可以，但不能把受版权保护的网页全文逐句翻译。下面按原页面结构，给你一份较完整的中文讲解版，公式、MATLAB操作和主要结论都保留。原文在密歇根大学 CTMS PID 教程。

1. PID 控制器简介

PID 是一种常见的反馈控制器，由三部分组成：

- P：比例 Proportional
- I：积分 Integral
- D：微分 Derivative

反馈控制系统的基本结构是：

```
目标值 r(t)
    ↓
计算误差 e(t) = r(t) - y(t)
    ↓
PID 控制器
    ↓
控制输入 u(t)
    ↓
被控对象 Plant
    ↓
实际输出 y(t)
    └──────────反馈──────────┘
```

PID 输出为：

$$u(t)=K_pe(t)+K_i\int_0^t e(\tau)d\tau+K_d\frac{de(t)}{dt}$$

其中：

- $e(t)$：目标值与实际值之间的误差
- K_p：比例增益
- K_i：积分增益
- K_d：微分增益

PID 的优势是结构简单、容易理解，并且无需特别精确的数学模型也能得到不错的控制效果。

2. P、I、D 分别做什么

比例项 P

$$P(t)=K_pe(t)$$

比例项根据当前误差直接输出控制量。

误差越大，控制器动作越强；误差越小，动作越弱。

增大 K_p 通常会：

- 缩短上升时间
- 加快系统响应
- 减小稳态误差

- 增大超调
- 增益过大时引起振荡或不稳定

例如机器人向前倾斜越多，轮子就需要越快或以更大力矩向前运动。

积分项 I

[I(t)=K_i\int_0^t e(\tau)d\tau]

积分项会累积过去的误差。

即使当前误差很小，只要它长时间存在，积分输出也会逐渐增大。它主要用于消除稳态误差。

增大 (K_i) 通常会：

- 消除稳态误差
- 加快对长期偏差的纠正
- 增大超调
- 延长稳定时间
- 过大时造成持续振荡和积分饱和

例如机器人因质心不完全居中，需要长期保持一个很小的电机力矩，积分项可以自动补偿这个偏差。

但双轮平衡机器人对积分项非常敏感，一般只使用很小的 (K_i)，甚至先设为零。

微分项 D

[D(t)=K_d\frac{de(t)}{dt}]

微分项关注误差变化速度。

如果机器人正在快速向前倾倒，即使当前角度误差还不大，微分项也能提前产生较大的纠正动作。

增大 (K_d) 通常会：

- 增加系统阻尼
- 减少超调
- 减少振荡
- 缩短稳定时间
- 对传感器噪声更加敏感

在平衡机器人中，D 项非常重要。实践中通常直接使用 IMU 陀螺仪测得的俯仰角速度，而不是对角度数值差分。

3. 各参数的一般影响

原教程给出的经验关系可以整理为：

参数增大	上升时间	超调	稳定时间	稳态误差	稳定性
(K_p)	通常减小	增大	小幅变化	减小	可能变差
(K_i)	减小	增大	增大	消除	变差
(K_d)	小幅变化	减小	减小	基本不变	通常改善

这些只是一般规律，不是所有系统都完全相同。三个参数之间会互相影响。

4. 教程使用的示例系统

教程使用了一个二阶被控对象：

[P(s)=\frac{1}{s^2+10s+20}]

MATLAB代码：

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
step(P);
```

这里：

- `tf('s')`：建立拉普拉斯变量 (s)
- `P`：定义系统传递函数
- `step(P)`：绘制单位阶跃响应

没有控制器时，该系统响应较慢，而且最终输出无法达到目标值，存在很大的稳态误差。

5. 只使用 P 控制

设控制器为：

[C(s)=K_p]

闭环系统：

[T(s)=\frac{K_pP(s)}{1+K_pP(s)}]

代入被控对象后：

[T(s)=\frac{K_p}{s^2+10s+20+K_p}]

MATLAB示例：

```
Kp = 300;

C = pid(Kp);
T = feedback(C * P, 1);

t = 0:0.01:2;
step(T, t);
```

这里：

- `pid(Kp)`：创建 P 控制器
- `feedback(C*P,1)`：建立单位负反馈闭环
- `step(T,t)`：观察闭环阶跃响应

结果大致是：

- 响应显著加快
- 稳态误差减小

- 超调增大
- 仍不能完全消除稳态误差

所以单独增加 P 虽然有效，但不能解决所有问题。

6. 使用 PD 控制

PD 控制器为：

[C(s)=K_p+K_ds]

MATLAB示例：

```
Kp = 300;
Kd = 10;

C = pid(Kp, 0, Kd);
T = feedback(C * P, 1);

step(T);
```

加入 D 项后通常能：

- 保留较快响应
- 减少超调
- 减少振荡
- 缩短稳定时间
- 提高阻尼

但 PD 仍不能完全消除稳态误差，因为当误差保持不变时，微分项为零。

对于双轮机器人，PD 往往已经能够完成基本平衡：

[\tau=K_p(\theta_{ref}-\theta)-K_d\dot{\theta}]

7. 使用 PI 控制

PI 控制器为：

[C(s)=K_p+\frac{K_i}{s}]

MATLAB示例：

```
Kp = 30;
Ki = 70;

C = pid(Kp, Ki);
T = feedback(C * P, 1);

step(T);
```

加入积分项后：

- 稳态误差可以消除
- 系统最终能达到目标值

- 超调可能明显增大
- 稳定时间可能变长
- 调参不合适时容易振荡

PI 很适合速度、电流等控制环，但不一定适合直接作为快速倒立摆姿态环。

8. 使用完整 PID

完整 PID：

[C(s)=K_p+\frac{K_i}{s}+K_ds]

示例参数：

```
Kp = 350;
Ki = 300;
Kd = 50;

C = pid(Kp, Ki, Kd);
T = feedback(C * P, 1);

step(T);
```

完整 PID 希望同时做到：

- P：提高响应速度
- I：消除稳态误差
- D：减少超调和振荡

但并不是说 PID 一定比 PI 或 PD 更好。很多实际系统只需要 PI 或 PD。

例如：

- 电机速度控制常用 PI
- 机器人姿态控制常用 PD
- 存在持续负载偏差时可能使用 PID
- 噪声很大时要谨慎使用 D

9. 自动整定

如果已知被控对象的数学模型，可以使用 MATLAB：

```
C = pidtune(P, 'PID');
```

然后查看闭环响应：

```
T = feedback(C * P, 1);
step(T);
```

也可以打开图形工具：

```
pidTuner(P, 'PID');
```

PID Tuner 可以在响应速度和鲁棒性之间调整，并观察：

- 上升时间
- 超调量
- 稳定时间
- 抗扰动能力
- 稳定裕度

但自动整定结果只是起点。真实机器人还会受到摩擦、延迟、采样周期、电机饱和和结构弹性的影响。