REGULATOR PID

Dokumentacja i opis

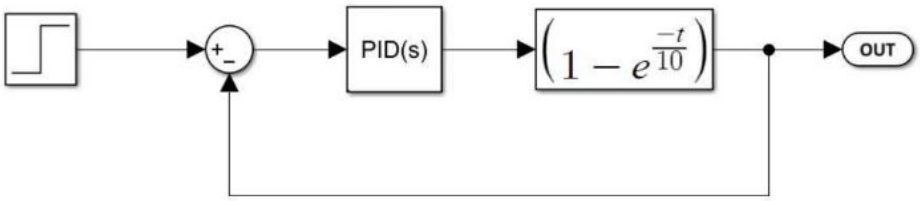
Wiktoria Sawicka
Informatyczne systemy automatyki
264179@student.pwr.edu.pl

Wydział informatyki i telekomunikacji,
Informatyczne systemy automatyki

29 października 2023

1 Istota zadania

Zadanie polegało na realizacji kontrolera PID ze sprzężeniem zwrotnym ujemnym w języku C. Kontroler sterować można sygnałami z przedziału od -5 do 5. Funkcja za argumenty przyjmować miała parametry: Kp, Ki, Kd, deltę czasu oraz wartość zadaną. Do wykonania zadania użyto bibliotek $< stdio.h >$ oraz $< sys/time.h >$.



Rysunek 1: Przedstawiony schemat systemu

2 Sposób wykonania

Aby lepiej zorganizować pracę, zadanie podzielono na dwa etapy:

- wyznaczenie nastaw regulatora (parametrów)
- zaimplementowanie kodu w języku C

2.1 Wyznaczenie nastaw regulatora

Aby obliczyć optymalne wartości parametów K_p, K_d oraz K_i, jako pierwszy krok obliczono transmitancję z podanego równania obiektu.

$$G(s) = \frac{\frac{1}{10}}{s(s + \frac{1}{10})}$$

Następnie korzystając z programu MATLAB wygenerowano charakterystyki modułową i fazową transmitancji $G(s)$, gdzie zapisano wartości modułu i fazy. Następnie stworzono trójkolumnową tablicę, której wiersze zawierały częstotliwość oraz wyżej wspomniane wartości modułu oraz fazy. Znaleziono następnie wartości, gdzie wartość fazy najbliżej odpowiadała $-180°$ - granicy stabilności.

$$\omega' = 15.9079\frac{rd}{s} \qquad M = 0.0063 \qquad F = -180.7853°$$

Następnie wyliczono wartości krytyczne: częstotliwość krytyczną ω_{kr} oraz okres krytyczny T_{kr}. Korzystając z tak wyznaczonych wartości, dalej wykorzystano regułę Zieglera-Nicholsa.

$$k_{p,kr} = \frac{1}{M} = \frac{1}{0.0063} = 158.73$$

Regulator	k_p	T_i	T_d
P	$0.5k_{p,kr}$	–	–
PI	$0.45k_{p,kr}$	$0.85T_{kr}$	–
PID	$0.6k_{p,kr}$	$0.5T_{kr}$	$0.125T_{kr}$

Rysunek 2: Reguła Zieglera-Nicholsa

$$T_{kr} = \frac{2\pi}{\omega'} = \frac{2\pi}{15.9079} = 0.395169(s)$$

$$k_p = 0.6k_{p,kr} = 0.6 \cdot 158.73 = 95.238$$

$$T_i = 0.5T_kr = 0.5 \cdot 0.395169 = 0.19755$$

$$T_d = 0.125T_{kr} = 0.125 \cdot 0.395169 = 0.0493875$$

Następnie wartości te przeliczone zostały na wartości nastaw.

$$K_p = k_p = 95.238$$

$$K_i = \frac{1}{T_i} = 5.062$$

$$K_d = T_d = 0.0493875$$

2.2 Zaimplementowanie kodu w C

Aby zachować czytelnośc dokumentu, podstanowiono nie wstawiać całego kodu, ale opisać dwie kluczowe funkcje w nim zawarte.

I Funkcja kluczowa - wyliczenie sygnału sterującego

```
double signal(double Kp, double Ki, double Kd, double setting, double time_delta)
{
    static double integral = 0;
    static double prevError = 0;
    error = (setting - current_signal);
    integral = integral + Ki * time_delta * (error + prevError);
    if (integral > up_lim_int)
    {
        integral = up_lim_int;
    }
    else if (integral < down_lim_int)
    {
        integral = down_lim_int;
```

```
    }
    double derivative = (error - prevError) / time_delta;
    double output = (Kp * error) + (Ki * integral) + (Kd * derivative);
    prevError = error;
    if (output > sig_max)
    {
        output = sig_max;
    } else if (output < sig_min)
    {
        output = sig_min;
    }
    return output;
}
```

Wyżej wskazana funckja odpowiedzialna jest za wyliczenie wartości sygnału sterujacego. Zgodnie z poleceniem, funkcja przyjmuje argumenty: Kp, Kp, Ki, $setting$, $time\_delta$. Funkcja ta wylicza wszystkie potrzebne parametry do wyliczenia poprawnego sygnału wyjściowego, to znaczy człon całkujący, człon proporcjonalny oraz człon różniczkujący. Dodatkowo funkcja ta ogranicza wartości wyliczonego członu całkującego żeby zapobiec przeregulowaniu, oraz organicza wartość sygnału sterującego, aby mieścił się on w przedziale od -5 do 5. Dodatkowo błąd poprzedni jest cały czas aktualniany w celu poprawnego wyliczenia członu całkującego oraz członu różniczkującego.

II funkcja kluczowa - symulacja

```
void simulate(double Kp, double Ki, double Kd, double setting, double time_delta)
{
    double reg_time = 1000;
    for (int i = 0; i <= reg_time; i++)
    {
        if (current_signal == setting) break;
        double output = signal(Kp, Ki, Kd, setting, time_delta);
        double input = output / 10.0;
        current_signal += input * time_delta;
    }
}
```

Powyższa funkcja symuluje pracę regulatora. Po podaniu symulacyjnego czasu regulacji reg_time symulacja programu odbywa się w pętli. Na początku sprawdzane jest, czy aktualny sygnał na wyjściu osiągnał ten zadany. Jest to sprawdzenie, potrzebne w razie gdyby wartość zadana została osiągnięta szybciej niż w ustalonym czasie regulacji. Następnie obliczany jest sygnał sterujący, który pomnożony przez deltę czasu dodawany jest do aktualnego sygnału.

2.3 Czas reakcji

Dodatkowo w kodzie zawarto również pomiar czasu, aby sprawdzić czas regulacji. Wykonano 10 pomiarów dla wartości zadanej $setting = 10$, a następnie wyniki uśredniono.

$$\Delta t_{reg} = 13\mu s$$

3 Podsumowanie

Regulacja przebiega pomyśle i w niedługim czasie. Regulator PID wydaje się pracować poprawnie. Link do github z kodem programu: `https://github.com/Wiktoria-S/PID_regulator.git`