Cơ chế hoạt động của PID – cốt lõi lý thuyết điều khiển

Bộ não thu nhỏ: xoay quay khái niệm “Sai số”

Đo lường – triệt tiêu sai số!

- PID là gì?
    - Ứng dụng trong ngành công nghiệp tự động hoá, việc nắm PID là vô cùng quan trọng, đây là thông số mọi thiết bị và là nguyên tắc cơ bản của quá trình điều chỉnh thông số ứng với mỗi mô hình và thiết kế để tối ưu chuyển động.
    - P – tỷ lệ : giá trị sai số hiện tại
    - I = tích phân : giá trị sai số quá khứ
    - D = (vi phân): giá trị sai số tương lai
    -
- Cách thức triển khai của mô hình này?
    - Cơ chế: hiệu chỉnh sai số.
    - Outp tỉ lệ thuận với sai số hiện tại, khác biệt giữa điểm đặt và biến số quá trình đo được.
    - Ki: loại bỏ sai số trạng thái ổn định.
    - Tích luỹ sai số theo thời gain, loại bỏ hiệu quả sai số trạng thái ổn định.
    - Kd: xet tốc độ thay đổi của tín hiệu lỗi, dự đoán tương lai.
    -
- Ứng dụng vào đâu và như thế nào?
    - Bắt đầu với 1 giá trị trung bình và quan sát sự thay đổi,
    - Tránh nhất:
        - Overshoot, vượt ngưỡng và dao động quanh giá trị đó
        - Yêu cầu thu về tính ổn đinh: xác định ngưỡng giá trị Kp
- Bài toán của mình: PID giải quyết vấn đề cho xe như thế nào? Giá trị thoả đáng nằm trong phạm vi nào??
    - Đối với xe tự hành, giá trị Kp, Kd ứng dụng ở đâu, quan sát sai số như thế nào?
    - Giống như vẽ đường cho hươu chạy, việc quan sát quá trình di chuyển của chúng sẽ diễn ra như thế nào??
    -
- Kd, Kp có vai trò gì với xe??
- Hình dung đơn giản như chạy trên máy chạy bộ:
- Điều chỉnh tốc độ để tránh nhanh hơn và chậm hơn để tránh trôi khỏi máy chạy bộ.
- P – phản xạ, sai số càng lớn phản ứng càng mạnh….
- I – nuôi lòng hận thù với sai số. => Mục tiêu triệt tiêu hoàn toàn sai số
- Quá ám ảnh với quá khứ => hiện tượng overshoot vọt lố => tiếp tục nóng lên ….. => sai số âm rồi dao động xung quanh mục tiêu

- D – liều thuốc, đang đến gần mục tiêu rồi: tốc độ thay đổi của sai số, giảm chấn (bộ giảm xóc) => giúp ổn định, sự điềm tĩnh của hệ thống
- Nhạy cảm với nhiễu, dao động nhỏ, nhiễu, khuếch đại (giật cục) => robot cân bằng => nhiễu loạn không hề tồn tại ở cảm biến

⇨ Zignernickle : tránh đoán mò

⇨ Chỉ Kp thôi, điểm tới hạn (không tắt dần và không ổn đinh…

⇨ => Tu đó =>

1. Thiết lập: Kết nối bộ điều khiển PID trong một vòng phản hồi với cảm biến nhiệt độ và bộ phận gia nhiệt của lò phản ứng.
2. Khởi động tính năng tự điều chỉnh:
    - Đặt bộ điều khiển ở chế độ phản hồi rơle.
    - Xác định biên độ rơ nhỏ (ví dụ: ±5% của phạm vi điều khiển).
3. Quan sát dao động:
    - Hệ thống sẽ bắt đầu dao động khi rơle chuyển đổi giữa trạng thái cao và thấp.
    - Ghi lại chu kỳ dao động (Tu) và biên độ (a).
4. Tính toán các tham số PID:
    - Xác định độ lợi cuối cùng: Ku = (4 * biên độ rơ) / (π * biên độ dao động)
    - Sử dụng quy tắc Ziegler-Nichols để tính tham số PID: Kp = 0,6 * Ku Ti = 0,5 * Tu Td = 0,125 * Tu
5. Triển khai và kiểm chứng:
    - Cập nhật bộ điều khiển PID với các thông số đã tính toán.
    - Kiểm tra phản ứng của hệ thống đối với các thay đổi điểm đặt và nhiễu loạn. c. Tinh chỉnh thủ công nếu cần thiết để đạt hiệu suất tối ưu.

Xác định PID không phải đoán mò có cách nào??

Dựa trên video "EEVacademy #6 - PID Controllers Explained", phương pháp xác định các thông số cho bộ điều khiển PID được gọi là quá trình **"tuning" (tinh chỉnh)**. Video đề xuất sử dụng phương pháp **Ziegler-Nichols** để tìm điểm khởi đầu, sau đó kết hợp với quy trình tinh chỉnh lặp lại (iterative process) để đạt hiệu quả tối ưu.

Dưới đây là chi tiết các bước xác định PID theo nguồn tin:

1. Phương pháp Ziegler-Nichols (Ultimate Cycle Method)

Đây là phương pháp giúp bạn tìm ra các giá trị khởi điểm cho các hằng số PID thay vì phải đoán mò,. Quy trình thực hiện như sau:

- **Bước 1: Thiết lập hệ thống chỉ dùng bộ điều khiển P (Proportional).** Bạn bắt đầu với hệ thống (ví dụ: lò nung) chỉ sử dụng thành phần tỷ lệ, loại bỏ thành phần tích phân (I) và đạo hàm (D).

- **Bước 2: Tìm độ lợi tới hạn (K_u).** Tăng dần giá trị hằng số tỷ lệ (K_P) cho đến khi đầu ra của hệ thống tạo ra một dao động hình sin ổn định (biên độ không đổi, không tăng cũng không giảm). Giá trị K_P tại thời điểm này được gọi là K_u (Ultimate Gain).
- **Bước 3: Đo chu kỳ dao động (T_u).** Khi hệ thống đang dao động ổn định ở bước trên, hãy đo chu kỳ của sóng hình sin đó. Giá trị này gọi là T_u.
- **Bước 4: Tính toán các thông số khởi điểm.** Sử dụng K_u và T_u để tính toán các hệ số theo công thức Ziegler-Nichols:
    - Hệ số Tỷ lệ (K_P) = $0.6 \times K_u$
    - Hệ số thời gian Tích phân (T_I) = $T_u / 2$
    - Hệ số thời gian Đạo hàm (T_D) = $T_u / 8$

Từ các giá trị này, bạn có thể chuyển đổi sang các hằng số K_I và K_D (nếu bộ điều khiển của bạn dùng định dạng đó) bằng cách: $K_I = K_P / T_I$ và $K_D = K_P \times T_D$ (lưu ý: transcript có đoạn nói công thức K_D hơi khó nghe nhưng ngữ cảnh gợi ý việc chuyển đổi dựa trên quan hệ giữa các hằng số này).

2. Phương pháp Tinh chỉnh Lặp lại (Iterative Tuning)

Sau khi có các giá trị khởi điểm từ Ziegler-Nichols, hoặc nếu bạn tinh chỉnh thủ công từ đầu, video hướng dẫn cách quan sát phản ứng của hệ thống để điều chỉnh từng thành phần,:

- **Điều chỉnh thành phần Tích phân (I):**
    - **Mục đích:** Để loại bỏ lỗi trạng thái ổn định (steady-state error) - tức là khi hệ thống ổn định nhưng không đạt đúng giá trị mong muốn (ví dụ: lò nung chỉ đạt 9 độ thay vì 10 độ),.
    - **Lưu ý:** Nếu tăng I quá cao, hệ thống sẽ bị **vọt lố (overshoot)** và dao động do hiện tượng "tích phân bão hòa" (integral wind-up), là sự tích lũy lỗi quá khứ quá lớn,.
- **Điều chỉnh thành phần Tỷ lệ (P):**
    - **Mục đích:** Cải thiện tốc độ phản hồi của hệ thống.
    - **Lưu ý:** Nếu P quá lớn, hệ thống sẽ trở nên bất ổn định và dao động liên tục. Nếu P quá nhỏ, hệ thống phản ứng chậm.
- **Điều chỉnh thành phần Đạo hàm (D):**

- **Mục đích:** Giảm vọt lố (overshoot) và giảm thời gian ổn định (settling time). Thành phần D kháng lại sự thay đổi, đóng vai trò như bộ giảm chấn (damper) trong hệ thống treo xe hơi, giúp làm phẳng các dao động,,.
- **Cách làm:** Khi thấy hệ thống dao động mạnh hoặc vọt lố sau khi tăng P và I, hãy tăng dần D để kìm hãm lại,.
- **Lưu ý:** D rất nhạy cảm với nhiễu (noise). Nếu cảm biến có nhiễu, D sẽ khuếch đại nhiễu đó làm hệ thống rung lắc hoặc mất ổn định.

Tóm tắt quy trình thực tế: Bạn quan sát biểu đồ phản hồi của hệ thống (step response). Nếu thấy lỗi sai lệch (không về đích), hãy tăng I. Nếu thấy dao động quá nhiều (overshoot), hãy giảm P hoặc I, hoặc tăng D để kháng lại sự thay đổi đó,. Quá trình này được lặp đi lặp lại cho đến khi đạt được hình dạng phản hồi mong muốn.

Ứng dụng với Line-following AGV

Theo phương pháp Ziegler-Nichols (Ultimate Cycle Method) được mô tả trong video:

- Bạn đặt robot lên đường line (thẳng).
- Bạn tăng dần K_p.
- Khi K_p đủ lớn, robot sẽ bắt đầu lắc qua lắc lại quanh đường line liên tục mà không dừng. **Chính cái sự "lắc qua lắc lại" ổn định đó được gọi là "stable sinusoid" (dao động hình sin ổn định)**.

Vì vậy, bạn tạo ra dao động này bằng cách tăng K_p, chứ không phải bằng cách vẽ đường đi ngoằn ngoèo.

2. Quy trình tìm K_p, K_d trên đường thẳng

Bạn có thể áp dụng phương pháp quan sát "Step Response" (Phản hồi bước) được nhắc đến trong nguồn theo cách thực tế sau:

- **Bước 1: Tạo dao động (Tìm giới hạn của P)**
  - Cho robot chạy trên đường thẳng.
  - Tăng dần K_p. Ban đầu robot sẽ bám line yếu.
  - Tiếp tục tăng đến khi bạn thấy robot bắt đầu **rung lắc liên tục** (lắc trái phải đều đặn) quanh đường line nhưng không bị văng ra ngoài.

- Đây chính là trạng thái "dao động ổn định" mà video nhắc tới. Lúc này, hệ thống đang hoạt động giống như một cái lò xo không có ma sát,.

- **Bước 2: Dập tắt dao động (Thêm D)**
  - Giữ nguyên K_p ở mức gây ra rung lắc đó.
  - Tăng dần K_d.
  - Thành phần D hoạt động như một "bộ giảm chấn" (damper). Nó sẽ kháng lại sự thay đổi và hấp thụ năng lượng của các cú lắc.
  - Bạn sẽ thấy robot từ trạng thái đang rung bần bật trở nên di chuyển mượt mà và "phẳng" hơn, bám chặt vào line.
- **Bước 3: Kiểm tra bằng "Step Input" (Tác động bất ngờ)**
  - Để kiểm tra xem bộ PID đã tốt chưa, hãy dùng tay đẩy nhẹ robot lệch khỏi line một chút hoặc nhấc robot đặt lệch sang một bên rồi thả ra (đây gọi là disturbance/nhiễu).
  - Nếu robot lao về line và **lắc lư vài cái** mới thẳng lại -> Tăng thêm K_d hoặc giảm chút K_p (đang bị overshoot/dao động dư).
  - Nếu robot quay về line **quá chậm** -> Tăng K_p.