PID control

(proportional, integral, derivative)

Esta é uma matéria vasta e complexa, que envolve conceitos de matemática (cálculo de integrais), para além do domínio de todas as variáveis onde o sistema vai operar e seu controlo, e ainda quando se pretende desenvolver um programa informático para um microcontrolador, cujo objectivo é exactamente garantir o automatismo e precisão de funcionamento desse sistema.

É uma técnica que se utiliza para atingir um eficaz controlo de funcionamento de sistemas aplicados em máquinas, automóveis, robots, mísseis, etc..

O conceito de base é bastante antigo e existiu ao longo dos tempos como um sistema simplesmente mecânico, hidráulico ou electrónico. Mais recentemente o PID digital, com a utilização de microcontroladores e respectiva programação, atingiu níveis de velocidade de resposta, precisão e segurança extraordinários com um custo muito mais reduzido.

PID são as iniciais de Proporcional, Integral e Derivativo sendo cada uma destas três funções as responsáveis pelo cálculo cujos resultados em conjunto garantem o efectivo funcionamento de um sistema.

Basicamente o PID aceita um *Input* e, considerando o valor que se

pretende (*Setpoint*), trata essa informação e produz o *Output* com os valores ajustados para actuar no mecanismo que se pretende controlar. Consegue-se assim um output capaz de induzir um input igual ao setpoint. Isto pressupõe que o processo funcione em modo repetitivo (em ciclo).

Por exemplo, num silo de cereais em que a temperatura deve ser mantida constante e automaticamente. Ignorando, por simplicidade, a complexidade de respostas a tratar, factores externos ao sistema, tipo de cereal, camadas intermédias, etc., as partes imediatamente a estudar serão:

1-O silo (designa-se por "*plant*")

2-Sensor para medir a temperatura (feedback)

3-Fornecedor de calor/frio (por ex. um aquecedor, ventilador, etc.)

Assim, será necessário desenhar um sistema que receba os inputs dos sensores, trate essa informação e de seguida gere os outputs (parâmetros) para o aquecedor/ventilador fornecer a quantidade de calor/frio estritamente necessária para se atingir com precisão e sem flutuações o nível de temperatura desejado.

Este é, em termos muitos simples, o conceito que se aplica a um dispositivo de controlo de velocidade do automóvel, a um robot para se manter na vertical ou seguir um traçado, um helicóptero para se manter no ar, ou para um míssil manter a sua trajectória e atingir o alvo.

Suponhamos que, no modelo acima de exemplo, se pretende que a temperatura deve ser rigorosamente sempre igual a 23º .

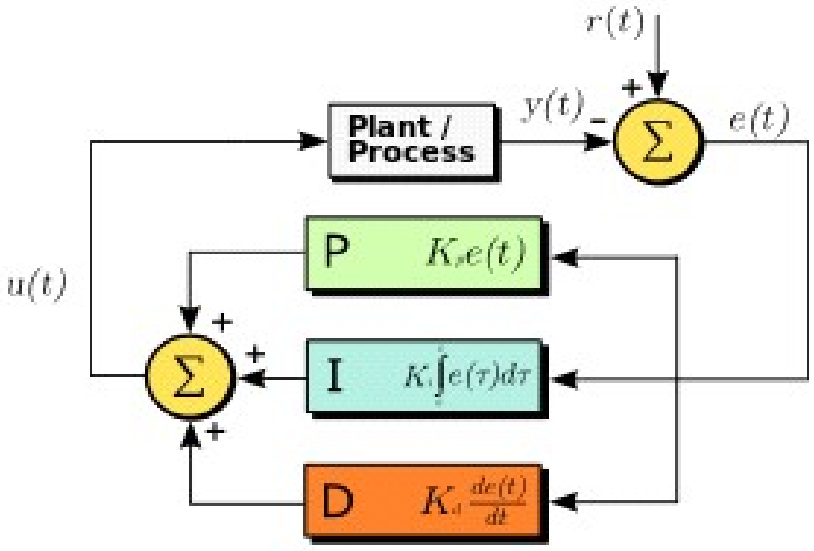
A este valor de 23º, que é o valor desejado, chama-se *Setpoint*.

A diferença entre o *Setpoint* e o valor medido (*Input*) é o *erro*, e este valor é muito importante no processo de cálculo do PID.

O princípio de funcionamento de um PID é exactamente fazer com que as variações dos erros se mantenham o mais possível na vizinhança do Setpoint e esse erro tenda (no verdadeiro sentido matemático) para zero.

Sendo s o valor que se pretende fixar (Setpoint), qualquer que seja o valor de e > 0, teriamos:

V_e (s) =] s - e, s + e [



A expressão matemática do PID é:

$$Output = U(t) = Kp\ e(t) + Ki \int_0^t e(t)\,dt + Kd \frac{d}{dt} e(t)$$

isto é, o valor de saída do controlador, em relação ao tempo, é igual à soma do produto dos erros por uma constante Kp, mais o produto do valor do integral dos erros por uma constante Ki, mais o produto do valor da derivada do erro pela constante Kd.

em que:

$P = K_p\ e\ (t)$

é o termo Proporcional e representa o valor da saída proporcional ao erro vindo da realimentação. Esta é a parte mais fácil de tratar e resume-se ao produto de uma constante pelo valor do erro:

// exemplo de programação

```
...
#define kp 2
double Setpoint, cha_roll;
...
void setup() {
        Setpoint = 0;
...
}
void loop() {
        leiturasensor();
        e  =  Setpoint - cha_roll;
        P =  e *  kp;
...
}
```

$$I = Ki \int_0^t e(t)\, dt$$

É o integral de e(t) no intervalo [0,t] multiplicado pela constante Ki.

Resulta daqui a soma dos sucessivos erros multiplicados pelo tempo decorrido entre a leitura do corrente intput do sensor e a leitura do input imediatamente anterior.

```
...
soma_erro  +=  erro * (dt);
if (soma_erro  >  i_maximo){
        soma_erro  =  i_maximo;
}
else if(soma_erro  <  i_minimo){
        soma_erro  =  i_minimo
{
I  =  soma_erro * ki;
...
```

$$D = Kd \frac{d}{dt} e(t)$$

é o termo derivativo e atrasa a taxa de variação de saída do controlador. Trata-se de calcular a derivada do erro tentando antecipar qual será o valor do próximo erro.

```
...
D  =  (erro  - erro_anterior) /  (dt) * Kd;
erro_anterior  =  erro;
...
```

sendo:

u(t) = a saída do controlador em relação ao tempo

Kp = a constante proporcional

Ki = a constante integral

Kd = a constante derivativa

e = ao erro

e = Setpoint - Input

dt = à diferença entre o tempo aquando da presente leitura e o da imediatamente anterior

Os valores de Kp, Ki e Kd são os definidos no programa, podendo ao longo do seu funcionamento efectuar inputs para os alterar.

Uma fase importante de um projecto que integre esta técnica é a realização do tuning do PID. Existem diversos métodos científicos sobre esta matéria (por exemplo o de Ziegler–Nichols), mas o tuning também se consegue realizar por tentativas, sem quaisquer necessidades de matemáticas complicadas e com os resultados aceitáveis. Usando esta via, a alteração dos valores sucessivos de Kp, Ki e Kd tem que ser efectuada por um processo qualquer que o permita, como por exemplo pela porta de comunicações serial, através de um interface que aceite os valores e os transfira para o microcontrolador via rádio, etc., etc..

Neste processo começa por se atribuir um valor baixo ao parâmetro Kp e ao Ki com o valor zero. Activar o sistema que tem o PID instalado, (por exemplo um quadcopter!) e verificar o desvio. Como P resulta proporcionalmente ao valor de Kp, variar o seu valor de 10% aumentará directamente numa substancial melhoria de desempenho, ou... agravará na mesma proporção a situação! Neste caso reduzir o valor para menos 5% e por este processo conseguir a melhor aproximação possível do valor do Setpoint, mas sem criar quaisquer oscilações.

De seguida abordamos o valor do Ki. I acumula e trata os valores da variável erro num determinado intervalo de tempo. Começa-se por atribuir o valor zero a Ki e um valor baixo a Kp. Depois aumentar em 10% o Kp e em função dos resultados, trabalhar com este método até se conseguir chegar à situação desejada.

Trabalhar com Kd implica um maior cuidado, pois as alterações aqui efectuadas tornam o sistema mais sensível. Por exemplo se o erro está a diminuir continuamente, resulta menor acção de controlo e se o erro sobe rapidamente contribuirá para uma elevada acção de controlo.

Resultado quando se aumenta o valor dos parâmetros:

Parâmetro	Tempo de subida	Overshoot	Tempo de estabelecimento	Erro
Kp	Diminui	Aumenta	Pequena variação	Diminiu
Ki	Diminui	Aumenta	Aumenta	Elimina
Kd	Pequena variação	Diminui	Diminui	Pequena variação

Este apontamente surge na sequência da necessidade de entender e implementar o algoritmo do PID no projecto em construção de um quadcopter (engenho de voar, drone, UAV-Unmanned aerial vehicle, etc.,) que vôa com base em quatro rotores controlados por um sistema computacional (microcontrolador Arduino Mega 2560) .

Um IMU (Inertial Measurement Unit) gera os ângulos (ângulos de Euler) dos eixos X, Y e Z, (φ, Θ, ψ) e estes serão os inputs para o PID, mapeados com valores mínimo e máximo, que gerará por sua vez valores para injectar nos controladores dos rotores, de modo a que o erro se aproxime constantemente do Setpoint, que neste caso concreto, é igual a zero, isto é os desvios dos eixos, horizontal, vertical e rotacional (roll, pitch e yaw), em vôo estabilizado, deve ser zero.