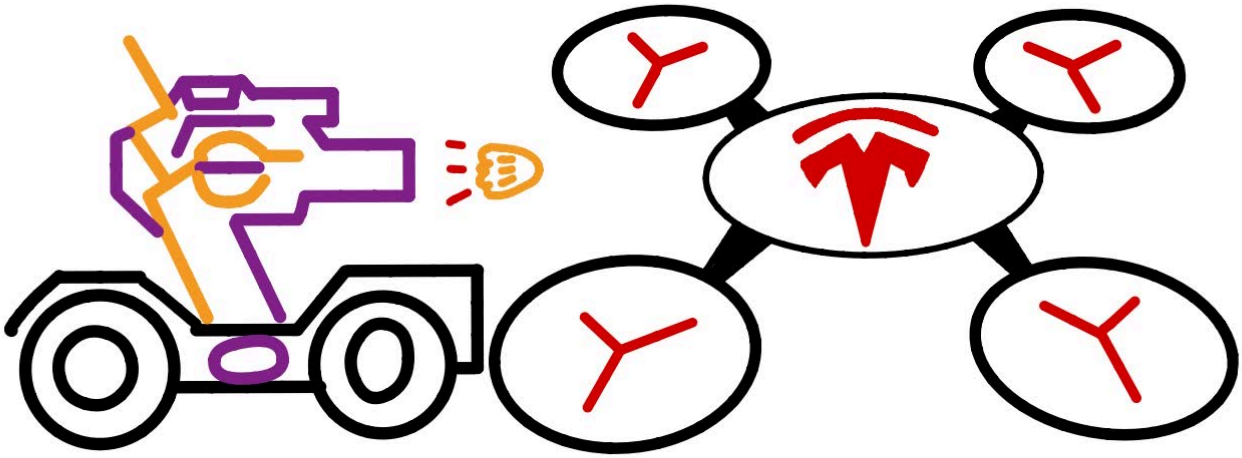
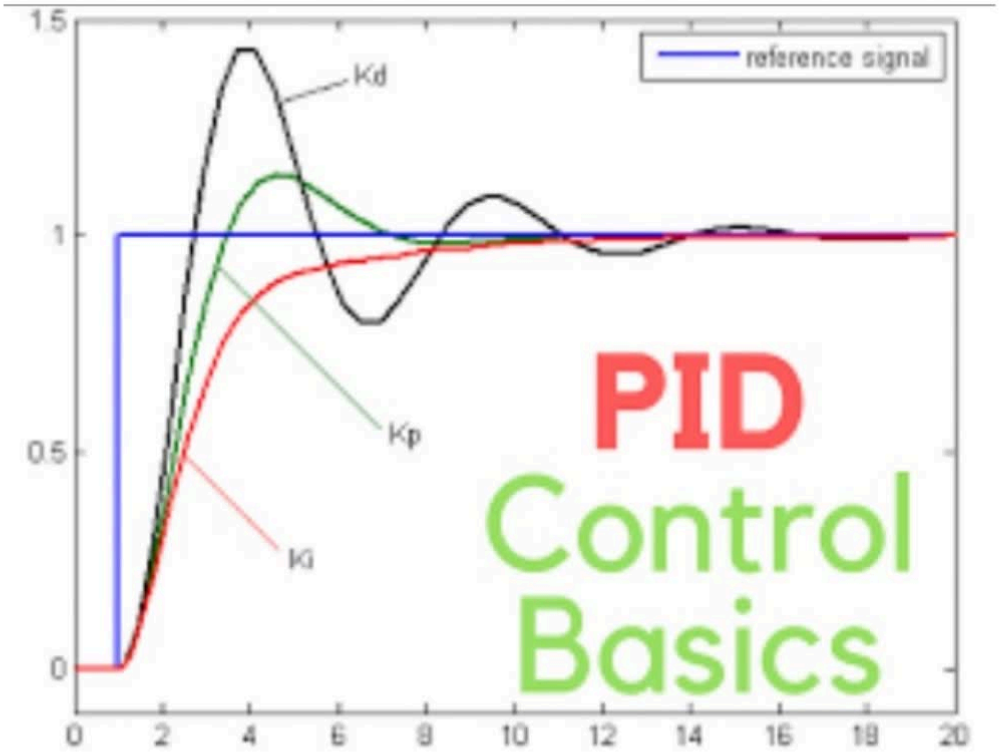
PID

从不懂到会用



P I D

从不懂到会用



Clang

PID Control Theory

Catalog

听懂

- 一. 不懂☆
  - 1. 引入
  - 2. 适用系统
  - 3. 宏观意义
- 二. 略懂☆☆
  - 1. 控制系统概述
  - 2. 连续与离散信号
- 三. 懂的都懂 ☆☆☆
  - 1. PID公式解释（抽象派）
  - 2. PID公式解释（形象派）
  - 3. PID参数整定（速讲）
  - 4. 其余相关控制知识

会用

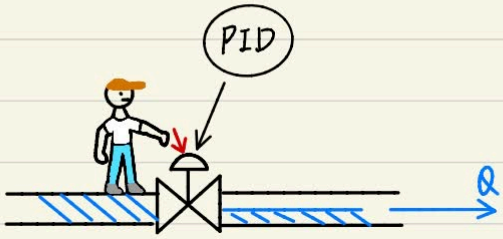
- 四. 会用就行 ☆☆☆☆
  - 0. 前期分析
  - 1. RM 电机 M3508 速度环
  - 2. RM 电机 [illegible]

一、不懂 ☆

1. 引入

① 流量稳定

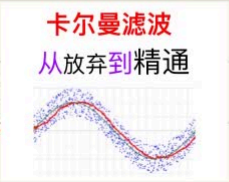
② 改变流量



升/秒 (L/s)

2. 适用系统

适用线性系统 (在个人空间



视频<<入门刀章提到.)

准确：二阶以内线性系统

① 齐次性

② 叠加性

$y = f(x)$

$y_1 = f(x_1)$

$y_2 = f(x_2)$

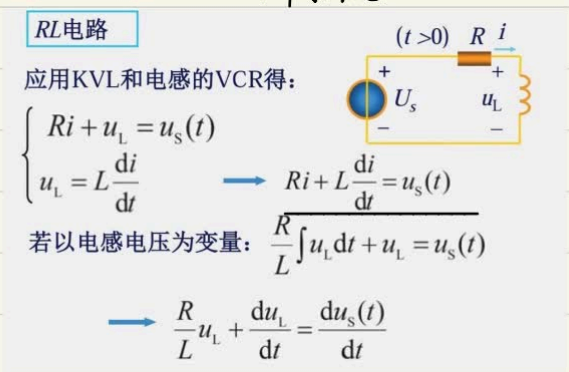
$y_1 + y_2 = f(x_1) + f(x_2) = f(x_1 + x_2)$

0阶 $y = kx$

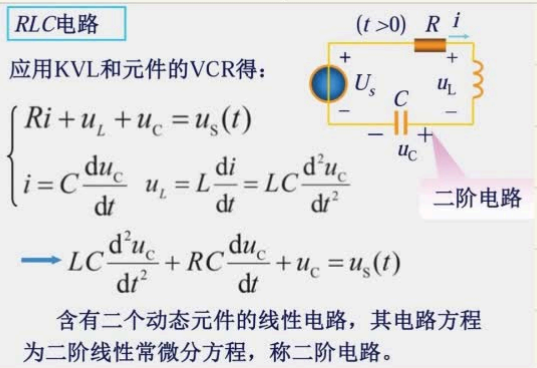
一阶 $T\frac{dy}{dt} + y = kx$

二阶 $T^2\frac{d^2y}{dt^2} + 2\zeta T\frac{dy}{dt} + y = kx$

一阶系统

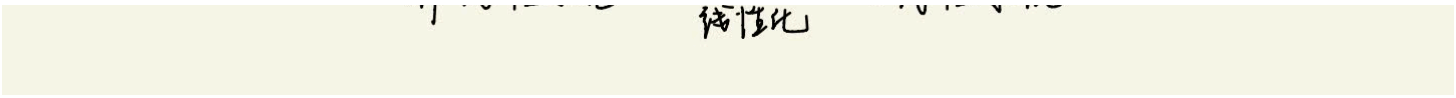


二阶系统



不适用系统：高阶系统 $\xrightarrow{\text{降阶}}$ 二阶系统

非线性系统 $\xrightarrow{\text{李雅普诺夫第一法}}$ 线性系统



3. 宏观意义

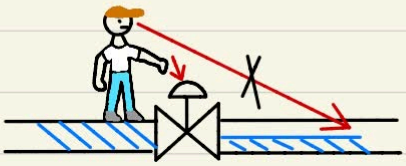
PID 95%

简单，无须精确建模

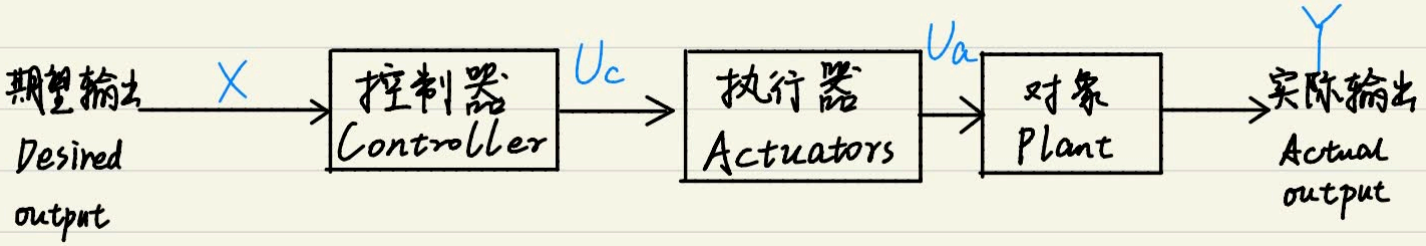
二. 略懂☆☆

1. 控制系统概述

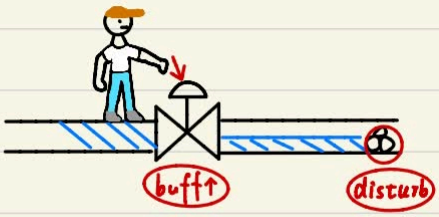
① 开环控制系统



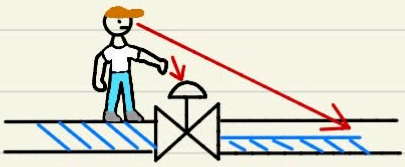
A、一般开环控制系统



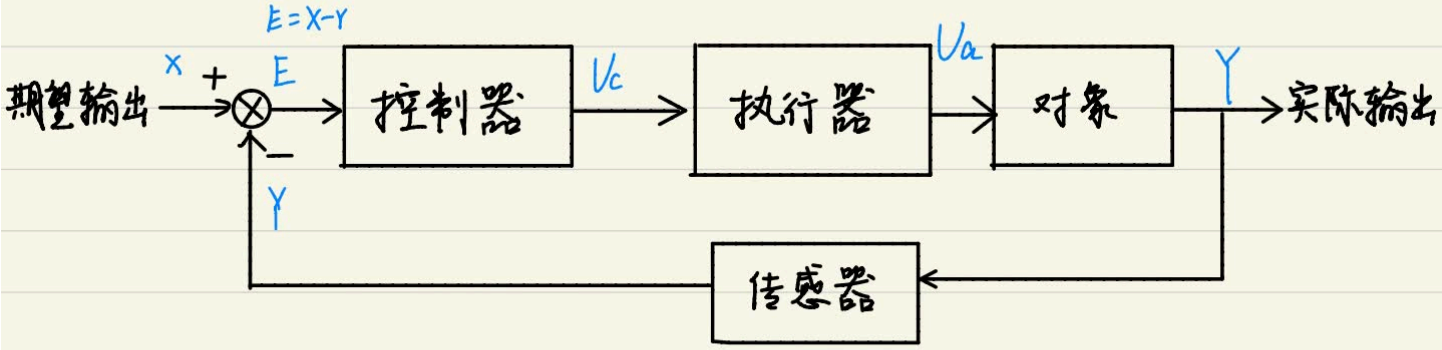
B、前馈控制系统



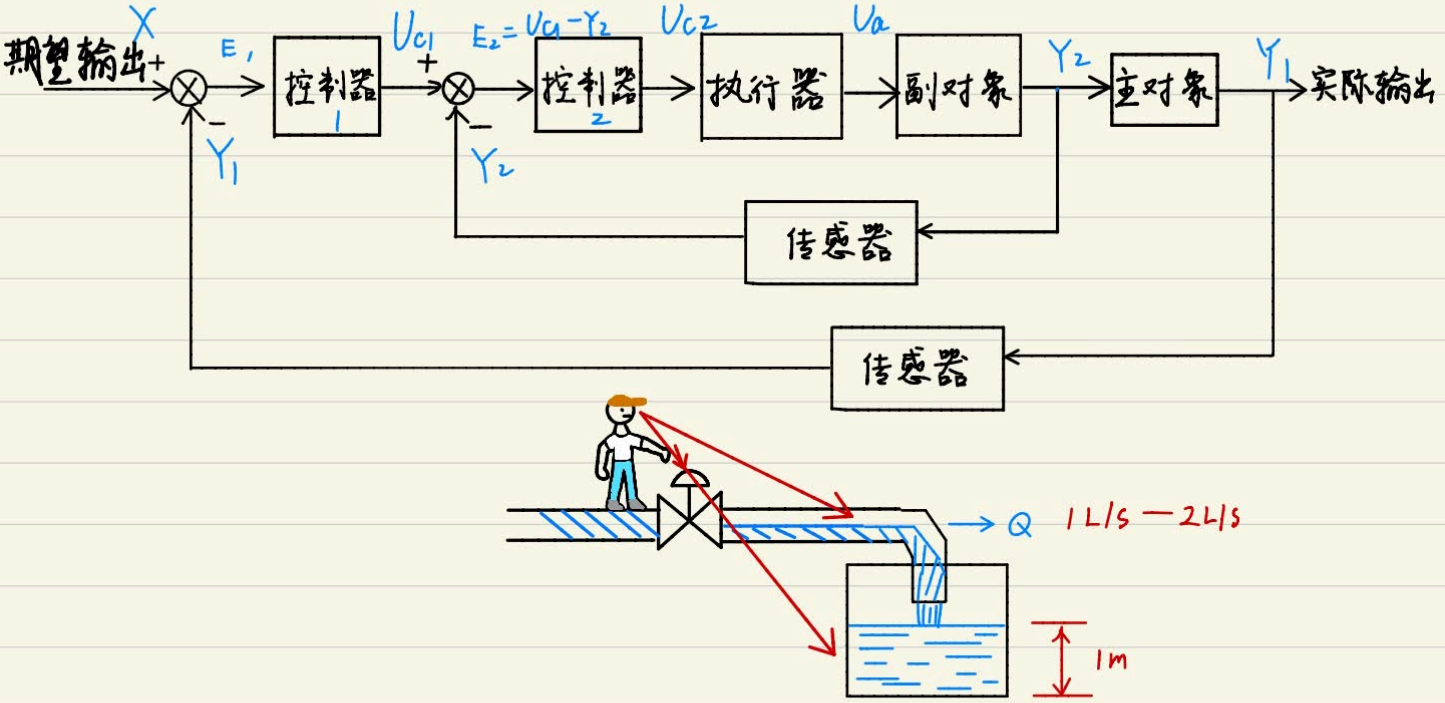
② 闭环控制系统



A、单闭环

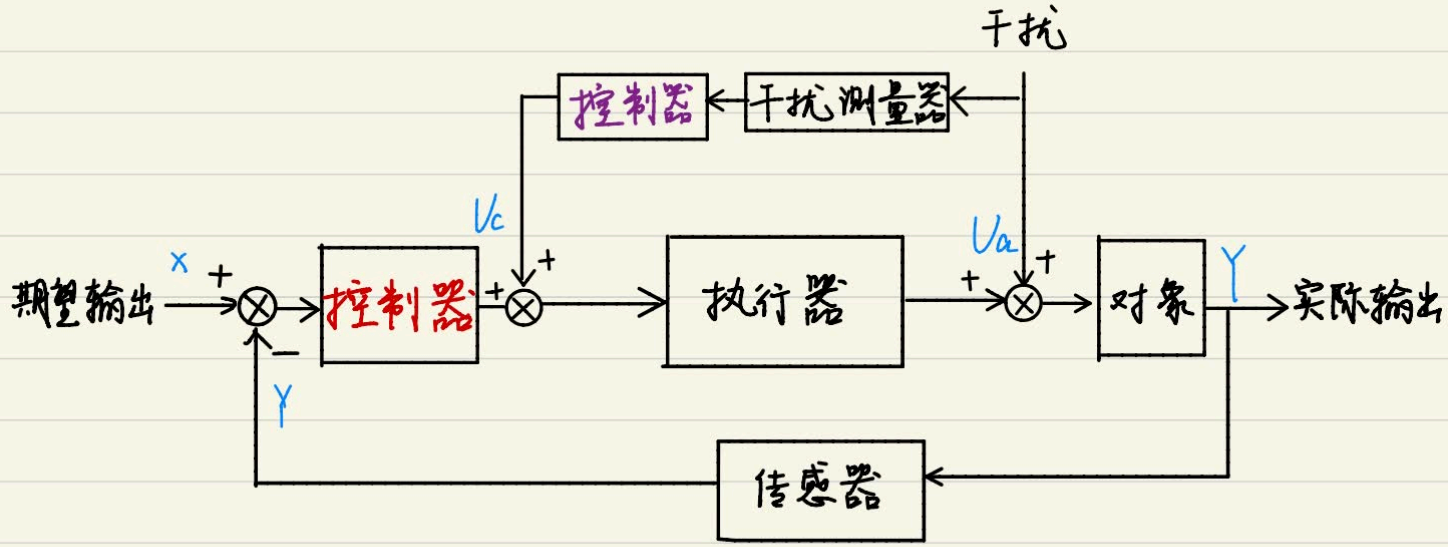


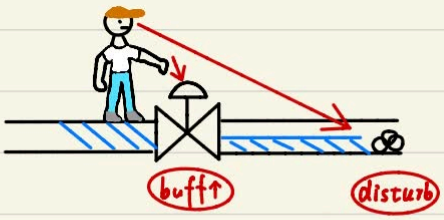
B、双闭环



③ 复合控制系统

前馈—反馈复合控制系统





③ 参数详解

a. 误差

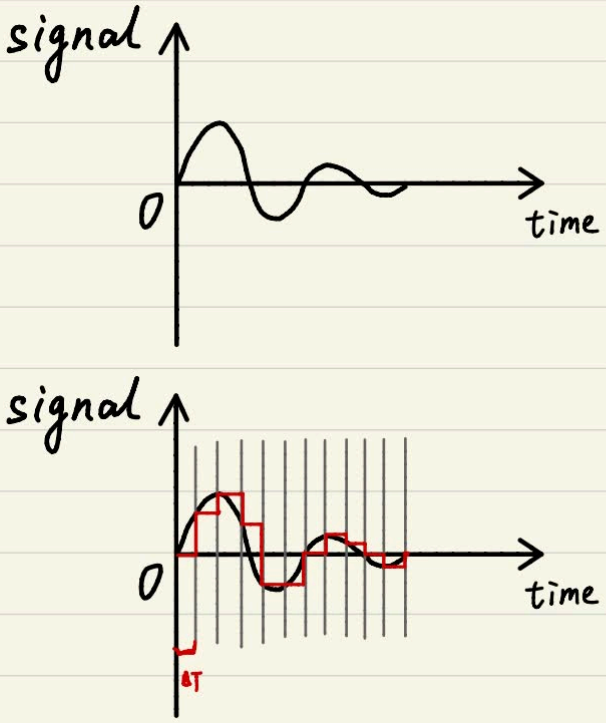
b. 控制器输出

c. 执行器输出

d. 系统输出

2. 连续与离散信号

① 图形表示



② 信号算式表示

$$\int_0^t x(t)\,dt \longrightarrow \sum_{n=0}^{N} X(n)$$

$$\frac{dx(t)}{dt} \longrightarrow \frac{X(n)-X(n-1)}{\Delta t}$$

三. 懂的都懂 ☆☆☆

1. PID公式解释（抽象派）

$$C = \frac{1}{P}\left(e + \frac{1}{T_i}\int_0^t e\,dt + T_d\frac{de}{dt}\right)$$

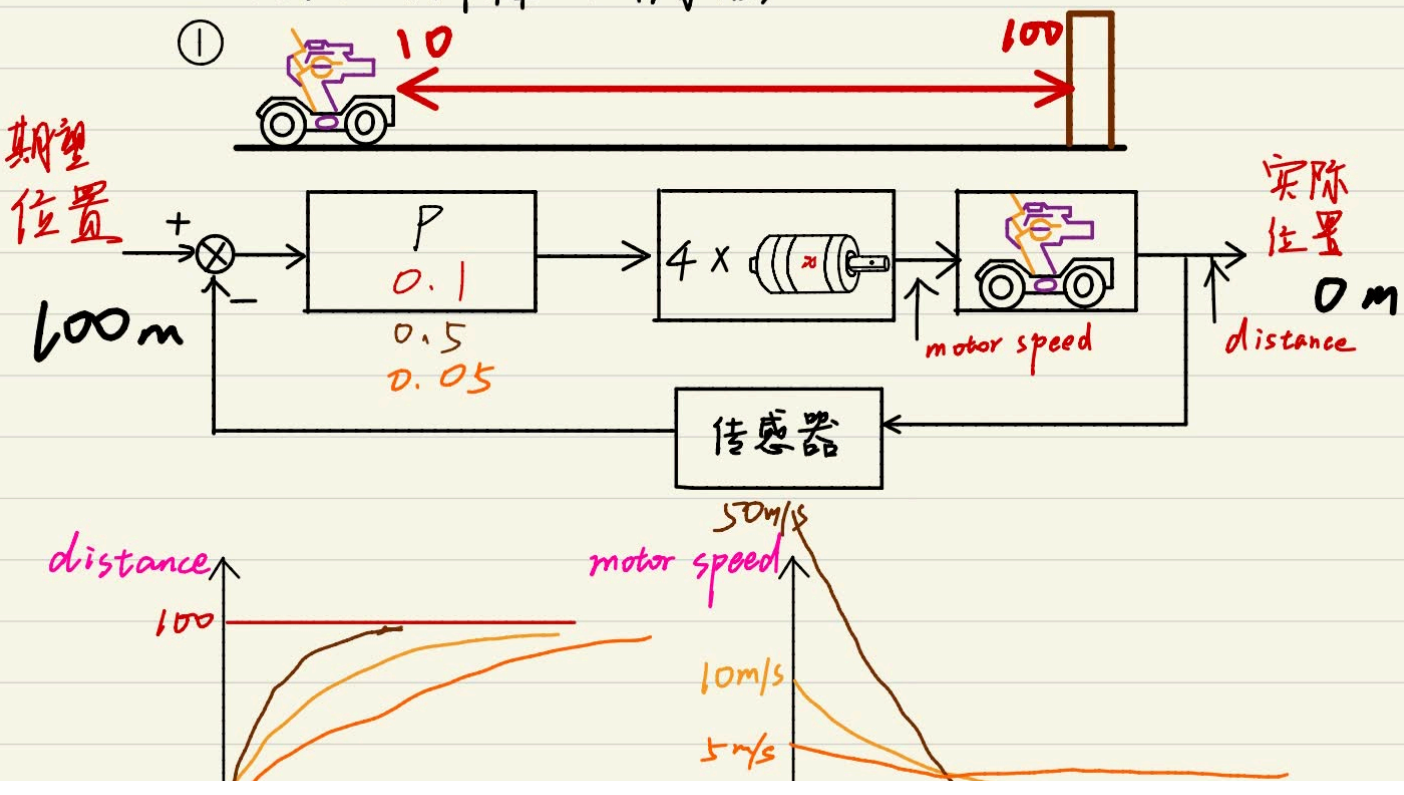
⇓ 实际常用

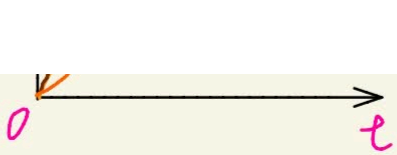
连续 $$C = K_p e + K_i\int_0^t e\,dt + K_d\frac{de}{dt}$$

离散 $$C = K_p e_i + K_i\sum_{i=1}^{N} e_i + \frac{K_d(e_i - e_{i-1})}{\Delta t}$$

$$= K_p e_i + K_i\sum_{i=1}^{N} e_i + K_d(e_i - e_{i-1})$$

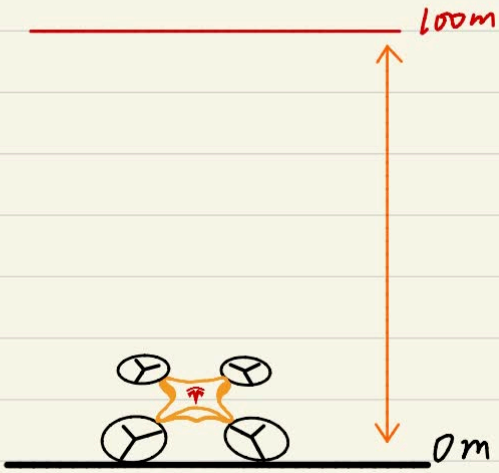
2. PID公式解释（形象派）

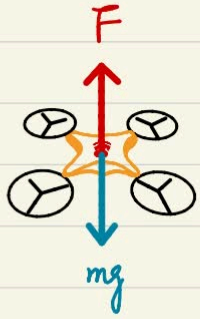



0
t


0
t

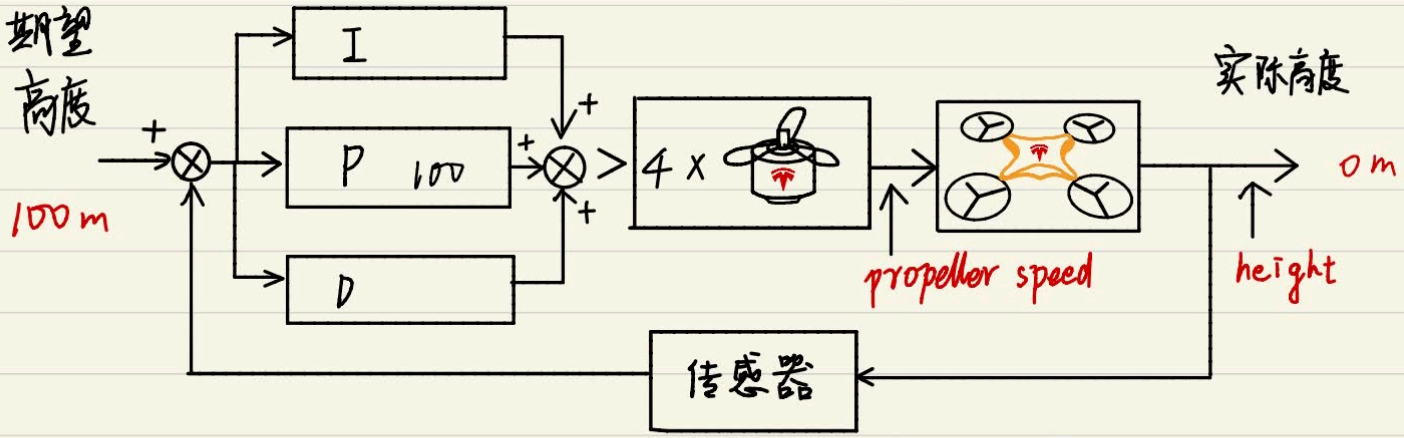
②



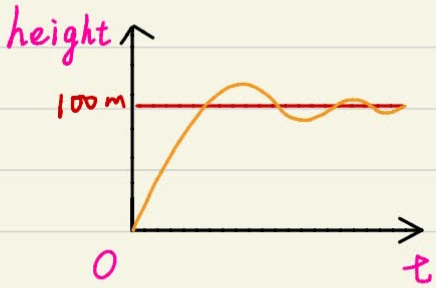


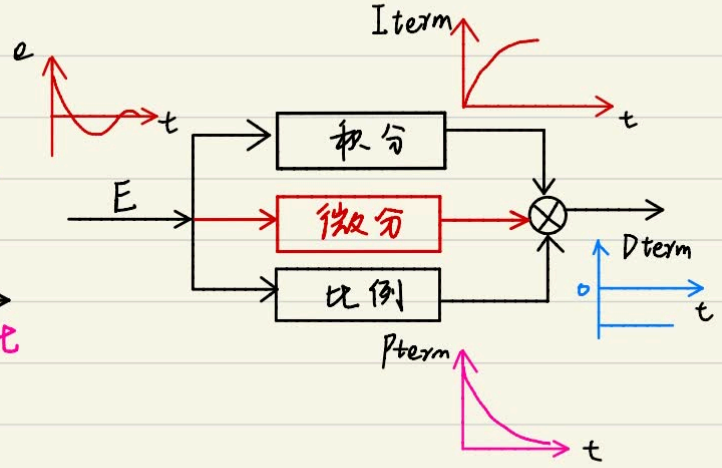
propeller speed

100 rpm

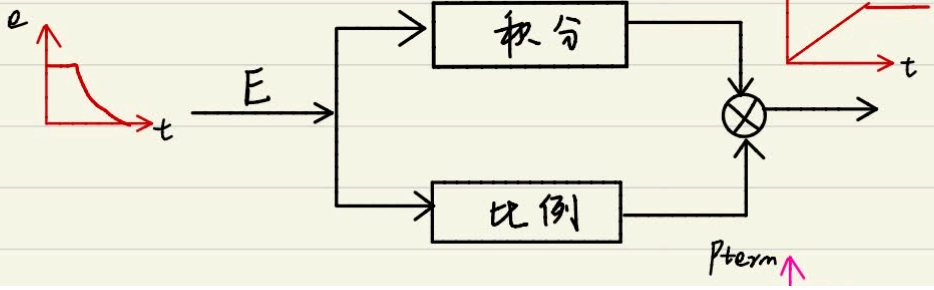


D项分析





I项分析

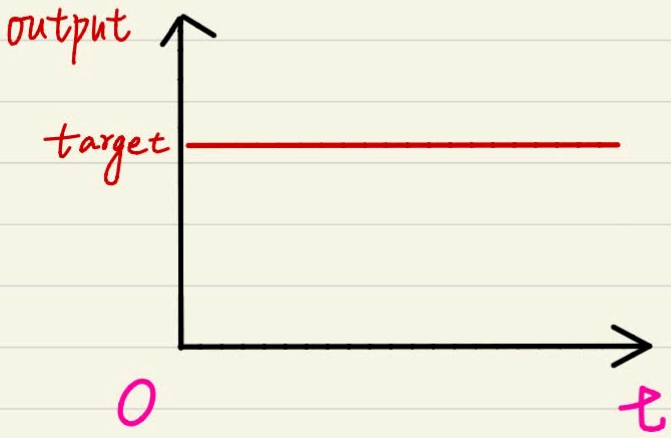



t

P 项分析

偏差（高度偏差）		P 值		控制信号
100	×	1	=	100
50	×	2	=	100
25	×	4	=	100
10	×	10	=	100
1	×	100	=	100

3. PID 参数整定（速讲）



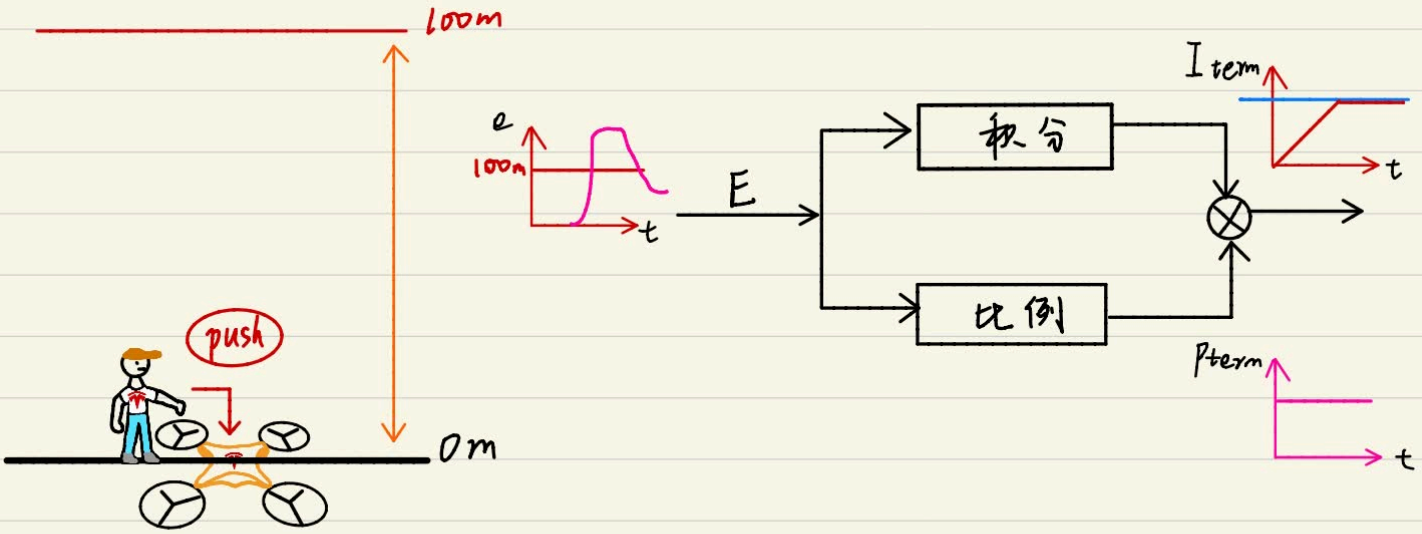
$K_p \cdot e_i$

$K_i \cdot \left(\sum_{i=0}^{N} e_i\right)$

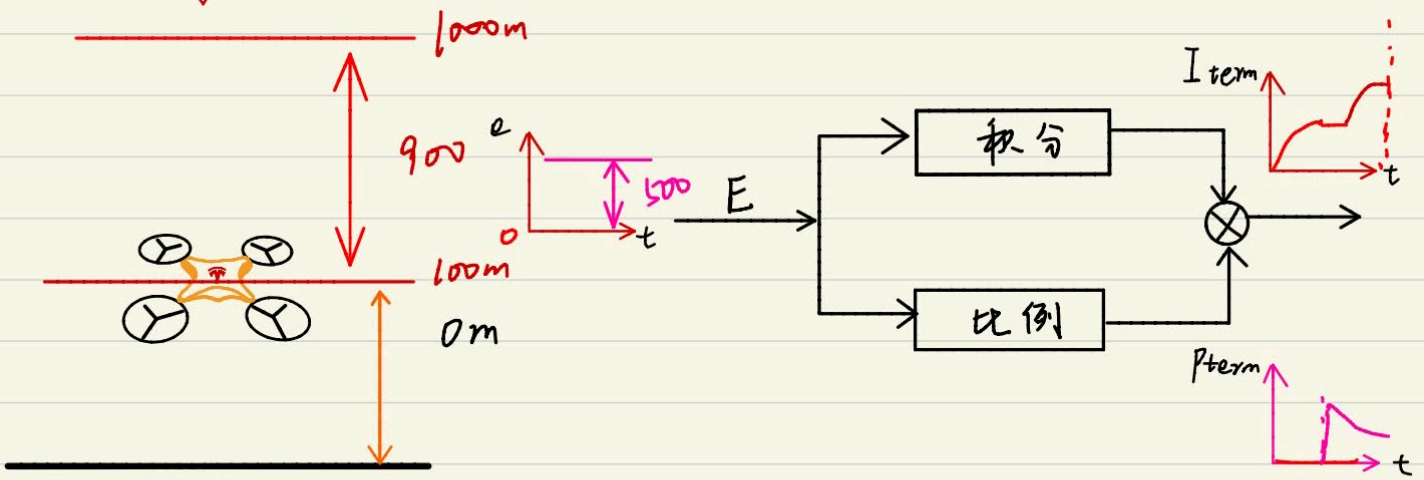
$K_d \cdot (e_i - e_{i-1})$

4. 其余相关控制知识

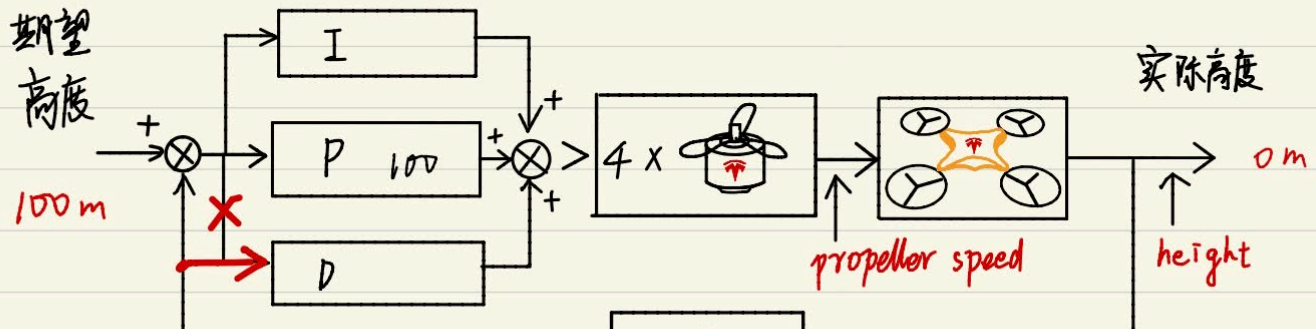
① 积分限幅

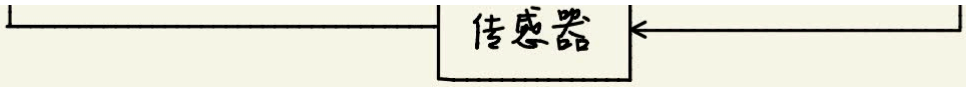


② 积分分离

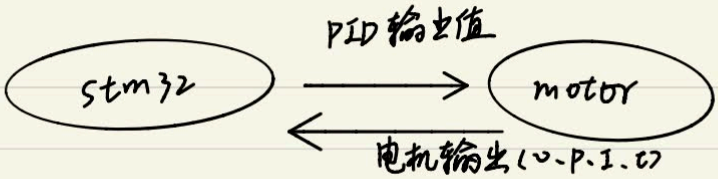


③ 微分先行




传感器

四、会用就行 ☆☆☆☆



0. 前期分析

主控发送值(控制器输出)

数据域	内容	电调 ID
DATA[0]	控制电流值高 8 位	1
DATA[1]	控制电流值低 8 位	
DATA[2]	控制电流值高 8 位	2
DATA[3]	控制电流值低 8 位	
DATA[4]	控制电流值高 8 位	3
DATA[5]	控制电流值低 8 位	
DATA[6]	控制电流值高 8 位	4
DATA[7]	控制电流值低 8 位	

主控接收值(实际电机输出)

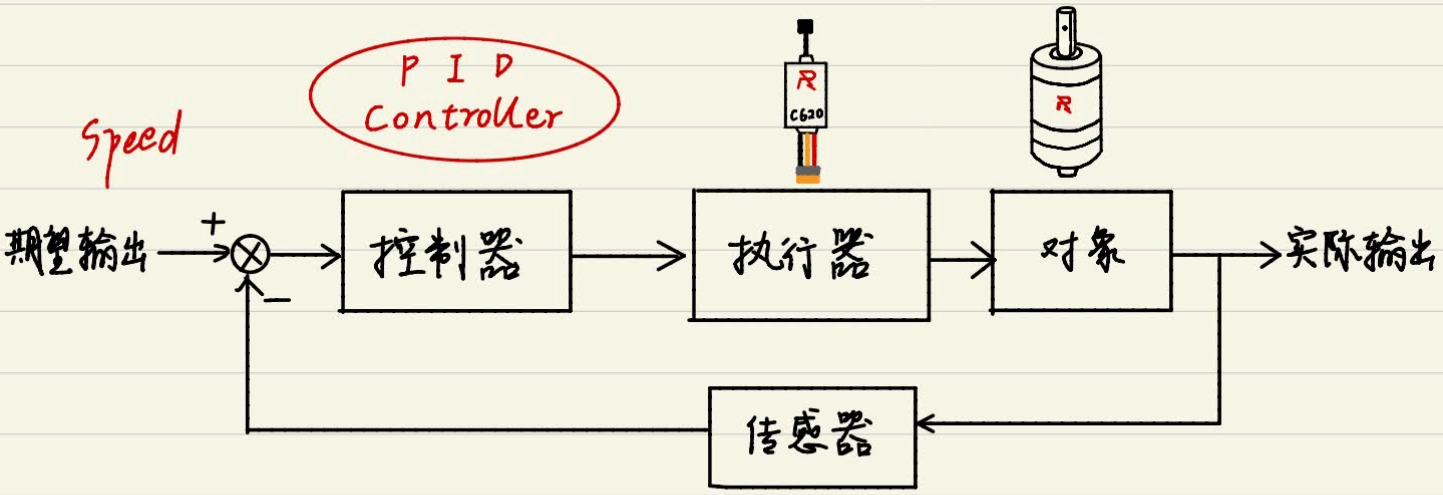
数据域	内容
DATA[0]	转子机械角度高 8 位
DATA[1]	转子机械角度低 8 位
DATA[2]	转子转速高 8 位
DATA[3]	转子转速低 8 位
DATA[4]	实际转矩电流高 8 位
DATA[5]	实际转矩电流低 8 位
DATA[6]	电机温度
DATA[7]	Null

上述值的发送与接收遵循CAN协议，

具体收发请于个人空间参考视频

1. RM 电机 M3508 速度单环

① 控制系统



② 代码讲解

chassis_motor_pid[i].SpeedPID.target = 500; (rpm/s) 要考虑减速比

```
chassis_motor_pid[i].SpeedPID.current = motor_chassis[i].speed_rpm;
pid_calc(&chassis_motor_pid[i].SpeedPID);
```

```
/*位置pid算法*/
void pid_calc(_pid* pid)
{
        pid->e = pid->target - pid->current;

        pid->p_out = (int32_t )(pid->Kp * pid->e);
        pid->i_out += (int32_t )(pid->Ki * pid->e);

        //积分限幅
        limit(&(pid->i_out),pid->IntegralLimit);

        //积分分离
        if (fabs(pid->e) < I_Band)
        {
            pid->i_out += (int32_t )(pid->Ki * pid->e);
        }
        else
        {
            pid->i_out = 0;
        }

        pid->d_out = (int32_t )(pid->Kd * (pid->e - pid->last_e));

        pid->total_out = pid->p_out + pid->i_out + pid->d_out;
        //pid输出限幅
        limit(&(pid->total_out), pid->MaxOutput);

        pid->last_e = pid->e;
}
```

积分限幅 ①

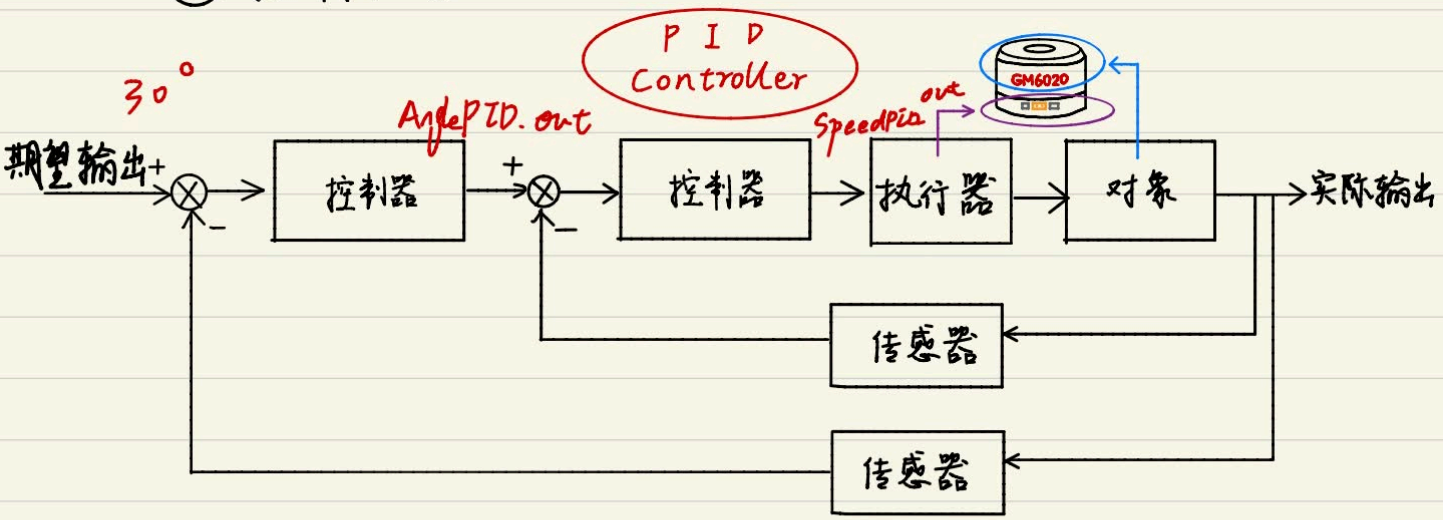
积分分离 ②

③ 积分分离和积分限幅

```
if (fabs(pid->e) < I_Band)
{   limit(&(pid->i_out),pid->IntegralLimit);
    pid->i_out += (int32_t )(pid->Ki * pid->e);
}
else
{
    pid->i_out = 0;
}
```

2. RM 电机 6020 角度双环

① 控制系统



② 代码讲解

manipulator_motor_pid[i].AnglePID.target = 30; (degree/s)

```
update_angle(&manipulator_motor_pid[i]._angle, motor_manipulator[i].angle);
manipulator_motor_pid[i].AnglePID.current = manipulator_motor_pid[i]._angle.angle;
pid_calc(&manipulator_motor_pid[i].AnglePID);

manipulator_motor_pid[i].SpeedPID.target = manipulator_motor_pid[i].AnglePID.total_out;
manipulator_motor_pid[i].SpeedPID.current = motor_manipulator[i].speed_rpm;
pid_calc(&manipulator_motor_pid[i].SpeedPID);
```

```
/*位置pid算法*/
void pid_calc(_pid* pid)
{
        pid->e = pid->target - pid->current;

        pid->p_out = (int32_t )(pid->Kp * pid->e);
        pid->i_out += (int32_t )(pid->Ki * pid->e);

        //积分限幅
        limit(&(pid->i_out),pid->IntegralLimit);

        //积分分离
        if (fabs(pid->e) < I_Band)
        {
            pid->i_out += (int32_t )(pid->Ki * pid->e);
        }
        else
        {
            pid->i_out = 0;
        }

        pid->d_out = (int32_t )(pid->Kd * (pid->e - pid->last_e));

        pid->total_out = pid->p_out + pid->i_out + pid->d_out;
        //pid输出限幅
        limit(&(pid->total_out), pid->MaxOutput);

        pid->last_e = pid->e;
```

积分限幅 ①

积分分离 ②

③ 积分分离和积分限幅

```
if (fabs(pid->e) < I_Band)
{   limit(&(pid->i_out),pid->IntegralLimit);
    pid->i_out += (int32_t )(pid->Ki * pid->e);
}
else
{
    pid->i_out = 0;
}
```

```
}
```

GM6020 角度更新 (degree/s)

```
void update_angle(motor_angle* _angle, uint16_t angle_fb)
{
    _angle->encoder = angle_fb;
    if(_angle->encoder_is_init)
    {
        if(_angle->encoder - _angle->last_encoder > 4096)
            _angle->round_cnt --;
        else if(_angle->encoder - _angle->last_encoder < -4096)
            _angle->round_cnt ++;
    }
    else
    {
        _angle->encoder_offset = _angle->encoder;
        _angle->encoder_is_init = 1;
    }
    _angle->angle_offset  = _angle->encoder_offset / 8192.0f * 360.0f;

    _angle->last_encoder = _angle->encoder;
    _angle->total_encoder = _angle->round_cnt * 8192 + _angle->encoder - _angle->encoder_offset;
    _angle->angle = _angle->total_encoder / 8192.0f * 360.0f;
}
```