Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Laboratorium Sterowania Robotów

Sterowanie PID robotem Lego Mindstorms NXT

Uwagi wstępne

1. Wszystkie przykłady i zadania wykonujemy w środowisku MATLAB z użyciem skrzynki narzędziowej `RWTHMindstormsNXT`.
2. Komunikację z robotem rozpoczynamy od podłączenia poprzez kabel USB. Następnie wykorzystujemy połączenie poprzez Bluetooth, zgodnie z zaleceniami prowadzącego.
3. Poniższy opis został przygotowany na podstawie informacji umieszczonych na stronach:
   (a) `http://www.mindstorms.rwth-aachen.de/`.
   (b) `http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html`

Budowa i oprogramowanie robota

1. Zbudować robota napędzanego dwoma silnikami. Dodatkowo wyposażyć robota w czujnik światła który będzie znajdował się w przedniej części robota i umieszczony na wysokości ok. 1 cm od powierzchni na której porusza się robot.
2. Podstawowym celem jest śledzenie grubej (ok.3 cm) linii czarnej namalowanej na białej powierzchni.
3. Używając czujnika światła, robot będzie się poruszał po krawędzi linii czarnej. W odróżnieniu od standardowego zadania śledzenia linii, gdzie właściwie śledzimy środek linii, tutaj śledzimy krawędź linii (zakładamy że będzie to lewa krawędź) aby w relatywnie prosty sposób móc dokonać detekcji wychylenia robota w lewą lub prawą stronę. Przykładowo, w przypadku sledzenia lewej krawędzi, wychylenie robota (czyli mamy błąd śledzenia linii) w prawą stronę będzie równoważne temu, że czujnik światła znajdzie się na polu czarnym. Analogicznie, wychylenie w lewą stronę będzie powodowało to, iż czujnik znajdzie się na polu białym.
4. Korzystając z poleceń do obsługi czujnika światła należy zapisać wartości zwracane przez ten czujnik odpowiednio dla 3-ech przypadków:
   - czujnik znajduje się nad powierzchnią o kolorze białym
   - czujnik znajduje się nad krawędzią czarnej linii
   - czujnik znajduje się nad czarną linią
5. W przypadku konieczności wykonania skrętu w prawą lub lewą stronę (czyli jesteśmy odpowiednio nad białą i czarną powierzchnią) wykorzystujemy fakt, iż robot skręca gdy koła obracają się z różną prędkością. Manipulując prędkościami silników (czyli ich mocą) dobrać odpowiednią prędkość skrętu. Przykładowo, przyjąć moc jedego z silników 50% a drugiego 30%.
6. Błąd śledzenia linii definiujemy jako
$$e = y - r \tag{1}$$
gdzie y to aktualna wartość odczytana z czujnika światła, r wartość odczytana z czujnika światła gdy znajduje się on dokładnie na krawędzi linii.
7. Narysuj wykres szybkości skrętu w odniesieniu do błedu śledzenia. Szybkość skrętu normalizujemy do zakresu $[-1, 1]$ gdzie wartość 1 oznacza najszybszy możliwy skręt w prawo a -1 najszybszy możliwy skręt w lewo. Oznacza to, że najmniejsza (ujemna) wartość błedu będzie odpowiadała szybkości 1 (czyli najszybszy skręt w prawo) a największa wartość błędu będzie odpowiadała szybkości -1 (czyli najszybszy skręt w lewo).
8. Sterowanie proporcjonalne (czyli typu P) to sterowanie proporcjonalne do wartości błędu czyli szybkość skrętu uzależnimy od wartości błędu. Oznacza to, iż będziemy sterowali mocą silnika i

naszym zadaniem będzie dobór zmiany mocy silników w zależności od wartości błędu śledzenia linii. Jest to równoważne z doborem współczynnika K_p w równaniu

$$\text{Szybkość\_skrętu} = K_p \times e$$

gdzie e oznacza błąd śledzenia (wzór (1)).

9. Zakładając, że w przypadku poruszania robotem w przód korzystamy z 50% mocy obu silników to domyślna wartość współczynnika mocy, oznaczanego jako T_p, będzie $T_p = 50$. Domyślną wartość współczynnika K_p możemy dobrać doświadczalnie ale możemy również przybliżyć następującym wzorem

$$K_p = \frac{0 - \text{avg}(T_p)}{\min(e) - 0} \tag{2}$$

gdzie $\min(e)$ minimalna wartość błędu, $\text{avg}(T_p)$ - domyślna wartość współczynnika mocy (przyjmujemy $T_p = 50$).

10. Prosty algorytm sterowania proporcjonalnego możemy zrealizować w następujący sposób (przyjmujemy, że wartość K_p wyznaczona jest ze wzoru (2) a wartość r jest taka sama jak we wzorze (1))

```
% wartości początkowe parametrów
Tp=50;            % domyślna moc
KaPe=Kp;      % domyślna wartość współczynika wzmocnienia proporcjonalnego
offset = r;       % czyli zakładamy, że na początku bład e=0

while(1)
        LightValue = odczytana aktualnie wartość z sensora światła
          e= LightValue - offset        % wyznaczamy błąd - wzór (1)
           Skręt= KaPe * e                    %  sterowanie proporcjonalne
           powerA = Tp + Skręt          % moc silnika A
           powerC = Tp - Skręt           % moc silnika C
end
```

Uwaga ! w powyższym algorytmie przyjęto, że lewy silnik jest podłączony do portu A a prawy silnik do portu C.

11. Sterowanie całkujące polega na wyznaczaniu sygnału sterującego na podstawie całki błędu. W przypadku jednak cyfrowych wartości błędu (a tak jest w naszym przypadku) całkę zastępujemy sumą, czyli

$$\text{Szybkość\_skrętu} = K_i \times \Sigma e$$

gdzie K_i jest wzmocnieniem części całkującej. Sumę błędów możemy policzyć w prosty sposób. Za każdym razem gdy obliczamy błąd śledzenia (e) to dodajemy go do zmiennej `integral`:

$$\text{integral} = \text{integral} + e$$

Teraz już możemy skorzystać ze sterowania całkująco-proporcjonalnego, czyli

$$\text{Szybkość\_skrętu} = K_p \times e + K_i \times \text{integral}$$

12. Sterowanie całkujące dąży do eliminacji błędu śledzenia trajektorii. W każdym wykonaniu pętli, jeśli występuje błąd, jest on sumowany a przez to występuje silniejsza korekta błędu. Jest to szczególnie przydatne dla małych wartości błędów.

13. W naszym przypadku program sterowania całkująco-proporcjonalnego będzie miał postać

```
% wartości początkowe parametrów
Tp=50;            % domyślna moc
KaPe=Kp;      % domyślna wartość współczynnika wzmocnienia proporcjonalnego, np. Kp=10
KaI=Ki             % domyślna wartość współczynnika wzmocnienia całkującego  np. Ki=1
offset = r;       % czyli zakładamy, że na początku bład e=0
integral = 0    % czyli całka jest na początku równa 0
while(1)
        LightValue = odczytana aktualnie wartość z sensora światła
          e= LightValue - offset        % wyznaczamy błąd - wzór (1)
           integral=integral+e             % wyznaczmy całkę
           Skręt= KaPe * e+KaI*integral    %  sterowanie proporcjonalno-całkujące
           powerA = Tp + Skręt          % moc silnika A
           powerC = Tp - Skręt           % moc silnika C
end
```

14. Ostatnia część naszego sterownika to część różniczkująca. Pozwala ona nam reagować na zmiany błędu śledzenia. Jeśli zmiany są duże (czyli przyrosty błędu są duże) to reakcja tej części będzie duża. Przyrost błędu wyznaczamy na podstawie kolejnych wartości błędu, czyli przyrost (zmienna `derivative`) wyznaczamy w następujący sposób

$$\text{derivative} = \text{e(aktualny)} - \text{e(w poprzedniej iteracji)}$$

15. Łącząc wszystkie 3 typy sterowania w jeden algorytm otrzymujemy sterownik PID. Implementacja tego algorytmu może wyglądać następująco:

```
% wartości początkowe parametrów
Tp=50;            % domyślna moc silników
KaPe=Kp;      % domyślna wartość współczynnika wzmocnienia proporcjonalnego, np. Kp=10
KaI=Ki             % domyślna wartość współczynnika wzmocnienia całkującego  np. Ki=1
KaDe=Kd       % domyślna wartość współczynnika wzmocnienia różniczkującego  np. Kd=100
offset = r;       % czyli zakładamy, że na początku bład e=0
integral = 0     % czyli całka jest na początku równa 0
derivative = 0 % czyli różniczka na początku równa jest 0
lasterror =0    % czyli ostatnia wartość błedu równa jest 0
while(1)
        LightValue = odczytana aktualnie wartość z sensora światła
          e= LightValue - offset        % wyznaczamy błąd - wzór (1)
           integral=integral+e             % wyznaczmy całkę
           derivative = error - lasterror % wyznaczamy różniczkę (czyli przyrost błędu)
           Skręt= KaPe * e+KaI*integral+KaDe*derivative    %  sterowanie PID
           powerA = Tp + Skręt          % moc silnika A
           powerC = Tp - Skręt           % moc silnika C
           lasterror=error
end
```

Zadania do wykonania

1. Zbudować robota według wskazówek umieszczonych w instrukcji oraz przekazanych przez prowadzącego. Robot powinien zawierać dwa silniki i 1 sensor światła.

2. Napisać program do precyzyjnego pomiaru zakresu wartości odczytywanych z sensora światła przy odchylaniu robotem w lewo i prawo od krawędzi linii czarnej.

3. Napisać program umożliwiający poruszanie się robota po krawędzi linii czarnej. W tym celu należy zakres wartości odczytywanych przez sensor światła podzielić na dwa podzakresy (małe i duże wartości co odpowiada znajdowaniu się czujnika nad polem czarnym lub białym). Gdy odczytana wartość jest z zakresu małych wartości to robot musi skręcać w lewo (ponieważ znajduje się nad linią a my śledzimy lewą krawędź). Analogicznie, robot skręca w prawo gdy odczytane wartości są duże (jesteśmy nad białą powierzchnią). Dobrać prędkość skrętu w odniesieniu do kształtu linii - np. duże prędkości skrętu dla linii zawierającej wiele ostrych zakrętów.

4. Napisać program umożliwiający poruszanie się robota po krawędzi linii czarnej. W odróżnieniu do powyższego zadania, bazując na naszym już doświadczeniu, dokonujemy podziału na 3 podzakresy odpowiadające odpowiednio skrętowi w lewo, poruszaniu się w przód i skrętowi w prawo.
   Czy uzyskane rozwiązanie jest lepsze od poprzedniego?

5. Zaimplementować algorytm sterowania proporcjonalnego robotem mobilnym. Zbadać wpływ zmian wartości parametru K_p (współczynnika wzmocnienia proporcjonalnego) i T_p (domyślnej mocy silnika) na jakość sterowania (np. szybkość eliminacji błędu śledzenia trajektorii).

6. Zaimplementować algorytm sterowania całkująco-proporcjonalnego (PI). Wzmocnienia K_p i K_i dobrać eksperymentalnie.

7. Zaimplementować algorytm sterowania PID. Wzmocnienia K_p, K_i i K_d dobrać eksperymentalnie.

8. Dobrać parametry sterownika PID, tj. K_p, K_i i K_d przy pomocy metody Zigler-Nichols'a. W tym celu należy dokonać pomiaru okresu drgań własnych ω_c, wzmocnienia krytycznego K_c oraz czasu wykonania jednego przebiegu pętli pomiarowej dt. Aby dokonać pomiaru, wybieramy tylko sterowanie proporcjonalne i zwiększamy wzmocnienie aż do momentu gdy będą występowały oscylacje niegasnące w ruchu robota, czyli wychylenie w jedną i drugą stronę będą takie same i nie będą zanikać. Wzmocnienie przy którym uzyskaliśmy takie oscylacje to właśnie jest K_c a ω_c to okres trwania tych oscylacji. Pomiar nie musi być szczególnie dokładny, wystarczy przybliżenie. Według

literatury, czas trwania oscylacji powinien wynosić od 0.5 do 2 sekund a K_c około 3. Dodatkowo musimy zmierzyć czas potrzebny na wykonanie pętli sterującej czyli czas pomiedzy dwoma kolejnymi pomiarami odczytanymi z czujnika światła. Najlepiej zmierzyć czas 1000 powtórzeń tej pętli a następnie podzielić go przez 1000 w celu uzyskania dokładnego wyniku. Parametry K_p, K_i i K_d dobieramy w następujący sposób:

Sterowanie	K_p	K_i	K_d
P	$0.5 \times K_c$	0	0
PI	$0.45 \times K_c$	$\frac{1.2 \times K_p \times dT}{\omega_c}$	0
PID	$0.6 \times K_c$	$\frac{2 \times K_p \times dT}{\omega_c}$	$\frac{K_p \times \omega_c}{8 \times dt}$

9. Zbadać wpływ zwiększenia wartości parametrów K_p, K_i i K_d na następujące wskaźniki jakości sterowania:

   (a) *przeregulowanie* - mierzone jako maksymalne wychylenie robota w prawą lub lewą stronę w reakcji na błąd regulacji (bład śledzenia linii).

   (b) *czas regulacji* - jest to czas mierzony od momentu powstania błędu (czyli robot przestaje dokładnie śledzić linie, np. gdy nagle będzie musiał wykonać zwrot) do momentu kiedy robot ponownie śledzi zadaną trajektorię bez widocznych błędów.

   (c) *błąd w stanie ustalonym* - jest to stała wartość błędu śledzenia linii, która pozostaje po upływie co najmniej 2× czasu regulacji.