《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 方尧

学号 190410102

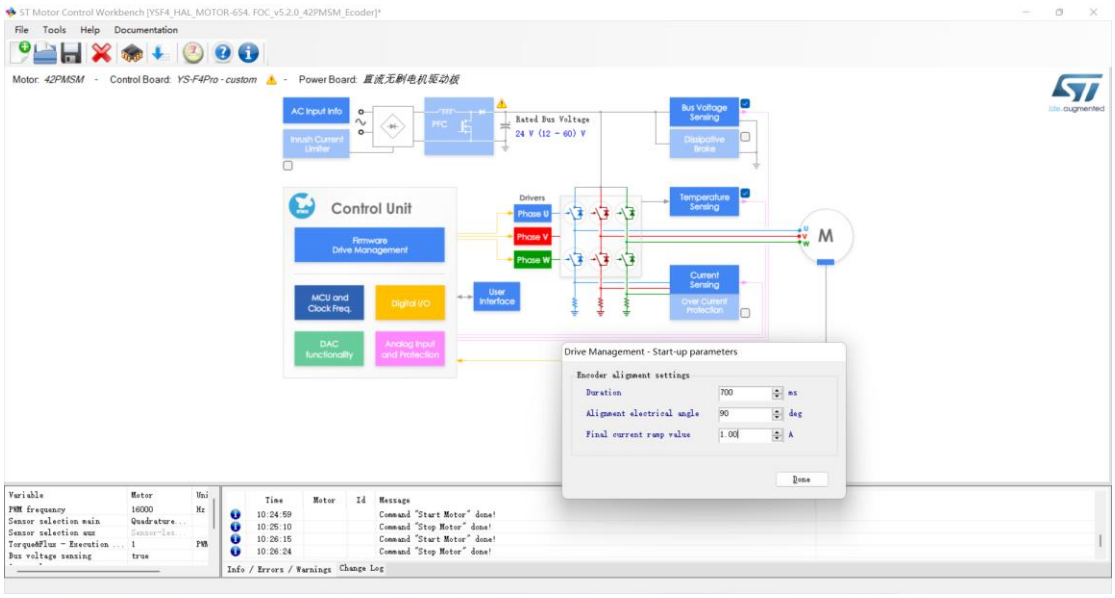
日期 2022 年 6 月 22 日

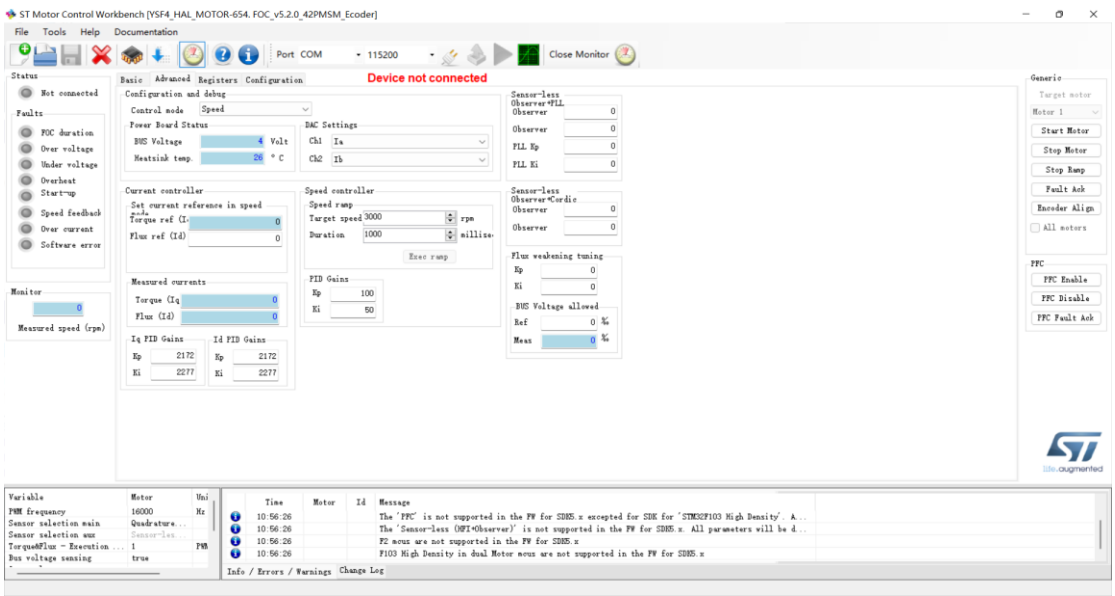
目录

任务一 电机控制库认知

1.1 Workbench 编码器对齐参数配置

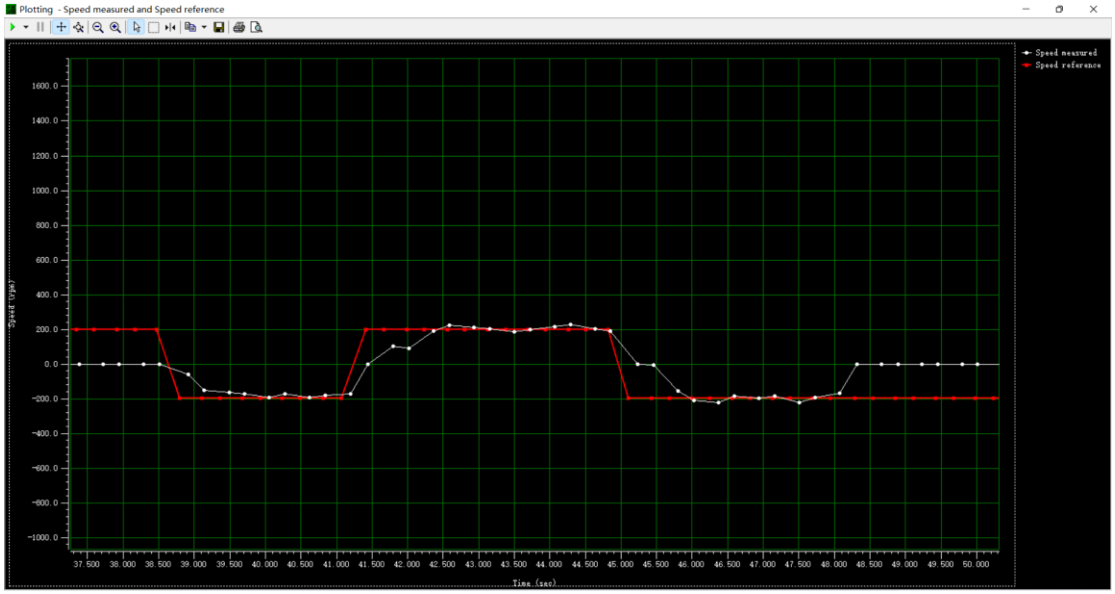
（截图即可）





1.2 速度波形显示界面

（速度曲线截图）



1.3 说说你对 ST MC SDK5.x（电机控制库）的认知

可以从电机控制库的开发背景、发展历程、组成结构、功能、软件使用等任意方面阐述自己的理解，字数不限。

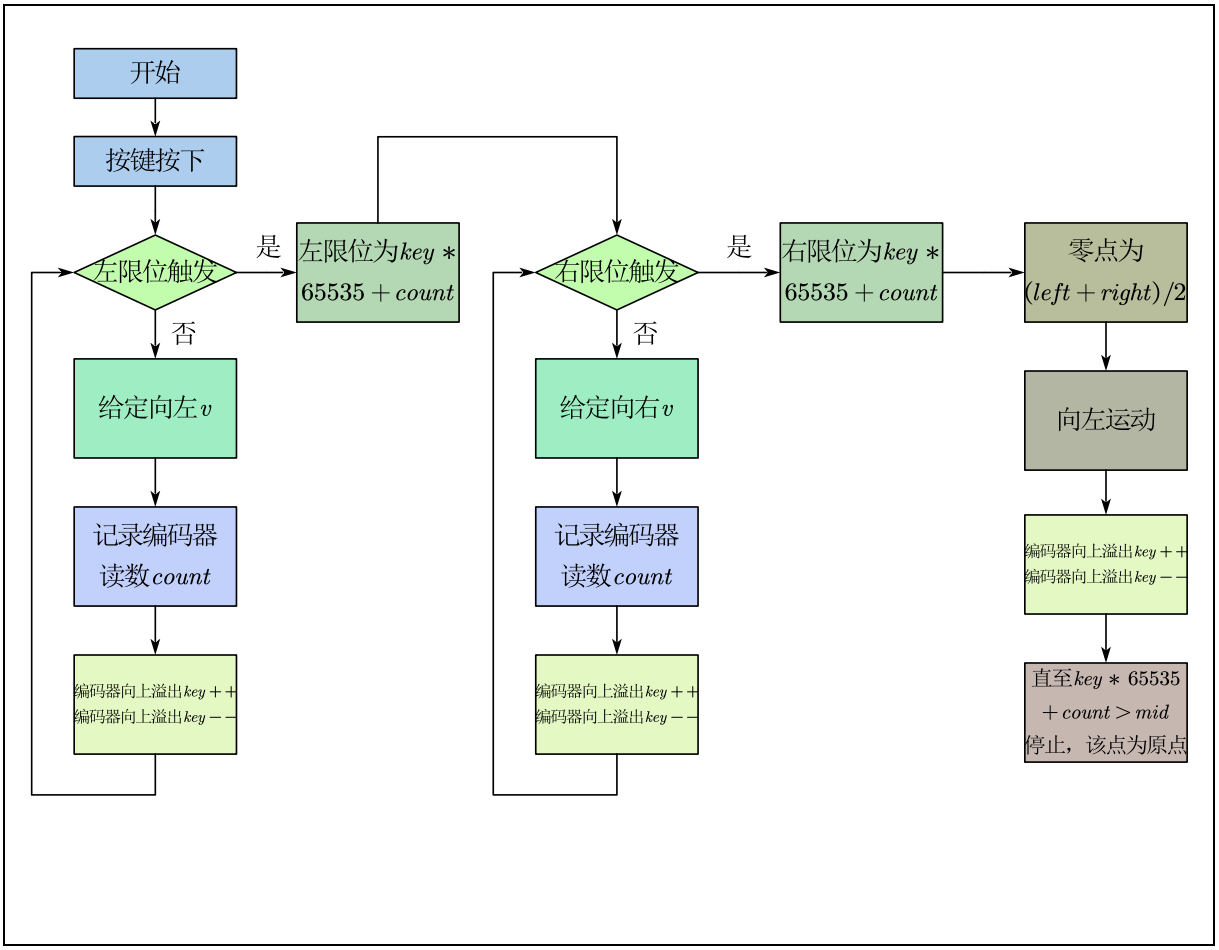
STM32 微控制器提供工业标准 Arm Cortex-M 内核的性能，可运行矢量控制控制或 FOC 模式，广泛应用于空调、家用电器、无人机、建筑和工业自动化、医疗和电动自行车等高性能驱动的应用领域。 X-CUBE-MCSDK PictureSTM32 MC SDK（电机控制软件开发套件）固件（X-CUBE-MCSDK 和 X-CUBE-MCSDK-FUL）包括永磁同步电机（PMSM）固件库（FOC 控制）和 STM32 电机控制 Workbench，以便通过图

形用户界面配置固件库参数。STM32 电机控制 Workbench 为 PC 软件，降低了配置 STM32 PMSM FOC 固件所需的设计工作量和时间。用户通过 GUI 生成项目文件，并根据应用需要初始化库。可实时监控并更改一些算法变量。支持单/双同步磁场定向控制（FOC），含有电机分析仪和一键式调整可用于快速启动未知的电机，以及基于 STM32Cube HAL/LL 库的简化固件架构；支持速度/位置传感器（编码器和霍尔）以及无传感器工作（状态观察器）支持速度和扭矩两种控制方式，其内还为特定应用实现了电机控制算法，包括 MTPA（最大转矩电流）、弱磁、前馈和带速启动，用户还可以通过 STM32 电机控制 Workbench PC 端软件实现完全定制化和实时通信等等。

任务二 按键控制滑台回零

2.1 程序流程图

画出滑台回零程序流程图。



2.2 功能代码

在此处粘贴自己编写的代码，并写好代码注释。

```
//测距归零类宏变量
int count_LR[2];//左右限位位置
int count[2]={0,0};//编码器读数，上一次读数和这一次读数
int center=0;//中心位置
int key=0;//向上向下溢出标记
double mile=0;//实际测距 mm
int p=0;//结合溢出后编码器读数

//回零函数
void Gohome()
{
  int v=10;
  int time=1;
  //左移
  //拨片触发 到负限位开关时，负限位引脚被拉低
  while(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_1)!=GPIO_PIN_RESET)//左限位未触发
  {

    MC_ProgramSpeedRampMotor1(v,time);
    MC_StartMotor1();
    count[0]=count[1];
    count[1]=__HAL_TIM_GET_COUNTER(&htim3);
    if(abs(count[1]-count[0])>60000)//减少返回 1
    {
      key++;//向左增大有溢出
      //HAL_Delay(10);
    }
    printf("count=%d,key=%d,p=%d \n",count[1],key,count[1]+key*65535);
    HAL_Delay(1);
  }
  count_LR[0]=count[0]+key*65535;
  MC_StopMotor1();

  //右移
  //拨片触发 到负限位开关时，负限位引脚被拉低
  while(HAL_GPIO_ReadPin(GPIOG,GPIO_PIN_0)!=GPIO_PIN_RESET)//左限位未触发
  {

    MC_ProgramSpeedRampMotor1(-v,time);
    MC_StartMotor1();
```

```
    count[0]=count[1];
    count[1]=__HAL_TIM_GET_COUNTER(&htim3);
    if(abs(count[1]-count[0])>60000)//减少返回 1
    {
      key--;//向右减小有溢出
      //HAL_Delay(10);
    }
    printf("L=%d,count=%d,key=%d,p=%d
\n",count_LR[0],count[1],key,count[1]+key*65535);
    //HAL_Delay(1);
  }
  count_LR[1]=count[0]+key*65535;
  MC_StopMotor1();
  //原点位置
  center=(int)(count_LR[1]+count_LR[0])/2;
  printf("l=%d,r=%d,center=%d \n",count_LR[0],count_LR[1],center);

  //HAL_Delay(1000);
  //回零
  while(count[1]+key*65535<center)
  {

    MC_ProgramSpeedRampMotor1(v,time);
    MC_StartMotor1();
    count[0]=count[1];
    count[1]=__HAL_TIM_GET_COUNTER(&htim3);
    if(abs(count[1]-count[0])>60000)//减少返回 1
    {
      key++;//向左增加有溢出
      //HAL_Delay(1);
    }
    printf("L=%d,R=%d,center=%d,key=%d,p=%d
\n",count_LR[0],count_LR[1],center,key,count[1]+key*65535);
    HAL_Delay(10);
  }
  MC_StopMotor1();
}

//测量位置函数
void Measure()
{
    count[0]=count[1];
    count[1]=__HAL_TIM_GET_COUNTER(&htim3);
    if(count[1]-count[0]<-60000)//65535-》0
```

```
    {
      key++;//向左增加有溢出
      HAL_Delay(10);
    }
    if(count[1]-count[0]>60000)//0-》65535
    {
      key--;//向右减小有溢出
      HAL_Delay(10);
    }
    p=count[1]+key*65535;
    mile=(p-center)*0.005;
}
```

2.3 实验总结

说说自己在开发中遇到的主要问题，及最终解决方法。

主要遇到的问题为：如果使用指导书中编码器计数减少函数标记上下溢出情况，会因为电机在上下溢出时刻点反复波动，导致上下溢出标记值出现异常；

解决方法：设置二维变量，记录上一次编码器读数和这一次编码器读数，作差得到的差值与-60000和60000比较，设置上下溢出标记值key++或key--，得到正确结果。

改正后代码正常，归零效果良好，测距效果良好。

任务三 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

计算联轴器的谐振频率；与控制系统要求的带宽进行对比，确定机械谐振模态是否可以忽略；写出计算过程。

计算电机侧转动惯量：

电机自身转动惯量为 $J_m = 2.8\times10^{-5} kg\cdot m^2$

联轴器质量均匀，转动惯量和惯性力矩相同， $J_Y = 2.26\times10^{-6} kg\cdot m^2$

丝杆部分 $m = 0.6kg, r = 7.5\times10^{-3} m, J_{SF} = 1/2mr^2 = 1.6875\times10^{-5} kg\cdot m^2$

平台部分由螺母和工作台组成，两者质量为0.225kg和0.335kg，采用指导书数据，

$J_M = M\left(\frac{h}{2\pi}\right)^2 = 1.368\times10^{-6} kg\cdot m^2$，总转动惯量为：

$$J = J_m + J_M + J_{SF} + J_Y = 4.85\times10^{-5} kg\cdot m^2$$

机械谐振频率：

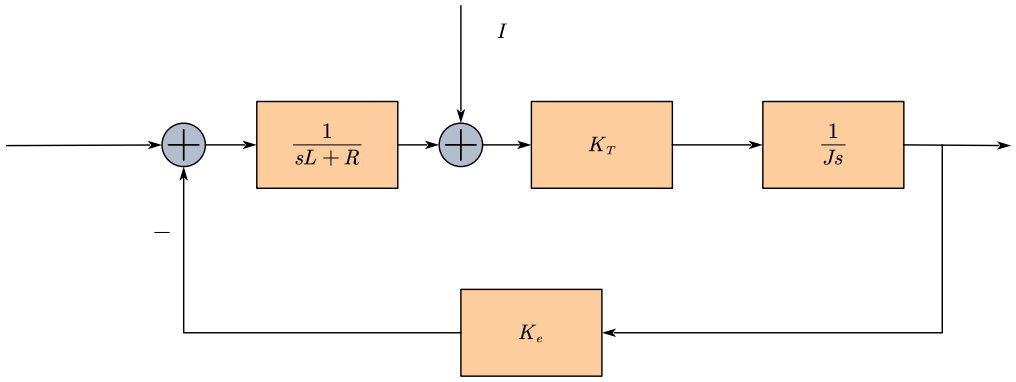
$$\omega_m = \sqrt{\frac{K}{J}} = 4425.8 rad/s$$，其中 K 取扭转刚度950Nm/rad

1Hz（ 2π ）远小于 ω_m，故设计时可以忽略机械谐振。

3.1.2 被控对象的系统建模

结合课上所学内容，画出控制系统方框图，对被控对象进行建模，并对模型进行简化处理。

一般采用控制电流方式控制电机转速，故对被控对象建模如下：



可以得到：

$$\frac{\dot{\theta}}{I}=\frac{K_T\left(sL+R\right)}{JLs^2+JRs+K_eK_T}$$

L一般很小，简化为：

$$\frac{\dot{\theta}}{I}=\frac{K}{\left(T_1s+1\right)\left(T_2s+1\right)}$$

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

将正弦扫频信号生成算法的源代码粘贴在下面的方框中，并写好代码注释。

```
//定义结构变量
typedef struct
{
  unsigned int t_0;
  unsigned int t_01;
  float f0;
  float f1;
  float k;
  float p;
  float A;
```

```
}my_sweep_t;

typedef struct
{
  uint8_t start_flag;
  uint8_t frame_len;
  uint8_t header_check;
  uint8_t data_buf[12];
  uint8_t frame_check;
}frame_matlab_t;
//离散控制器设计
double e,e_1;
double u,u_1;
//MATLAB 传输数据使用
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;
}
//MATLAB 传输数据使用
void send_data_2_matlab(float data1, float data2, float data3)
{
  frame_matlab_t frame = {0};
  int i = 0;
  frame.start_flag = 0xAA;
  frame.frame_len = sizeof(frame);
  frame.header_check = get_uint8_sum_check((uint8_t *)&frame, 2);
  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);
  frame.frame_check = get_uint8_sum_check((uint8_t *)&frame,
frame.frame_len-1);
  HAL_UART_Transmit(&huart5, (uint8_t *)&frame,frame.frame_len,0xffff);
}
//初始化扫频信息
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))
```

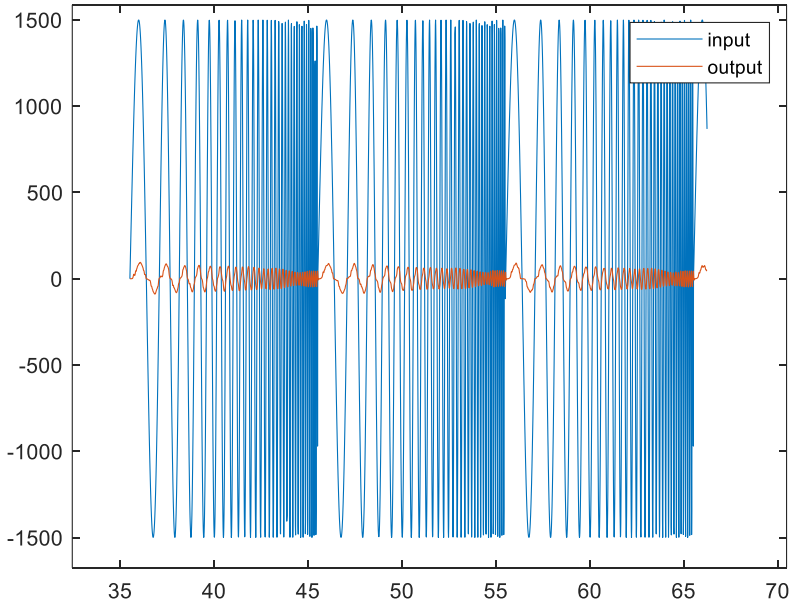
```
  {
    return -1;
  }
  sweep->t_0 = t_0;
  sweep->t_01 = t_01;
  sweep->f0 = f0;
  sweep->f1 = f1;
  sweep->A = A;
  sweep->k = exp(1/(t_01*0.001f)*log((f1/f0)));
  sweep->p = 2*PI*f0/log(sweep->k);
  return 0;
}
//计算扫频信号输出
float run_my_sweep(my_sweep_t *sweep, unsigned int t_now)
{
  float t = 0.0f;
  float y = 0.0f;
  if (!sweep)
  {
    return 0.0f;
  }
  if (t_now < sweep->t_0)
  {
    return 0.0f;
  }
  t = (t_now - sweep->t_0) % sweep->t_01;
  t = t * 0.001f;

  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

1) 将扫频辨识功能源代码粘贴在下面的方框中，并写好代码注释。

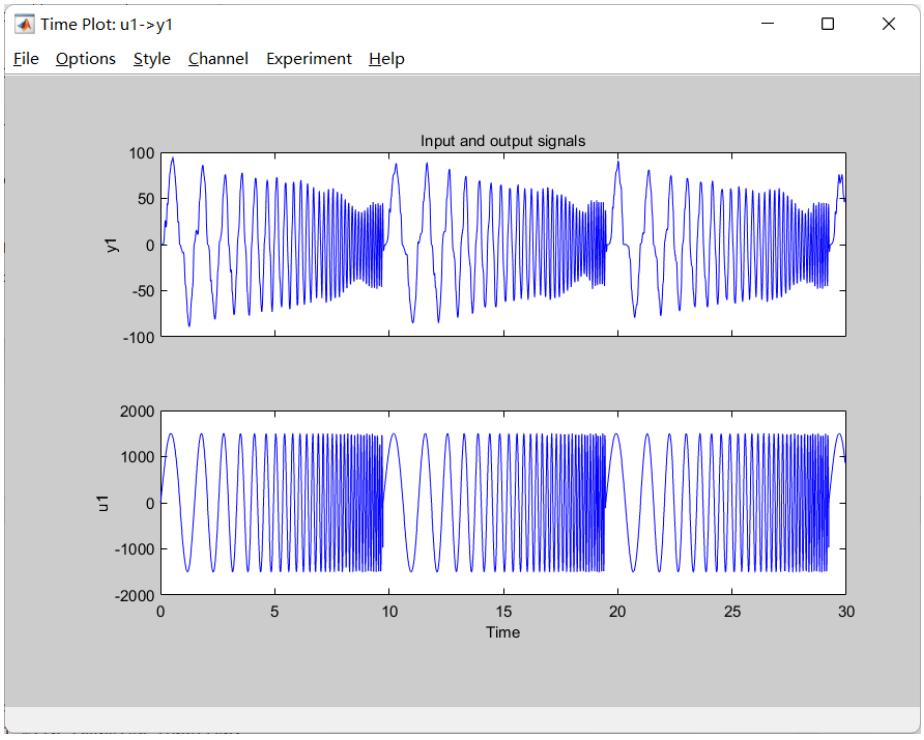
```
//扫频生成函数
void function_sweep_identification(void)
{
  static my_sweep_t sweep = {0};
  int16_t sweep_input = 0;

  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;
  uint32_t t_period_ms = 10*1000;
  float f0 = 0.5;
  float f1 = 10;
  float Amp = 1500.0f;
  float time = 0.0f;
  sys_tick = HAL_GetTick();
  time = 0.001f * sys_tick;
  MC_StartMotor1();
  if ((find_home_flag == 1) && (MC_GetSTMStateMotor1() == RUN))
  {
    if (last_sys_tick != sys_tick)
    {
      last_sys_tick = sys_tick;
      if (sys_tick % 10 == 0)
      {
        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;
        }
        sweep_input = (int16_t)run_my_sweep(&sweep, sys_tick);
        MC_ProgramTorqueRampMotor1(sweep_input,0);
        sweep_output = MC_GetMecSpeedAverageMotor1();
        send_data_2_matlab(time,(float)sweep_input,(float)sweep_output);
      }
    }
  }
}
```

2) 将扫频辨识过程中 Matlab 采集波形保存并粘贴在下方。
（扫频辨识波形数据，参考图 3.22）



3.2.3 使用 matlab 系统辨识工具箱，获得辨识的系统模型

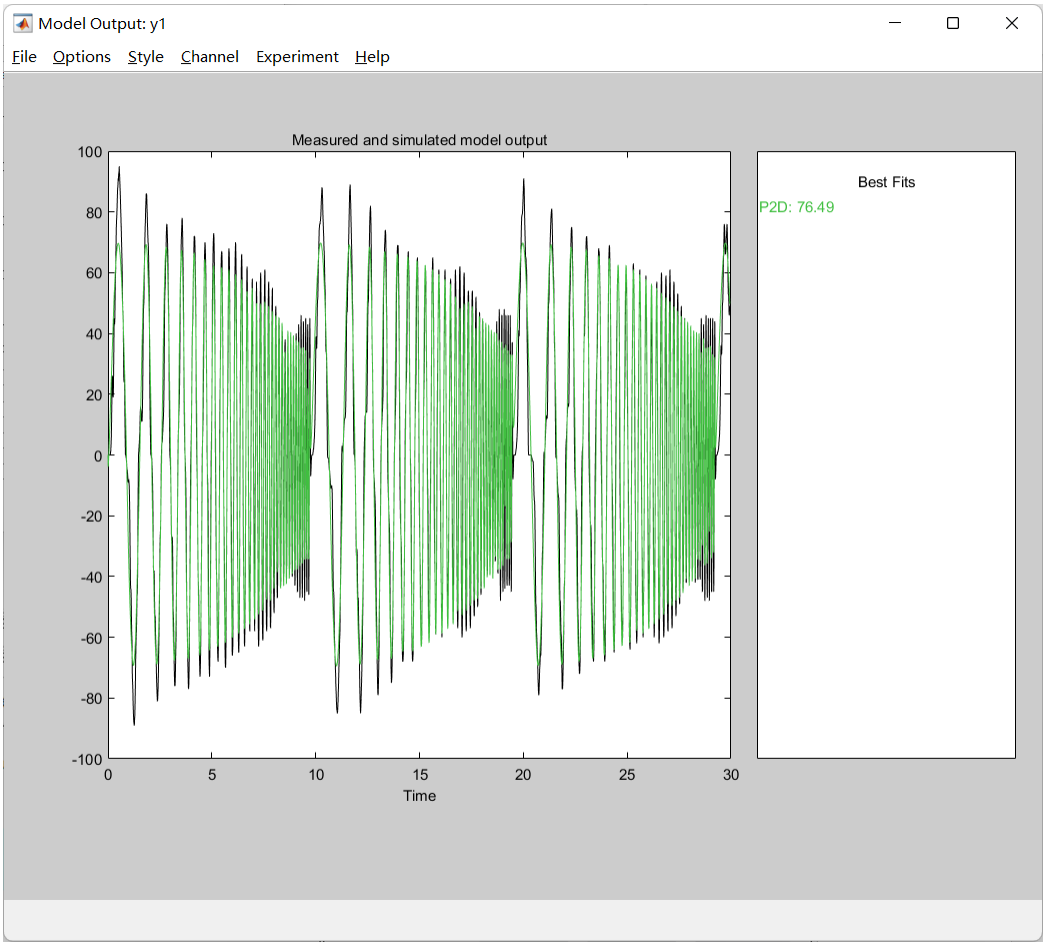
1) 将输入输出数据导入 Matlab 系统辨识工具箱后，Time Plot 查看输入输出曲线，将曲线截图粘贴在下方。
（输入输出曲线截图，参考图 3.26）



2) 参数辨识结束后，将参数辨识计算过程截图粘贴在下方。
（参数辨识计算过程截图，参考图 3.29）

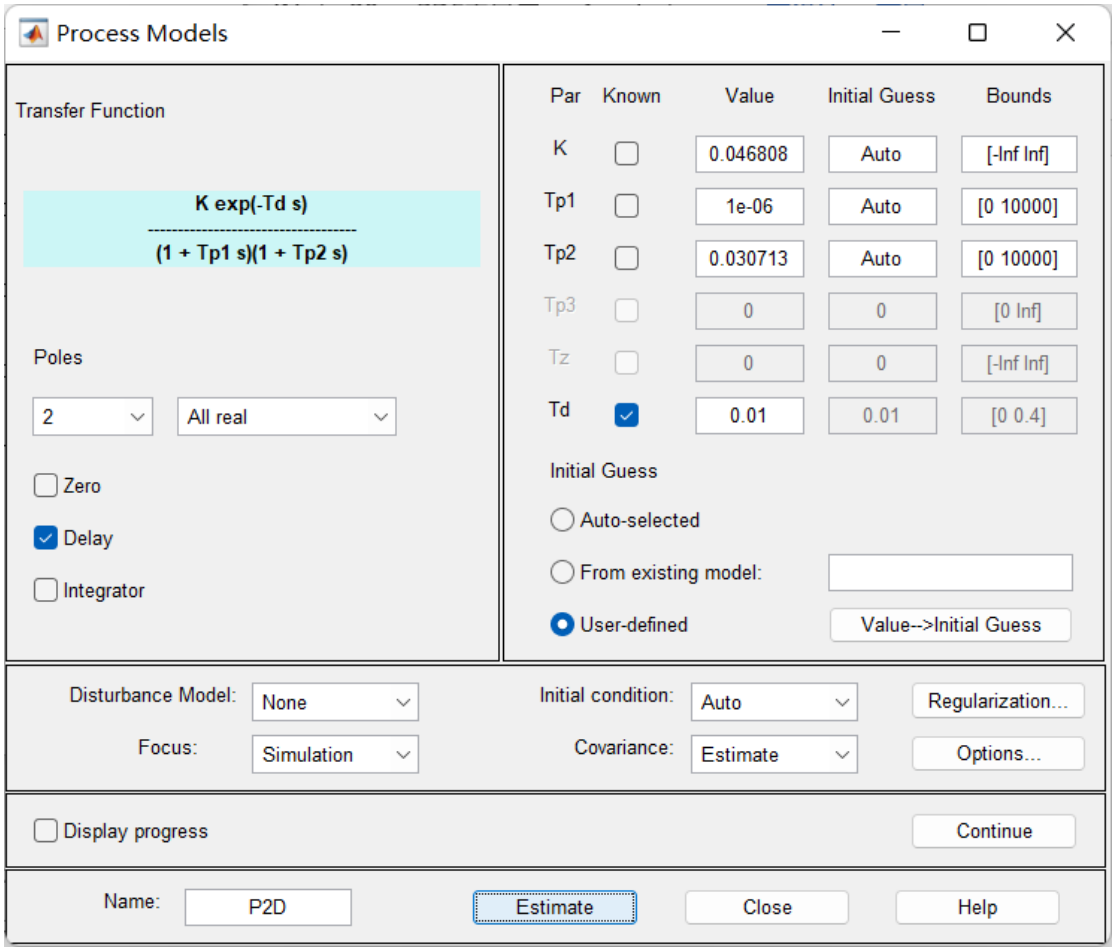
```
Process model with transfer function:
                 Kp
  G(s) = ----------------- * exp(-Td*s)
         (1+Tp1*s)(1+Tp2*s)

          Kp = 0.046808
         Tp1 = 1e-06
         Tp2 = 0.030713
          Td = 0.01

Name: P2D
Parameterization:
    {'P2D'}
   Number of free coefficients: 3
   Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "mydata".
Fit to estimation data: 76.49%
FPE: 107.5, MSE: 107.2
```

3) 参数辨识结束后，勾选 Model output,可以看到辨识的模型输出和实际的输出，将曲线截图并粘贴在下方。
（模型输出与实际输出截图，参考图 3.30）



4) 将参数辨识结果截图并粘贴在下方。

（参数辨识结果截图，参考图 3.31）



5) 写出系统开环传递函数。

$$G(s) = \frac{0.046808}{(1 + 1e - 6s)(1 + 0.0307s)}$$

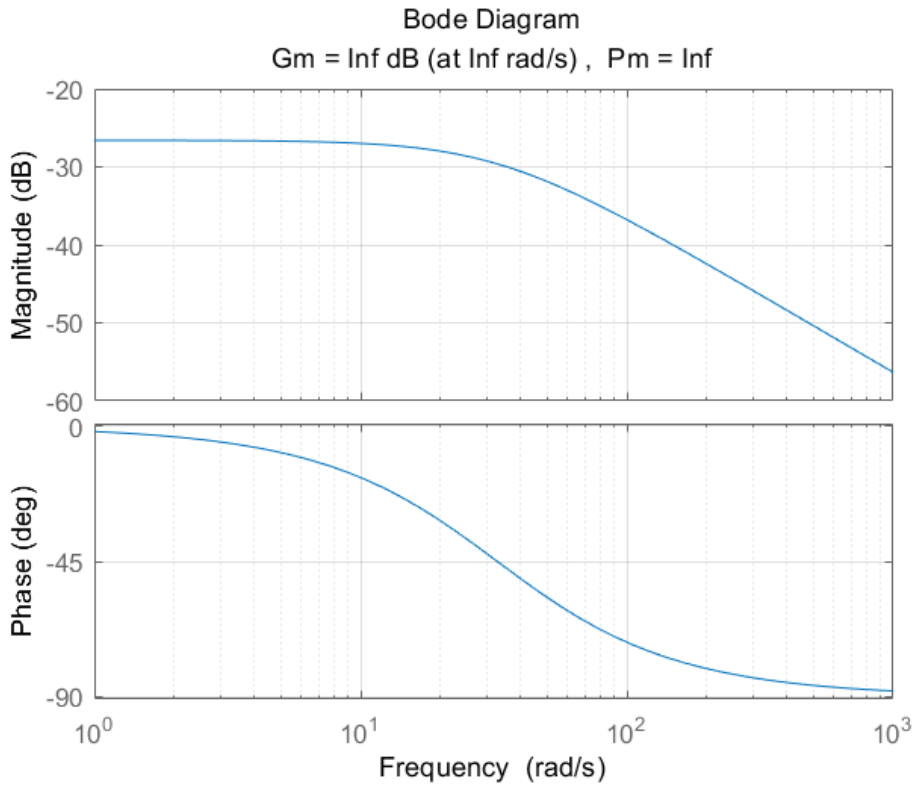
3.3 控制器设计

推荐使用：频域的方法进行控制器的设计，使得控制器的带宽，相位裕度等满足控制系统任务指标。

请同学在这里给出控制器设计的详细过程（可以是理论推导设计过程，或者 matlab 辅助设计过程），并结合 bode 图给出相位裕度和幅值裕度的情况。

1）未校正系统

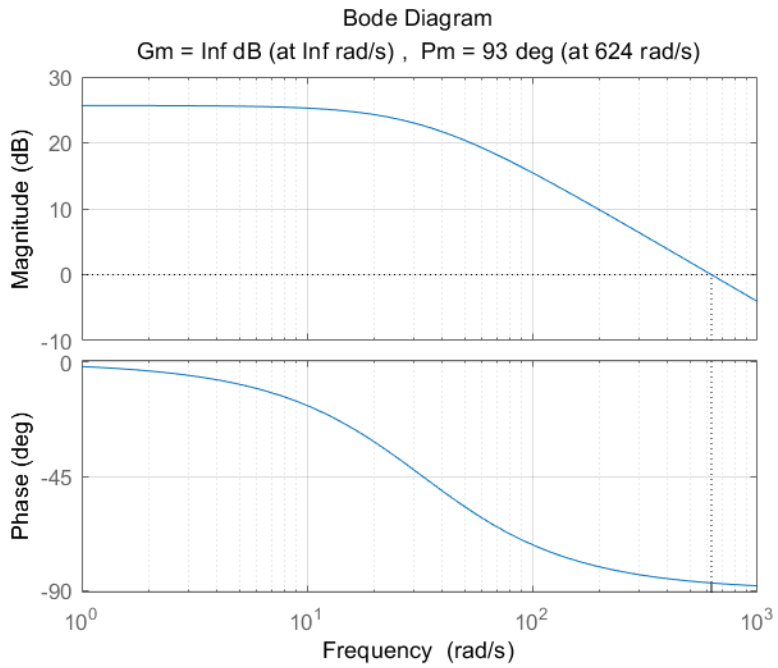
未校正系统 bode 图如下：



2）稳态误差设计

稳态误差设计：$\frac{20}{1+0.046808K}<1$ ，得 K>405.98,取 K=410

带增益后系统 bode 图如下：



设计要求的超调十分小，无法通过经验公式计算得到相角裕度。但相角裕度和超调成反比，可将相角裕度调大。可知此时相角裕度 93°已经很大了，基本满足要求。简单设计串联超前校正环节。

3）设计超前校正环节

$$\phi_m = 135^\circ - 93^\circ + 3^\circ = 45^\circ$$

$$\alpha = \frac{1+\sin\phi_m}{1-\sin\phi_m} = 5.83$$

$$20\lg\left|G_0(j\omega_c)\right| = -10\lg\alpha$$

得

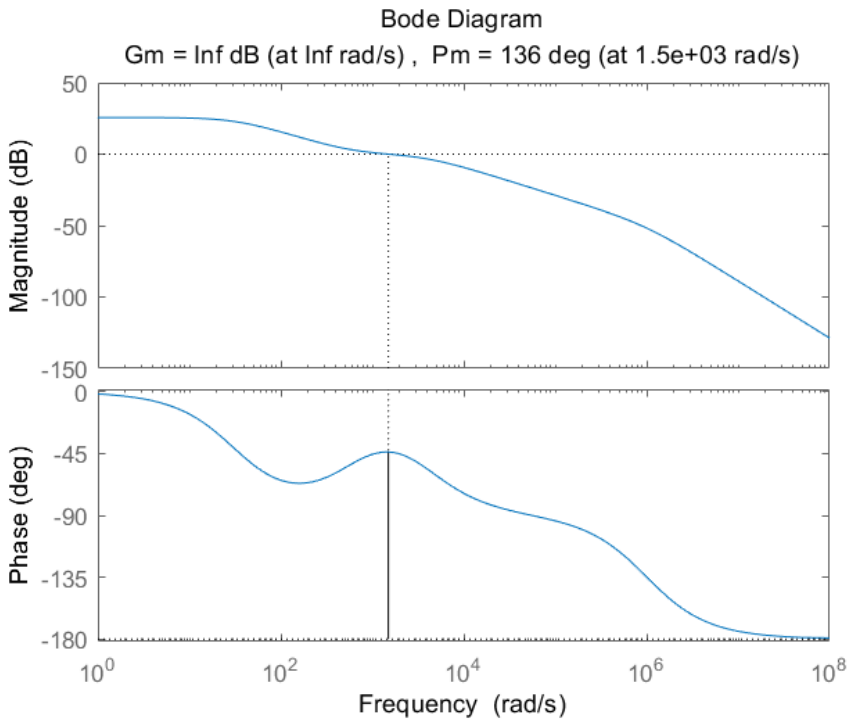
$$\omega_c = 1510 rad/s$$

$$T = \frac{1}{\omega_m\sqrt{\alpha}} = 2.74\times10^{-4}$$

故超前校正环节传递函数为：

$$G_{c1}(s) = \frac{5.83\times2.74\times10^{-4}s+1}{2.74\times10^{-4}s+1}$$

附加串联超前校正后系统开环 bode 图如下：



基本可满足要求。

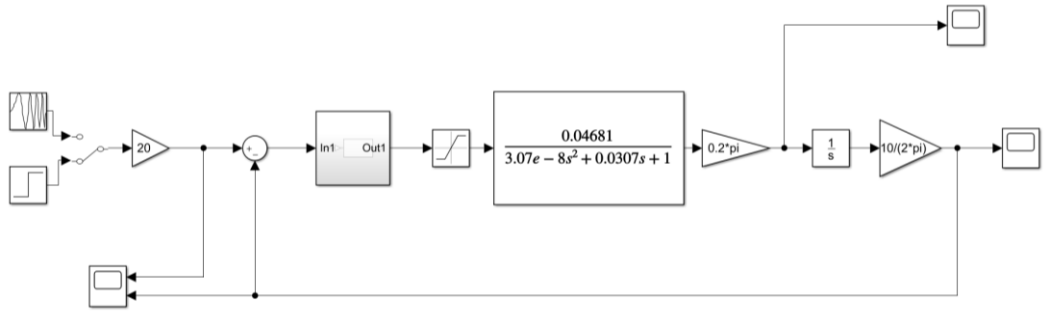
连续控制器方程为：

$$G_c(s) = K\cdot G_{c1}(s) = 410\frac{5.83\times2.74\times10^{-4}s+1}{2.74\times10^{-4}s+1}$$

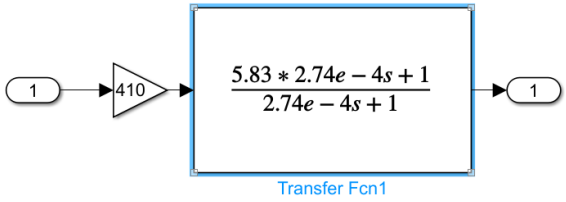
3.4 控制器仿真验证

在 simulink 中搭建被控对象和控制算法的仿真系统，对控制算法进行快速的验证。使得仿真环境下，控制系统的任务指标满足要求。记录控制器仿真验证过程。

① Simulink 仿真框图

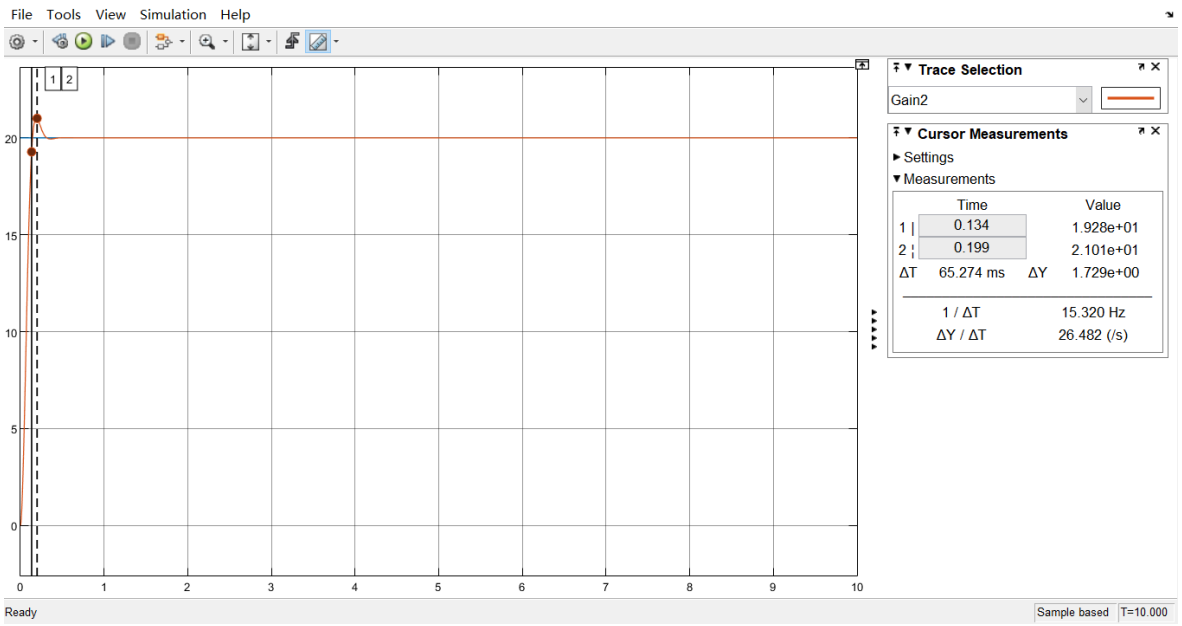


闭环系统框图



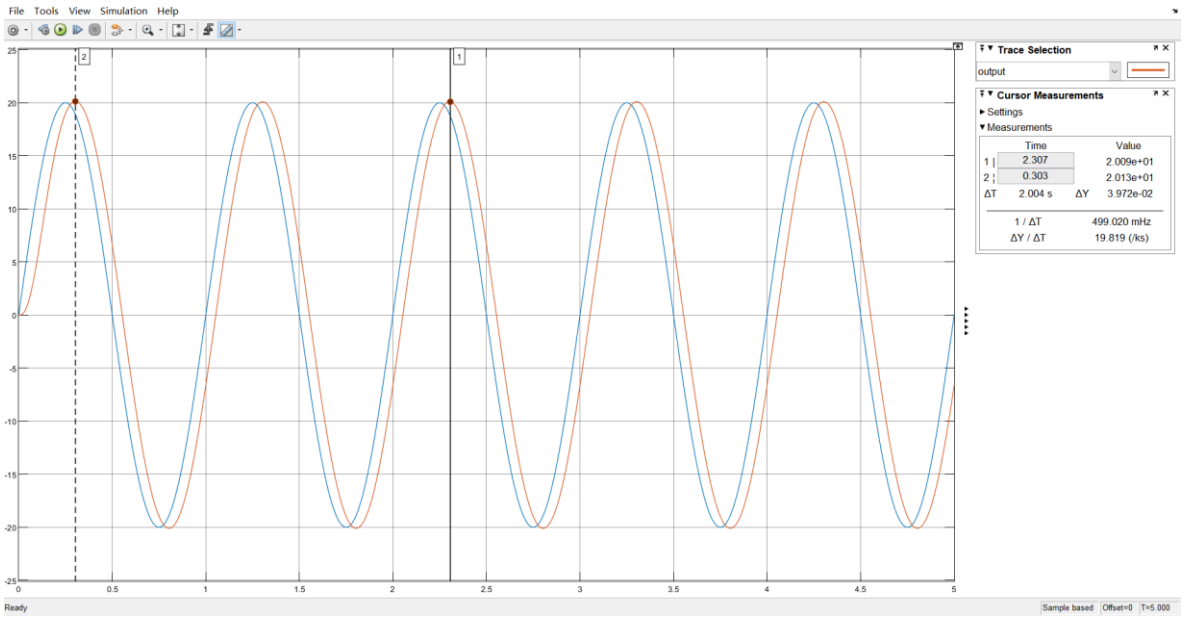
控制器

② 阶跃响应测试结果



满足性能指标要求。

③ 扫频跟随测试结果



满足性能指标要求。

3.5 控制程序开发

1) 把上一节中验证好的控制器，转变为离散化的控制器。写出控制器离散化推导过程与结果；若用 Matlab 推导，则粘贴 Matlab 代码即可。

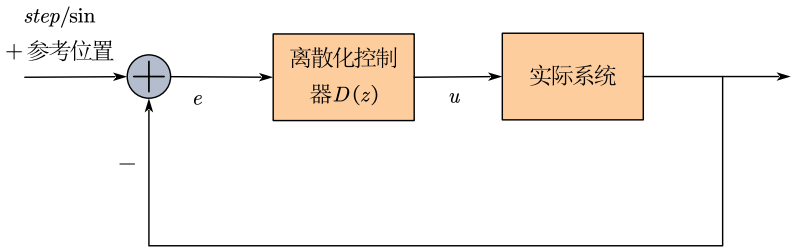
离散化 matlab 代码如下：

```
%连续控制器离散化
s = zpk('s') ;
ks = 410*(5.83*2.74e-4*s+1)/(2.74e-4*s+1) %设置控制器
Ts = 0.01; %控制器周期  Ts
%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)
```

得到离散化结果如下：

```
kz_tf =

   512.9 z + 264.5
   ---------------
      z + 0.8961

Sample time: 0.01 seconds
Discrete-time transfer function.
```

2) 将控制器源代码粘贴在下方，并写好注释。




```
//控制器如下
e_1=e;//残差信号
 e=ref-mile;//ref 为参考输出位置
 u_1=u;
 u=-0.8961*u_1+512.9*(e+0.5157*e_1);//控制器
//输入信号饱和
 if(u>4997)u=4997;
if(u<-4997)u=-4997;
```

3.6 控制系统调试

1) 将阶跃响应测试源代码粘贴在下方，并写好注释。

```
/*实现一个阶跃测试函数，放在 main 的 while(1)中进行阶跃测试*/
void function_step_test(void)
{
  /*1.定义局部变量和静态变量*/
  static uint32_t last_sys_tick = 0;
  static uint32_t init_flag = 0;
  uint32_t sys_tick = 0;
  double Amp=20;
  static double ref=0;
  printf("step test \n");
  int16_t input = 0;

  /*2.初始化你的控制器*/
  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
  MC_StartMotor1();
  if ((find_home_flag == 1) && (MC_GetSTMStateMotor1() == RUN))
  {
    if(init_flag == 0)
    {
      Measure();
```

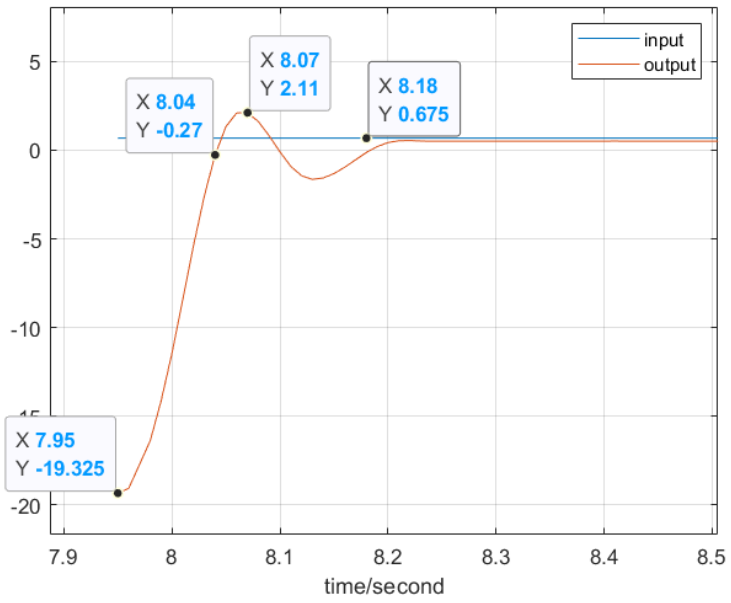
```
      ref=Amp+mile;
      init_flag=1;
      e_1=0;e=0;u=0;u_1=0;
    }
    if (last_sys_tick != sys_tick)/*4.分频器确定当前 tick 为控制器运行的 tick*/
    {
      last_sys_tick = sys_tick;
      if (sys_tick % 10 == 0) //降频
      {
        /*5.获取光栅尺的当前位置 mile (mm)*/
        Measure();

        /*8.将 ref 作为你的位置控制器的输入命令，调用控制器*/
        e_1=e;//残差信号
        e=ref-mile;//ref 为参考输出位置
        u_1=u;
        u=-0.8961*u_1+512.9*(e+0.5157*e_1);//控制器
        if(u>4997)u=4997;
        if(u<-4997)u=-4997;
        MC_ProgramTorqueRampMotor1(u,0);
        /*9.使用 send_data_2_matlab，把时间，ref，mile 发送到 matlab 进行显示*/
        send_data_2_matlab(time,(float)ref,(float)mile);
      }
    }
    else
    {
      /*10. do some thing if need*/
    }
  }
  else
  {
    /*11. do some thing if need*/
  }
}
```

2) 将扫频跟随测试源代码粘贴在下方，并写好注释。

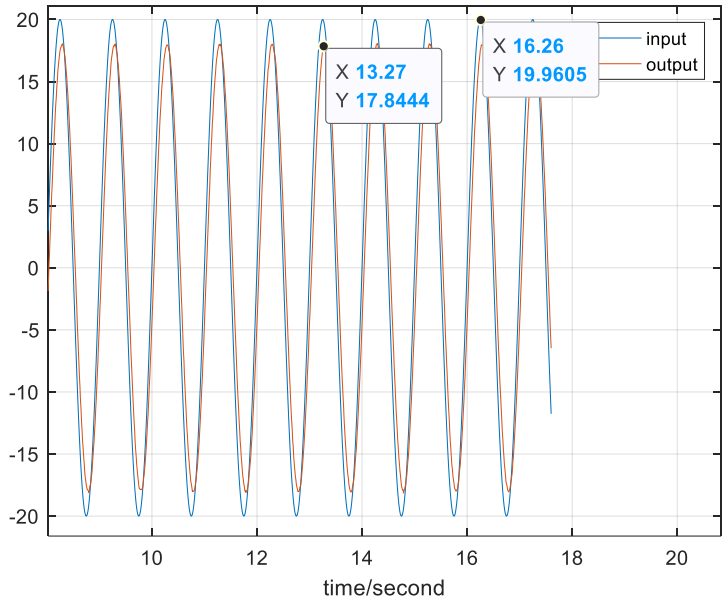
```
//扫频跟随
void function_sin_test(void)
{
  static my_sweep_t sweep = {0};
  double sweep_input = 0;
  int16_t sweep_output = 0;
  uint32_t sys_tick = 0;
```

```
  static uint32_t init_flag = 0;
  static uint32_t last_sys_tick = 0;
  static uint32_t start_sys_tick = 0;
  double f0=1;
  sys_tick = HAL_GetTick(); //获取当前时刻，单位 ms
  time = 0.001f * sys_tick; //单位 s

  static double ref=0;

  MC_StartMotor1();
  if ((find_home_flag == 1) && (MC_GetSTMStateMotor1() == RUN))
  {
    //初始化扫频配置
    if (init_flag == 0)
    {
      Measure();
      ref=mile;
      init_flag = 1;
      e=0;e_1=0;u_1=0;u=0;
    }
    if (last_sys_tick != sys_tick)
    {
      last_sys_tick = sys_tick;
      if (sys_tick % 10 == 0) //降频
      {
        //获取正弦扫频信号
        Measure();
        sweep_input = 20*sin(2*PI*f0*((double)sys_tick-sweep.t_0)/1000);
        e_1=e;
        e=ref+sweep_input-mile;
        u_1=u;
        u=-0.8961*u_1+512.9*(e+0.5157*e_1);
        if(u>4997)u=4997;
        if(u<-4997)u=-4997;
        MC_ProgramTorqueRampMotor1(u,0);
        //把时间，input，output 发送到 matlab
        send_data_2_matlab(time,(float)ref+sweep_input,(float)mile);
      }
    }
  }
}
```

3) 将调优后的阶跃响应曲线粘贴在下方，对于 2cm 的阶跃响应，要求 95%的上升时间不超过 0.1s；超调量不大于 10%；稳态误差小于 1mm;



4) 将调优后的扫频跟随曲线粘贴在下方，要求对于幅值为 2cm 的正弦信号，以-3dB 为截止带宽标准的闭环带宽不小于 1hz;



5) 写出经过调试后，最终的控制器传递函数。

$$G_c(s) = 410\frac{5.83\times 2.74\times 10^{-4}s+1}{2.74\times 10^{-4}s+1}$$

3.7 控制器设计的不足与改进

说说你的控制器设计的不足之处，以及该如何改进？

该模型未考虑噪声和扰动的影响。
对于噪声，应该考虑等效噪声带宽，进行系统设计；
对于扰动，可以设计扰动观测器进行扰动抑制，或者进行扰动辨识然后再进行前馈设计，抑制扰动信号对输出的影响。

3.8 实验总结

说说自己在整个控制系统设计过程中遇到的主要问题，及最终解决方法。

由于要求的超调很小，无法通过经验公式转化为开环频域指标，故设计系统时仅在趋势上将相角裕度调大，但最后在串联超前校正环节作用下，系统响应满足了要求的性能指标。