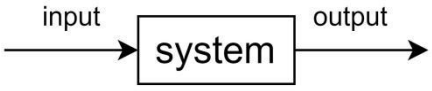
电机控制系统仿真实验

哈尔滨工程大学创梦之翼战队电控组

1 认识系统

1.1 系统的基本概念

总的来说，系统是由相互联系的部分组成的一个更大更复杂的整体。为了直观，我们一般通过一个带有向内和向外箭头的方框表示系统：

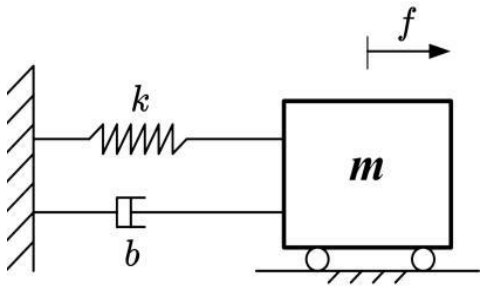


其中指向方框的箭头表示系统的输入，由方框指向外的箭头表示系统的输出。系统的输出与输入并非相互

独立，相反的，系统在输入信号的激励下会产生相应的输出响应。上图表示的系统只有单个的输入与输出，我们一般称这种系统为单输入单输出（single-input single-output SISO）系统。相应的，存在多个输

入输出的系统被称之为多输入多输出（multi-input multi-output MIMO）系统：

以一个弹簧阻尼系统为例：



一个质量为 m 的光滑物块通过劲度系数为 k，阻尼系数为 b 的弹簧与数值墙面连接。施加给物体 f 的外力，以向右为正方向。物块位移也以向右为正方向。对物块使用受力分析：

其中：

$$\begin{cases} v = \dot{x} \\ a = \ddot{x} \end{cases}$$

即：

$$f = m\ddot{x} + b\dot{x} + kx$$

不难发现，在这个系统中，力 引起位移 的变化。因此力 便是系统的输入，系统在力 f 的激励下会产生相应地输出响应，即位移 的变化。

不难发现，在这个系统中，力 f 引起了位移 x 的变化。因此力 f 就是系统的输入，系统在 f 的激励下会产生相应的输出响应，即位移 x 的变化。

1.2 线性时不变系统

以下讨论均建立在系统是线性时不变系统的前提上，需先了解线性时不变系统（linear time-invariant LTI）的概念。

线性指的是系统输入输出之间的关系是线性映射：如果输入信号 $x_1(t)$使系统产生输出响应为 $y_1(t)$，输入信号 $x_2(t)$使系统产生的响应为 $y_2(t)$，则输入 $a*x_1(t)+b*x_2(t)$会使系统产生 $a*y_1(t)+b*y_2(t)$的输出响应。即系统满足叠加定理。

时不变指系统输入延迟τ秒则其输出响应延迟τ秒：如果输入信号 x(t)使系统产生输出响应 y(t)，则输入信号 x(t+τ)的输出响应为 y(t+τ)。

1.3 系统建模

系统建模即通过牛顿定律、基尔霍夫电压定律等物理定律建立描述动态系统的微分方程，上文中二阶系统的例子便是这个过程。对于一阶二阶线性微分方程而已，求其解析解尚且不是难事，但对于阶数更高的微分方程则难以直接求解，因此可以通过拉普拉斯变换将其由关于时间 t 的线性微分方程变换为关于复数 $s = \sigma + j\omega$ 的多项式代数方程：

$$\mathcal{L}\{f\} = \mathcal{L}\{m\ddot{x} + b\dot{x} + kx\}$$

其中：

$$\begin{cases} \mathcal{L}\{f(t)\} = F(s) \\ \mathcal{L}\{x(t)\} = X(s) \\ \mathcal{L}\{\dot{x}(t)\} = sX(s) \\ \mathcal{L}\{\ddot{x}(t)\} = s^2X(s) \end{cases}$$

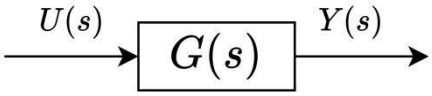
即：

$$F(s) = ms^2X(s) + bsX(s) + kX(s)$$

根据力 f 引起位移 x 的变化这一因果关系，可以确定系统输入 U(s)=F(s)，确定系统输出 Y(s)=X(s)。将上述表达式变换为输出/输入的形式，从而体现输入输出的因果关系，从而得到了系统的传递函数：

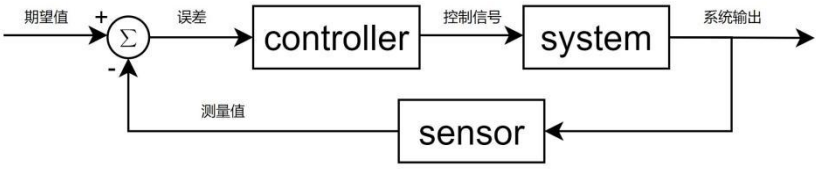
$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{ms^2 + bs + k}$$

不难看出，传递函数描述的便是这个系统输入输出的关系，即：



1.4 系统控制

系统控制关注什么样的输入信号能使系统产生符合我们期望的输出响应。这要求我们设计合适的控制器和系统结构以得到恰当的系统输入。要得到符合我们期望的系统输出除了考虑动态系统本身，还需要考虑外界扰动的影响，因此反馈控制应用最为广泛。反馈控制系统的结构可表示为：

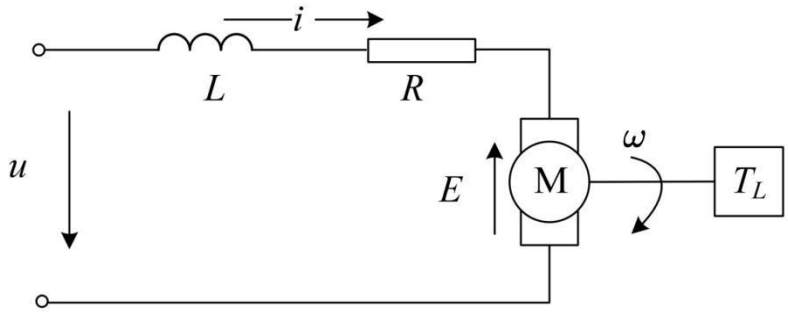


通过反馈回路，我们得以将系统输出的测量值与期望值作比较，以得到误差。误差反应的是当前系统实际表现与我期望的表现两者的差距，通过设计的控制器可以利用误差信号得到恰当的控制信号以减小误差信号，即使系统输出跟随期望值。

2 系统组成与基本原理

2.1 电机系统建模

采用电压控制的直流电机等效电路图如下：



图中 L 和 R 分别表示电机绕组的电感和电阻，T_L 表示负载转矩。在本实验中假设电机为空载，即 $T_L=0$。绕组的电流 i 产生电磁转矩 $T=K_t i$，从而使电机转动。其中 K_t 为转矩常数。另外，电机在转动时会产生感应电动势 $E=Ke\omega$，其中 Ke 称为反电动势常数。反电动势 E 的方向与电流 i 方向相反。

根据基尔霍夫电压定律，有：

$$u(t) = Ri(t) + L\frac{\mathrm{d}i(t)}{\mathrm{d}t} + E(t)$$

根据牛顿定律，有：

$$T = J\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} + b\omega$$

其中 b 为阻尼系数。带入反电动势方程 $E = K_e\omega$与 电磁转矩方程 $T = K_t i$，有：

$$u(t) = Ri(t) + L\frac{\mathrm{d}i(t)}{\mathrm{d}t} + K_e\omega(t)$$

$$K_t i(t) = J\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} + b\omega(t)$$

通过拉普拉斯变换得到其 s 域表达式：

$$\Omega(s) = \frac{K_t I(s)}{Js+b}$$

$$I(s) = \frac{U(s)-K_e\Omega(s)}{Ls+R}$$

两式合并可得到电压到转速的传递函数 $G_v(s)$：

$$G_v(s) = \frac{\Omega(s)}{U(s)} = \frac{K_t}{JLs^2 + (JR+Lb)s + Rb + K_eK_t}$$

2.2 电机模型单片机仿真代码

代码包含 motor_simulation.lib，motor_simulation.h，bsp_dwt.c，bsp_dwt.h 四个文件。

Motor_simulation.h:

```
typedef struct motorparam
{
    float J;
    float b;
    float Kt;
    float Ke;
    float R;
    float L;
    float constFriction;
} motorParam_t;

typedef struct motor
{
    float U;
    float I;
    float Velocity; // rad/s
    float Angle;    // rad
    float dI;
    float dV;
    float lastdI;
```

```
    float lastdV;
    float lastVelocity;
    float MeasureI;
    float MeasureVelocity; // rad/s
    float MeasureAngle;    // rad
    float maxU;
    motorParam_t motorParam;
} motorObject_t;//电机结构体 包含了各种电机信息

/**
 * @brief Init motor object
 * @param[in]       *motor points to the motorObject_t struct
 */
void Motor_Object_Init(motorObject_t *motor);//传入电机结构体变量 初始化电机

/**
 * @brief Motor simulation implementation
 * @param[in]       *motor points to the motorObject_t struct
 * @param[in]       input voltage
 * @param[in]       simulation period in s
 */
void Motor_Simulation(motorObject_t *motor, float input, float dt);//传入电机结构体
变量，输入电压，控制周期
/**
 * @brief Get motor current
 * @param[in]       *motor points to the motorObject_t struct
 * @param[out]      motor current
 */
float Get_Motor_Current(motorObject_t *motor);//传入电机结构体 返回电机电流值
/**
 * @brief Get motor velocity
 * @param[in]       *motor points to the motorObject_t struct
 * @param[out]      motor velocity
 */
float Get_Motor_Velocity(motorObject_t *motor);//传入电机结构体 返回电机速度值
/**
 * @brief Get motor angle
 * @param[in]       *motor points to the motorObject_t struct
 * @param[out]      motor angle
 */
float Get_Motor_Angle(motorObject_t *motor);//传入电机结构体 返回电机角度值
```

main.c（例）：

```
motorObject_t Motor; //创建电机结构体变量
```

```c
pid_t PID;//创建 PID 结构体变量
uint32_t DWT_CNT;
float dt;
float t;

int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
```

```c
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  DWT_Init(???);//DWT 定时器初始化 请自行思考需要填入几
  Motor_Object_Init(&Motor);//电机初始化
  PID_Init(???);//请自行编写 PID 初始化函数
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    dt = DWT_GetDeltaT(&DWT_CNT);//利用 DWT 定时器获得仿真周期
    t += dt;
    Current = Get_Motor_Current(&Motor);//读取电机电流
    Velocity = Get_Motor_Velocity(&Motor);//读取电机转速
    Angle = Get_Motor_Angle(&Motor);//读取电机角度

    Input = PID_Calculate(???);//自行编写 PID 计算函数，计算出输入电压

    Motor_Simulation(&Motor,Input,dt);//将输入电压值传给电机，使电机运转
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
    HAL_Delay(1);
  }
```

```
  /* USER CODE END 3 */
}
```

2.3 PID 控制

请大家自行查阅资料，可参考：

1. RoboMaster 电控入门（4）PID 控制器 - sasasatori - 博客园 (cnblogs.com)
2. PID 库与 PID 基本优化（一） - WangHongxi - 博客园 (cnblogs.com)
3. PID 控制器开发笔记之一：PID 算法原理及基本实现 - Moonan - 博客园 (cnblogs.com)
4. 串级 PID 为什么外环输入是内环的期望？ - 知乎 (zhihu.com)
5. PID 的 TRICK(一)简述五种 PID 积分抗饱和（ANTI-Windup）方法 - 知乎 (zhihu.com)
6. STM32 应用(十)经典控制算法 PID(单级和串级)原理与代码实现三木今天学习了嘛の博客-CSDN 博客 stm32pid 算法

3 实验内容与要求

3.1 控制系统设计与实现

3.1.1 速度闭环

设计反馈控制系统实现电机速度闭环控制。要求掌握闭环控制基本原理与代码实现、三项参数如何影响系统表现。并了解参数整定基本方法。

1. 阶跃响应：输入阶跃期望信号 velocityRef = 10 rad/s，检验闭环系统性能
2. 斜坡响应：输入斜坡期望信号 velocityRef = t rad/s，检验系统闭环性能
3. 频率响应：输入正弦期望信号 velocityRef = 10sin(ωt)rad/s，其中ω能使输出的振幅为系统输入振幅的 0.707 倍。即 velocity = 7.07sin(ωt + φ) rad/s。

3.1.2 角度闭环

分别设计单级反馈控制系统与串级反馈控制系统实现电机角度闭环控制，并对比分析两种控制系统优缺点（通过响应速度、精度与抗扰性能三方面分析）。要求明确串级控制系统结构，掌握串级控制系统相比单级反馈控制系统优缺点。并了解参数整定基本方法。

1. 阶跃响应：输入阶跃期望信号 angleRef = 2π rad，检验闭环系统性能
2. 频率响应：输入正弦期望信号 angleRef = 2πsin(ωt)rad，其中ω能使输出的振幅为系统输入振幅的 0.707 倍。即 angle = 1.414πsin(ωt + φ) rad。
3. 抗扰性能测试：设定期望 angleRef = 0 rad，在电机电压输入端加入扰动信号 disturbance = 10V，对比单级反馈控制和串级反馈控制对扰动输入的响应。

3.2 效果呈现

利用串口调试助手，ozone 等手段，绘制期望曲线与响应曲线，并进行对比研究。

4 扩展任务

4.1 传递函数推导与仿真

通过研究电机系统的响应，推导出电机的传递函数，并在 matlab 中进行仿真。用根轨迹，伯德图等知识对电机系统进行分析，并设计控制器。

4.2 复合控制

自行查阅资料，采用能想到的各种手段设计更好的控制器（PID 优化，前馈等），提高系统的性能。进行 Matlab 仿真和单片机程序仿真，并与之前的反馈控制进行性能对比。

4.3 滤波

在 Matlab 仿真中给系统的反馈信号叠加噪声，并自行设计滤波器滤除噪声，比较滤波前后的控制效果。