学院：数据科学与计算机学院 专业：计算机科学与技术
姓名：郑 康 泽 学号：17341213

智能控制与计算智能

第二章作业

一. 题目

求二阶传递函数

$$G_p(s) = \frac{133}{s^2 + 25s}$$

的阶跃响应。取采样时间$1ms$进行离散化。参照程序chap2_1.m，设计专家PID控制器，并进行Matlab仿真。

二. 分析

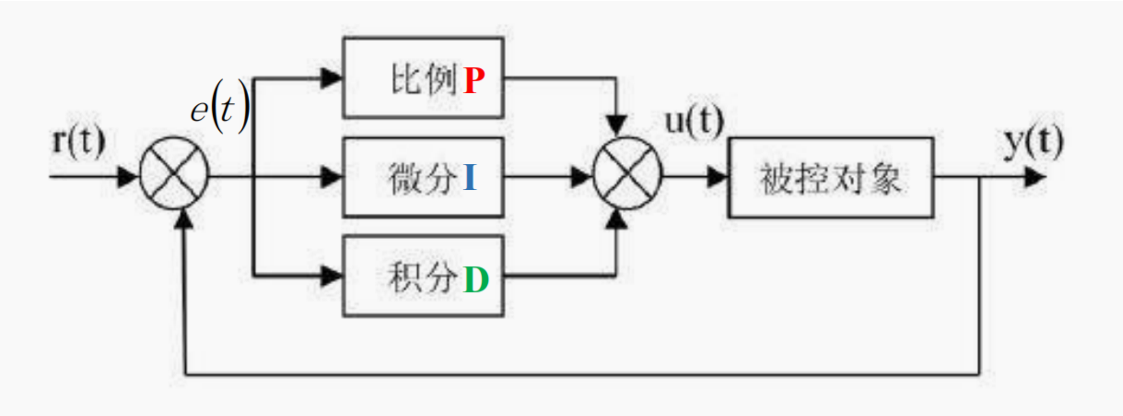
1. PID控制器是一种线性控制器，它将给定的$r(t)$的值与实际输出的$y(t)$的偏差的比例(P)、积分(I)、微分(D)通过线性组合构成控制量，对控制量进行控制。PID调节器公式如下：

   $$u(t) = K_p\left[e(t-1) + \frac{1}{T_I}\int_0^{t-1} e(k)dt + T_D\frac{de(t-1)}{dt}\right]$$

   其中，$e(t) = r(t) - y(t)$。其离散形式有多种，这里我们采用其位置型离散形式，也即

   $$u(t) = k_p e(t-1) + k_i\sum_{i=0}^{t-1} e(i) + k_d\left[e(t-1) - e(t-2)\right]$$

2.



在上图中，$r(t)$是系统的期望输出，$u(t)$是系统的实际输入，也是PID控制器的输出，$y(t)$是系统的实际输出，$e(t)$是系统的期望输出与实际输出之差，PID控制器的任务是根据$e(t)$，调节出一个合适的$u(t)$，使得$y(t)$收敛于$r(t)$。根据PID调节器公式可知，要想得到$u(t)$，我们需要知道$e(t)$，也即知道$r(t)$和$y(t)$，因为想要的响应为阶跃响应，所以$r(t)=1$，所以目前的未知量是$y(t)$。此时就需要利用题目给出的另外一个条件，二阶传递函数为$G_p(s)=\dfrac{133}{s^2+25s}$，首先利用Matlab中的tf函数并离散化，建立传递函数的离散模型

$$G(z)=\frac{\mathrm{num}(2)z+\mathrm{num}(3)}{z^2+\mathrm{den}(2)z+\mathrm{den}(3)}。$$

因为$G_p(s)$中分母的阶数一定大于分子的阶数，所以有$\mathrm{num}(1)=0$和$\mathrm{den}(1)=1$。由传递函数定义：$G(z)=\dfrac{Y(z)}{U(z)}$可得，

$$\begin{aligned}Y(z)=&-\mathrm{den}(2)z^{-1}Y(z)-\mathrm{den}(3)z^{-2}Y(z)\\&+\mathrm{num}(2)z^{-1}U(z)+\mathrm{num}(3)z^{-2}U(z)。\end{aligned}$$

再通过z逆变换，从复数域回到时域，得到以下差分公式：

$$\begin{aligned}y(t)=&-\mathrm{den}(2)y(t-1)-\mathrm{den}(3)y(t-2)\\&+\mathrm{num}(2)u(t-1)+\mathrm{num}(3)u(t-2)。\end{aligned}$$

得到了$y(t)$的值，就可以得到$u(t+1)$的值，之后再应用五条专家规则即可完成对系统输入的控制。

3. 可以看出，计算$y(t)$和$u(t)$需要之前的数据，但一开始我们并没有任何数据，并且传递函数要求是零初始条件，所以我们可以初始化$y(t-1)$，$y(t-2)$，$u(t-1)$，$u(t-2)$，$e(t-1)$，$\sum_{i=0}^{t-1}e(i)$，$e(t-1)-e(t-2)$为零，然后在之后的迭代中，逐步更新这些值。
4. 五条专家规则如下：
    1. 当$|e(t)|>M_1$，说明误差的绝对值很大，令$u(t+1)=$定值。
    2. 当$e(t)\Delta e(t)>0$或$\Delta e(t)=0$时，说明误差的绝对值在增大，或者误差不变，分为以下两种情况：
        1. 当$|e(t)|\geq M_2$，令$u(t+1)=u(t)+k_1*k_p*e(t)$，其中$k_1>1$。

2. 当$|e(t)| < M_2$，令$u(t+1) = u(t) + k_2 * k_p * e(t)$，其中$k_2 < 1$。

3. 当$e(t)\Delta e(t) < 0$且$e(t)\Delta e(t-1) > 0$或者$\Delta e(t) = 0$时，说明误差的绝对值在减少，或者误差不变，此时保持控制器的输出不变，也即利用PID控制器的位置型离散形式

$$u(t+1) = k_p e(t) + k_i \sum_{i=0}^{t} e(i) + k_d \left[e(t) - e(t-1)\right]。$$

4. 当$e(t)\Delta e(t) < 0$且$e(t)\Delta e(t-1) < 0$时，说明误差处于极值状态$e_m(t)$，分为以下两种情况：
   1. 当$|e(t)| \geq M_2$，令$u(t+1) = u(t) + k_1 * k_p * e_m(t)$，其中$k_1 > 1$。
   2. 当$|e(t)| < M_2$，令$u(t+1) = u(t) + k_2 * k_p * e_m(t)$，其中$k_2 < 1$。
5. 当$e(t) \leq \epsilon$，说明误差的绝对值已经很小了，加入积分调节，减少稳态误差。

三. 代码及结果

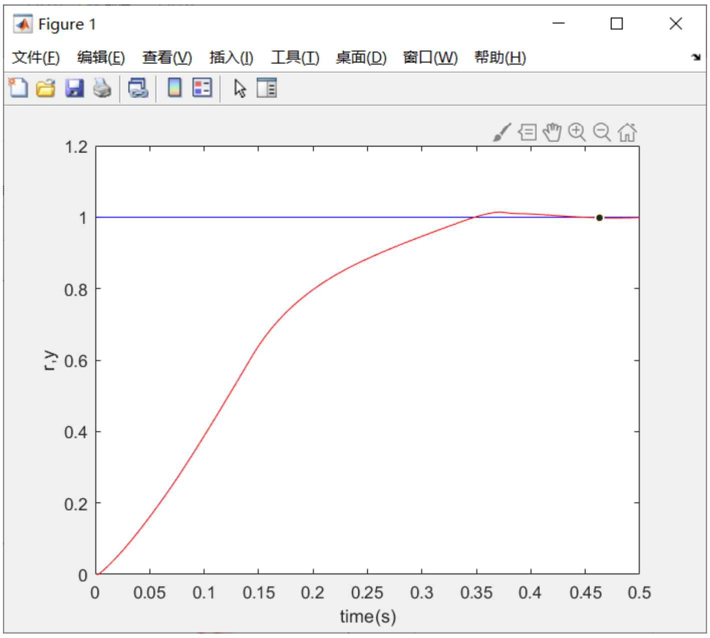
1. 代码：

```
%Expert PID Controller
clear;
clc;

ts = 0.001;                         % 采样时间
step = 500;                         % 迭代次数
r = ones(step, 1);                  % 系统期望输出
time = zeros(step, 1);              % x轴
u = zeros(step, 1);                 % 系统实际输入/PID控制器输出
y = zeros(step, 1);                 % 系统实际输出
error = zeros(step, 1);             % 系统期望输出与实际输出之差

u_1 = 0; u_2 = 0;                   % 前两步系统的输入
y_1 = 0; y_2 = 0;                   % 前两步系统的输出

e = [0, 0, 0]';                     % kp、ki、kd对应的e
e2_1 = 0;                           % 上一步的e(2)

% kp、ki、kd
kp = 5;
ki = 0.03;
kd = 0.01;

sys = tf(133, [1, 25, 0]);          % 建立传递函数模型
dsys = c2d(sys, ts, 'z');           % 离散化该模型
[num, den] = tfdata(dsys, 'v');     % 获取分子分母的系数

for k = 1:step
```

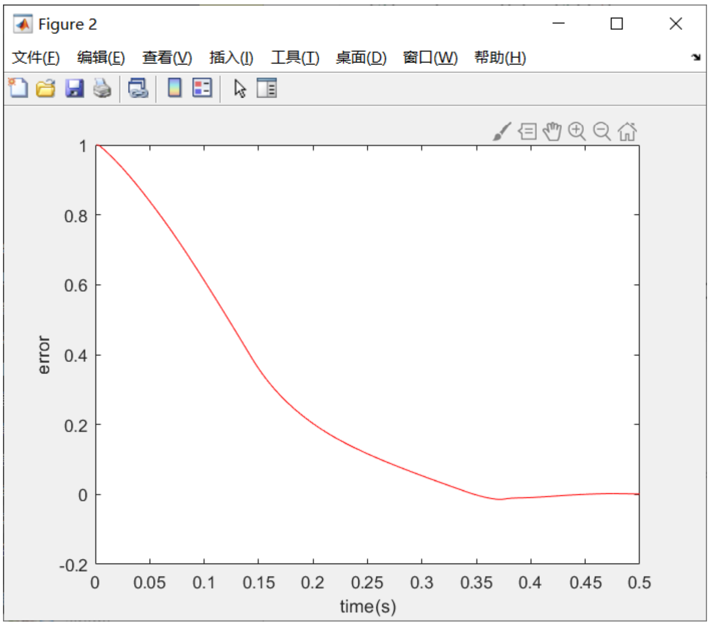
```
    time(k) = k*ts;

    u(k) = kp*e(1) + kd*e(2) + ki*e(3);   % 位置型离散形式

    % 规则一
    if abs(e(1)) > 0.8
       u(k) = 0.45;
    elseif abs(e(1)) > 0.4
       u(k) = 0.5;
    elseif abs(e(1)) > 0.2
       u(k) = 0.12;
    elseif abs(e(1)) > 0.01
       u(k) = 0.1;
    end

    % 规则二
    if (e(1)*e(2)>0) || (e(2)==0)
       if abs(e(1)) >= 0.0125
          u(k) = u_1 + 2*kp*e(1);
       else
          u(k) = u_1 + 0.4*kp*e(1);
       end
    end

    % 规则三
    if (e(1)*e(2)<0 && e(2)*e2_1>0)||(e(1)==0)
        u(k) = 2*u(k);
    end

    % 规则四
    if (e(1)*e(2)<0) && (e(2)*e2_1<0)
       if abs(e(1)) >= 0.0125
          u(k) = u_1 + 2*kp*e(1);
       else
          u(k) = u_1 + 0.6*kp*e(1);
       end
    end

    % 规则五
    if abs(e(1)) <= 0.001
       u(k) = 0.5*e(1) + 0.010*e(3);
    end

    % 限制控制器的输出
    if u(k) >= 10
       u(k) = 10;
    end
    if u(k) <= -10
       u(k) = -10;
    end

% 系统实际输出以及误差
y(k) = -den(2)*y_1 - den(3)*y_2 + num(2)*u_1 + num(3)*u_2;
error(k) = r(k) - y(k);

% 更新参数
u_2 = u_1; u_1 = u(k);
y_2 = y_1; y_1 = y(k);
```

```
e2_1 = e(2);
e(2) = (error(k)-e(1))/ts;
e(1) = error(k);
e(3) = e(3)+error(k)*ts;
end

figure(1);
plot(time,r,'b',time,y,'r');
xlabel('time(s)');ylabel('r,y');
figure(2);
plot(time,error,'r');
xlabel('time(s)');ylabel('error');
```

2. 结果

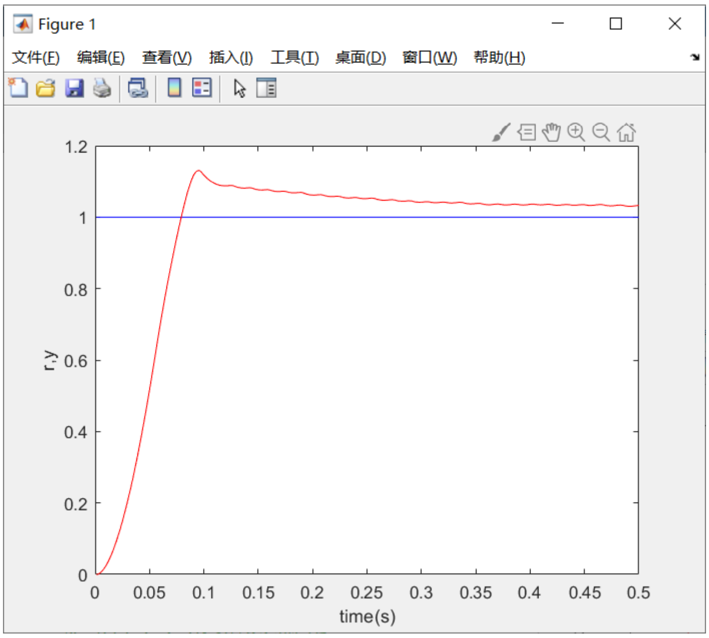


可以看到，time(s)大概为0.35时，系统的实际输出收敛于期望输出。



四. 调参过程

如果直接用chap2_1.m的话，似乎是要迭代将近1000次才能收敛，但是chap2_1.m用于三阶传递函数时，收敛得很快，迭代50次就行了。于是就调参调了一下午，尝试了很多方案，例如修改规则一、二、四中的阈值M_1、M_2，修改k_p、k_i、k_d等等，都没什么太大效果。不过适当提高k_p确实加快了一点收敛。最终，将目标转向了从未修改的规则三，也就是`u(k) = u(k)`，尝试给它加了个系数，也即`u(k) = k*u(k)`，发现当`k`大于1的时候，能明显加快收敛速度，并且`k`越大，收敛越快，但是收敛时的误差也会越来越大，下图是`k=5`的收敛情况：



对比第三部分的图，可以发现确实是如此。不过我感到疑惑的是，规则三说明的是由于误差的绝对值在减少或者不变，所以保持控制器的输出不变，而我在这里给了一个大于1的系数，发现促进了收敛，这个能如何解释呢？