) R];

end

% 控制量ut的上下限

lb=-10\*ones(Np,1);

ub=10\*ones(Np,1);

% 转换后的优化目标函数矩阵，循环优化函数中H后的表达式为优化目标的另一项

H=2\*(Bt'\*Qt\*Bt + Rt);

% 转换后的优化中的不等式约束左边系数矩阵，后面循环中的bi为不等式右边

Ai= eye(Np);

u=[]; % u用来保存每一步采用的控制量

x=x0; % x用来保存状态变量

xk=x0; % 保存当前时刻的状态

%此处为增加的梯度投影下降法使用到的矩阵

G=zeros(Np);

W=[20;10;20;zeros(7,1)];%此处的含义是为了满足Gz<=W+Sx，但是为了补足可乘，所以为10\*1，多余补零

S=[eye(3);zeros(7,3)];%注意此处为了配合S\*x，需要补成10\*3的矩阵

for k=1:50

% 进行二次优化

C=(2\*At'\*Qt\*Bt)';%与之前不同的是，本处因为采用到一阶系数矩阵，所以不能有xk，故不乘xk（测试过程中如果加上xk结果会出错）

[ut]=quickqp2(H,C,xk,G,W,S);

%此处为QP以加速梯度投影法写成的求解器

% 采用优化得到的控制量的第一个元素作为实际作用的控制量，代入到原系统中得到下一个时刻的状态量

u(k) = ut(1);

x(:, k+1) = A\*x(:, k) + B\*u(k);

xk = x(:, k+1);

end

%打出图像

figure();

plot(x');

legend('x\_1','x\_2','x\_3');

figure();

plot(u);

legend('u');

1. **quickqp.m**

%2150248 姚天亮 自动化

function [z]=quickqp(Q,c,x,G,W,S)

% H为二次项Q

% f为一次项c’

%A，l，u为约束条件 H即是Q

n = size(Q,1);

% w = ones(n,1);

% z = ones(n,1);

% s = ones(n,1);

y=ones(n,1);

maxiter = 10000;

k=0; %迭代计数

K=inv(Q)\*G';

J=inv(Q)\*c;

L=max(eig(G\*inv(Q)\*G'));

y\_1=y; %记录上一次y

while k<maxiter

beta=max((k-1)/k+2,0);

w=y+beta\*(y-y\_1);

z=-K\*w-J\*x;

s=(1/L)\*G\*z-(1/L)\*(W+S\*x);

y\_1=y;

y=max(w+s,0);

k=k+1;

end