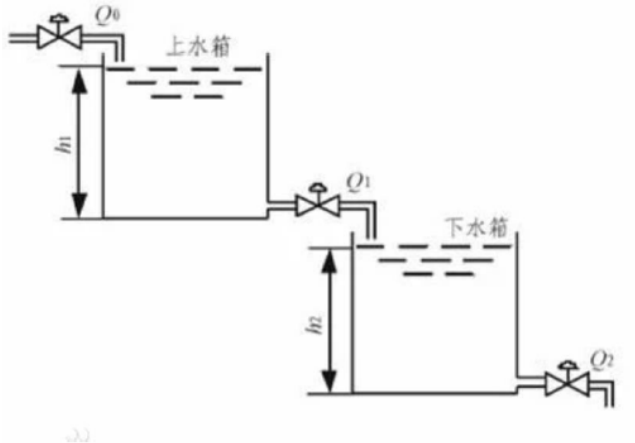
一：基本题目：

7.1 工业双容水箱液位控制系统如下图所示：



上位水箱传递函数为：

$$G_1(s) = \frac{1}{80s+1}e^{-2s}$$

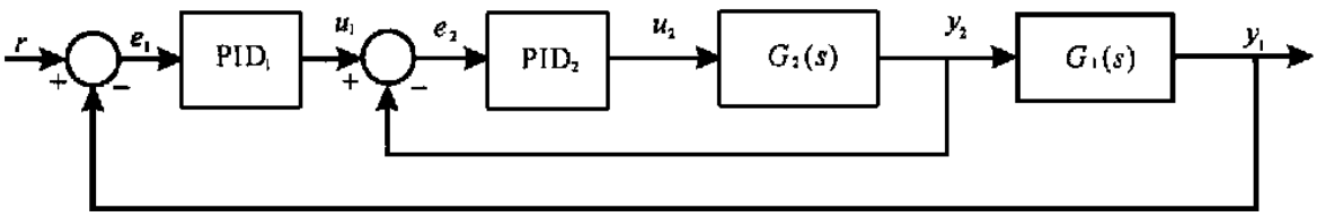
下位水箱传递函数为：

$$G_2(s) = \frac{1}{60s+1}e^{-8s}$$

系统输出期望为单位阶跃。

二：串级 PID 控制器设计：

使用 Matlab 编写脚本 Double_PID.m 对系统进行仿真模拟，设计基本流程图思路如下所示：



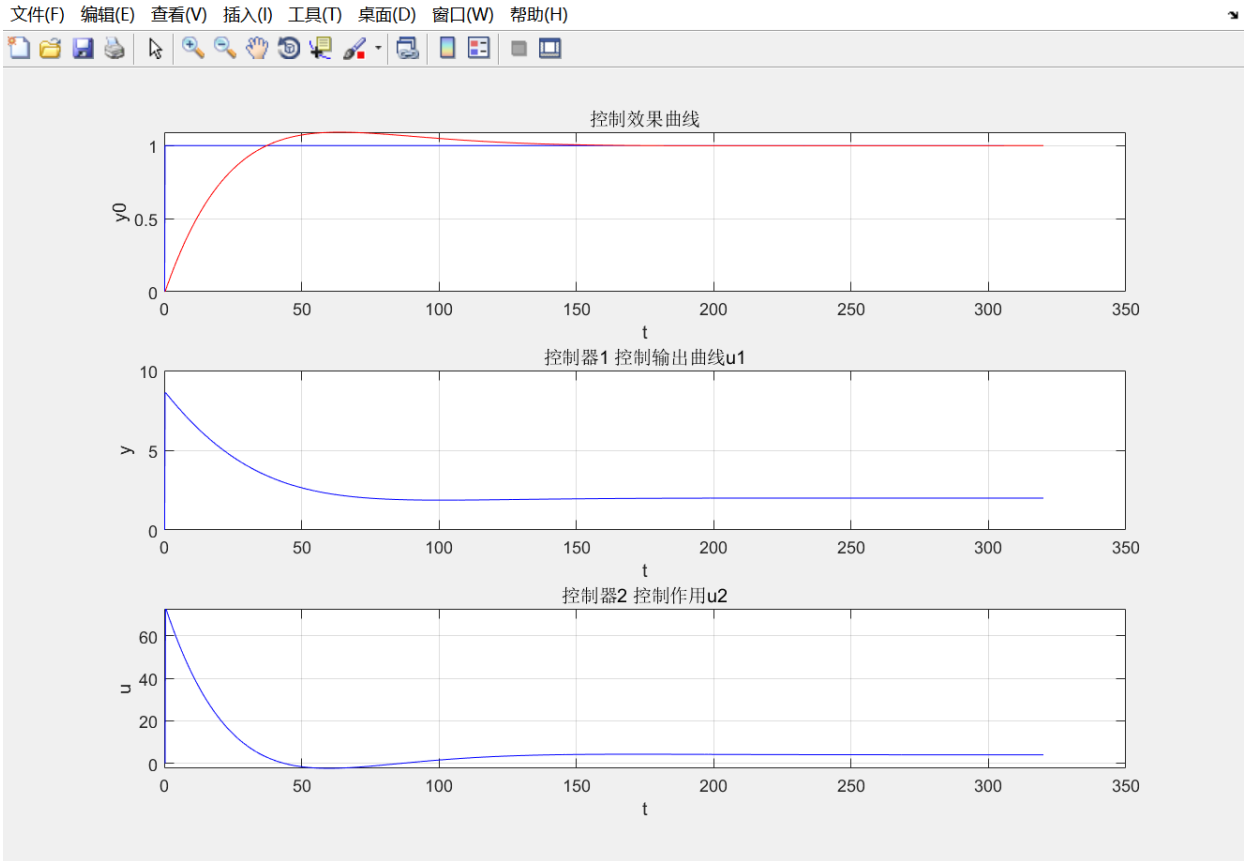
其中，r 为输入的阶跃响应。

根据串级控制器的设计思路，先断开回路，整定副环，尽可能获取较大的比例增益，考虑副环的快速性，我选择了 PI 调节器进行调节，并在前期使用 pid_tune 的可视化功能进行快速高效的调节。

在完成副环副环的整定后，接入副回路，采用单回路的整定方法进行 PID 参数的整定，由于本系统在副环外容积数目不多，选用 PI 控制器即可。这里我选择试凑法进行参数的整定。

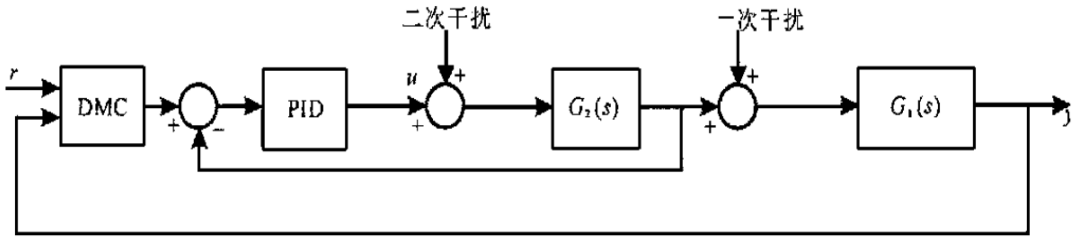
最终效果图如下所示，最终效果图如下所示，图中第一张图像为输出曲线，第二张为 PID1 输出 u1，第三张为 PID2 输出 u2。

从图可以看到，使用串级 PID 系统可以快速地完成阶跃控制，且超调相对较小。



三：DMC-PID 控制器设计：

使用 Matlab 编写脚本 DMC_PID.m 对系统进行仿真模拟，设计基本流程图思路如下所示：



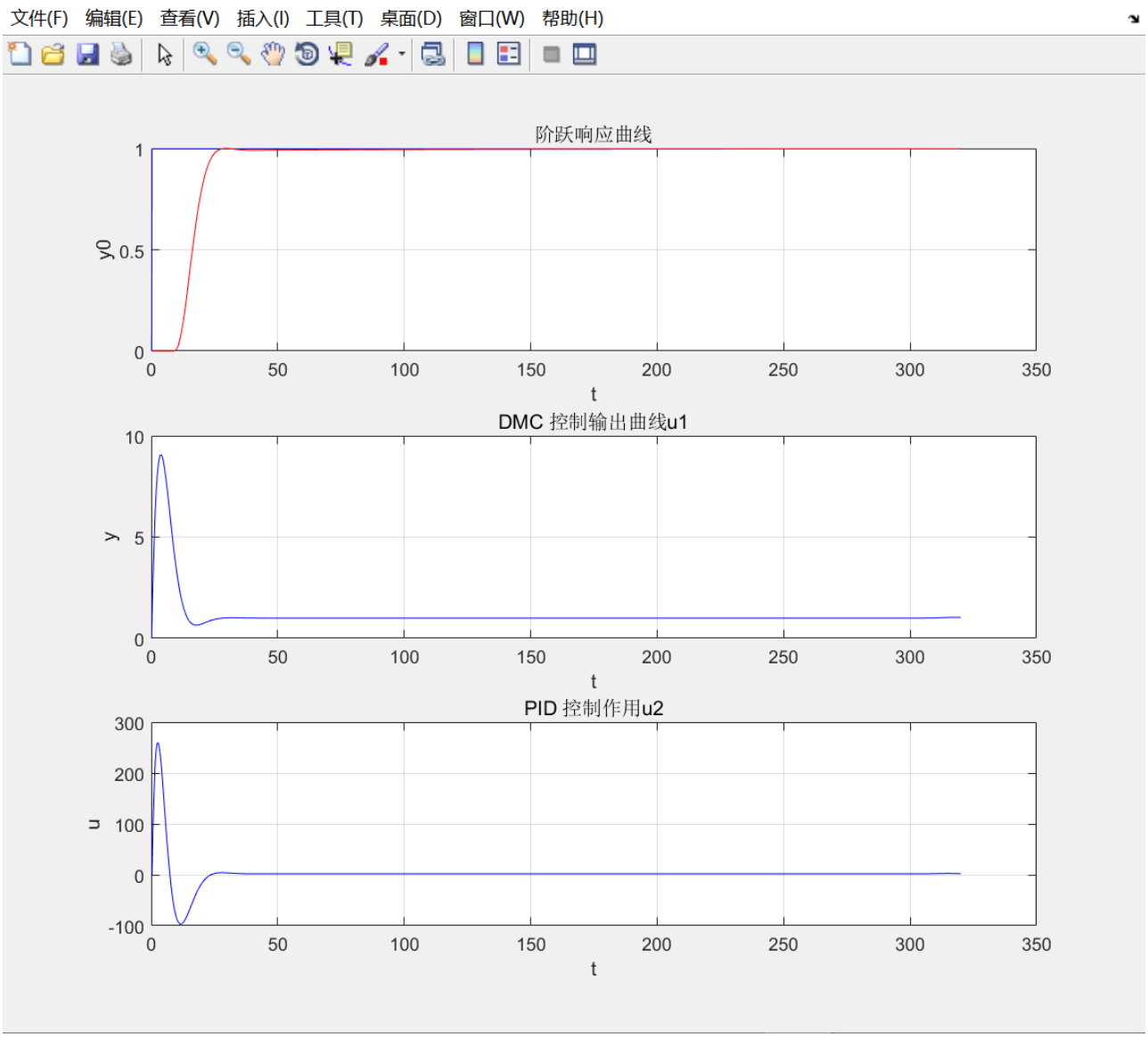
其中，r 为输入的阶跃响应。

根据 DMC_PID 控制器的设计思路，副回路中包含系统的主要扰动，我使用 PID_tune 进行了初步参数的调节。

之后，我参考 DMC.m 进行了 DMC 控制器的设计，根据上课学习的工程参数的整定原则，进行了 P、M 等参数的选择，并将其与 PID 控制器串联构成 DMC-PID 控制系统。

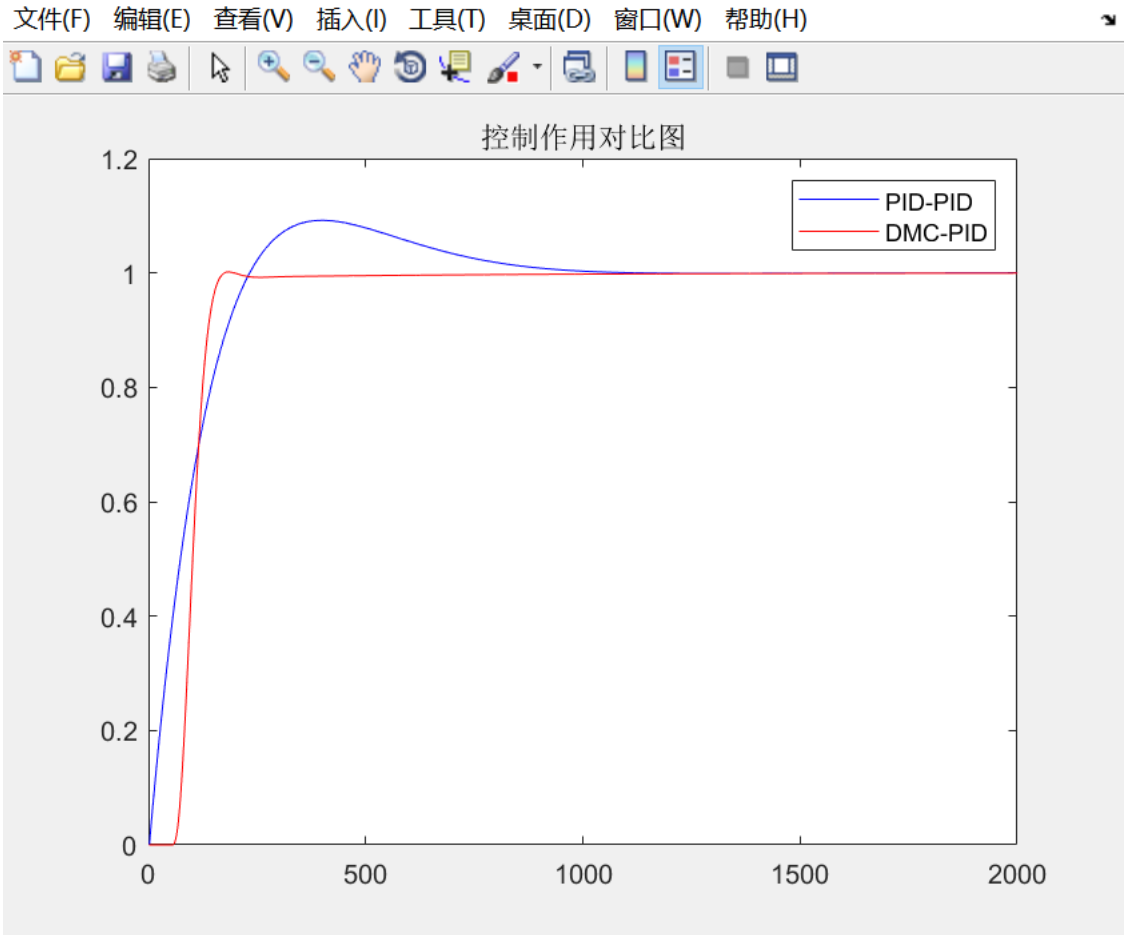
最终效果图如下所示，图中第一张图像为输出曲线，第二张为 DMC 输出 u，第三张为 PID 输出 u2。

可以看到，使用 DMC-PID 系统可以快速地完成对水箱系统的阶跃控制，且超调基本为 0。


文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
阶跃响应曲线
y0
t
DMC 控制输出曲线u1
y
t
PID 控制作用u2
u
t

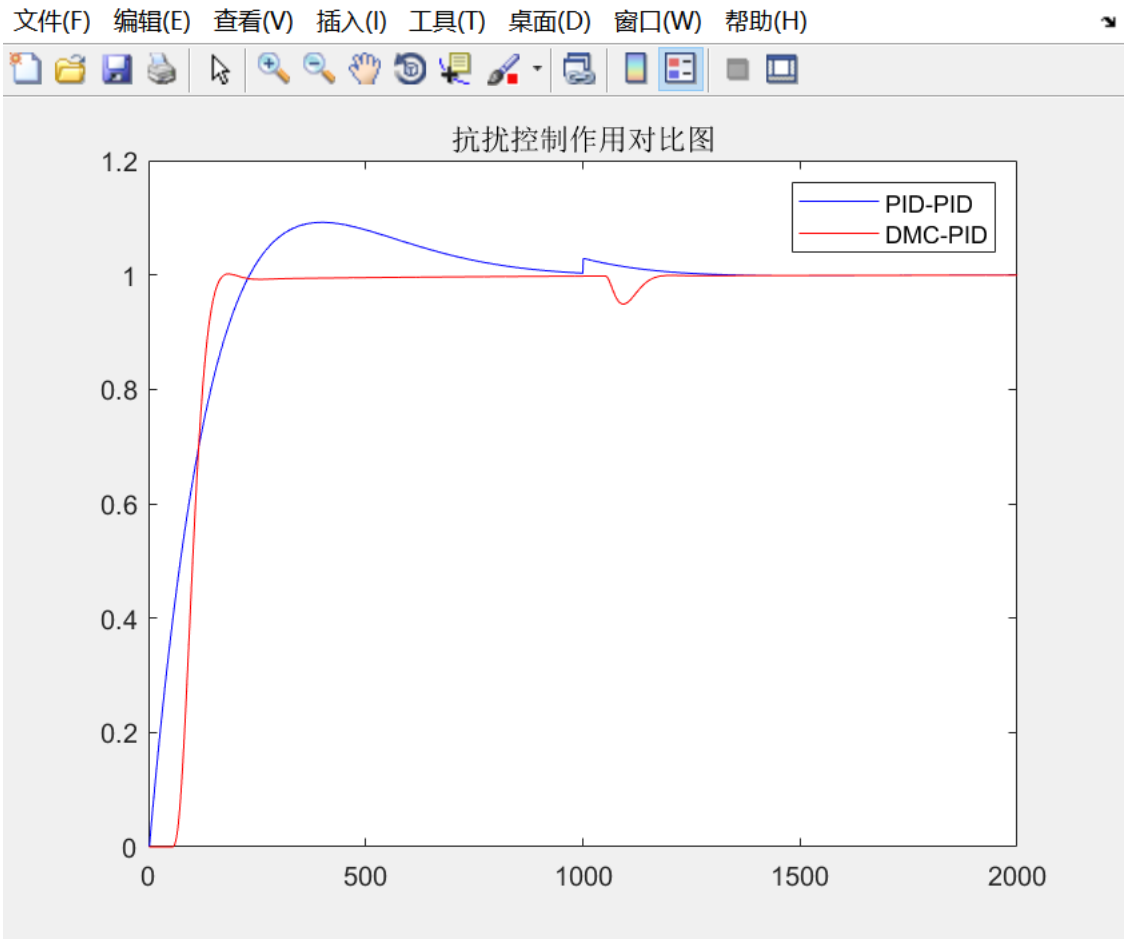
四：两种控制对比

将两种控制效果图放置在同一坐标系进行对比：



显然可以看到 DMC-PID 的控制效果远好于 PID-PID 控制系统，超调比 PID-PID 低了 8.7%，实际测得的上升时间与峰值时间也远小于 PID-PID 控制器，有着更好的动态性能与稳态性能。

接下来，简单探究一下两种方法的抗扰动特性，在某一时刻同时加入阶跃后，两种控制系统的输出效果图如下所示，相比而言，DMC-PID 可以更快的回复到稳定状态。


文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
抗扰控制作用对比图
PID-PID
DMC-PID
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000

```
%%
    %%% DMC_PID 控制器
    %%% 2021.5.28
    %%% @C_zihao
    clear;clc;
    %%  模型对象
    G2 = tf(1, [80 1], 'inputdelay', 2);
    G = tf(1, [60 1], 'inputdelay', 8);
    Ts = 0.16; %  采样时间
    dsys = c2d(G2, Ts, 'zoh');
    [num, den] = tfdata(dsys, 'v');

    %%  设置 DMC 参数
    P = 120; %    预测步长
    M = 50; %  控制步长
    umax = 5000; %  控制量最大值
    delta_umax = 1000; %控制量变化最大值
    N = 2000; %截断步长
    Yrk1 = zeros(1, P);
    yr0 = 1000;
    startvalue = 0; %系统初始输出值
    x1 = startvalue;
    x2 = 0;
    kp = 43; %1.539;
    ki = 0.07; % 0.0399;
    % sys = tf( 1 ,[60 1],'inputdelay',8);
    % [C_pi,info] = pidtune(sys,'Pid');
    for n = 1:P
        x2 = 1;
        x1 = x2;
        Yrk1(n) = x2;
    end
    %%  建立系统的阶跃响应模型
    [y0, x0] = step(G, 0:Ts:Ts * N);

    %初始化 DMC 矩阵 A
    A = zeros(P, M);
    for i = 1:N
        a(i) = y0(i);
    end
```

```
for i = 1:P
    for j = 1:M
        if i - j + 1 > 0
            A(i, j) = a(i - j + 1);
        end
    end
end


A0 = zeros(P, N - 1);
%构造矩阵 A0
for i = 1:P
    for j = N - 2:-1:1
        if N - j + 1 + i - 1 <= N
            A0(i, j) = a(N - j + 1 + i - 1) - a(N - j + i - 1);
        else
            A0(i, j) = 0;
        end
    end
    A0(i, N - 1) = a(i + 1);
end
%% 初始化参数
ys = ones(N, 1);
y = zeros(N, 1);
u = zeros(N, 1);
y_1 = zeros(N, 1);
y_1(2:N) = 1;
e = zeros(N, 1);
y2 = zeros(N, 1);
u2 = zeros(N, 1);
error = zeros(N, 1);
error_i = 0;
pre_y2 = 0;
pre_pre_y2 = 0;
pre_u2 = 0;
pre_pre_u2 = 0;
%% DMC-PID 控制
for k = 2:N
    Uk_1 = zeros(N - 1, 1);
    for i = 1:N - 1
        if k - N + i <= 0
```

```
            Uk_1(i) = 0;
        else
            Uk_1(i) = u(k - N + i);
        end
    end
    Y0 = A0 * Uk_1;
    e(k) = y(k - 1) - Y0(1);

    %期望输出
    Yr = zeros(P, 1);
    for i = 1:P
        Yr(i) = Yrk1(i);
    end
    Ek = zeros(P:1);
    for i = 1:P
        Ek(i) = e(k);
    end
    %求逆
    A2 = zeros(P:1);
    for i = 1:P
        A2(i) = 1 - 0.9^i;
    end
    A3 = A2 * (yr0 - y(k - 1));
    Q = eye(P); Q(1, 1) = 0; Q(2, 2) = 0; Q(3, 3) = 0; Q(4, 4) = 0;
    delta_u = inv(A' * Q * A + eye(M)) * A' * (Yr - Y0); %控制增量计算
    for i = 1:M
        if delta_u(i) > delta_umax
            delta_u(i) = delta_umax;
        end
    end
    for i = 1:M
        if k + i - 1 <= N
            u(k + i - 1) = u(k + i - 1 - 1) + delta_u(i); %控制率的计算
            if u(k + i - 1) > umax
                u(k + i - 1) = umax;
            end
        end
    end


    %PID 反馈回路
```

```
        error = u(k - 1) - y2(k - 1);
        error_i = error_i +error;


        %PID 控制器输出
        u2(k) = pid_solve(kp, ki, 100, error, error_i);
        %离散模型输出
        y2(k) = -den(2) * pre_y2 +num(2) * pre_u2;
        pre_u2 = u2(k);
        pre_y2 = y2(k);
        temp = 0; %设置在 k-j+1 时刻以点的控制率
        for j = 1:N - 1
            if k - j <= 0
                temp;
            else
                if k - j - 1 <= 0
                    temp = temp + a(j) * y2(k - j);
                else
                    temp = temp + a(j) * (y2(k - j) - y2(k - j - 1));
                end
            end
        end
        if k - N <= 0
            y(k) = temp + e(N);
        else
            y(k) = temp + a(N) * y2(k - N) + e(N);
        end
    end
    %% 画图
    t = Ts * (1:N);
    subplot(311);
    plot(t, y_1(1:N), 'b');
    title('阶跃响应曲线');
    xlabel('t');
    ylabel('y0');
    grid on
    hold on
    plot(t, y, 'r')

    subplot(312);
    plot(t, u, 'b');
```

```
title('DMC  控制输出曲线 u1');
xlabel('t');
ylabel('y');
grid on


subplot(313);
plot(t, u2, 'b');
title('PID  控制作用 u2');
xlabel('t');
ylabel('u');
grid on
% load('y.mat')
% plot(y1, 'b')
% hold on
% plot(y, 'r')
% legend('PID-PID','DMC-PID')
% title('控制作用对比图');


%% PID_Demo
function [pid_out] = pid_solve(kp, ki, i_max, error, i_error)
    p_out = kp * error;
    i_out = ki * i_error;
    i_out = constrain(-i_max, i_out, i_max);
    pid_out = p_out + i_out;
end
function [output] = constrain(min, input, max)
    if (max < min)
        disp("minmax wrong")
    end
    if (input > max)
        output = max;
    elseif (input < min)
        output = min;
    else
        output = input;
    end
end
```

```
%%
    %%% DMC_PID控制器
    %%% 2021.5.28
    %%% @C_zihao
    clear;clc;

    %% 模型
    G2=tf( 1 ,[80 1],'inputdelay',2);
    G1=tf( 1 ,[60 1],'inputdelay',8);

    %采样周期
    Ts = 0.16;
    %离散模型
    dsys = c2d(G2,Ts,'zoh');
    [num,den] = tfdata(dsys,'v');
    kp2 = 8.38;
    ki2= 0.0335;
    %离散模型
    dsys = c2d(G1,Ts,'zoh');
    [num2,den2] = tfdata(dsys,'v');
    kp1 = 8.539;
    ki1= 0.0399;
    %PID _tune
    % sys = tf( 1 ,[60 1],'inputdelay',8);
    % [C_pi,info] = pidtune(sys,'Pid');

    N=2000;%截断步长
    %% 初始化
    y=zeros(N,1);
    u=zeros(N,1);
    y1=zeros(N,1);
    y2=zeros(N,1);
    u2=zeros(N,1);
    error=zeros(N,1);
    y0 = zeros(N,1);
    y0(2:N) = 1;
    error_i2 = 0;
    pre_y2 = 0;
    pre_u2 = 0;
    pre_y1 = 0;
    pre_u1 = 0;
```

```
error_i = 0;
pre_y = 0;
pre_u = 0;

%% 串级PID-PID程序
for k=2:N
    % 反馈回路1
    Y0 = 1;
    error = Y0 - y1(k -1);
    error_i = error_i + error;
    %主环PID控制器输出
    u(k) = pid_solve(kp1, ki1 ,100 , error , error_i);
    %反馈回路2
    error2= u(k-1) - y2(k - 1);
    error_i2 = error_i2 +error2;
    %副环PID控制器输出
    u2(k) = pid_solve(kp2, ki2 ,100 , error2, error_i2);
    %由离散模型求解
    y2(k) = -den(2)*pre_y2 +num(2)* pre_u2 ;
    pre_u2 = u2(k);
    pre_y2 = y2(k);
    %离散模型求出输出
     y1(k) = -den(2)*pre_y1 +num(2)* pre_u1 ;
     pre_u1 = u(k);
     pre_y1 = y1(k);
end

%% 画图
t=Ts*(1:N);
subplot(311);
plot(t,y0(1:N),'b');
title('控制效果曲线');
xlabel('t');
ylabel('y0');
grid on
hold on
plot(t,y1,'r')

subplot(312);
plot(t,u,'b');
```

```
title('控制器1  控制输出曲线u1');

xlabel('t');
ylabel('y');
grid on

subplot(313);
plot(t,u2,'b');

title('控制器2  控制作用u2');

xlabel('t');
ylabel('u');
grid on

%% PID_Demo
function [pid_out] = pid_solve(kp ,    ki,    i_max , error, i_error)

    p_out = kp * error;

    i_out = ki * i_error;
    i_out = constrain(-i_max , i_out , i_max);

    pid_out = p_out + i_out;
end

function [output] = constrain(min, input , max)
      if(max<min)
            disp("minmax wrong")
      end
      if(input>max)
            output = max;
      elseif(input<min)
            output = min;
      else
            output = input;
      end
end
```