4. Matlab代码实现

> DR_CAN代码

之前我们说到，MPC的研究对象一般是离散形式的状态空间方程，因为计算机程序的计算是离散形式的。

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k) \qquad (1)$$

一般情况下，系统的控制目标是使 $k+1$ 时刻，系统的状态最接近期望状态，这时候我们引入误差的概念：

$$e=x_d-x_{k+1} \qquad (2)$$

详细matlab代码见Github仓库。

在上一小节，我们最终推导出的代价函数为：

$$J=\boldsymbol{x}_k^T\boldsymbol{G}\boldsymbol{x}_k+2\boldsymbol{x}_k^T\boldsymbol{E}\boldsymbol{U}_k+\boldsymbol{U}_k^T\boldsymbol{H}\boldsymbol{U}_k \qquad (3)$$

在MATLAB中，我们使用 `quadprog()` 函数来求解这个二次规划问题。

> MATLAB 中的 `quadprog` 函数用于解决二次规划问题，这类问题的一般形式如下：
>
> $\min \frac{1}{2}x^THx+f^Tx$
>
> 这里 `x` 是需要找到的解向量，`H` 是一个对称正定的矩阵，也称为 Hessian 矩阵，`f` 是与 `x` 同维度的向量。

于是，在 `Prediction()` 函数中的最关键一步，求解 $\min J$ 的代码为：

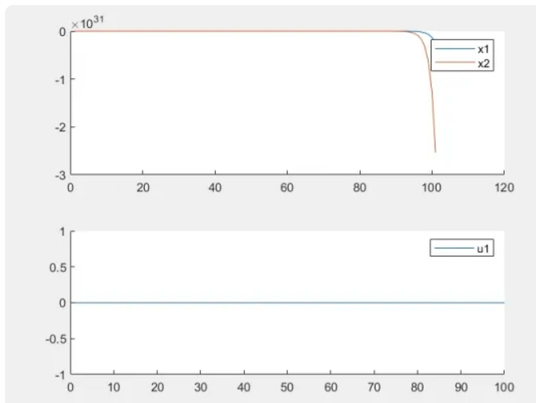
Prediction.m MATLAB

```matlab
    % Solve the quadratic programming problem
    U_k = quadprog(2*H, 2*E'*x_k);
```

在代码中，我们设置 x_1, x_2 的期望值都为0。x1的初值为20，x2的初值为−20。

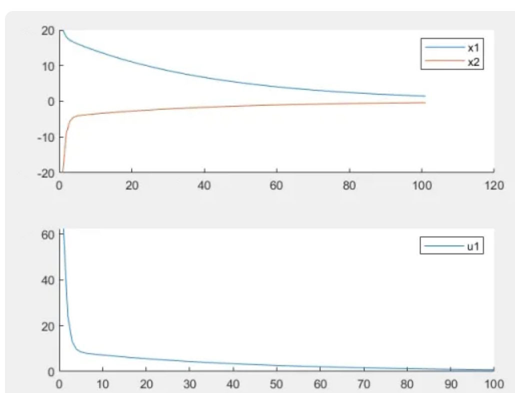
第一种情况：

不进行MPC控制，当 $\boldsymbol{u}(k)=0$ 时，系统的状态：



第二种情况：

1. 预测步长 $N=5$ ：



第三种情况：

我们将系统变为一个多输入的系统：

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.2 & 1 \\ 0.5 & 2 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

同时在代码中修改A, B, R矩阵：

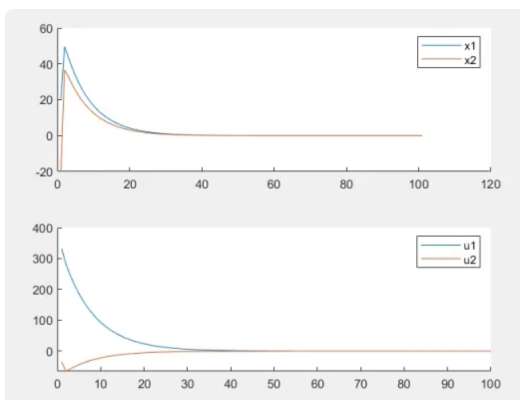
MPC_Test.m MATLAB

```matlab
%% 第一步，定义状态空间矩阵
% 定义状态矩阵 A, n x n 矩阵
A = [1 0.1 ; -1 2];
n = size(A, 1);

% 定义输入矩阵 B, n x p 矩阵
B = [0.2 1 ; 0.5 2];
p = size(B, 2);

Q = [1 0 ; 0 1];   % Q:状态变量权重矩阵, n x n 矩阵
F = [1 0 ; 0 1];   % F:终端误差权重矩阵, n x n 矩阵
R = [0.1 0 ; 0 0.1];         % R:控制变量权重矩阵, p x p 矩阵
```

效果：



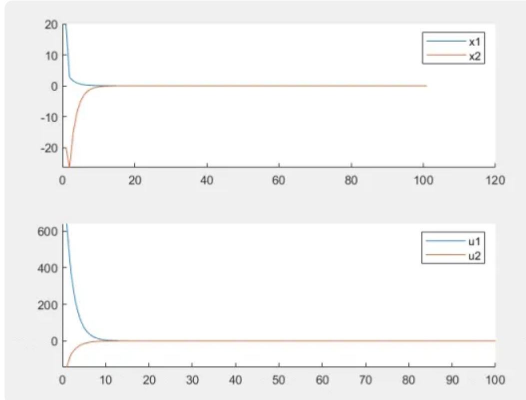
第四种情况：

在上面的基础上，修改权重系数，例如：

MPC_Test.m MATLAB

```matlab
Q = [100 0 ; 0 1];   % Q:状态变量权重矩阵, n x n 矩阵
F = [100 0 ; 0 1];   % F:终端误差权重矩阵, n x n 矩阵
```

这时候，我们更注重状态x1的变化。



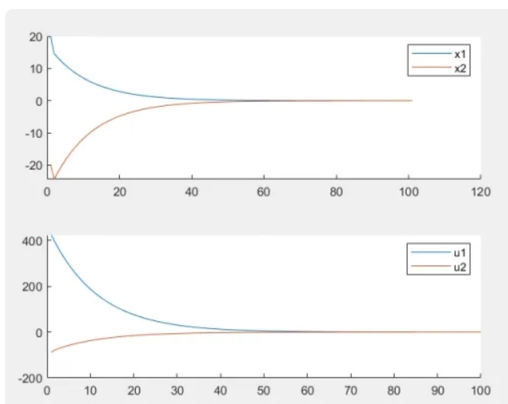
由上图可知，由于我们增大了x1的状态权重，x1在迭代过程中会更加迅速地趋近于0，且在开始时，u1的控制输入远大于u2。

第五种情况：

在上面的基础上，我们修改R矩阵：

MPC_Test.m MATLAB

```matlab
R = [1 0 ; 0 0.1];          % R:控制变量权重矩阵, p x p 矩阵
```

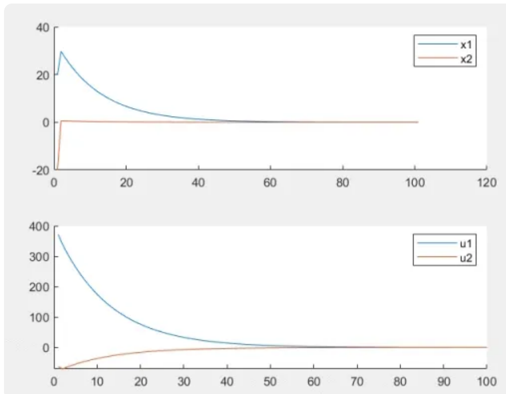


增大了控制矩阵中u1的权重系数，u1的收敛速度变慢了，但u1的输入也变慢了，这减小了系统的能耗（控制量越大，系统越耗能）。

第六种情况：

我们加大状态量x2的权重系数：

```
MPC_Test.m                                                MATLAB
1 Q = [1 0 ; 0 100];  % Q:状态变量权重矩阵，n x n 矩阵
2 F = [1 0 ; 0 100];  % F:终端误差权重矩阵，n x n 矩阵
3 R = [0.1 0 ; 0 0.1];          % R:控制变量权重矩阵，p x p 矩阵
```

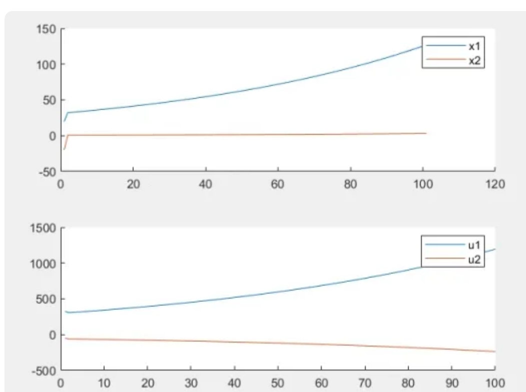


可以看到，这时候，x2的收敛速度明显快于x1。

第七种情况：

如果在**情况7**中，我们还是觉得控制量u2的值太高，希望降低u2的值，这时候我们可以通过增加R中u2的权重：

```
MPC_Test                                                  MATLAB
1 Q = [1 0 ; 0 100];  % Q:状态变量权重矩阵，n x n 矩阵
2 F = [1 0 ; 0 100];  % F:终端误差权重矩阵，n x n 矩阵
3 R = [0.1 0 ; 0 1];          % R:控制变量权重矩阵，p x p 矩阵
```



这时候，x2可以收敛到期望状态 $x_2^{desired}=0$ ，但x1却发散了。