哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

《自动控制实践 B》
综合实验 实验报告

学院 机电工程与自动化学院

姓名 吴俊达

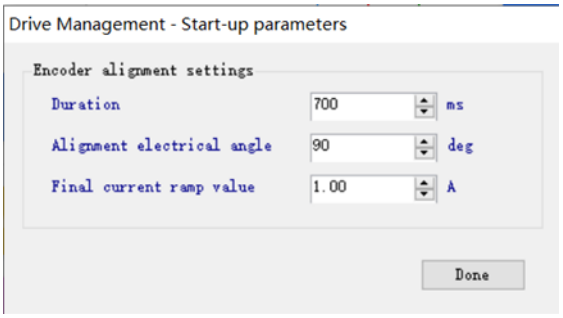
学号 **210320621**

日期 **2024.7.5**

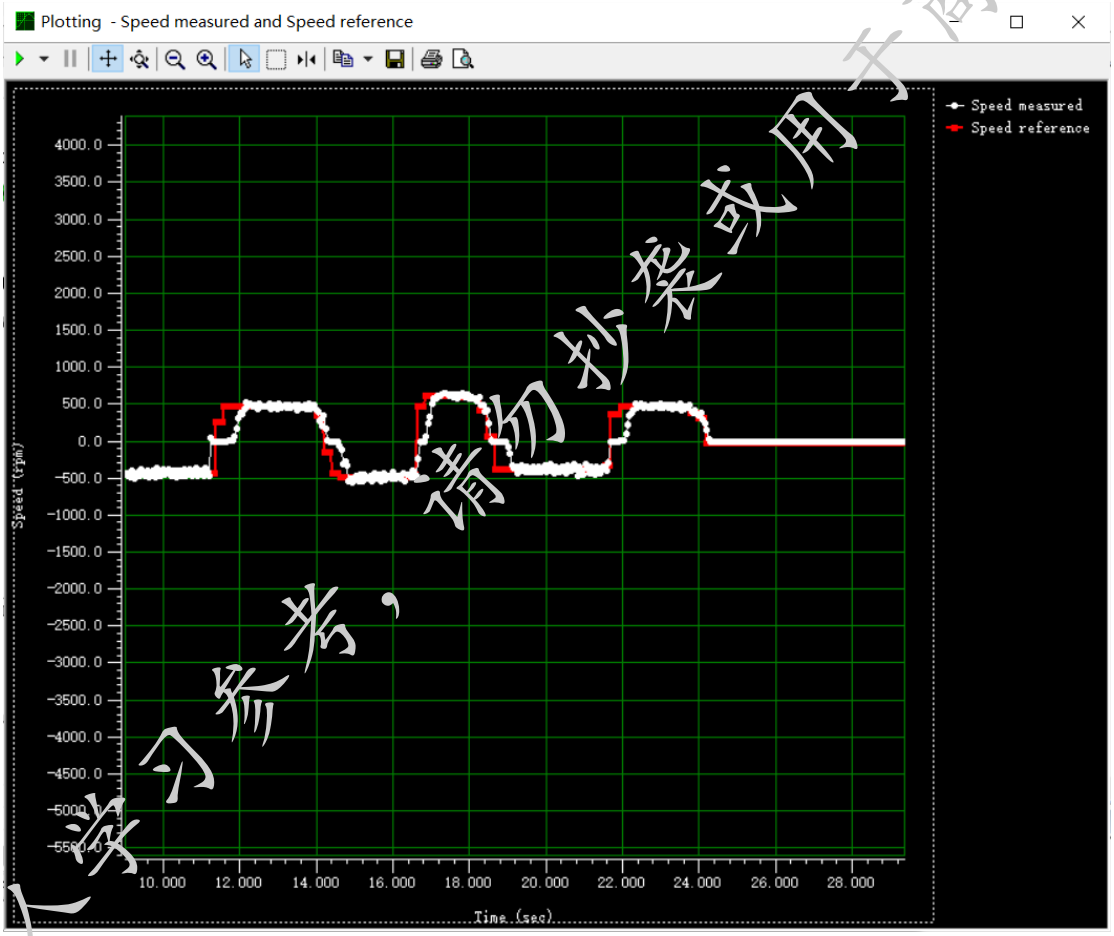
目录

任务1　　电机控制库认知

1.1　Workbench 编码器对齐参数配置



1.2　速度波形显示界面



1.3　说说你对 ST MC SDK5.x（电机控制库）的认知

STMicroelectronics (ST) 的 MC SDK (Motor Control Software Development Kit) 是一套专门用于电机控制的开发工具和库。它的一些主要特性和功能概述如下：

1. 支持多种电机类型

MC SDK 支持无刷直流电机 (BLDC)、永磁同步电机 (PMSM) 和感应电机 (IM) 等多种电机类型，提供多种控制算法以满足不同应用需求。

2. 高效控制算法

MC SDK 包含各种高效的控制算法，如：

- 矢量控制（Field-Oriented Control, FOC）
- 六步换相控制
- 直接转矩控制（Direct Torque Control, DTC）

3. 硬件抽象层

MC SDK 提供了硬件抽象层，使得用户可以在不修改上层应用代码的情况下，更换不同的硬件平台。这一特性使得开发更加灵活和高效。

4. 调试与监控工具

MC SDK 包含多个调试和监控工具，如：

- ST Motor Profiler：用于自动化电机参数测量和控制参数生成。
- ST Motor Pilot：实时监控电机运行状态和调试控制算法。

5. 兼容 STM32 系列微控制器

MC SDK 专门针对 ST 的 STM32 系列微控制器进行优化，提供了对这些 MCU 的全面支持。用户可以利用 STM32 的高性能计算能力和丰富的外设资源，实现复杂的电机控制应用。

6. 扩展和定制

MC SDK 提供了广泛的 API 和示例代码，用户可以基于这些示例进行扩展和定制，满足特定应用需求。

7. 社区和支持

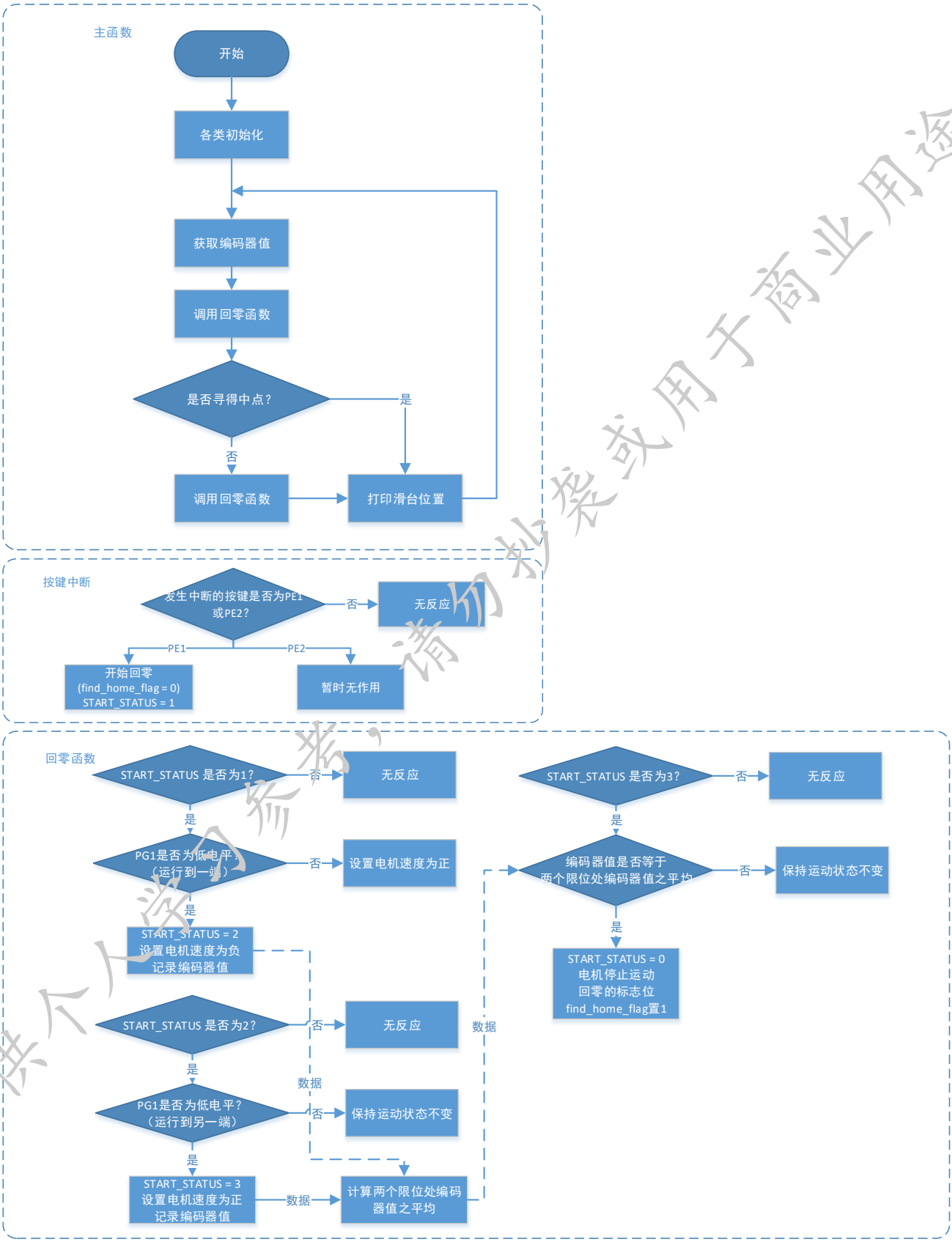
ST 还提供了丰富的文档、教程和社区支持，帮助用户快速上手并解决开发过程中遇到的问题。

8. 集成开发环境

MC SDK 可以与多种集成开发环境（如 STM32CubeIDE、IAR Embedded Workbench 等）配合使用，使开发流程更加顺畅。

任务2　按键控制滑台回零

2.1 程序流程图



2.2 功能函数

此处略去大多数的注释，因为在上述流程图中已经写得非常清楚。

2.2.1 主函数（仅展示 while(1)部分）

```
while (1)
  {
        RegulatedEncoderVal = (short)__HAL_TIM_GET_COUNTER(&htim3); // 获取 TIM3 编码器值，ReturnToZero()函数记录左、右位置时会用到。强制类型转换会使其成为有符号数而无需考虑溢出问题
        ReturnToZero(); // 调用回零函数
        if(find_home_flag == 0 && MotorSpeed != 0){ // 若未寻得中点
            MC_ProgramSpeedRampMotor1(MotorSpeed,5);  // 设置电机速度，速度在 ReturnToZero 函数中指定
            MC_StartMotor1(); % 启动电机
        }
        float pos = RegulatedEncoderVal * 0.005f; // 将编码器值转化为位置
        printf("%f\r\n", pos); // 传回电脑
  }
```

2.2.2 按键中断

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin == GPIO_PIN_1){
    for(uint32_t i = 0; i<500000; ++i); // 软件消抖
    if(HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_1)==GPIO_PIN_RESET){
        START_STATUS = 1; // 回零开始运行
        find_home_flag = 0;
        }
    }
    if(GPIO_Pin == GPIO_PIN_2){
    for(uint32_t i = 0; i<500000; ++i); // 软件消抖
    if(HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_2)==GPIO_PIN_RESET){
        START_STATUS = 4;
        }
    }
}
```

2.2.3 回零函数

```
void ReturnToZero(void){
    if(START_STATUS == 1){
        MotorSpeed = 65; // 朝一方向（左）运动
    }
```

```
    if(HAL_GPIO_ReadPin(GPIOG, GPIO_PIN_1) == GPIO_PIN_RESET && START_STATUS == 1){ //
左限位
        START_STATUS = 2;
        MotorSpeed = 0;
        MC_StopMotor1();
        EncoderLeft = RegulatedEncoderVal;
        MotorSpeed = -65; // 朝另一方向（右）运动
    }
    if(HAL_GPIO_ReadPin(GPIOG, GPIO_PIN_0) == GPIO_PIN_RESET && START_STATUS == 2){ //
右限位
        START_STATUS = 3;
        MotorSpeed = 0;
        MC_StopMotor1();
        EncoderRight = RegulatedEncoderVal;
        EncoderMid = EncoderLeft - (EncoderLeft - EncoderRight)/2;
        MotorSpeed = 65; // 又朝一方向（左）运动
    }
    gap = abs(RegulatedEncoderVal - EncoderMid); // 计算中点与当前编码器值的差异
    if(START_STATUS == 3 && gap<5){
        MotorSpeed = 0;
        MC_ProgramSpeedRampMotor1(0,1);
        MC_StartMotor1();
        START_STATUS = 0;
        find_home_flag =1; // 找到中点
    }
}
```

2.3 实验总结

运行该代码，滑台先向左运动，达到限位后向右运动，达到限位后又向左运动，最后停在中点，则达成了预期的回零目的。

主要问题：溢出的处理方法本是使用 HAL_TIM_PeriodElapsedCallback 回调函数来对溢出的编码器数值进行增减，但没有发挥预期的作用。后来在老师和同学的建议下使用了现在的方法。

任务3　控制系统设计

3.1 控制系统建模

3.1.1 机械谐振模态分析

固有频率 $\omega_{\mathrm{m}}=\sqrt{\dfrac{K}{J}}$，其中 J 为转动惯量。滑台的转动惯量是

$$J_{\text{滑台}}=540\times10^{-3}\times\left(\frac{10\times10^{-3}}{2\pi}\right)^2=1.37\times10^{-6}\,\mathrm{kg\cdot m^2}$$

丝杠的转动惯量是

$$J_{\text{丝杆}}=\frac{1}{2}MR^2=\frac{1}{2}\times600\times10^{-3}\times\left(\frac{15\times10^{-3}}{2}\right)^2=16.875\times10^{-6}\,\mathrm{kg\cdot m^2}$$

又知电机的转动惯量是 $J_{\text{电机}}=28\times10^{-6}\,\mathrm{kg\cdot m^2}$，则总转动惯量是

$$J=J_{\text{滑台}}+J_{\text{丝杆}}+J_{\text{电机}}=46.245\times10^{-6}\,\mathrm{kg\cdot m^2}$$

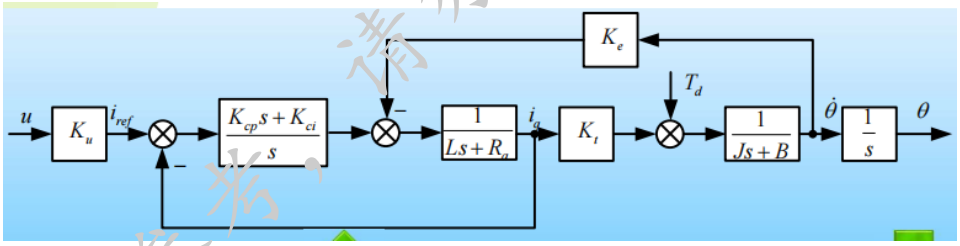
则由 K=950Nm/rad，有

$$\omega_{\mathrm{m}}=\sqrt{\frac{K}{J}}=\sqrt{\frac{950}{46.245\times10^{-6}}}=4.53\times10^{3}\,\mathrm{rad/s}$$

该系统所要求的闭环带宽数量级在 10Hz 以下，机械谐振频率距离截止频率很远，可以不用考虑。

3.1.2 被控对象的数学建模

方框图如下：



交流永磁同步电机伺服系统

经过降阶与非线性处理、平均化等，可以得出系统的传递函数大致为

$$P(s)=\frac{K_0}{s(\tau_e s+1)(\tau_m s+1)}$$

若两极点相距很远，也可以用一阶模型来等效。

3.2 控制系统辨识

3.2.1 在主控板上实现正弦扫频信号生成算法

```
int init_my_sweep(my_sweep_t *sweep, unsigned int t_0, unsigned int t_01, float f0,
float f1, float A)
{
    if ((t_01 == 0) || (f0 <=0.0f) || (f1 <= 0.0f) || (f0 == f1) || (A == 0) ||
(!sweep))
    {
```

```
        return -1; /*非法入参*/
    }
    sweep->t_0 = t_0;
    sweep->t_01 = t_01;
    sweep->f0 = f0;
    sweep->f1 = f1;
    sweep->A = A;

    sweep->k=exp(1/(0.001*t_01)*log(f1/f0));/*计算指数函数的底数 k，时间的单位转换为秒*/
    sweep->p = 2*3.1415926*f0/(log(sweep->k));/*计算系数 p*/
    return 0;
}

float run_my_sweep(my_sweep_t *sweep, unsigned int t_now)
{
    float t = 0.0f; //相对时间 t
    float y = 0.0f; //扫频信号
    if (!sweep) return 0.0f; /*非法入参*/
    if (t_now < sweep->t_0)
    {
        return 0.0f; /*时间还未得到*/
    }
    t = (t_now - sweep->t_0) % sweep->t_01; /*通过求余操作实现，周期性扫频的过程*/
    t = t * 0.001f; /*将单位转换为 s*/
    y = sweep->A * sin(sweep->p*(pow(sweep->k,t)-1));
    return y;
}
```

3.2.2 在主控板上实现扫频辨识功能

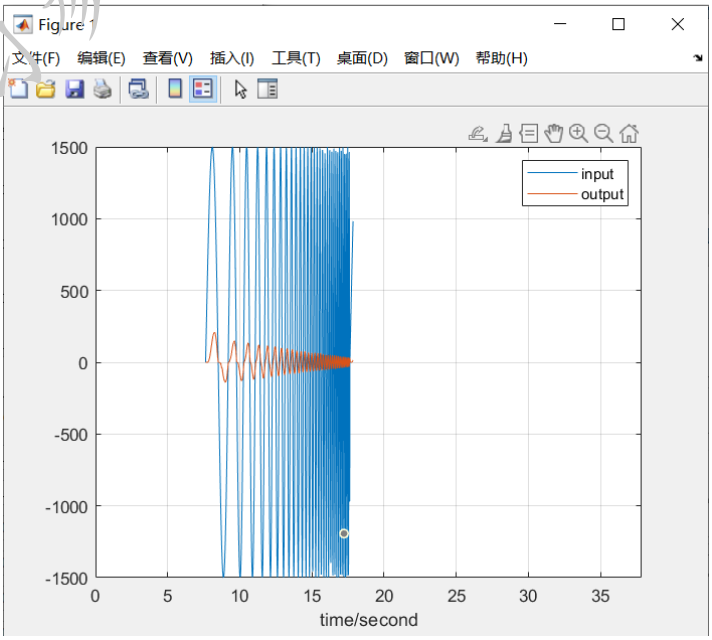
```
//*实现扫频辨识功能，该功能将被放置在 main 函数的 while(1)中运行*/
void function_sweep_identification(void)
{
    static my_sweep_t sweep = {0};
    int16_t sweep_input = 0;
    int16_t sweep_output = 0;
    uint32_t sys_tick = 0;
    static uint32_t init_flag = 0;
    static uint32_t last_sys_tick = 0;
    static uint32_t start_sys_tick = 0;
```

```
    // 频率在 10s 内 ，从 0.5hz 变化到 10hz，幅度为 1500 digit current
    uint32_t t_period_ms = 10*1000; //10s
    float f0 = 0.5;
    float f1 = 10;
    float Amp = 1500.0f;
    float time = 0.0f;
    sys_tick = HAL_GetTick(); //获取当前时刻，单位 ms
    time = 0.001f * sys_tick; //单位 s
    /*进入的条件是回零成功，且按了运行键（与提供的代码不同，此处改用 K3，见 2.2.2 节） */
    if ((find_home_flag == 1) && (MC_GetSTMStateMotor1() == RUN))
    {
        if (last_sys_tick != sys_tick) //如果当前时刻发生了变化，这个条件每 ms 都成立一次
        {
            last_sys_tick = sys_tick;
            if (sys_tick % 10 == 0) //通过 % 把频率从 1000hz 降低到 100hz，即每 10ms 发
生一次
            {
            //初始化扫频配置
                if (init_flag == 0)
                {
                    init_my_sweep(&sweep, sys_tick, t_period_ms, f0, f1, Amp);
                    printf("sweep-
init:k=%.5f,p=%.5f\r\n",(float)sweep.k,(float)sweep.p);
                    init_flag = 1;
                }
                //获取正弦扫频信号
                sweep_input = (int16_t)run_my_sweep(&sweep, sys_tick);
                //将正弦扫频信号输入到 ST MC SDK 的力矩控制 API 中
                MC_ProgramTorqueRampMotor1(sweep_input,0);
                //获取丝杆的转速信息，单位为 0.1hz
                sweep_output = MC_GetMecSpeedAverageMotor1();
                //把时间，input，output 发送到 matlab
                send_data_2_matlab(time,(float)sweep_input,(float)sweep_output);
            }
        }
    }
}
```

用于将数据发送到 MATLAB 上位机的代码是：

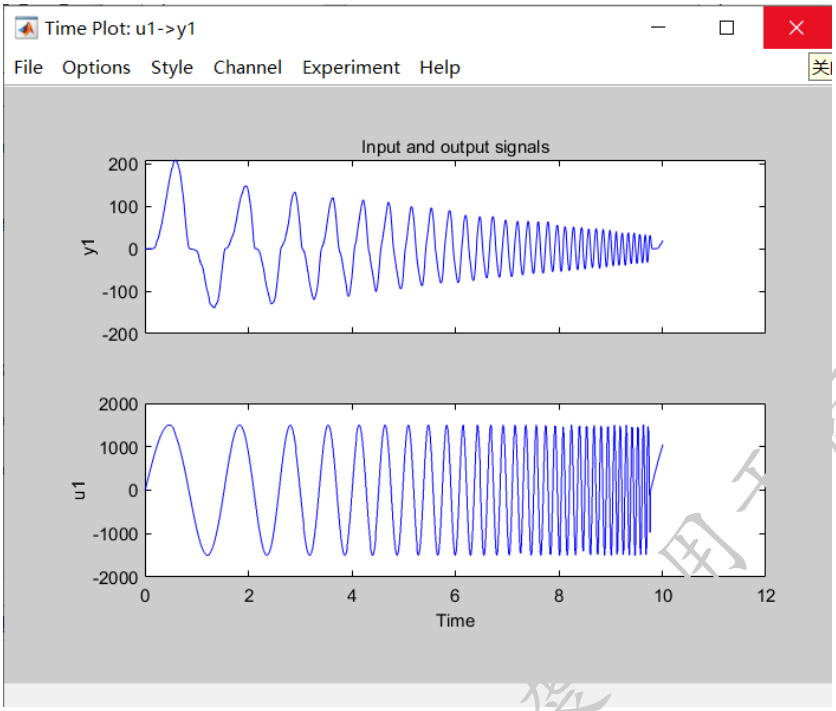
```
/* 对 uint8_t 数值进行求和 ,获得这组数据一个 uint8_t 特征*/
uint8_t get_uint8_sum_check(uint8_t *data, int len)
{
    int i = 0;
    uint8_t sum = 0;
    for (i = 0; i < len; i++) sum += data[i];
    return sum;
}
/*
函数：send_data_2_matlab
功能：往 matlab 发送三个浮点数
输入：三个浮点数
输出：无
*/
void send_data_2_matlab(float data1, float data2, float data3)
{
    frame_matlab_t frame = {0};
    /*填充帧头*/
    frame.start_flag = 0xAA;
    frame.frame_len = sizeof(frame);
    frame.header_check = get_uint8_sum_check((uint8_t *)&frame, 2);
    /*填充数据*/
    memcpy((uint8_t *)&frame.data_buf[0], (uint8_t *)&data1, 4);
    memcpy((uint8_t *)&frame.data_buf[4], (uint8_t *)&data2, 4);
    memcpy((uint8_t *)&frame.data_buf[8], (uint8_t *)&data3, 4);
    /*计算数据求和值,用于接收方校验数据的完好性*/
    frame.frame_check = get_uint8_sum_check((uint8_t *)&frame, frame.frame_len-1);
    /*通过 串口发送到电脑 */
    HAL_UART_Transmit(&huart1, (uint8_t *)&frame,frame.frame_len,0xffff);
}
```

扫频辨识过程中 Matlab 采集波形：



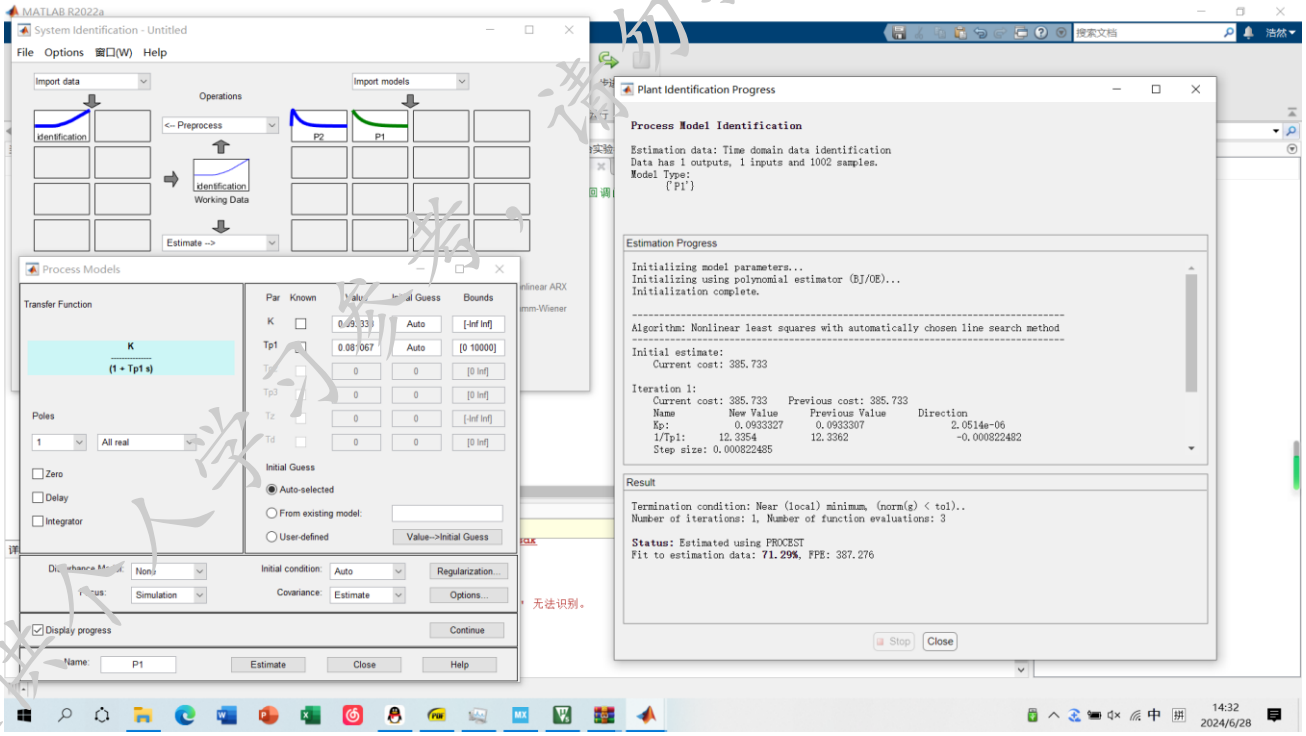
3.2.3　使用 MATLAB 系统辨识工具箱，获得辨识的系统模型

一、将输入输出数据导入 MATLAB 系统辨识工具箱后，Time Plot 查看输入输出曲线：

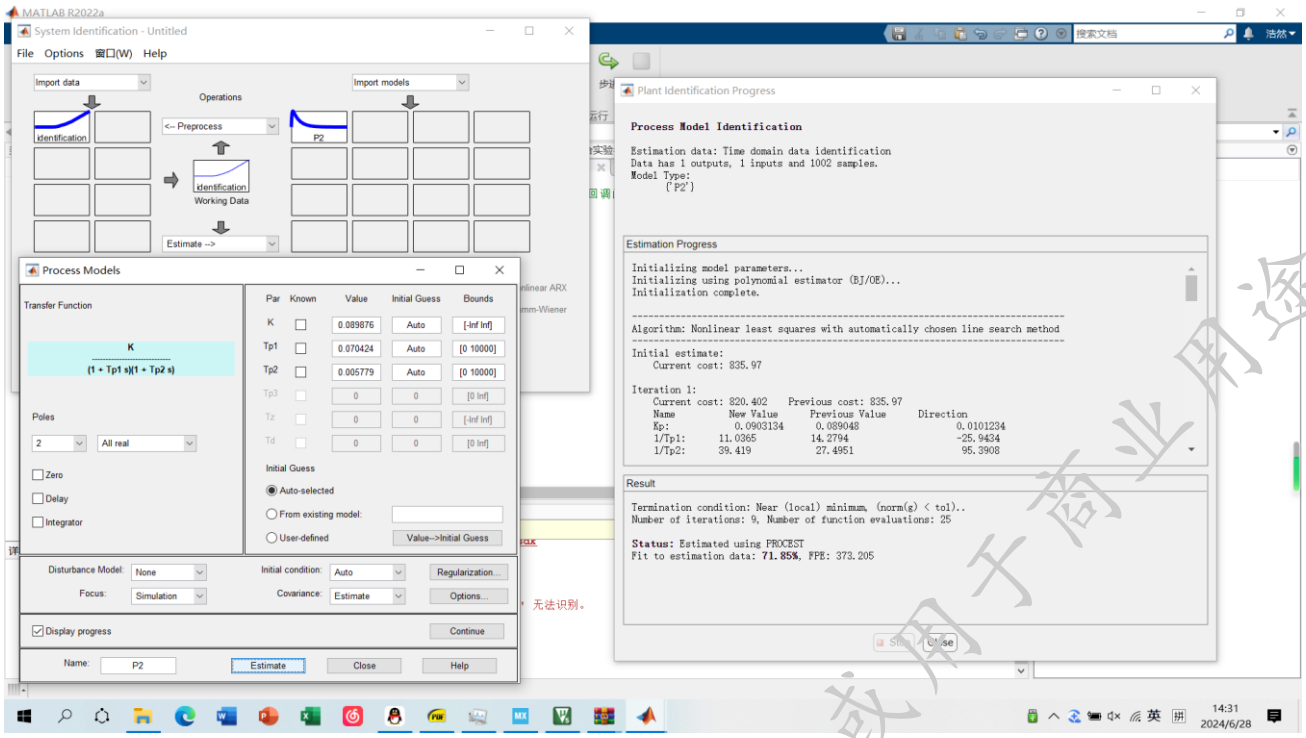


上方为输出 下方为输入

二、参数辨识计算过程截图：

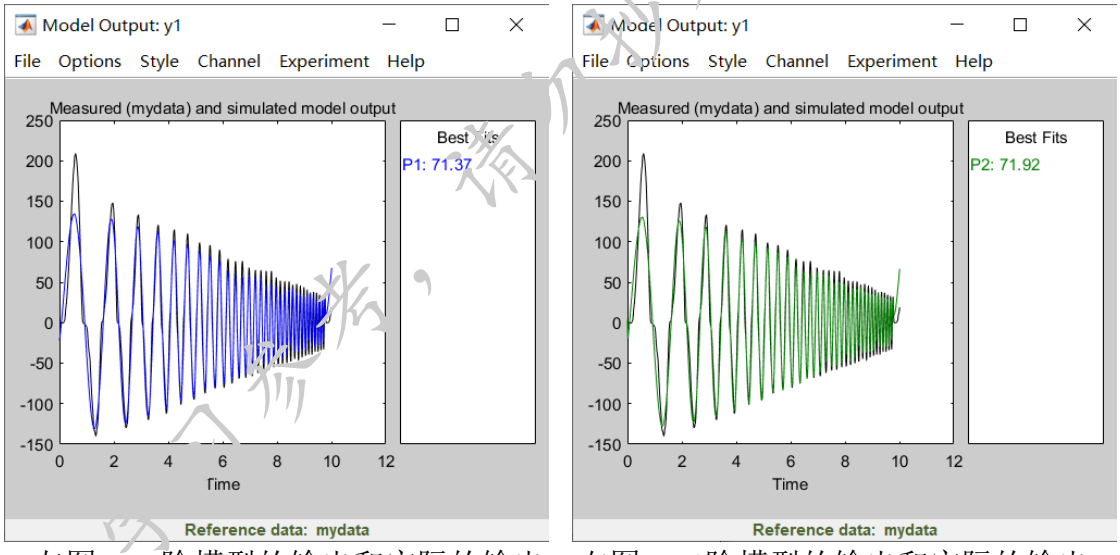


一阶模型辨识结果（吻合度 71.29%）



二阶模型辨识结果（吻合度 71.85%）

三、辨识的模型输出和实际的输出：



左图：一阶模型的输出和实际的输出；右图：二阶模型的输出和实际的输出

四、参数辨识结果：

$$G(s) = \frac{Kp}{1+Tp1*s}$$

Kp = 0.093333
Tp1 = 0.081067

$$G(s) = \frac{Kp}{(1+Tp1*s)(1+Tp2*s)}$$

Kp = 0.089876
Tp1 = 0.070424
Tp2 = 0.005779

左图：一阶模型；右图：二阶模型

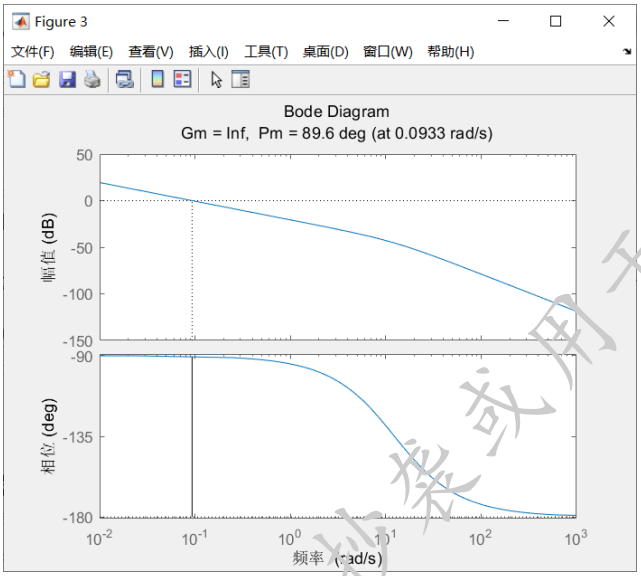
五、写出系统的开环传递函数：

一阶模型：$\boxed{G_0(s)=\dfrac{0.0933}{s(0.081067s+1)}}$；二阶模型：$\boxed{G_0(s)=\dfrac{0.08988}{s(0.07042s+1)(0.00578s+1)}}$

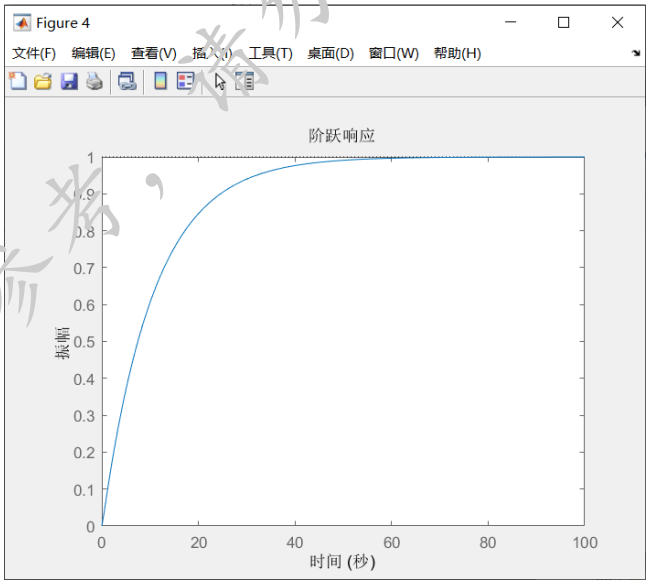
注意**须加入积分**。

3.3 控制器设计

选取辨识所得的一阶系统来进行设计。首先绘制原系统的 Bode 图：



系统剪切频率极小，动态性能差。从下面的阶跃响应曲线也可看出：



系统的带宽也不足（-3dB 带宽仅有 0.133rad/s），输入 1Hz 正弦波将会被严重削去。则设计的目标是，将剪切频率增大，并保持相位裕度在 30°至 60°之间。拟设计串联 PD 控制律，形式如下：

$$G_c(s)=P+D\frac{N}{1+N\dfrac{1}{s}}=\frac{sND}{s+N}+P=P\frac{\dfrac{P+ND}{PN}s+1}{\dfrac{s}{N}+1}$$

其中 N 为滤波系数，N 趋于∞时，第二项趋于理想的微分环节。此处取为 100。

附加此 PD 控制律后，系统的开环传递函数应为

$$G_c(s)=\frac{0.0933P}{s(0.081067s+1)}\frac{\frac{P+100D}{100P}s+1}{0.01s+1}$$

则 PD 控制器给该传函附加了一个零点 $\frac{100P}{P+100D}$，另附加了一个频率较高的极点，在系统高频段，提高抗噪声能力，对于动态性能影响相对较小。

系统为 I 型，理论上对于单位阶跃响应的误差为 0，不妨取 $P=500$。求取系统的剪切频率：系统开环对数幅频特性如下（注意到 $\frac{100P}{P+100D}<100$）

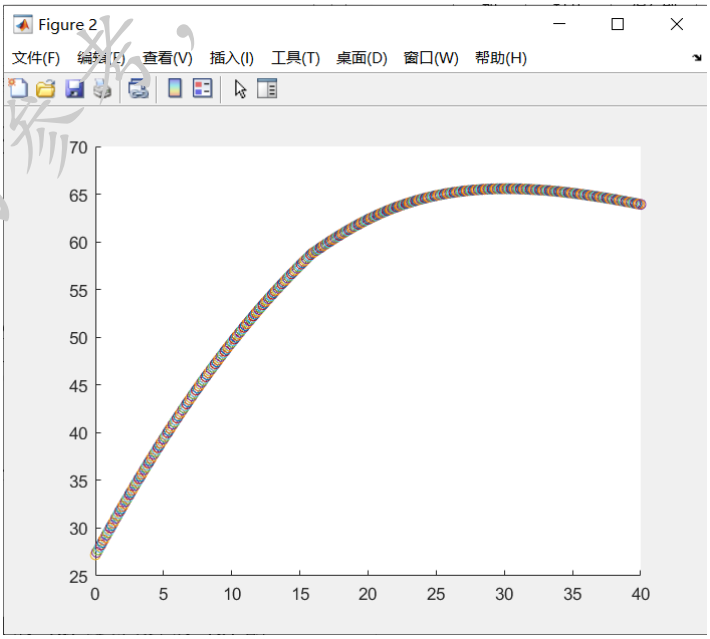
$$L(\omega)=\begin{cases}20\lg 0.0933P-20\lg\omega, \omega<12.335\\ 20\lg 0.0933P-20\lg\omega-20\lg 0.081067\omega, 12.335<\omega<\frac{100P}{P+100D}\\ 20\lg 0.0933P-20\lg\omega-20\lg 0.081067\omega+20\lg\frac{P+100D}{100P}\omega, \frac{100P}{P+100D}<\omega<100\\ \cdots\end{cases}$$

可知剪切频率落在第 2 段（此时 D 较小）时，剪切频率满足 $\omega_c=\sqrt{\frac{0.0933P}{0.081067}}=23.99\,\text{rad/s}$；剪切频率落在第 3 段（此时 D 较大）时，满足 $\frac{0.0933P\times\frac{P+100D}{100P}}{0.081067\times\omega_c}=1$，即 $\omega_c=0.0115(P+100D)$。

求取相角裕度的公式：

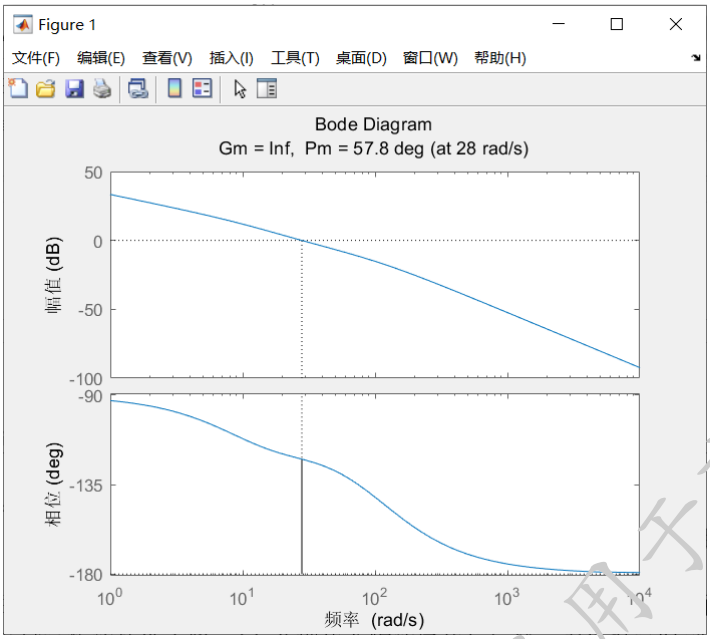
$$\gamma=90°-\arctan 0.081067\omega_c+\arctan\frac{P+100D}{100P}\omega_c-\arctan 0.01\omega_c$$

利用此式绘制出相角裕度的图线，可得



一般来说，相角裕度在 30°至 60°之间比较合理。考虑到此系统对于快速性的要求较高，所以取 $D=16$，则可由图得相角裕度为 59° 左右；根据上面推导的公式，计算得此时剪切频率大约是 24rad/s。

再绘制出 $P=500$、$D=16$ 时的精确 Bode 图并验算剪切频率和相位裕度：



可得近似计算的结果还是较为准确的；系统性能得到了改善，-3dB 的带宽约达到了 36rad/s。综上，$P=500$、$D=16$。在下一节中，将通过 Simulink 仿真来分析系统的阶跃响应和正弦响应。

该步骤用到的 MATLAB 代码如下：

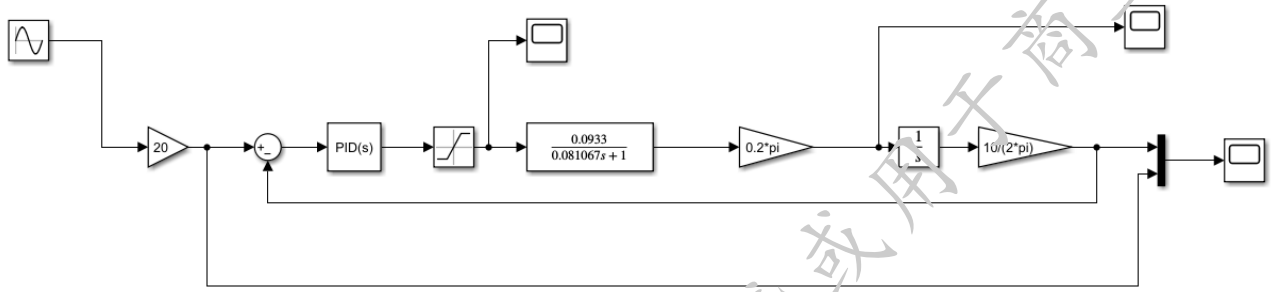
```
P = 500;
D = 16;
num = [0 0 (P+100*D)/100*0.0933 0.0933*P];
den = conv([1,0],conv([0.081067 1],[0.01 1]));
figure(1)
g = tf(num,den);
margin(g);
figure(2)
h = feedback(g,1);
step(h);

figure(3)
hold on
for D = 0:0.1:40
    a = (P+100*D)*0.0115;
    if D>15.9
        scatter(D,90-atand(0.081067*a)+atand((500+100*D)/50000*a)-atand(0.01*a));
    else
        scatter(D,90-atand(0.081067*23.99)+atand((500+100*D)/50000*23.99)-
atand(0.01*23.99));
    end
end
hold off
```

```
figure(4)
num = 0.0933;
den = conv([1,0],[0.081067 1]);
g = tf(num,den);
margin(g);
figure(5)
h = feedback(g,1);
step(h);
```

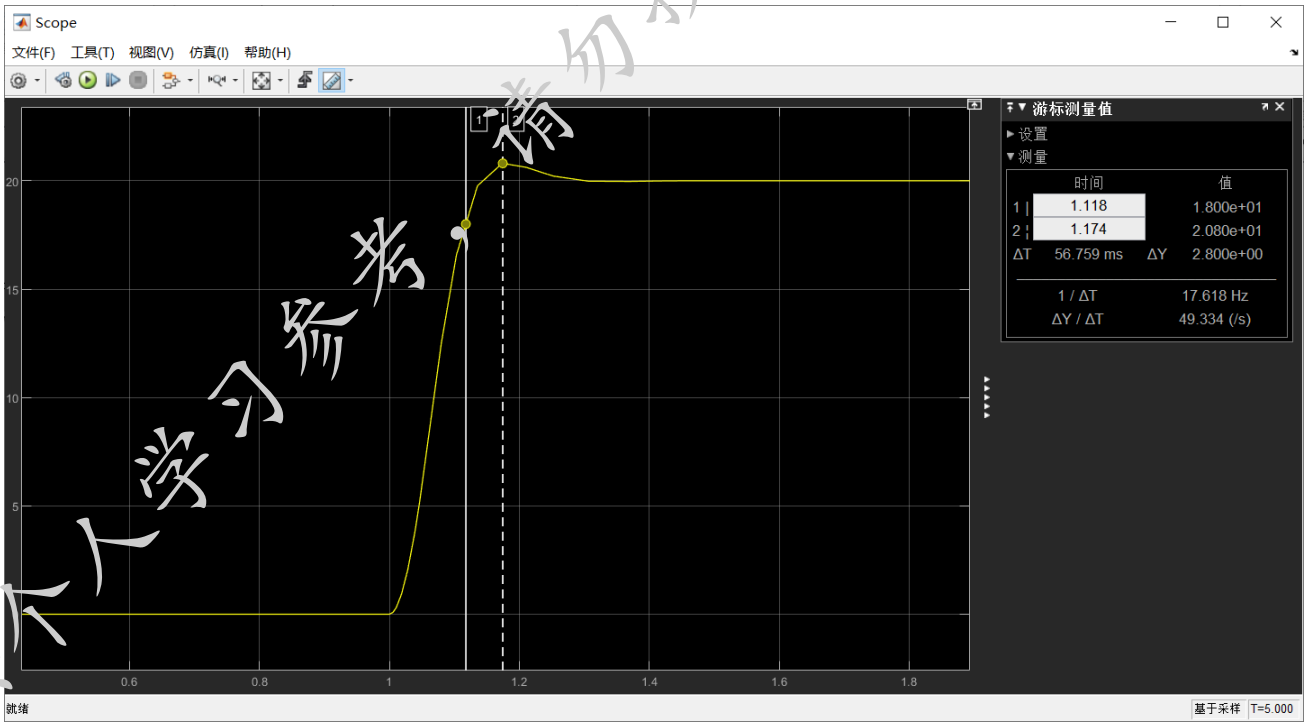
3.4 控制器仿真验证

3.4.1 Simulink 仿真框图

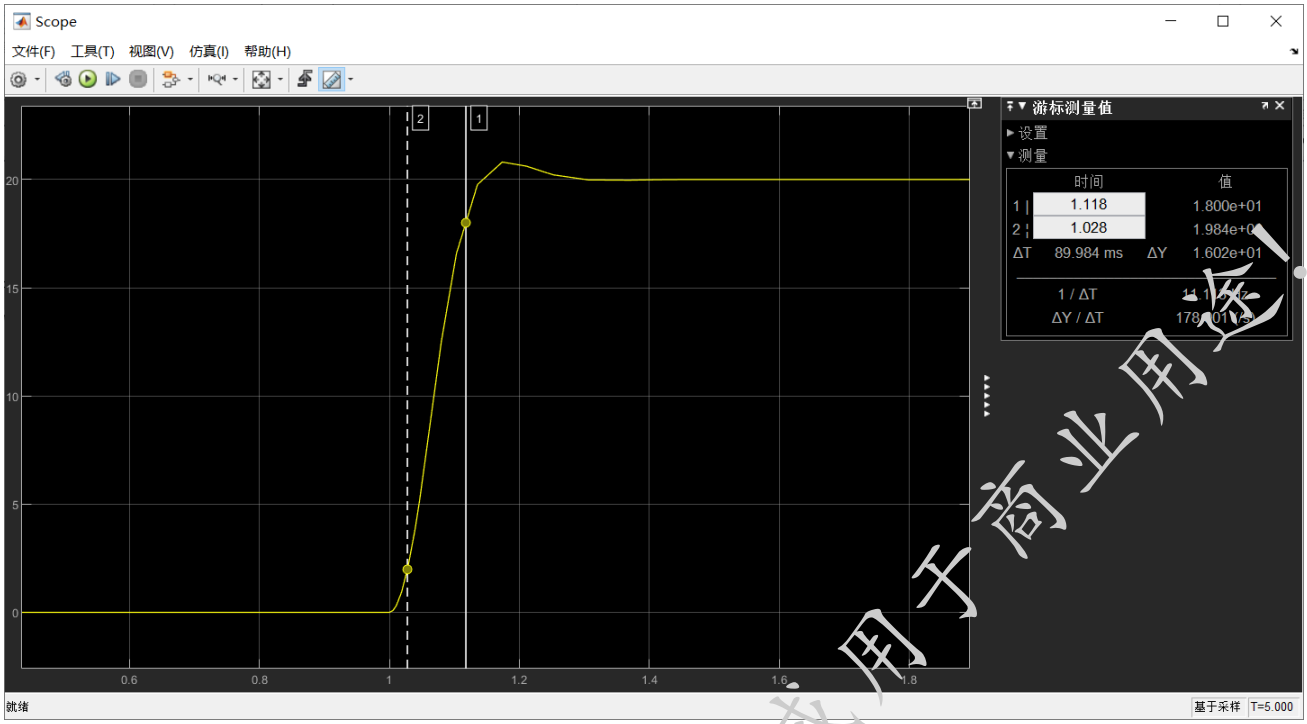


3.4.2 阶跃响应测试结果

只有一路输出是因为在测试阶跃信号时示波器只接了输出。

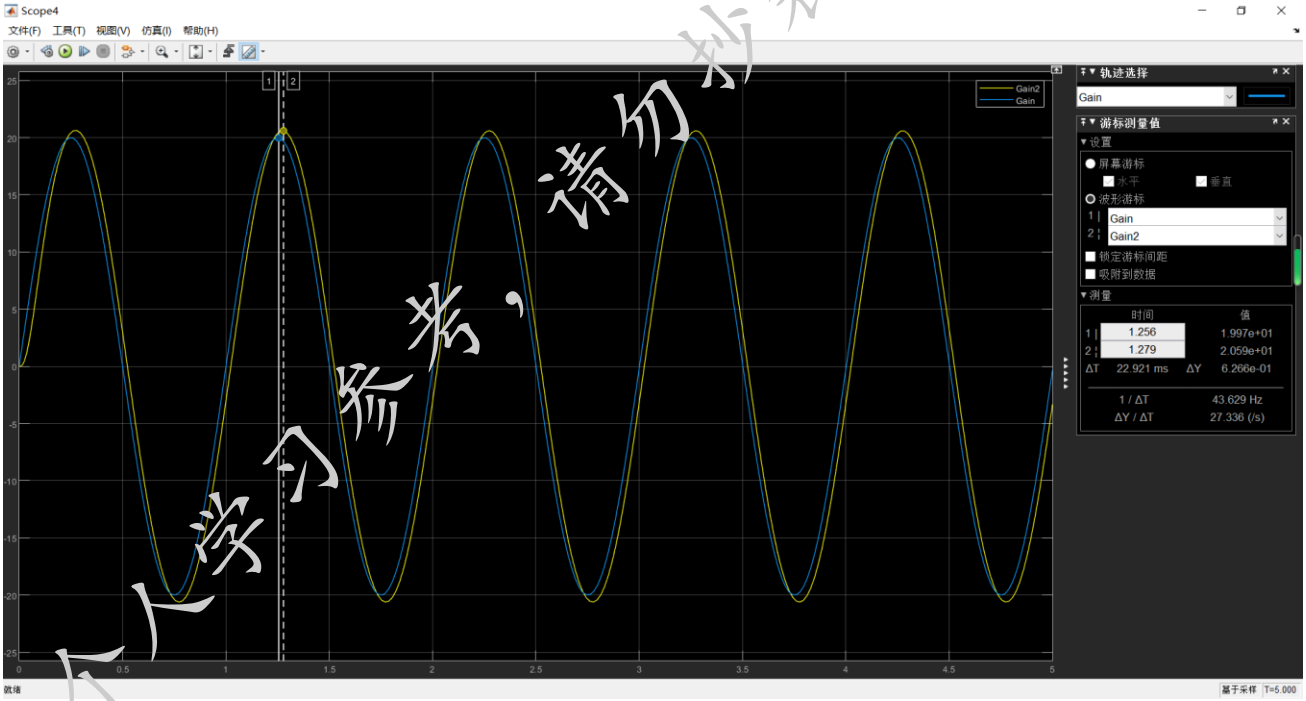


超调量约 4%



10% - 90% 上升时间约 0.9s

3.4.3 扫频跟随测试结果



黄色线为输出，蓝色线为输入。可见黄色线相比于蓝色线没有衰减。

3.5 控制程序开发

3.5.1 控制器离散化

使用 MATLAB 完成，源代码为：

```
s = zpk('s') ;
%设置控制器
ks = 500 + 16*100/(1+100/s);
%设置控制器周期 Ts
Ts = 0.01;
%使用 c2d 进行双线性变换；
kz = c2d(ks,Ts,'tustin')
%使用 tf 转变为离散的传递函数
kz_tf = tf(kz)
```

运行结果是：

```
kz_tf =

 1567 z - 1233
 -------------
  z - 0.3333

采样时间: 0.01 seconds
离散时间传递函数。
```

即控制规律：$u(k)=1567e(k)-1233e(k-1)+0.3333u(k-1)$。

3.5.2 控制器源代码

没有专门实现控制器，而是将其融入了下方的阶跃响应测试和扫频响应测试中。关键的代码是：

```
            pos=RegulatedEncoderVal * 0.005f; // 获取当前位置
            error=ref-pos; // 计算误差
            control=1567*error-1233*last_error+0.3333*last_control; // 控制量
            last_control=control; // 将本时刻控制量保存为上一次控制量
            last_error=error; // 将本时刻误差保存为上一时刻误差
            control_int=(int16_t)control; // 将控制量转化为整数
           // limit the amplitude
            if(control_int>4997)  control_int=4997;
            if(control_int<-4997)  control_int=-4997;
            MC_ProgramTorqueRampMotor1(control_int,0);
```

3.6 控制系统调试

3.6.1 阶跃响应测试源代码

```
void function_step_test(void)
{
  static uint32_t last_sys_tick2 = 0;
```

```
  static uint32_t sys_tick2 = 0;
    static float time2 = 0;
    float ini_pos = 0;

    sys_tick2=HAL_GetTick();
    time2=0.001*sys_tick2;
    if(step_init_flag==0)
    {
     ini_pos=RegulatedEncoderVal * 0.005f;
     ref=ini_pos+20;
     step_init_flag=1;
    }
    if(START_STATUS == 4)
    {
        if(sys_tick2%10==0)
        {
             pos=RegulatedEncoderVal * 0.005f; // 获取当前位置
             error=ref-pos; // 计算误差
             control=1567*error-1233*last_error+0.3333*last_control; // 控制量
             send_data_2_matlab(time2,ref,pos);
             last_control=control; // 将本时刻控制量保存为上一次控制量
             last_error=error; // 将本时刻误差保存为上一时刻误差
             control_int=(int16_t)control; // 将控制量转化为整数
            // limit the amplitude
             if(control_int>4997)  control_int=4997;
             if(control_int<-4997)  control_int=-4997;
             MC_ProgramTorqueRampMotor1(control_int,0);
        }
    }
}
```

3.6.2 扫频测试源代码

```
void function_chirp_test(void)
{
    static my_sweep_t sweep={0};
    static uint32_t last_sys_tick3=0;
    static uint32_t sys_tick3=0;
    static float time3=0;
    static uint32_t init_flag3=0;
```

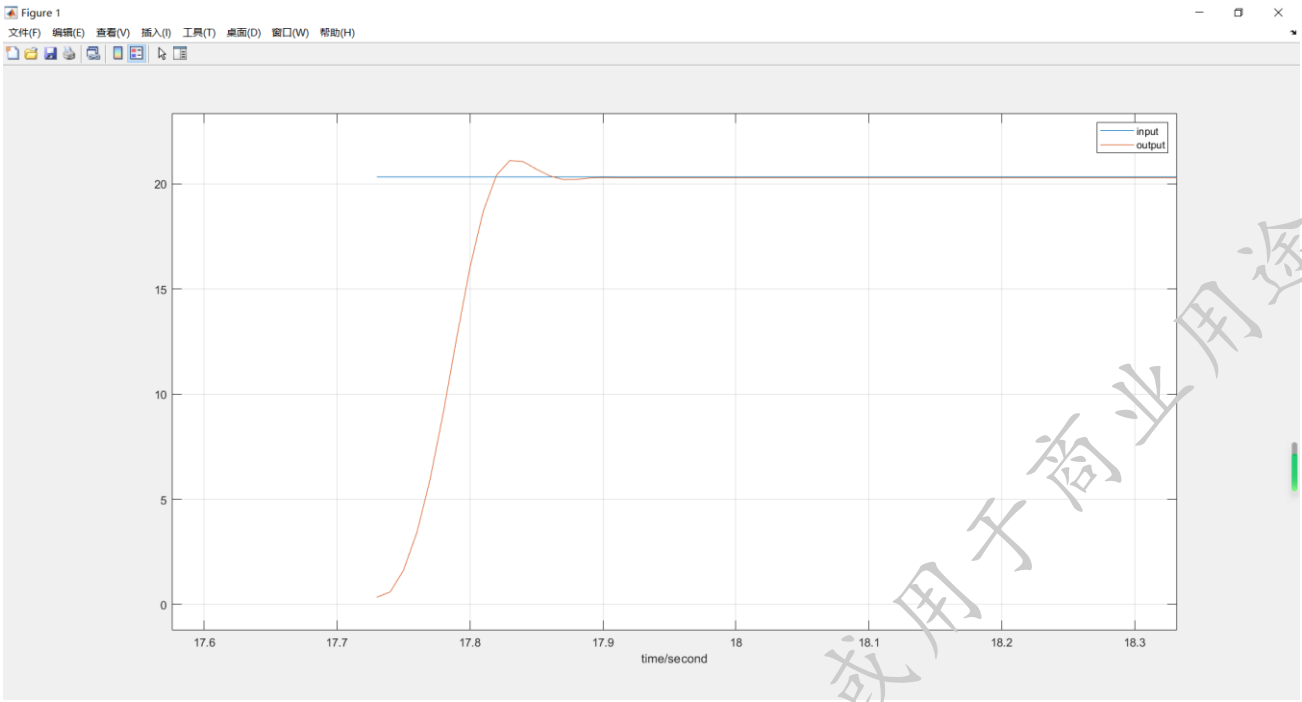
```
    static float ref=0;
    static float fmk=0;
    sys_tick3=HAL_GetTick();
    time3=0.001*sys_tick3;

    if(last_sys_tick3!=sys_tick3)
    {
        last_sys_tick3=sys_tick3;

        if(sys_tick3%10==0)
        {
         if(chirp_init_flag == 0)
         {
             init_my_sweep(&sweep,sys_tick3,10000,1,1.001,20);
             fmk=RegulatedEncoderVal * 0.005f;
             chirp_init_flag = 1;
         }

         pos=RegulatedEncoderVal * 0.005f;
         ref=fmk+run_my_sweep(&sweep,sys_tick3);
         error=ref-pos;
         control=1567*error-1233*last_error+0.3333*last_control;
         send_data_2_matlab(time3,ref,pos);
         last_control=control;
         last_error=error;
         control_int=(int16_t)control;
         // limit the amplitude
         if(control_int>4997)  control_int=4997;
         if(control_int<-4997)  control_int=-4997;
         MC_ProgramTorqueRampMotor1(control_int,0);
        }
    }
}
```

3.6.3 调优后的阶跃响应曲线



output input
100x1 double

	1	2
1	20.3400	
2	20.3400	
3	20.3400	
4	20.3400	
5	20.3400	
6	20.3400	
7	20.3400	
8	20.3400	
9	20.3400	
10	20.3400	
11	20.3400	
12	20.3400	
13	20.3400	
14	20.3400	
15	20.3400	
16	20.3400	
17	20.3400	
18	20.3400	
19	20.3400	
20	20.3400	

output input
100x1 double

	1	2
1	0.3450	
2	0.6000	
3	1.6250	
4	3.4100	
5	5.9500	
6	9.1300	
7	12.6700	
8	16.0250	
9	18.7000	
10	20.4150	
11	21.1150	
12	21.0600	
13	20.7050	
14	20.3850	
15	20.2100	
16	20.2150	
17	20.28[illegible]	
18	20.3000	
19	20.2900	
20	20.3850	

左图中从左到右分别是输入和输出的幅值。

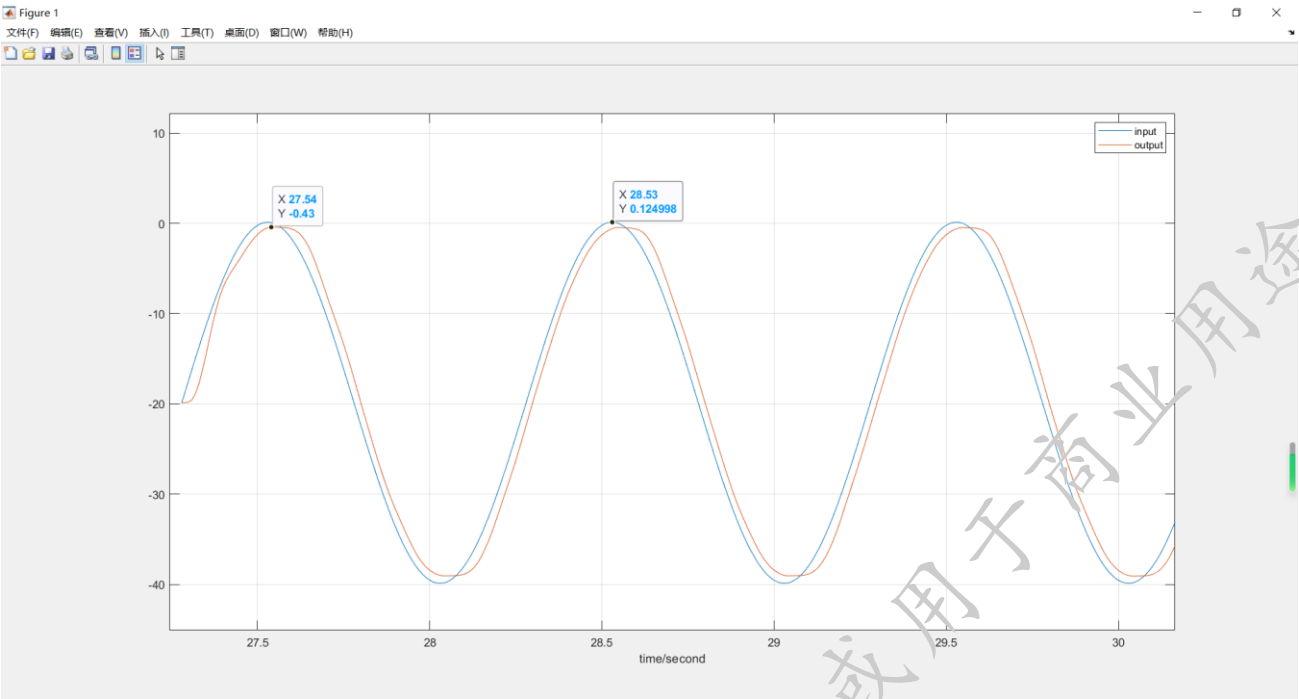
第一个数据是阶跃信号开始作用的瞬间发出的，此后每 0.01s 发一次数据。初始值在 0.34，阶跃信号作用的末值是 20.34。

比较输出信号的最大幅值（21.12-0.34=20.78）和输入信号的幅值 20，可知超调约是 3.9%；

输出信号在 0.09s（第 10 号数据）处幅值约是 20.08（20.42-0.34）。可见 95%上升时间不到 0.09s；

，最终的输出幅度是 19.94（20.28-0.34），稳态误差在 1mm 之内。

3.6.4　调优后的扫频跟随曲线



横轴为-20，则输入最大值实为 20.125，输出最大值实为 19.57，而

$$20\lg\left(\frac{19.57}{20.125}\right)=-0.243\,\mathrm{dB}$$

衰减小于 3dB。

3.6.5　最终的控制器传递函数

连续时间传递函数

$$G_c(s)=500+16\frac{100}{1+100\dfrac{1}{s}}=500\frac{\dfrac{s}{23.81}+1}{s/100+1}$$

离散化传递函数

$$H_c(z)=\frac{1567z-1233}{z-0.3333}$$

3.7　控制器设计的不足与改进

正弦输出波形出现削顶失真，尚可继续优化。摩擦阻力在低速运行时对滑台运动的干扰还是很明显的，可尝试对摩擦力进行建模，把系统辨识得更加准确。

3.8　实验总结

此实验是一个真正的控制系统设计实践，使我能将理论知识应用于现实。这也是本课程的亮点所在。