无人机编程基础

21工作室出品

无人机稳定在空中需要解决哪些问题

- 首先，多轴飞行器有3个运动轴：立轴、横轴、纵轴，6个自由度。前后左右上下。
- 其次，除了会运动之外还要会静止，如何保持悬停，悬停在什么位置。这就需要传感器来探测环境，获得参数，根据反馈数据进行运算、然后下达调整指令。
- 无人机靠的是桨叶旋转提供升力，作用力与反作用力，有正旋转就会有反旋转，如何让无人机不会自转？
  - 保证不自转是相邻两个轴旋转方向相反来抵消反扭矩

传感器

- 保持自稳悬停需要飞控计算陀螺仪，角速度传感器，gps，加速度计等等传感器提供的数据。
- 之后控制电调(转换成电机的pwm量)，再控制电机转速，来调整飞行姿态。
  - 初中物理s(距离)=v (速度)×t(时间)
  - 高中物理W(功)=F(力)× s(距离/位移)=P(功率) × t(时间)
  - 我们这里需要考虑花多少能量搞定无人机飞到指定的位置的问题（首先要考虑你有足够的动能追得上变化）

举个例子

人工智能利用PID算法可以解决很多生活中的问题

- AI泊车，把车停到指定的位置上需要几步《大象放入冰箱问题》
- AI划船，在河流中，把船划到对岸目标点。《合力抵消问题》
- AI温控，控制火候，把水加热到40度整。《散热问题》
- 无人机的飞控

停车问题

- 起始点在六环，把车停到二环小区
  - 第一个小时，你可能90公里/小时的车速
  - 第二个小时，你可能70公里/小时的车速
  - 第三个小时，你可能5公里/小时的车速
- 如上，把大象装入冰箱，需要按步骤（打开冰箱门、装入、关上冰箱门），需要分解大问题变成小问题（切成块）。不同的历史时期要有不同的表现形式（设置车速）。
- 这里我们将历史时期定义为采样周期，并且设定为1小时。
- 像庖丁解牛一样，离关节（难点）越近你得越小心。现实中跑高速很惬意、找车位、停好车很费劲。
- 从发现问题到落实行动，需要解决的是探测器获取数据、运算结果执行到位、验收合格等问题。
- 如果你设定只有90公里/小时的车速，你每小时睁开眼看一下距离，你会发现，你要停到45公里的那个位置，始终在你前后摇摆。

采样周期

- 对于停车这种简单问题，我们采样周期设定1小时，也可以是10分钟，也可以是1分钟，甚至是一秒钟。
- 那么过于频繁的探测剩余距离，对于你的体力（中断开车操作，去进行探测操作）和你的脑力（运算油门大小）都会有很大的代价。
  - 每分钟都发现前一个小时在高速上，大脑通过观察运算发现可以不用减速。白白在心中算了很久。
- 过于稀疏的采样，又会耽误了及时调整的最佳时期。
- 我们一般考虑能耗、算力、传感器的承受能力、以及可容许可接受的姿态调整效率，来决定采样率。

章节

- 比例调节
- 积分调节
- 微分调节

一、比例调节

- 我们在每次探测数据之后，按固定的比例参数进行调整。称之为：比例调节。
- 每次测量一下当前位置到目标位置的距离差值，然后乘以比例参数作为转速的依据，作为指令的数据基础。
- 这将是一个收敛的过程。
- 这里采样周期设为1小时、比例系数设为2/3、车子有点儿毛病，实际转速总比显示的转速高1公里/小时
  - 第一次，距离目的地实测60公里，获得转速 (60+1)2/3=40.66，
    1小时之后距离目的地s=60-40.66=19.33公里。
  - 第二次，距离目的地实测19.33公里，获得转速(19.33+1)2/3=13.55
    1小时之后距离目的地s=19.33- 13.55= 5.77公里。
  - 50次之后：

19.333333333333336
5.777777777777779
1.2592592592592595
0.24691358024691334
0.5843621399176955
0.4718792866941015
0.5093735711019661
0.4968754762993447
0.5010415079002183
0.49965283069992716
0.5001157231000243
0.49996142563332524
0.500012858122225
0.49999571395925835
0.5000014286802472
0.49999952377325085
0.5000001587422497
0.49999994708591655
0.5000000176380278
0.4999999941206573
0.5000000019597808
0.49999999934673967
0.5000000002177534
0.4999999999274154
0.5000000000241949

0.5000000002177534
0.4999999999274154
0.5000000000241949
0.4999999999919349
0.5000000000026883
0.49999999999910383
0.5000000000002986
0.4999999999999003
0.5000000000000332
0.499999999999989
0.5000000000000036
0.49999999999999867
0.5000000000000004
0.4999999999999998
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5

惯性、阻力

- 实际上，电机开始转动，带动轮胎、带动螺旋桨在空气或者水中运动，都会有惯性问题。
  - 不是你想停，想停它就停。它总会在结尾多转几下，或者在开头慢上半拍
- 实际上，电机开始转动，带动轮胎、带动螺旋桨在空气或者水中运动，都会有阻力问题。
  - 万事开头难，第一步需要克服的阻力永远比运动起来维持时，需要克服的阻力大。并且你还无法准确计算开始多出来的那部分力道有多大。

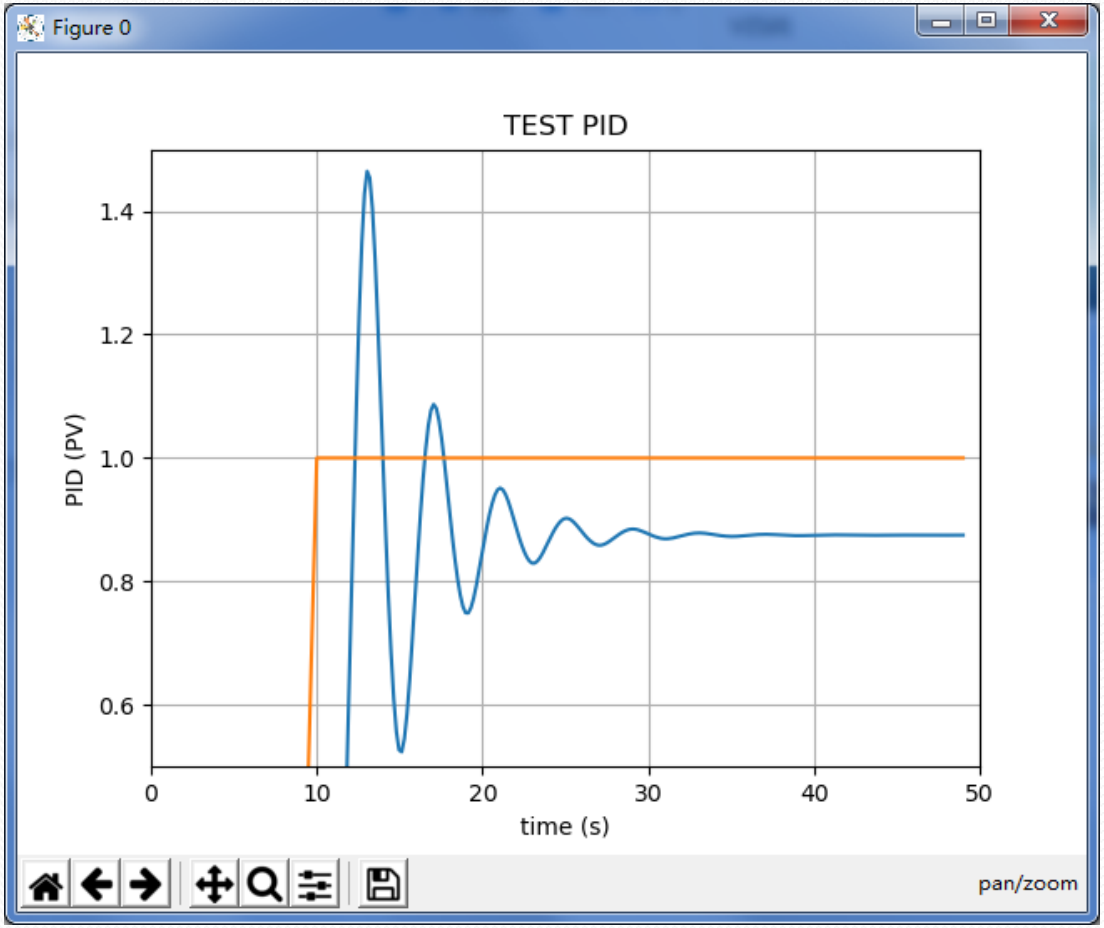
散热、微风

- 另外刚才说到温控，有一个现实问题，当室温25度时候，在把水加热到40度的过程中，除了需要把水加热到40度的能量之外，在加热的这一段时间内，水会散热到室内其他地方。所以你给的炉火要大于你的预想值
- 同理，现实中，划船到对岸的时候你可能发现，最终你把客人送到下游去了。
- 同理，微风中，你的无人机觉得自己已经按照规定把到达了既定的位置，却实际被微风带跑偏了。

稳态误差

- 以上的几个例子，都是系统中不可避免的稳态误差。
- 要消除这种误差，需要积分调节
- 纯比例调节不能消除的稳态误差，也叫静差。典型的是散热、微风。
- 系统部件中的缺陷（如摩擦、间隙、不灵敏区等）所造成的稳态误差不是静差。如惯性、阻力。但是也能用积分调节好。

图像化理解——稳态误差演示



二、积分调节

- 积分控制的作用是：只要系统存在误差，积分控制作用就不断地积累，输出控制量以消除误差。
- 因此只要有足够的时间，积分控制将能完全消除误差
- 但是积分作用太强会使系统超调加大，甚至使系统出现振荡

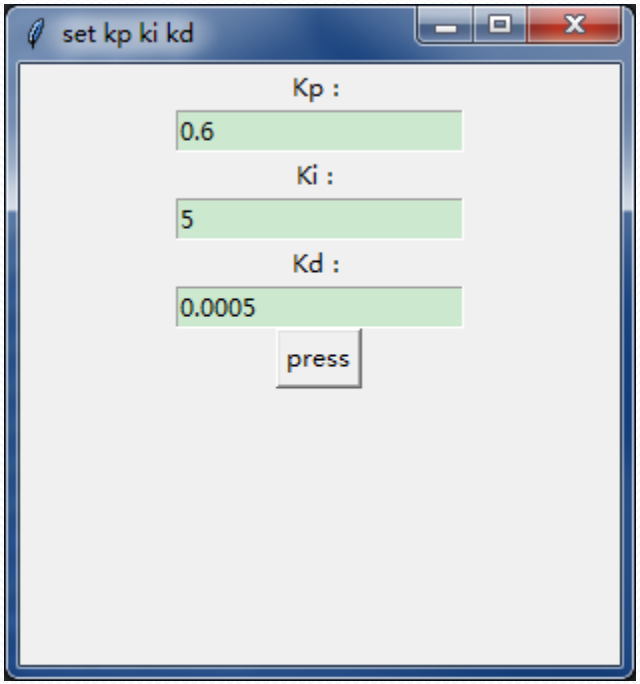
解决快速稳定问题，增加阻尼

- 在湖里开电动船的同学会有这种感觉，一把方向，发现船头歪了，然后反过来一把方向，船头回正之后，又歪了，你不断地改方向，船头不断的歪左歪右。俗称：画龙。发生了超调问题。

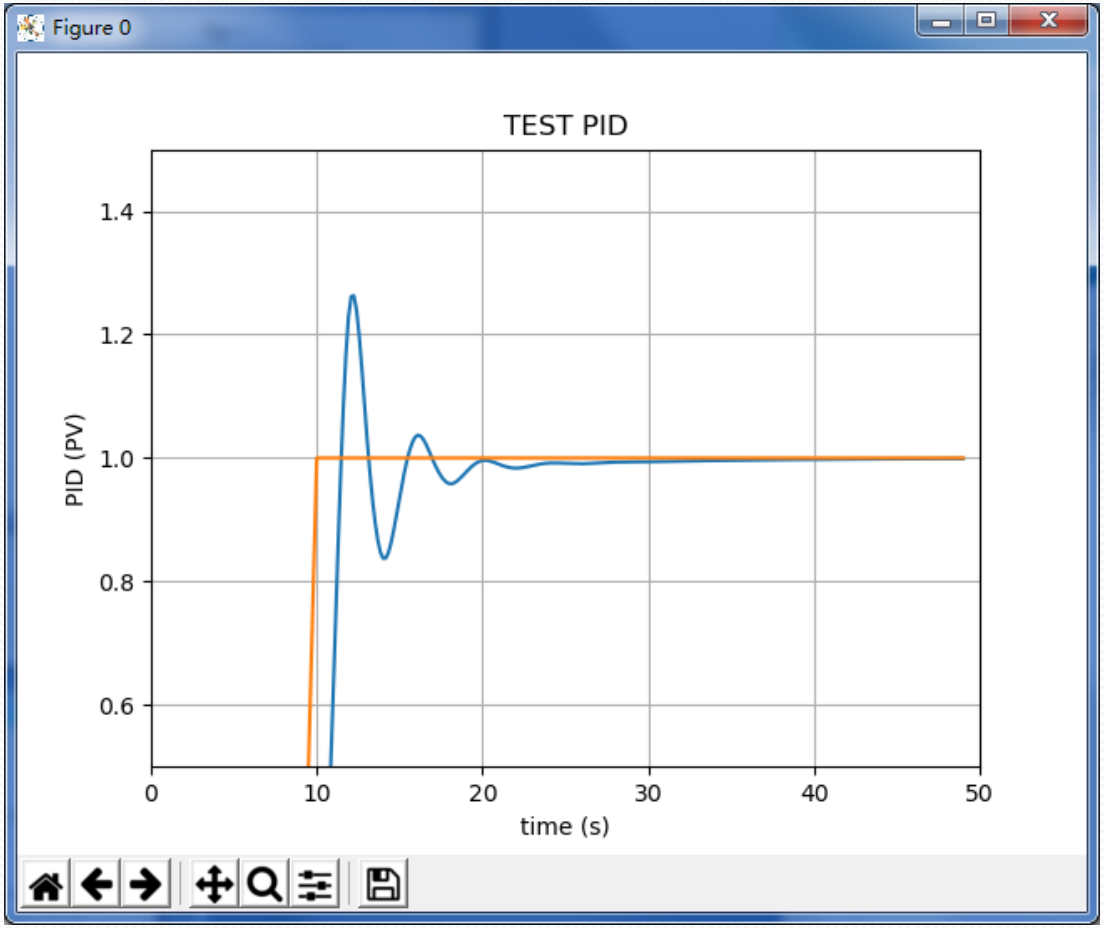
三、微分调节

- 微分控制：可以减小超调量，克服振荡，使系统的稳定性提高，同时加快系统的动态响应速度，减小调整时间，从而改善系统的动态性能。

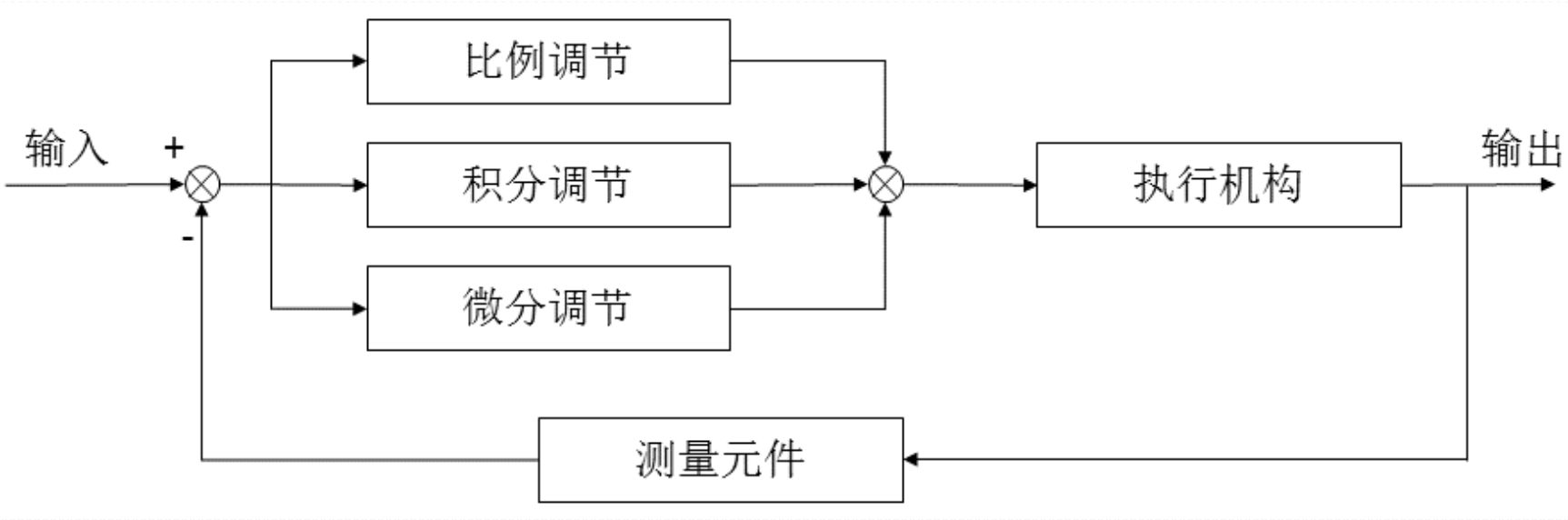
Kp=0.6 Ki=5 Kd=0.0005



图像化理解——PID算法



原理图



公式

1.1 PID算法公式

$$u(x) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)d(t) + T_d\frac{de(t)}{dt}\right)$$

1.2 PID离散化算法公式

1.2.1 位置式pid算法

$$u(k) = K_p\left(e(k) + \frac{T}{T_i}\sum_{n=0}^{k} e(n) + \frac{T_d}{T}\big(e(k) - e(k-1)\big)\right)$$

$$u(k) = K_p e(k) + \frac{K_p T}{T_i}\sum_{n=0}^{k} e(n) + \frac{K_p T_d}{T}\big(e(k) - e(k-1)\big)$$

$$u(k) = K_p e(k) + K_i\sum_{n=0}^{k} e(n) + K_d\big(e(k) - e(k-1)\big)$$

公式2

$$u(k) = K_p e(k) + K_i \sum_{n=0}^{k} e(n) + K_d\big(e(k) - e(k-1)\big) \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ①$$

1.2.2 增量式pid算法

$$u(k-1) = K_p e(k-1) + K_i \sum_{n=0}^{k-1} e(n) + K_d\big(e(k-1) - e(k-2)\big) \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ②$$

②-①可得

$$\Delta u(k) = u(k) - u(k-1) = K_p\big(e(k) - e(k-1)\big) + K_i e(k) + K_d\big(e(k) - 2e(k-1) + e(k-2)\big)$$

其中

$$\begin{cases} K_i = K_p \frac{T}{T_i} \\ K_d = K_p \frac{T_d}{T} \end{cases}$$

公式3

为了编程方便，根据误差测得的结果error(t)，将系数合并

$$\Delta u(k) = u(k) - u(k-1) = q_1(e(k) + q_2 e(k-1) + q_3 e(k-2)$$

其中

$$\begin{cases} q_1 = & K_p(1 + \frac{T}{T_i} + \frac{T_d}{T}) \\ q_2 = & -K_p(1 + \frac{2T_d}{T}) \\ q_3 = & K_p \frac{2T_d}{T} \end{cases}$$

增量式PI算法、没有微分项

代码实现

```
int Incremental_PI (int Encoder,int Target)
{
  static float Bias,Pwm,Last_bias;
  Bias=Encoder-Target;                                  //计算偏差
  Pwm+=Velocity_KP*(Bias-Last_bias)+Velocity_KI*Bias;   //增量式控制器PI
  Last_bias=Bias;                                       //保存上一次偏差
  return Pwm;                                           //增量输出
}
```

位置式PD算法，没有积分项

```
double BalanceCar::BalancePwm(SI_PID pid,float Angle, float Gyro)
{
    float Bias = 0;
    double balance;
    balance = pid.p * (Angle - Bias) + pid.d * Gyro;
    return balance;
}
```

```python
def update(self, feedback_value):
    error = self.SetPoint - feedback_value
    self.current_time = time.time()
    delta_time = self.current_time - self.last_time
    #print(delta_time) 大于采样周期，才会调整
    delta_error = error - self.last_error
    if (delta_time >= self.sample_time):
        self.PTerm = self.Kp * error#比例
        self.ITerm += error * delta_time#积分
        if (self.ITerm < -self.windup_guard):
            self.ITerm = -self.windup_guard
        elif (self.ITerm > self.windup_guard):
            self.ITerm = self.windup_guard
        self.DTerm = 0.0
        if delta_time > 0:
            self.DTerm = delta_error / delta_time#微分
        self.last_time = self.current_time
        self.last_error = error
        self.output = self.PTerm + (self.Ki * self.ITerm) + (self.Kd * self.DTerm)
```

$$u(k) = K_p e(k) + K_i \sum_{n=0}^{k} e(n) + K_d\big(e(k) - e(k-1)\big)$$

参考资料

- PID参数解析（一文读懂PID并会调试kp，ki，kd）（位置式+增量式PID）
- https://blog.csdn.net/yunddun/article/details/107720644

- 飞行控制PID算法——无人机飞控
- https://www.sohu.com/a/319975022_257861

- 一文读懂PID控制算法（抛弃公式，从原理上真正理解PID控制）
- https://blog.csdn.net/qq_25352981/article/details/81007075?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.compare&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.compare