Model kinematyki samochodu demonstracyjnego EVE i jego identyfikacja – raport

Adrian Jałoszewski, Tomasz Kotowski,
Dorota Kowalik, Michał Krent

9 maja 2016

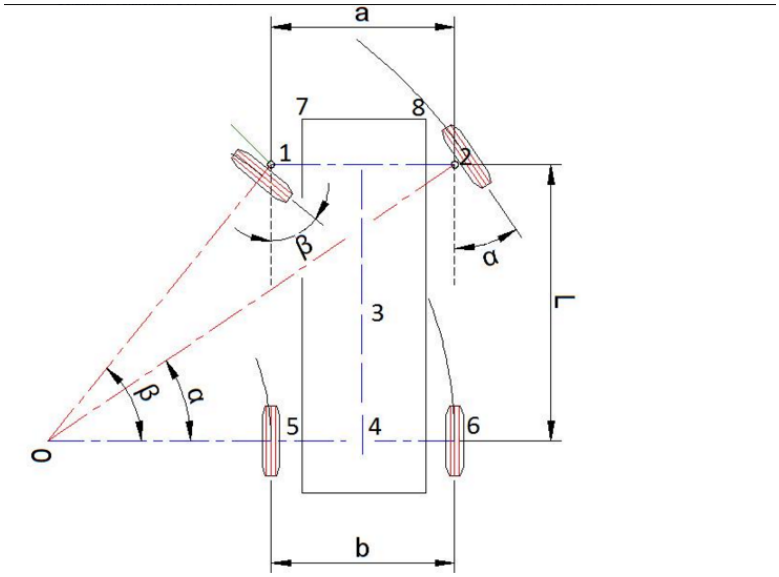
Spis treści

Rozdział 1

Model matematyczny

1.1 Sterowanie Ackermana

Do naszych zastosowań wybraliśmy sterowanie Ackermana. Jest to model specjalnie stworzony pod sterowanie pojazdami, które posiadają koła skrętne. Aby jednak móc sterować tym pojazdem musieliśmy tak dobrać punkty okręgu po którym nastąpiłoby przemieszczenie, aby jego tylna oś była prostopadła do okręgu po którym się przemieszcza.



Rysunek 1.1: Sterowanie Ackermana

Punktami gwarantującymi, że tak będzie jest dowolny punkt samochodu nie leżący na tylnej osi oraz jego odbicie względem tej osi. Trzecim punktem potrzebnym do wyznaczenia okręgu po którym będzie się poruszał pojazd jest kolejny punkt, w którym ma się znaleźć uprzednio wybrany punkt samochodu. Ponieważ naszym celem jest symulacja ruchu środka ciężkości pojazdu, to tym punktem samochodu będzie jego środek ciężkości.

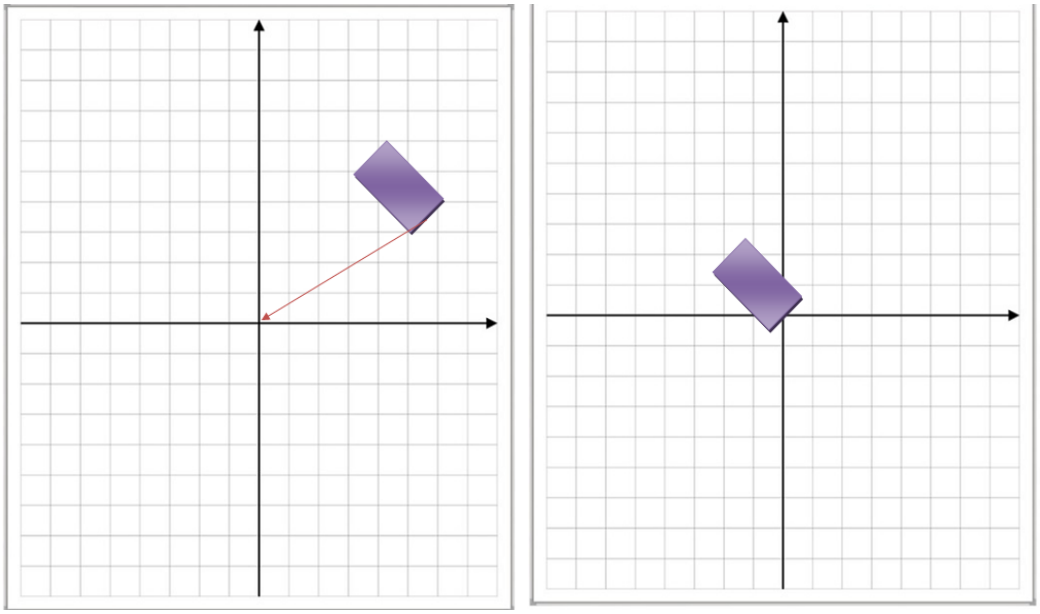
Pomiary samochodu potrzebne do modelu Ackermanna:

- odległość między osią tylną, a przednią: 1,8 m
- rozstaw osi: 143,5 cm
- szerokość: 55 cm
- skrętność kół 15^o–18^o (problemy z precyzyjnym zmierzeniem)

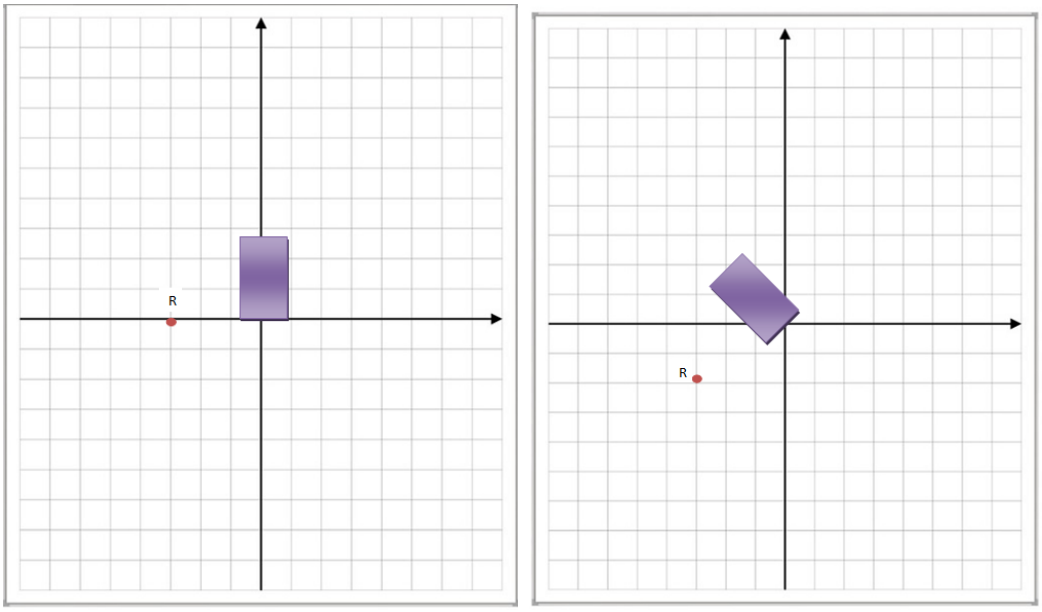
1.2 Kinematyka prosta

Celem kinematyki prostej jest wyznaczenie poruszania się pojazdu w zależności od zastosowanego na nim sterowania. W naszym przypadku jako sterowanie przyjmujemy wartość kąta na lewym kole, gdyż nie ma potrzeby generowania tych wartości dla dwóch kół – łączy je sztywna zależność. Dla uproszczenia założyliśmy, że samochód porusza się ze stałą prędkością – używamy stałego kroku o długości 10 cm. Samochód zaczyna swoją trasę z punktem nr 4 w punkcie (0.0), skierowany zgodnie z wektorem (0,1).

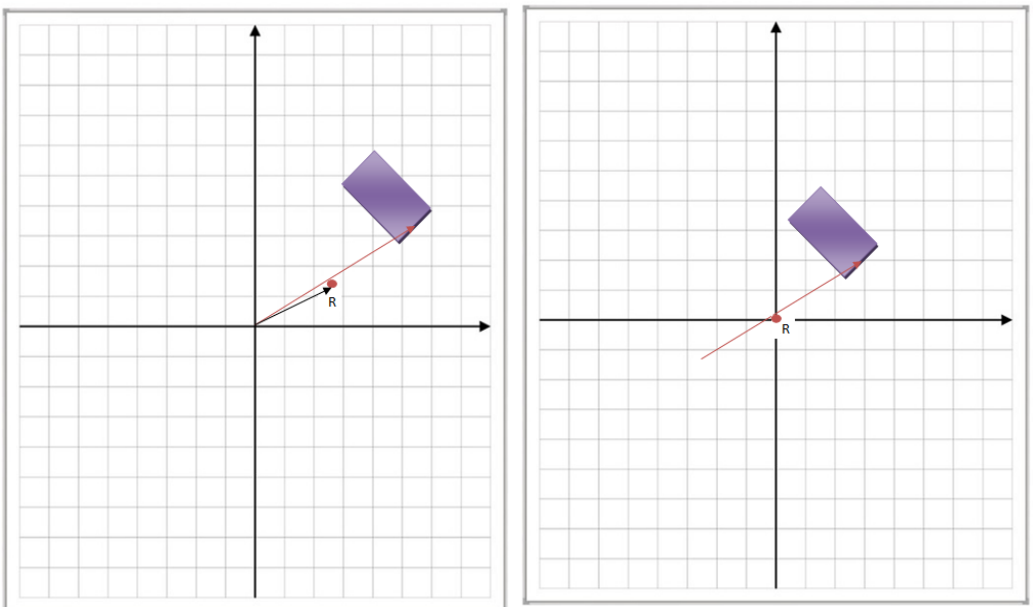
Wyznaczanie kinematyki prostej rozpoczynamy od przeniesienia punktu nr 4 pojazdu do środka układu współrzędnych.



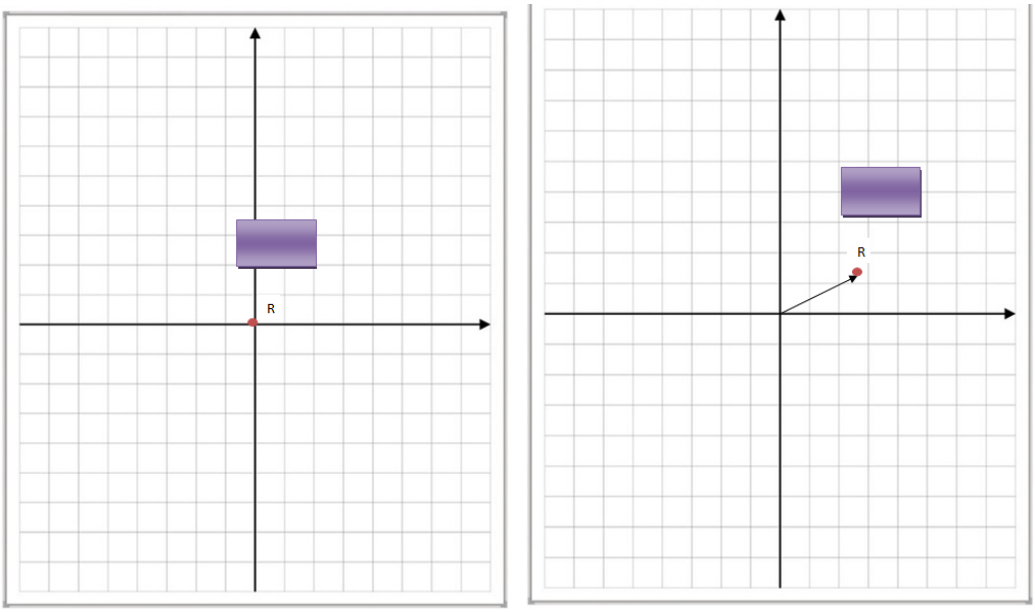
Następnie dokonujemy obrotu w celu wyprostowania samochodu i w układzie związanym z samochodem wyznaczamy środek obrotu z zależności trygonometrycznych, a następnie dokonujemy obrotu odwrotnego do tego, co go uprzednio wykonaliśmy.



Należy następnie cofnąć translację, wracając do układu globalnego związanego z podłożem, a następnie dokonać translacji do układu, którego środkiem jest uprzednio wyznaczony środek obrotu.



Ostatnimi dwoma krokami jest obrót samochodu względem środka obrotu o 10 centymetrów i powrót do układu globalnego.



Ze względu na dokonywane translacje, wszystkie zastosowane rotacje względem punktu są przekształceniami liniowymi wektorów, a dodanie kolejnego wymiaru wypełnionego jedynkami do wektorów zamienia również translacje w przekształcenia liniowe. Pozwala to na zastosowanie macierzy obrotu i macierzy translacji do wszystkich manipulacji.

Oddzielnie należy potraktować przypadek kiedy koła są wyprostowane – wtedy należy przesunąć samochód o zadany wektor o długości 10 centymetrów zgodnie z kierunkiem ruchu.

Całokształt operacji można zapisać jako:

$$X_{n+1} = T_{Rn}^{-1} R_{Rn} T_{Rn} T_{1n}^{-1} R_{1n}^{-1} \{\text{obliczenia}\} R_{1n} T_{1n} X_n$$

Gdzie T_{Rn} to macierz translacji przesuwająca środek obrotu do środka układu współrzędnych, macierz R_{Rn} to macierz obrotu o dany kąt (tak aby przesunąć o 10 cm), macierz T_{1n} to macierz translacji sprowadzająca środek tylnej osi do środka układu współrzędnych, a R_{1n} to macierz dokonująca rotacji prostującej samochód. Obliczenia mają na celu wyznaczenie środka obrotu i dokonania na nim pozostałych przekształceń.

1.3 Kinematyka odwrotna

Zadaniem kinematyki odwrotnej jest wyznaczenie sterowania na podstawie zadanej ścieżki po jakiej się porusza pojazd. Podobnie jak w kinematyce prostej przyjmujemy tu jako źródło sterowania lewe koło.

Całość należy rozpocząć od wyznaczenia środka obrotu lub podania kąta 0^o w wypadku jeżeli ten środek nie istnieje (jedziemy linią prostą). Do wyznaczenia tego używamy następującego układu równań:

$$\begin{cases}(x_1-a)^2+(y_1-b)^2=r^2\\(x_2-a)^2+(y_2-b)^2=r^2\\(x_3-a)^2+(y_3-b)^2=r^2\end{cases}$$

Gdzie (a, b) to środek okręgu po którym ma się poruszać środek ciężkości, a r to jego promień. Można to inaczej zapisać jako:

$$\begin{bmatrix}2x_1-2x_3 & 2y_1-2y_3\\2x_2-2x_3 & 2y_2-2y_3\end{bmatrix}\cdot\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}x_1^2-x_3^2+y_1^2-y_3^2\\x_2^2-x_3^2+y_2^2-y_3^2\end{bmatrix}$$

Odpowiednie punkty to:

(x_1, y_1) – środek masy

(x_2, y_2) – odbicie symetryczne środka masy względem tylnej osi

(x_3, y_3) – kolejny punkt trasy

Jeżeli wyznacznik macierzy jest równy zero, to znaczy, że poruszamy się po prostej, w przeciwnym wypadku wyznaczamy przy pomocy iloczynu wektorowego i skalarnego potrzebny nam kąt i dokonujemy obrotu na samochodzie do następnego punktu.

$$X_{n+1}=T_{Rn}^{-1}R_{Rn}T_{Rn}X_n$$

T_{Rn} to macierz przesuwająca punkt obrotu do środka układu współrzędnych, a macierz R_{Rn} to macierz dokonująca obrotu o zadany kąt. W obliczeniach iloczyn skalarny służy nam do wyznaczenia wartości kąta (arcus cosinus ma odpowiedni zakres), a iloczyn wektorowy służy do określania kierunku obrotu.

1.4 Reprezentacja pojazdu autonomicznego EVE

Pojazd autonomiczny EVE jest reprezentowany macierzą:

$$\begin{bmatrix}x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8\\y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8\\1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

Gdzie (x_i, y_i) to odpowiednie punkty oznaczone zgodnie ze schematem w podpunkcie o sterowaniu Ackermana.

Rozdział 2

Implementacja

2.1 Kinematyka prosta

```
function [wektor_odpowiedzi] = eve_forward(A)
    ilosc_katow=length(A);
    wektor_odpowiedzi=[];
    car=create_eve();
    for i=1:ilosc_katow
        if(A(i)==0)
            wektor=[car(1,3)-car(1,4); car(2,3)-car(2,4)];
            dlugosc=vect_len(wektor);
            wektor = wektor/dlugosc * 0.1;
            car=translate(wektor(1,1),wektor(2,1))*car;
            wektor_odpowiedzi=[wektor_odpowiedzi,car(1:2,3)];
        else
            go_back_matrix = translate(-car(1,4),-car(2,4));
            car=translate(-car(1,4),-car(2,4))*car;
            wektor_pierwszy=[car(1,3)-car(1,4); car(2,3)-car(2,4)];
            kat=get_angle(wektor_pierwszy, [0;1]);
            car=rotate(kat)*car;
            r=-1.435*cot(A(i))-0.55;
            R=[r ;0 ;1];
            car=rotate(kat)\car;
            R=rotate(kat)\R;
            car = go_back_matrix \ car;
            R = go_back_matrix \ R;
            R2=R;
            R=translate(-R(1,1),-R(2,1))\R;
            car=translate(-R2(1,1),-R2(2,1))*car;
            rot = get_angle(car(:,4), car(:,3));
            car=rotate(sign(rot)*0.1/abs(r))*car;
            car=translate(-R2(1,1),-R2(2,1))\car;
            wektor_odpowiedzi=[wektor_odpowiedzi,car(1:2,3)];
        end
    end
end
```

Rysunek 2.1: Kinematyka prosta

2.2 Kinematyka odwrotna

```
function alpha = eve_reverse(road, restrictions)
    stalk = [];
    car = create_eve();
    tr = road(:,1);
    rot = get_angle([0; 1; 1], road(:,2) - road(:,1));
    car = translate(0,-1.435/2) * car;
    car = rotate(rot) * car;
    car = translate(tr(1), tr(2)) * car;
    alpha = [];
    for i = 1:length(road)
      » p1 = car(:,3);
        p2 = car(:,3) + 2 * (car(:,4) - car(:,3));
        p3 = road(:,i);
        centre = circ_centr(p1, p2, p3);
        if isnan(centre)
            alpha = [alpha, 0];
            vect = p3 - p1;
            car = translate(vect(1), vect(2)) * car;
        else
            car = translate(-centre(1),-centre(2)) * car;
            p3 = translate(-centre(1), -centre(2)) * p3;
            angle = get_angle(car(:,4), car(:,1));
            if abs(angle) > 0.30
                msgbox('angle too big');
                break;
            end
            alpha = [alpha, angle];
            angle = get_angle(car(:,3), p3);
            car = rotate(angle) * car;
            car = translate(-centre(1),-centre(2)) \ car;
        end
        stalk = [stalk;p1(1), p1(2)];
        in_pol = inpolygon(car(1,:),car(2,:),restrictions(1,:), restrictions(2,:));
        in_pol = prod(in_pol);
        if in_pol == 0
            msgbox('out of the restrictions');
            break;
        end
    end
    plot(stalk(:,1), stalk(:,2))
end
```

Rysunek 2.2: Kinematyka odwrotna

W zadaniu kinematyki odwrotnej mieliśmy dodatkowo sprawdzić, czy pojazd mieści się w zadanych ograniczeniach oraz czy manewr jest wykonywalny. W przypadku kiedy manewr jest awykonalny lub nie da się go wykonać w polu wyznaczonym do manewru (wielokąt), to wyskakuje okno dialogowe i przerywa działanie skryptu zwracając do tej pory wyznaczone wartości.

W przypadku jakby się chciało podobny efekt uzyskać dla kinematyki prostej należy wynik z obliczeń kinematyki prostej przeprowadzić przez funkcję kinematyki odwrotnej.

Funkcja rozpoczyna swoje działanie od utworzenia i umieszczeniu pojazdu na trasie.

2.3 Funkcje pomocnicze

2.3.1 Macierz rotacji i translacji

```
function rot = rotate(phi)
    cp = cos(phi);
    sp = sin(phi);
    rot = [
        [cp, -sp, 0];
        [sp, cp, 0];
        [0, 0, 1]
    ];
end

function tr = translate(x, y)
    tr = [
        [1, 0, x];
        [0, 1, y];
        [0, 0, 1]
    ];
end
```

2.3.2 Tworzenie pojazdu autonomicznego macierzy EVE

```
function ev = create_eve()
    ev = [-0.55; 1.435];
    ev = [ev, [0.55; 1.435]];
    ev = [ev, [0; 1.435 / 2]];
    ev = [ev, [0; 0]];
    ev = [ev, [-0.55; 0]];
    ev = [ev, [0.55; 0]];
    ev = [ev, [-0.55; 1.8]];
    ev = [ev, [0.55; 1.8]];
    ev = [ev; ones(1,8)];
end
```

2.3.3 Iloczyn skalarny, wektorowy, długość wektora, kąt między wektorami

```
function prod = dot_product(a, b)
    prod = a(1) * b(1) + a(2) * b(2);
end

function prod = cross_product(a, b)
    prod = a(1) * b(2) - a(2) * b(1);
end

function len = vect_len(a)
    len = dot_product(a,a);
    len = sqrt(len);
end

function angle = get_angle(v, w)
    abs_v = vect_len(v);
    abs_w = vect_len(w);
    cos_angl = dot_product(v, w) / (abs_v * abs_w);
    angle = acos(cos_angl);
    if cross_product(v, w) <= 0
        angle = -angle;
    end
end
```

2.3.4 Wyznaczenie środka okręgu

```
function centre = circ_centr(a, b, c)
    x = [a(1), b(1), c(1)];
    y = [a(2), b(2), c(2)];
    A = 2*[[x(1)-x(3),y(1)-y(3)];
        [x(2)-x(3), y(2)-y(3)]];
    b = [x(1)^2-x(3)^2+y(1)^2-y(3)^2;
        x(2)^2-x(3)^2+y(2)^2-y(3)^2];
    if abs(det(A)) < 10e-4
        centre = [NaN; NaN];
    else
        centre = A \ b;
    end
end
```

Rozdział 3

Testy

Testy są podzielone na dwie kategorie, ze względu na to, jakie funkcje miały za zadanie sprawdzić. Jednak każda z kategorii testów zawiera w sobie od razu przykłady testów z drugiej kategorii, gdyż wyjście z jednej funkcji mogło służyć jako wejście do innej. Ponieważ zadanie kinematyki prostej jest odwrotne do zadania kinematyki odwrotnej mogliśmy w ten sposób dokonać walidacji wyników, gdyż przepuszczając najpierw serię kątów przez funkcję odpowiedzialną za kinematykę prostą mogliśmy wynik z niej wykorzystać jako wejście do kinematyki odwrotnej.

Ponieważ jest bardzo mało prawdopodobne, że obydwie funkcje zostały wykonane wadliwie w ten sposób aby wady z jednej funkcji znosiły się z wadami z drugiej funkcji uznaliśmy to za wiarygodny sposób walidacji poprawności naszych funkcji. Jednak ze względu na to, że krok stały jest szczególnym przypadkiem kroku zmiennego działa to tylko w jedną stronę – z kinematyki prostej, gdzie mieliśmy krok stały, wynoszący 10 cm. w kinematykę odwrotną.

Mechanizm testów został przez nas w pełni zautomatyzowany tak aby z łatwością móc włączać i wyłączać testy przez wykomentowanie odpowiednich linii. Wyłączanie testów jest koniecznością ze względu na ich czasochłonność – każdy test dokonywał zapisu na dysku trzech wykresów. Jest to przez nas stworzony mechanizm testów jednostkowych, pozwalający nam na tworzenie naszych funkcji na zasadach TDD. Ponieważ kod odpowiedzialny za sprawdzanie czy pojazd znajduje się w konturze jest funkcją wbudowaną MATLABa, postanowiliśmy go nie testować.

```
function test()
    close all
    test_1()
    test_2()
    test_3()
    test_4()
    test_5()
    test_6()
    test_7()
    test_8()
    test_9()

    rev_test_1()
    rev_test_2()
    rev_test_3()
    rev_test_4()
    rev_test_5()
    rev_test_6()
    rev_test_7()
    rev_test_8()
end
```

3.1 Kinematyka Prosta

Wyniki testów to trzy wykresy. Pierwszy pokazuje ścieżkę wyznaczoną przy pomocy kinematyki prostej, drugi przedstawia ścieżkę zakreśloną przez środek masy w trakcie wykonywania obliczeń dla kinematyki odwrotnej, a trzeci przedstawia kąty w stopniach, jakie zostały zwrócone przez funkcję kinematyki odwrotnej w zależności od iteracji. Pierwsze dwa przebiegi są wyskalowane w metrach.

Poniższy fragment kodu przedstawia schemat jakim się kierowaliśmy przy tworzeniu naszych testów jednostkowych dla kinematyki prostej. Dokonywał on inicjalizacji danych wejściowych w postaci wektora o długości n, a następnie przekazywał je do funkcji odpowiedzialnej za kinematykę prostą. Stamtąd były pobierane dane w postaci macierzy $3 \times n$ i przekazywane do funkcji odpowiedzialnej za kinematykę odwrotną.

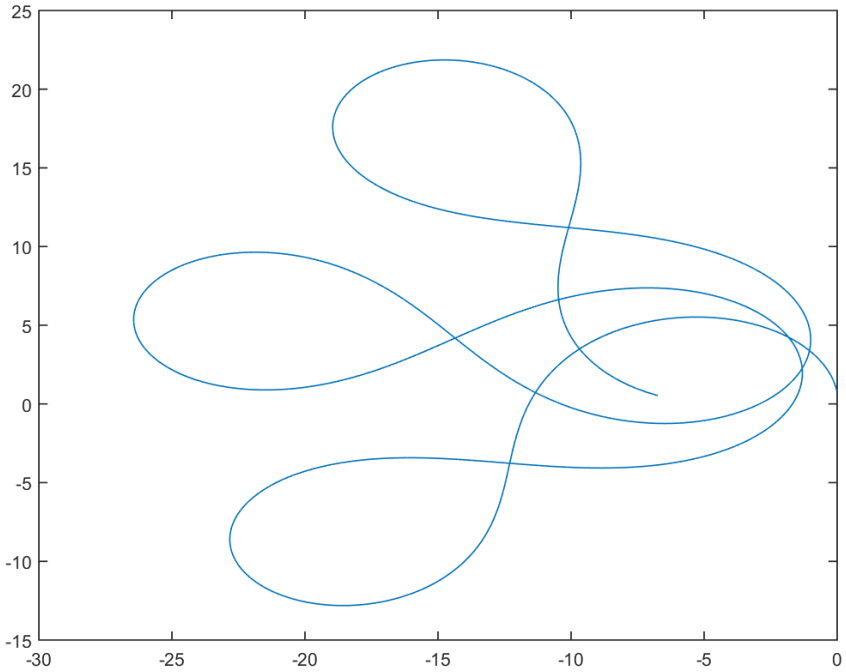
```
function test_1()
    figure(1)
    t = eve_forward(sin(1:0.01:20)*0.3);
    plot(t(1,:), t(2,:))
    print('img/sin_in_1.png','-dpng')
    figure(2)
    t = [t;ones(1,length(t))];
    restr = [10e5, 10e5, -10e5, -10e5;
        10e5, -10e5, -10e5, 10e5];
    t = eve_reverse(t,restr);
    print('img/sin_in_2.png','-dpng')
    figure(3)
    plot(1:length(t), t*180/pi)
    print('img/sin_in_3.png','-dpng')
end
```

Taki pojedynczy test dawał nam w odpowiedzi trzy wykresy, które były następnie zapisywane na dysku i były dostępne do ręcznego porównania.

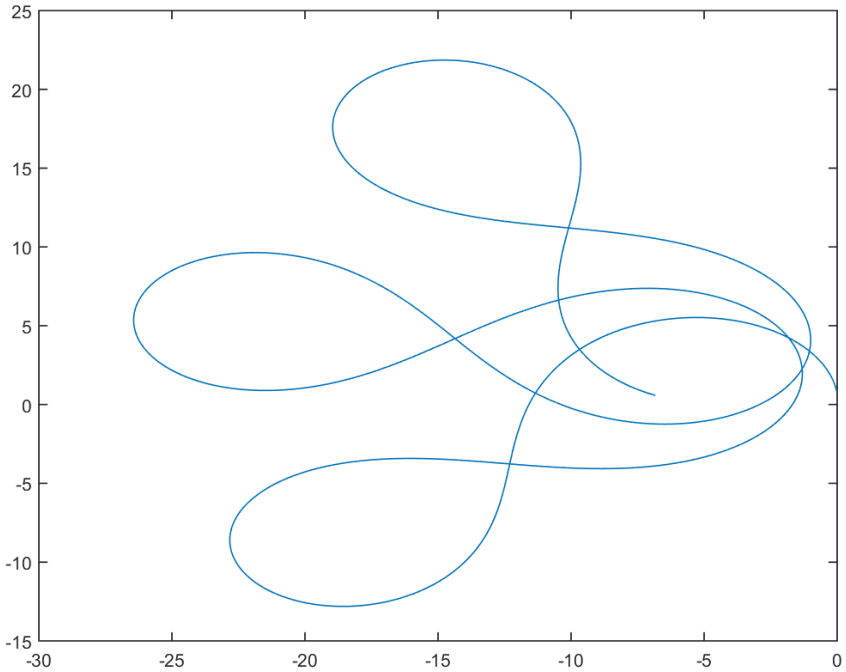
Z początku przebiegów kąta w funkcji iteracji można dosyć często zauważyć różne wartości, które mogą się zdawać niezgodne z przewidywaniami teoretycznymi. Nie jest to jednak ani błąd symulacji, ani błąd modelu – efekt ten bierze się stąd, że zgadzać się ma położenie środka ciężkości, a kierunek pojazdu nie musi być styczny do trasy po której pojazd miałby się poruszać. Jednak po pewnym czasie pojazd się stabilizuje i zaczyna poprawnie śledzić ścieżkę tak, aby kierunek jego pokrywał się ze styczną do toru.

3.1.1 Sinusoida

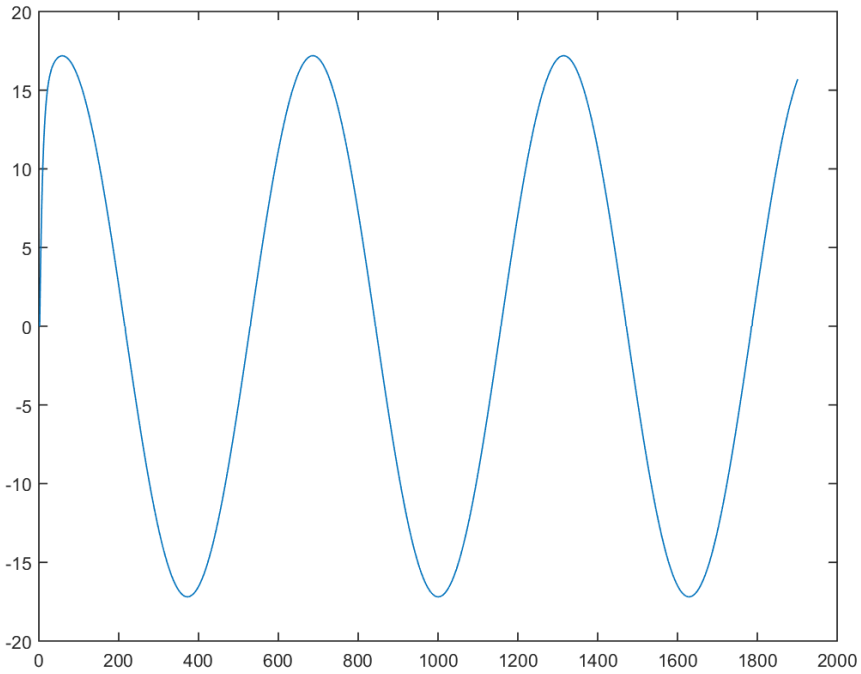
Obydwie ścieżki są ze sobą zgodne. Po kształcie krzywych można rozpoznać, że zachowują się jakby na wejściu miały zadaną pewną sinusoidę.



Rysunek 3.1: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.2: Ścieżka którą podążał pojazd

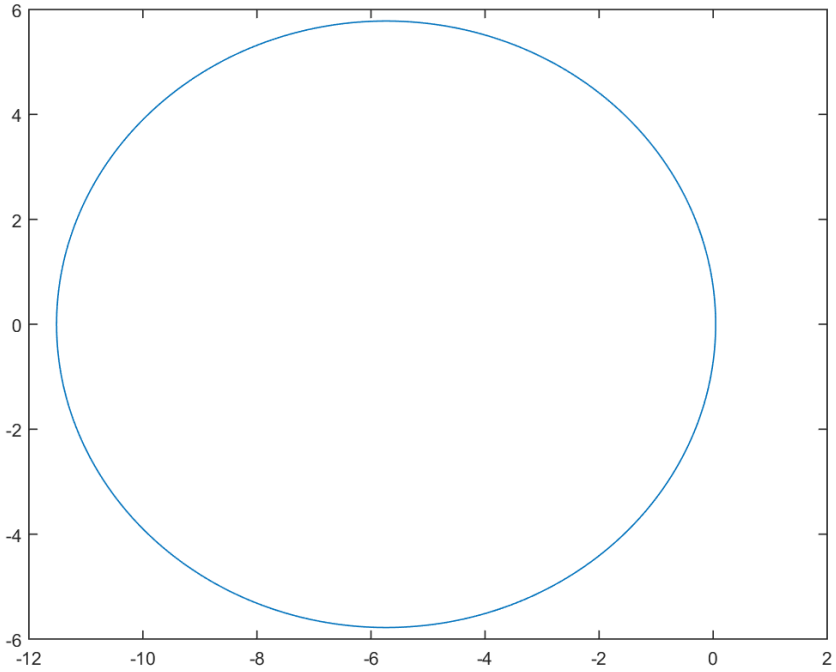


Rysunek 3.3: Kąt wyznaczony przez kinematykę odwrotną

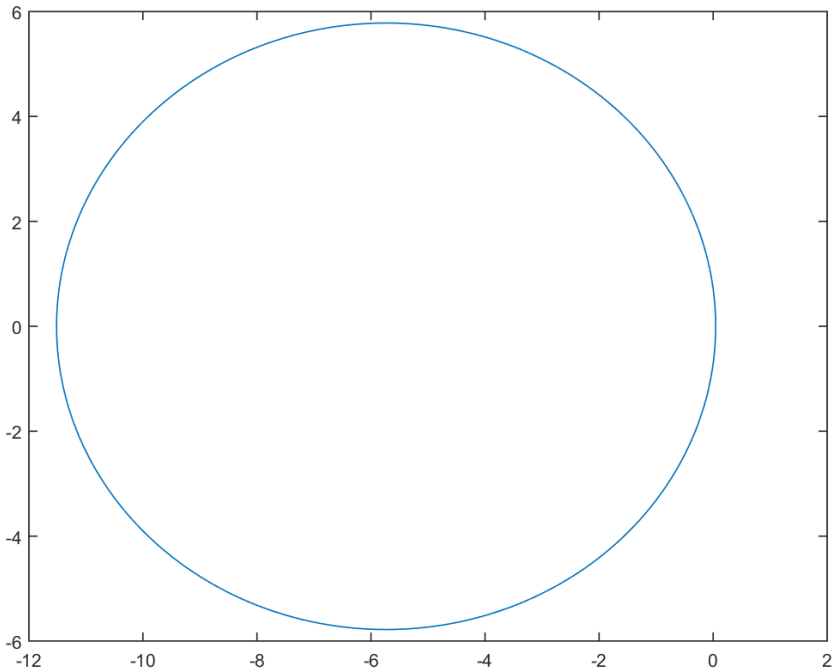
Ponieważ na wyjściu otrzymaliśmy tę samą sinusoidę z drobnymi zawahaniami w początkowych fazach (pojazd musiał ustawić się na torze) wnioskujemy, że test został przeprowadzony poprawnie.

3.1.2 Skok zadany, dodatni

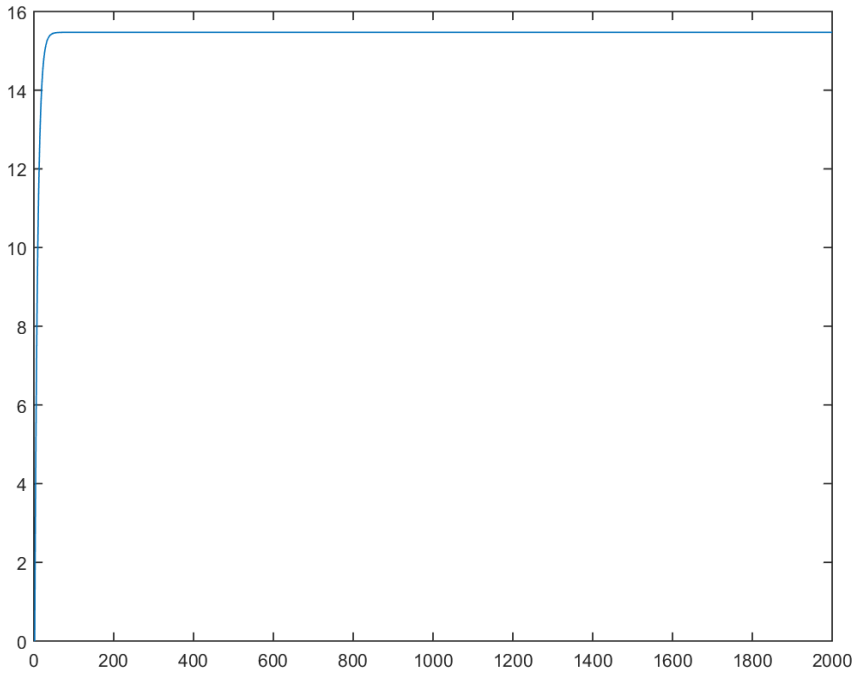
Obydwie ścieżki są ze sobą zgodne. Zgodnie z przewidywaniami są to okręgi (kąt skrętu był stały), które zaczynają się w punkcie (0,0). Samochód poruszał się po nich zgodnie z dodatnim kierunkiem trygonometrycznym po okręgu o promieniu zgodnym z przewidywaniami.



Rysunek 3.4: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.5: Ścieżka którą podążał pojazd

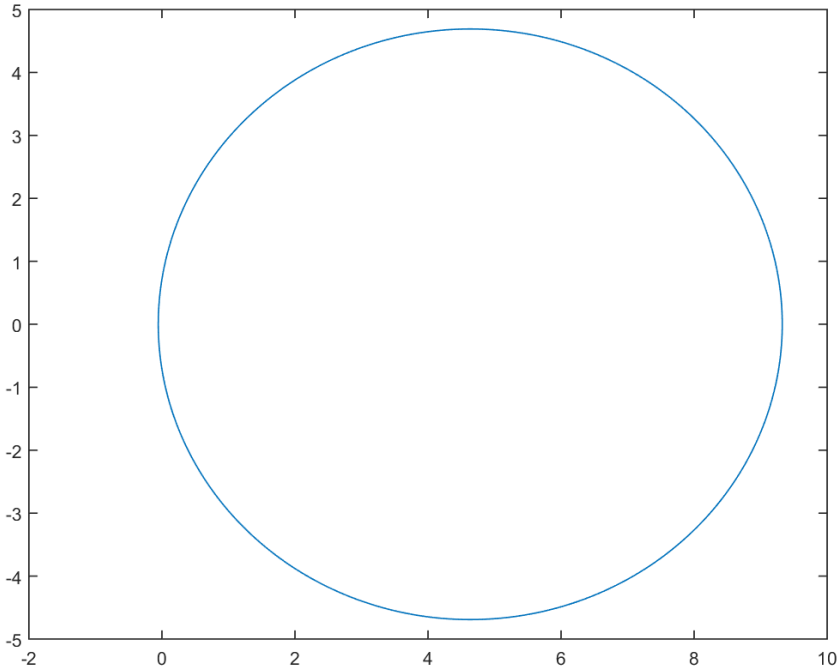


Rysunek 3.6: Kąt wyznaczony przez kinematykę odwrotną

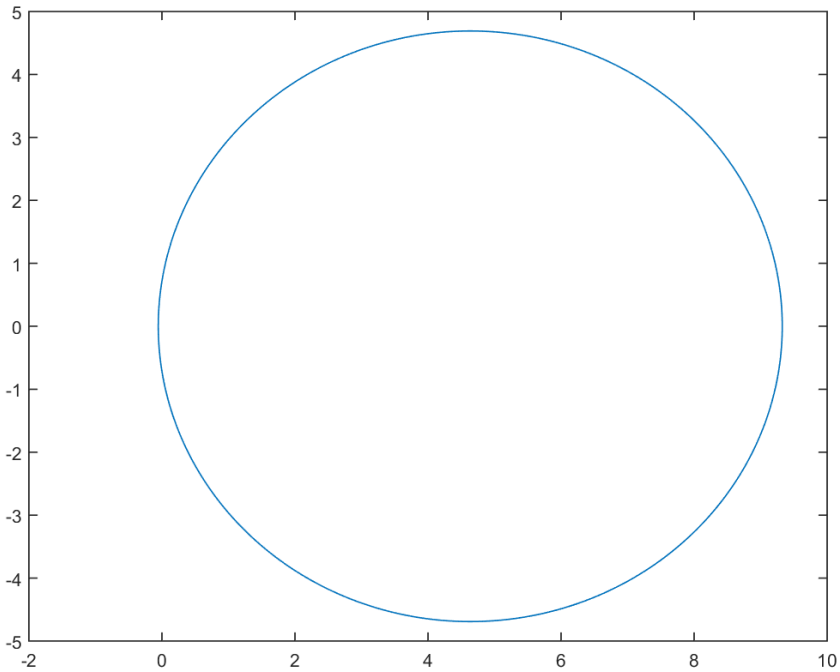
Na wykresie pozyskanym z kinematyki odwrotnej można zauważyć fazę początkową jak pojazd próbuje wjechać na koło i na nim pozostać (nierówne ustawienie go na początku linii) oraz to, że kąt ten po ustabilizowaniu się jest równy kątowi, który został zadany w teście.

3.1.3 Skok zadany, ujemny

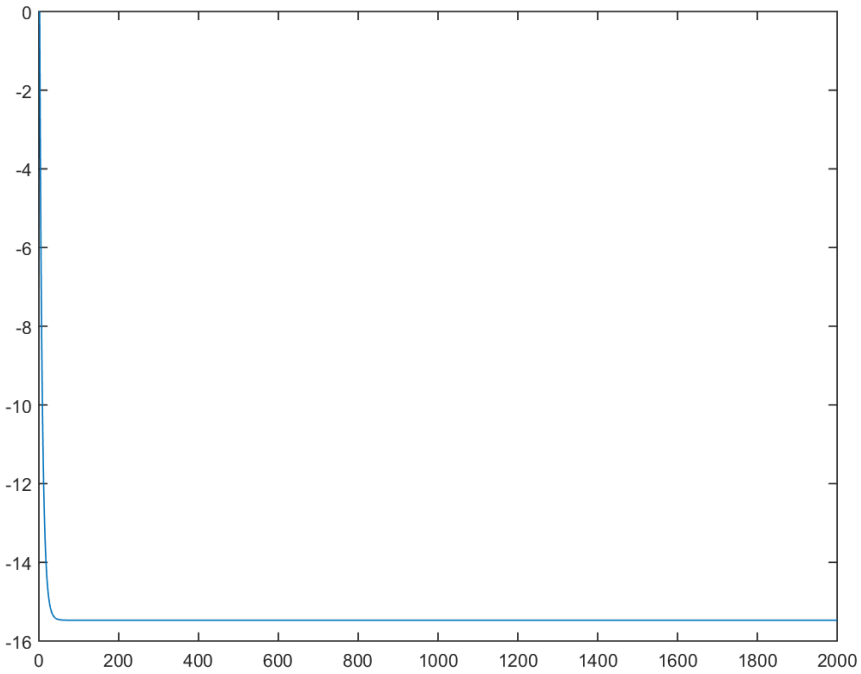
Obydwie ścieżki są ze sobą zgodne. Zgodnie z przewidywaniami są to okręgi (kąt skrętu był stały), które zaczynają się w punkcie (0,0). Samochód poruszał się po nich zgodnie z ujemnym kierunkiem trygonometrycznym po okręgu o promieniu zgodnym z przewidywaniami.



Rysunek 3.7: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.8: Ścieżka którą podążał pojazd

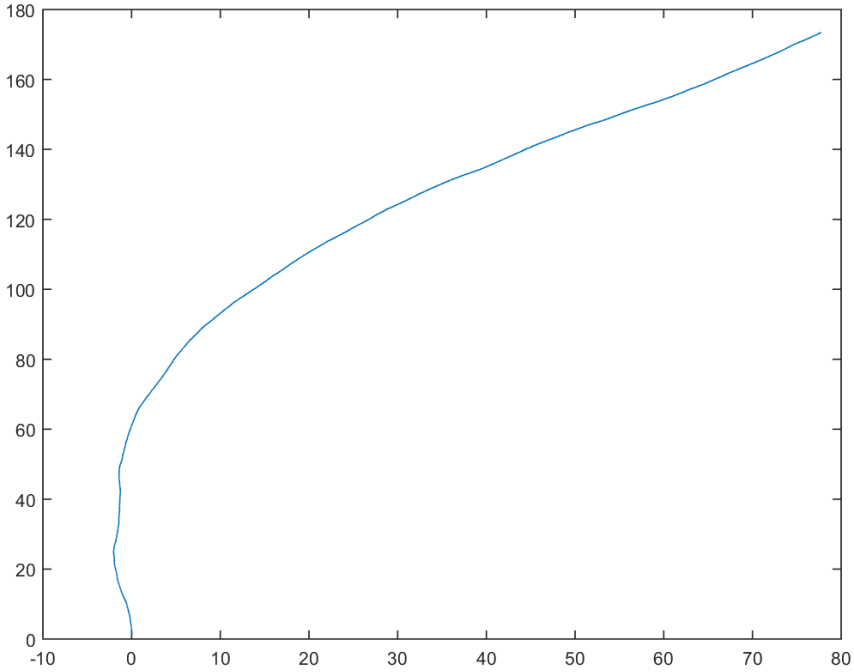


Rysunek 3.9: Kąt wyznaczony przez kinematykę odwrotną

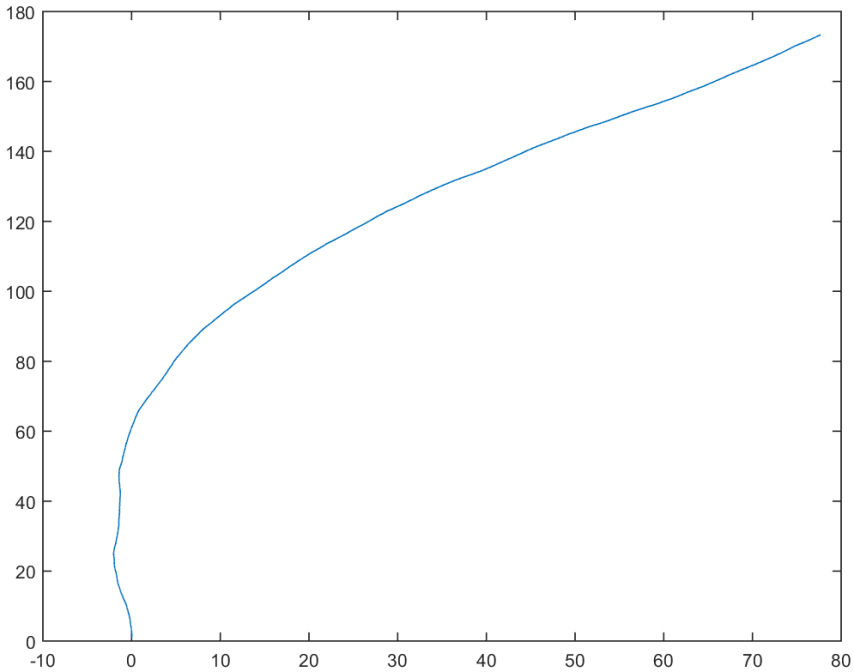
Na wykresie pozyskanym z kinematyki odwrotnej można zauważyć fazę początkową jak pojazd próbuje wjechać na koło i na nim pozostać (nierówne ustawienie go na początku linii) oraz to, że kąt ten po ustabilizowaniu się jest równy kątowi, który został zadany w teście.

3.1.4 Proces gaussowski

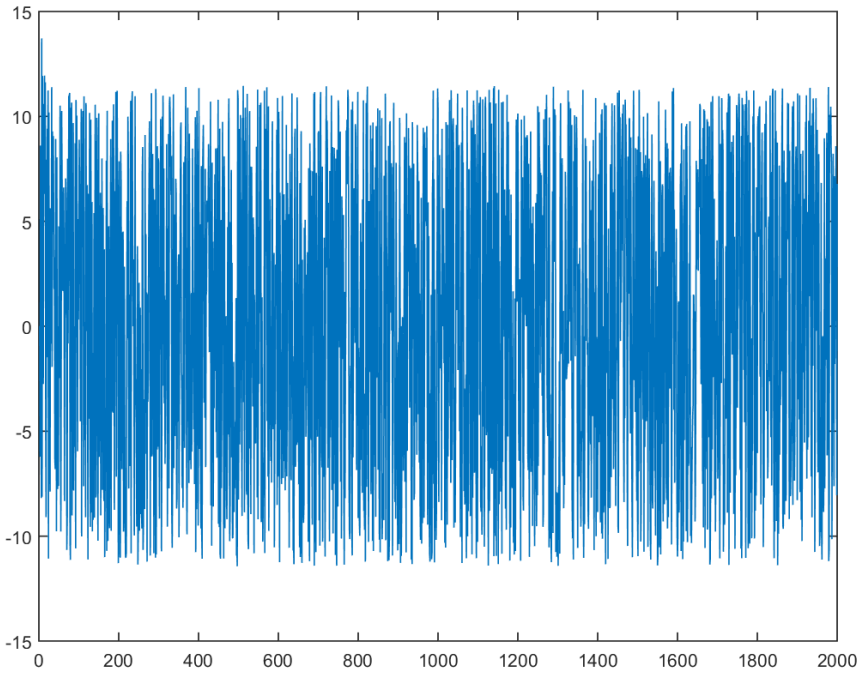
W tym teście wektorem wprowadzonym przez nas na wejście jest proces gaussowski taki, że każdy kolejny krok jest punktem spełniającym rozkład normalny przekształcony w pewien sposób ograniczający jego wartość do dopuszczalnego zakresu. Obydwie ścieżki są ze sobą zgodne.



Rysunek 3.10: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.11: Ścieżka którą podążał pojazd

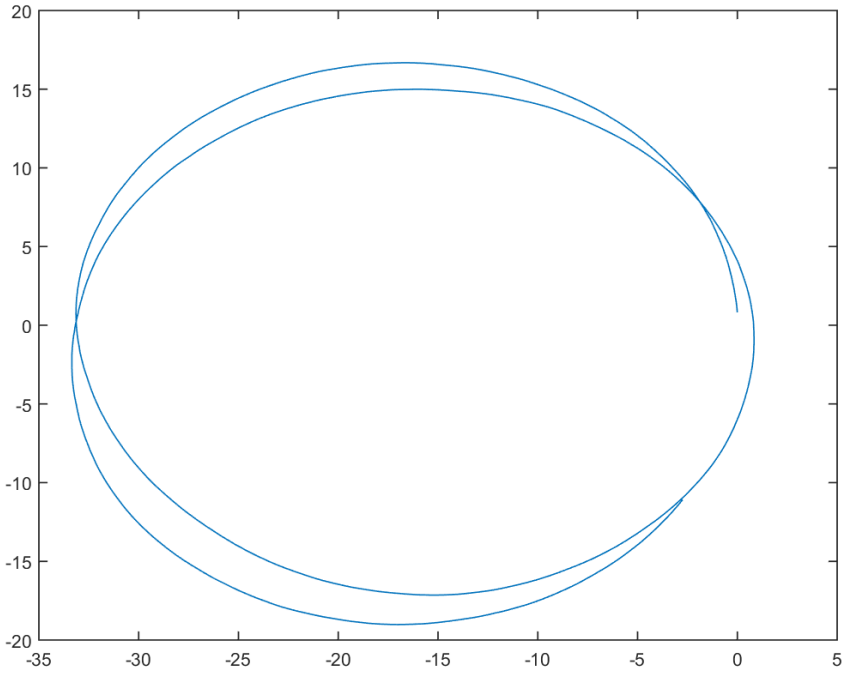


Rysunek 3.12: Kąt wyznaczony przez kinematykę odwrotną

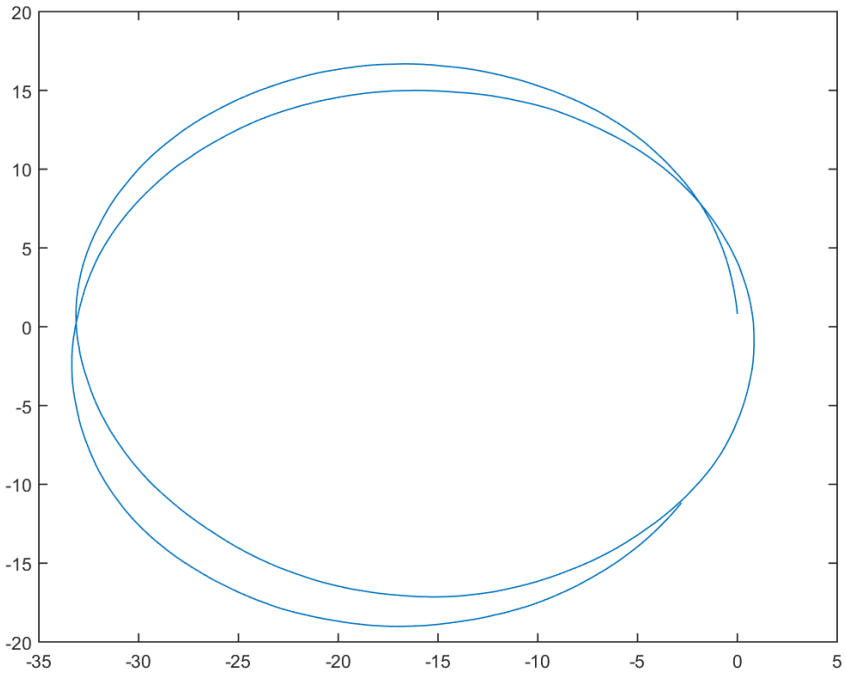
Pomimo tego, że kąt który był wykorzystywany do testu był z zakresu (-11,5, 11,5) stopni można na przebiegu zauważyć, że wartość ta jest nieco przekraczana. Proces gaussowski dla wartości bliższych maksymalnego zakresu kąta doprowadzał do wyskoczenia komunikatu o przekroczeniu maksymalnego zakresu. Są to jednak błędy numeryczne wynikające z liczby iteracji.

3.1.5 Wartości losowe dodatnie

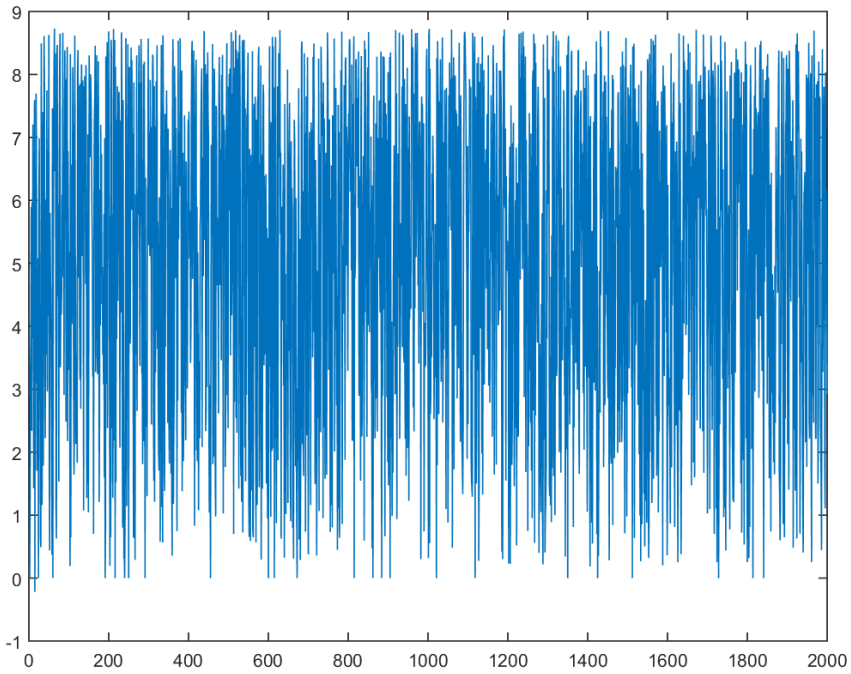
Ścieżki zgadzają się ze sobą. Pojazd porusza się zgodnie z dodatnim kierunkiem trygonometrycznym po różnych krągłych krzywych. Dzieje się tak ponieważ zmieniając losowo kąt zmienia się również losowo środek obrotu, jednak wartości tylko dodatnie gwarantują, że obrót będzie się zawsze odbywać w dodatnim kierunku trygonometrycznym.



Rysunek 3.13: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.14: Ścieżka którą podążał pojazd

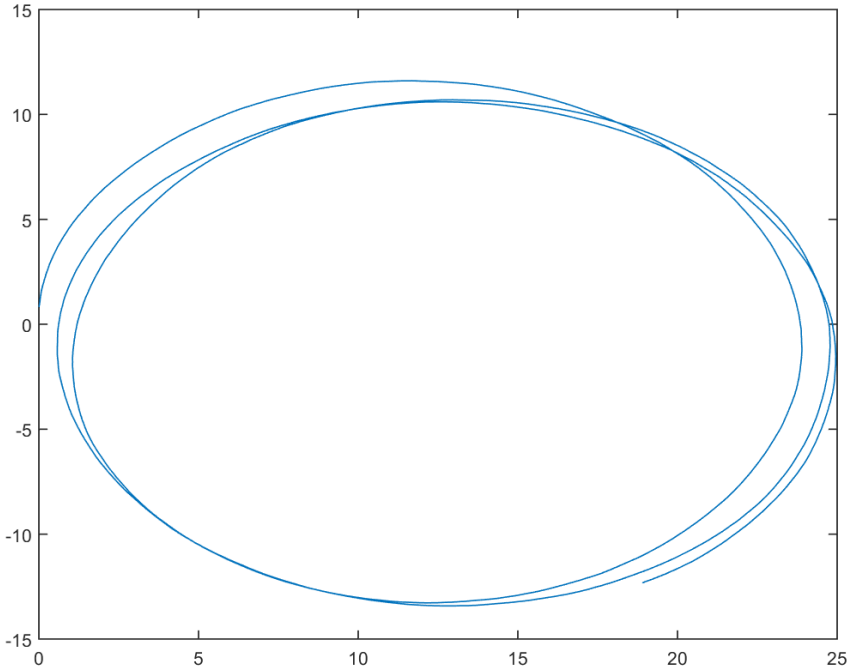


Rysunek 3.15: Kąt wyznaczony przez kinematykę odwrotną

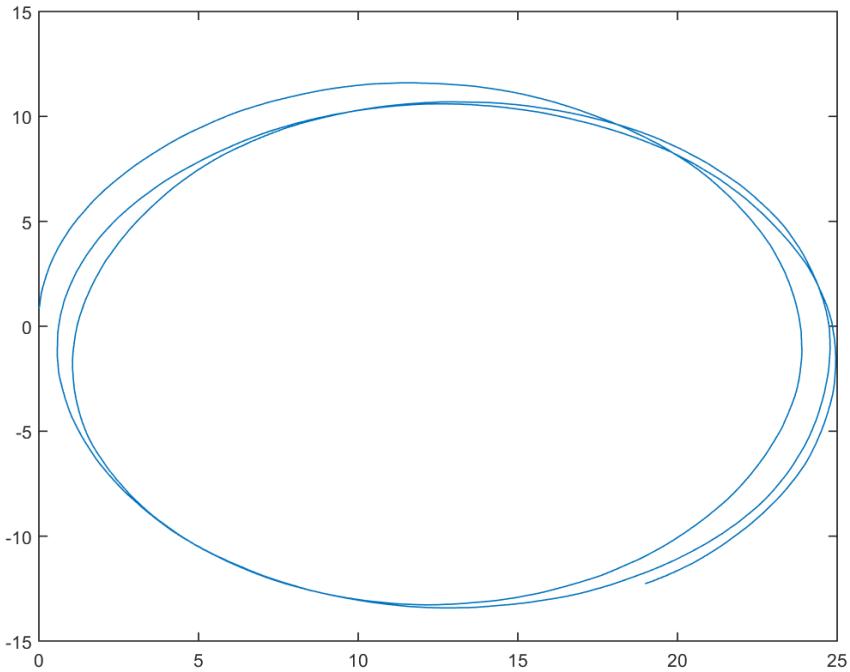
Na podstawie rozkładu wartości kąta, jaki sczytała kinematyka odwrotna można zauważyć funkcję, która była przez nas używana do ograniczenia wartości zmiennej losowej. Używaliśmy do tego tangensa hiperbolicznego, który generuje zagęszczenie występowania wartości wyższych. Na podstawie tego i ścieżki którą podążał pojazd możemy stwierdzić, że test został zakończony poprawnie.

3.1.6 Wartości losowe ujemne

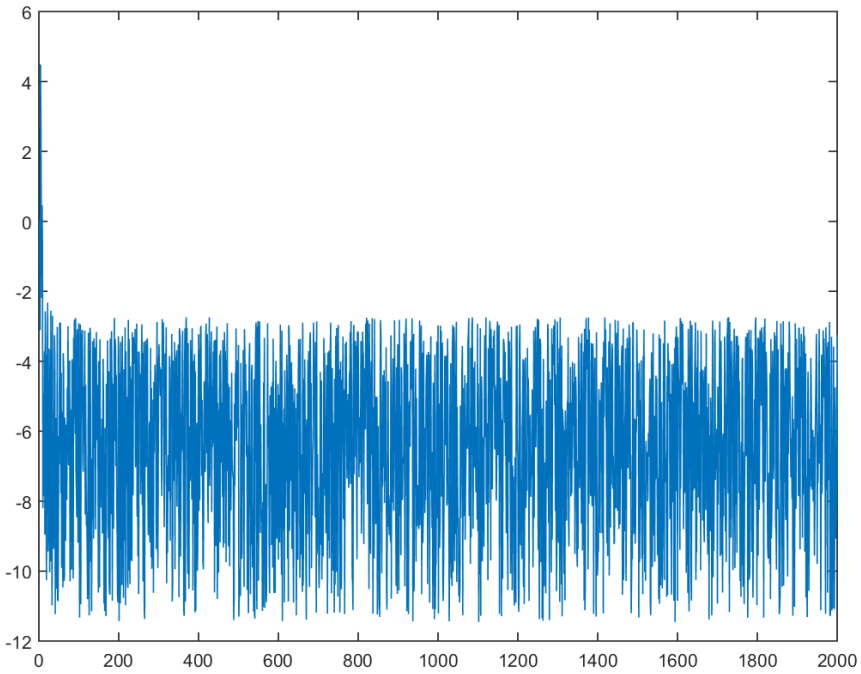
Ścieżki zgadzają się ze sobą. Podobnie jak dla wartości losowych dodatnich krzywe są krągłe. Jednak w tym przypadku mają ujemny kierunek trygonometryczny – pojazd porusza się zgodnie ze wskazówkami zegara.



Rysunek 3.16: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.17: Ścieżka którą podążał pojazd

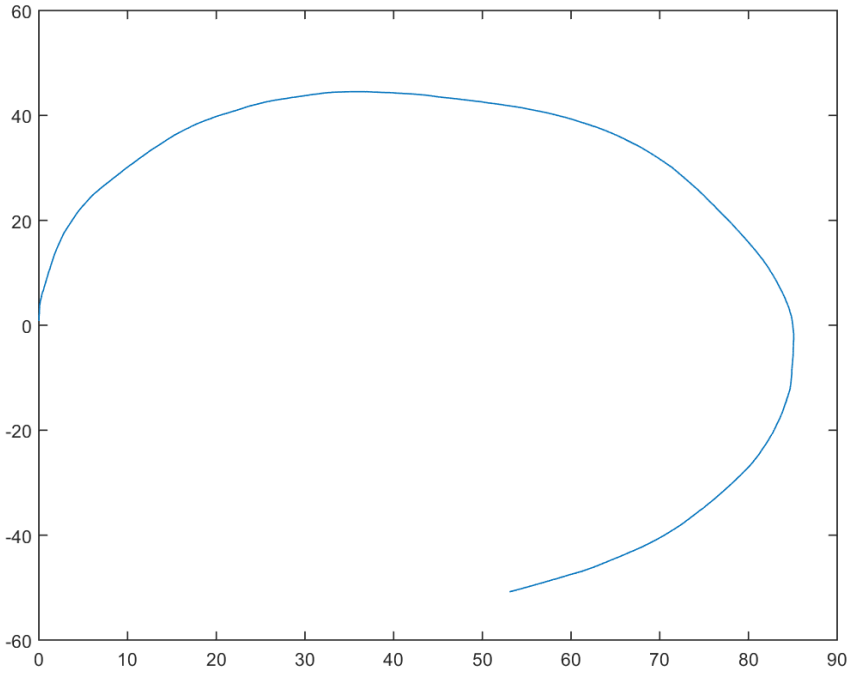


Rysunek 3.18: Kąt wyznaczony przez kinematykę odwrotną

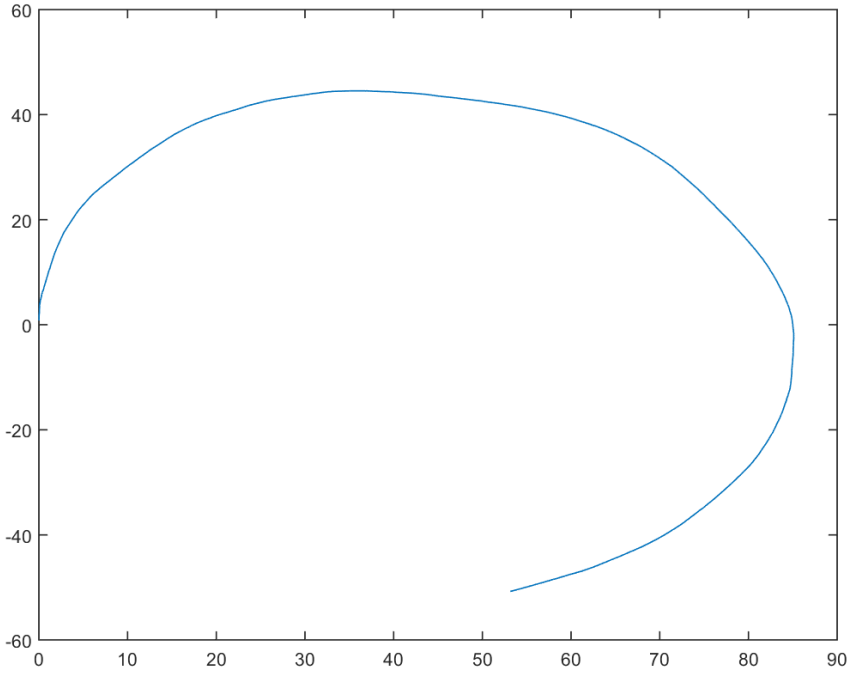
Odczytane wartości kątów zgadzają się z tymi zadanymi. Z początku jednak można zauważyć krótką chwilę kiedy samochód poruszał się w kierunku zgodnym ze wskazówkami zegara, próbując zacząć się poruszać zgodnie z krzywą.

3.1.7 Wartości losowe

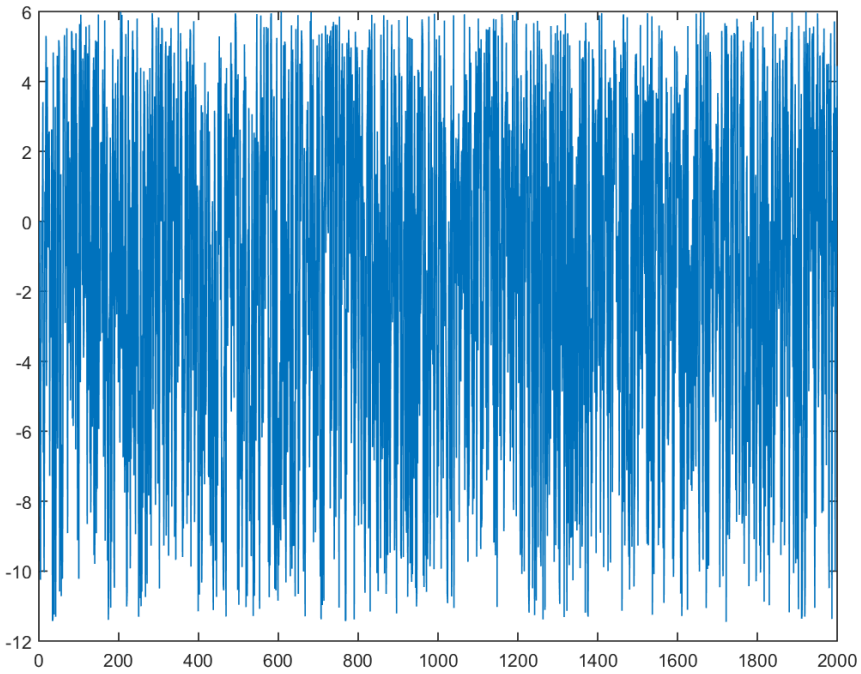
Ścieżki zgadzają się ze sobą. Pojazd porusza się losowo, można tu zauważyć, że nasz mechanizm ograniczania liczb pseudolosowych faworyzował liczby ujemne – pojazd poruszał się zgodnie ze wskazówkami zegara. Jednak nie robił tego aż tak intensywnie jak robił to dla wartości losowych ujemnych.



Rysunek 3.19: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.20: Ścieżka którą podążał pojazd

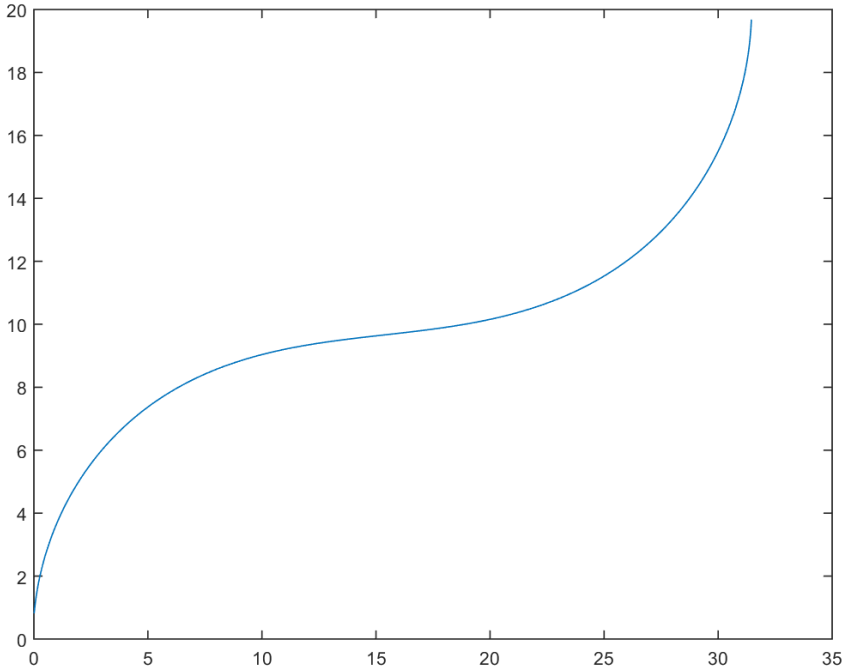


Rysunek 3.21: Kąt wyznaczony przez kinematykę odwrotną

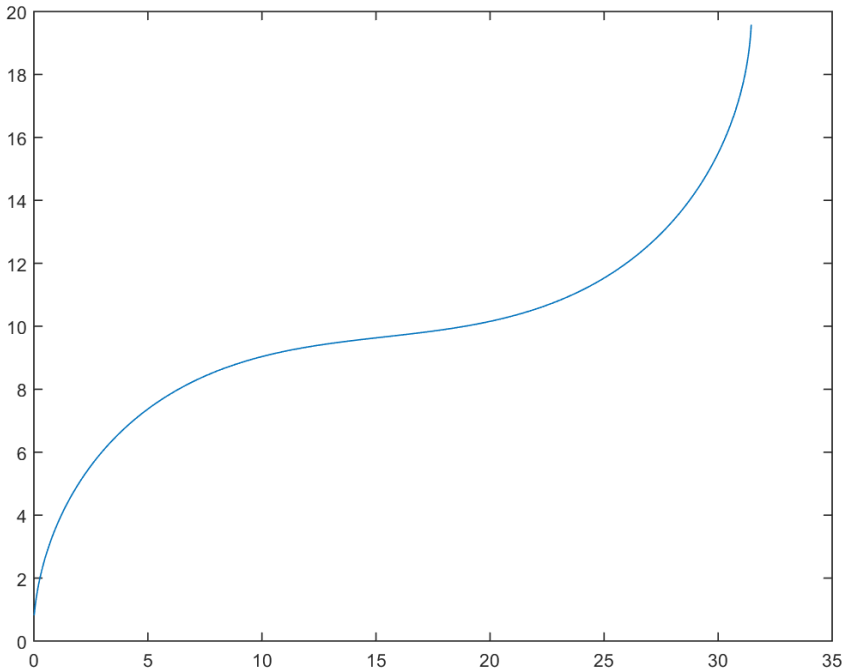
Odczytane wartości kątów zgadzają się z rozkładem wartości ogranicznika. Można odczytać, że wartości ujemne są nieco bardziej faworyzowane od wartości dodatnich, gdyż był to odpowiednio przesunięty tangens hiperboliczny.

3.1.8 Liniowa zmiana kąta

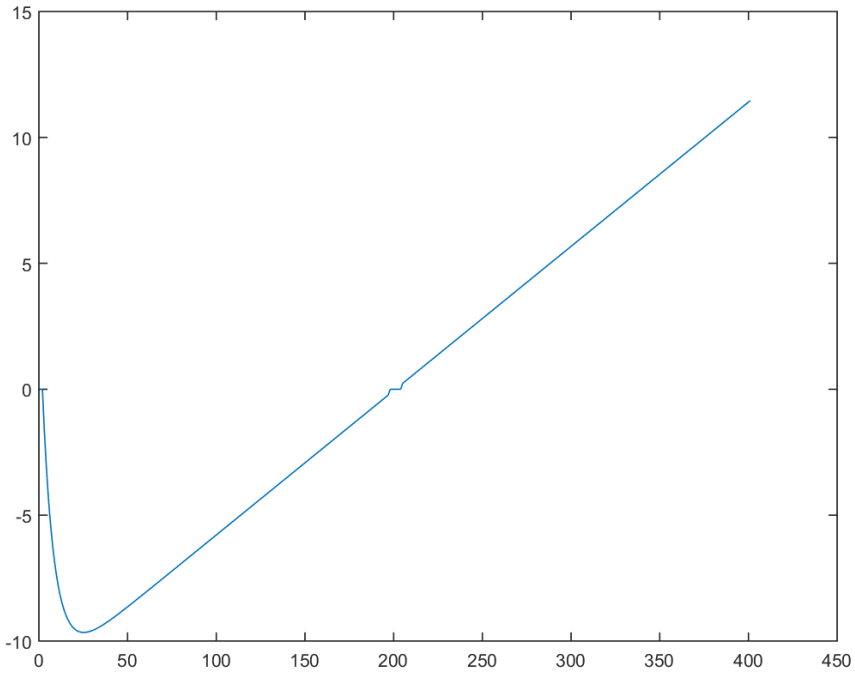
Ścieżki zgadzają się. Przy zmianie kąta w sposób ciągły można dostrzec miejsce, gdzie pojazd skręca w lewo, zaczyna jechać prosto, a następnie skręcać w prawo. Ścieżka jest punktowo symetryczna względem swojego środka, co jest zgodne z naszymi przewidywaniami.



Rysunek 3.22: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.23: Ścieżka którą podążał pojazd

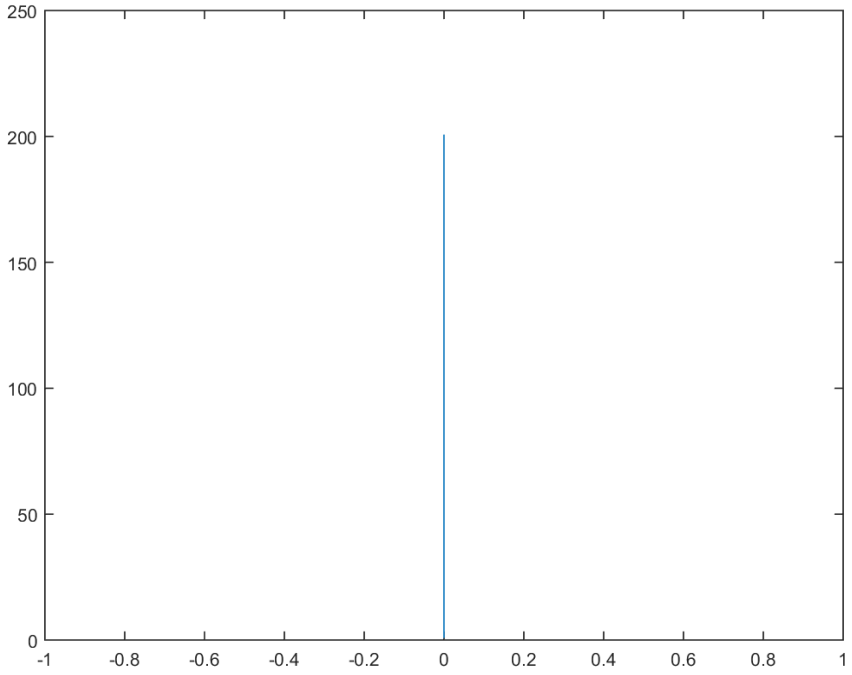


Rysunek 3.24: Kąt wyznaczony przez kinematykę odwrotną

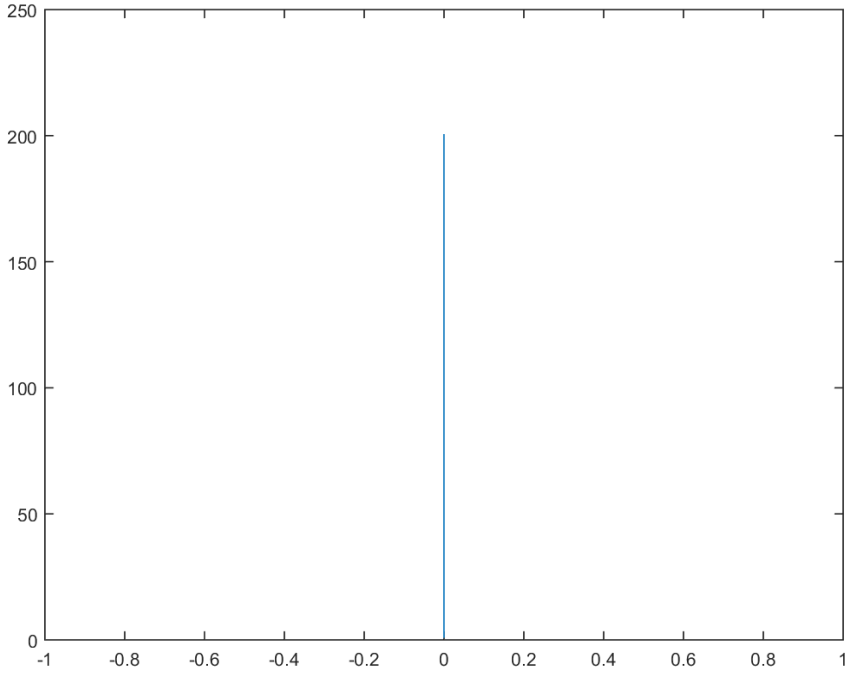
Jest to przykład, gdzie najlepiej widać jak pojazd próbuje się ustawić tak, aby jechać stycznie do toru. Następnym ważnym spostrzeżeniem jest widoczność niewielkiej strefy nieczułości, jaka musieliśmy zastosować w celu likwidacji błędów numerycznych.

3.1.9 Jazda prosta

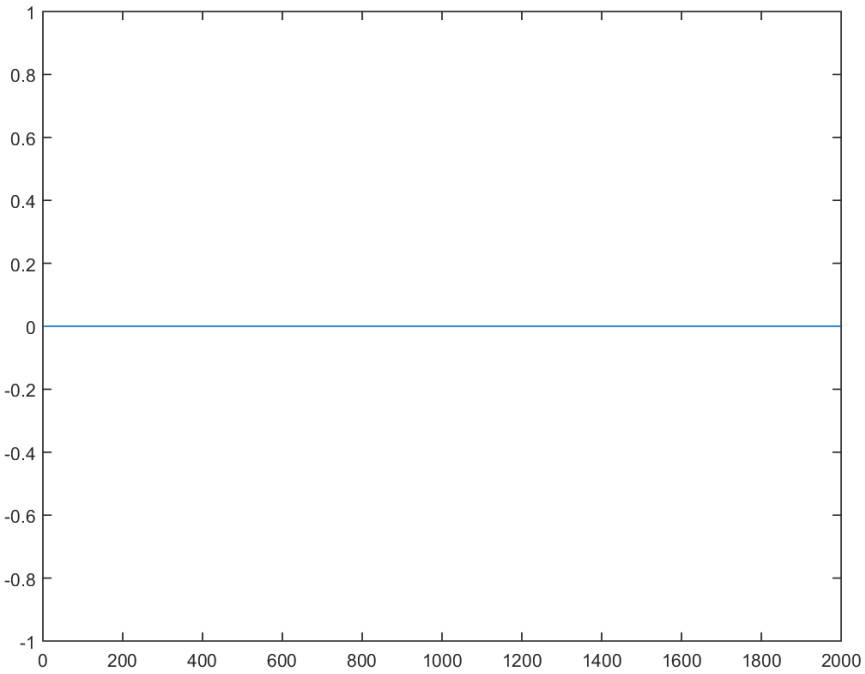
Ścieżki zgadzają się. Zgodnie z przewidywaniami, jeżeli podamy na wejście wektor zerowy, bo pojazd będzie się poruszał po linii prostej.



Rysunek 3.25: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.26: Ścieżka którą podążał pojazd



Rysunek 3.27: Kąt wyznaczony przez kinematykę odwrotną

Kąt pozyskany przez kinematykę odwrotną zgadza się z kątem przez nas zadanym. Nie występuje tu spotykane wcześniej „wjeżdżanie pojazdu na tor", gdyż mamy do czynienia z linią prostą, do której można łatwo stycznie ustawić pojazd.

3.2 Kinematyka Odwrotna

Podobnie jak w testach dla kinematyki prostej mamy do czynienia z trzema wykresami. Dla prostoty zachowaliśmy ten sam system oznaczeń oraz te same zasady nazewnictwa. Przebiegi są wyskalowane na tej samej zasadzie.

Poniższy kawałek kodu przedstawia schemat testów wykorzystanych do modelu kinematyki odwrotnej. Test składa się z dwóch części – w pierwszej inicjujemy krzywą pewnymi wartościami. Następnie skrypt przystępuje do symulacji kinematyki odwrotnej, której wynik jest wykorzystywany do symulacji kinematyki prostej. Kolejność dokonywania zmian jest tak przesunięta aby dane mogły zostać bezpośrednio przekazane dalej.

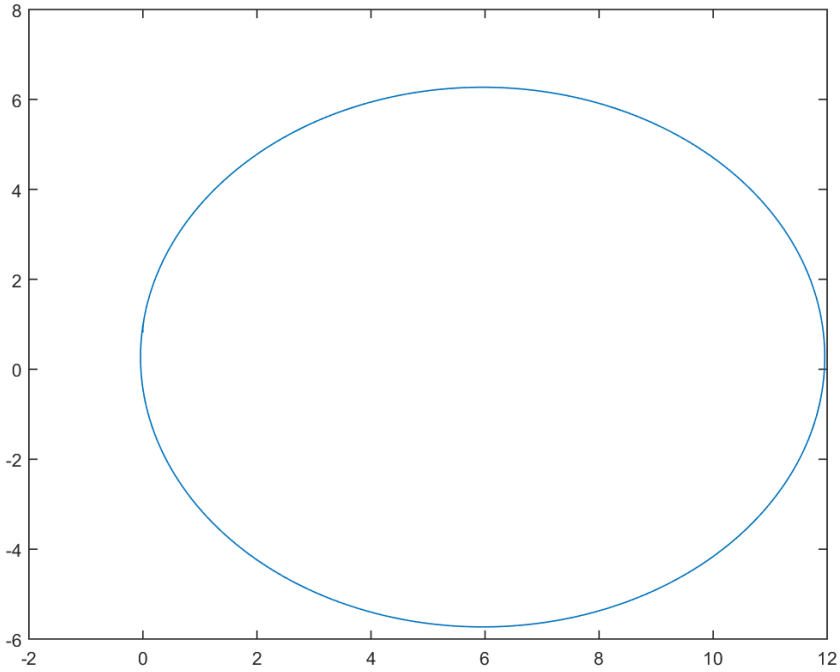
```
function rev_test_1()
    t = 0:0.1:1000;
    a = 6*sin(t);
    b = 6*cos(t);
    c = ones(1,length(a));

    inpt = [a;b;c];
    restr = [10e5, 10e5, -10e5, -10e5;
        10e5, -10e5, -10e5, 10e5];
    figure(2)
    t = eve_reverse(inpt, restr);
    print('img/rev/right_circle_in_2.png','-dpng')
    figure(3)
    plot(1:length(t), t*180/pi)
    print('img/rev/right_circle_in_3.png','-dpng')
    figure(1)
    t = eve_forward(t);
    plot(t(1,:), t(2,:))
    print('img/rev/right_circle_in_1.png','-dpng')
end
```

Zwrócone przez skrypt trzy wykresy są zapisywane na dysku w osobnym katalogu i są dostępne do ręcznej walidacji.

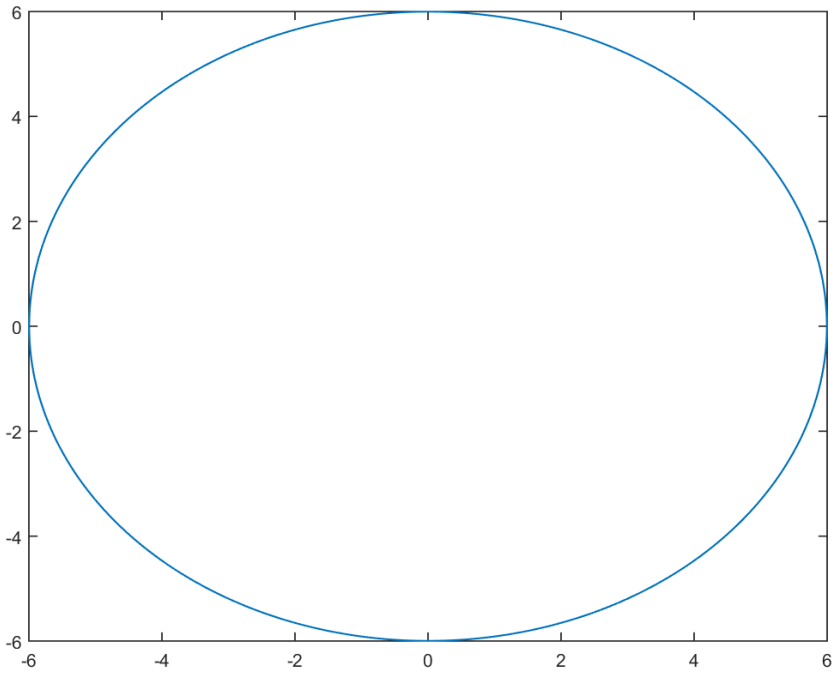
Ze względu na to, że dane są przekazywane z funkcji, która jako argument mogła przyjąć ścieżkę o zmiennym kroku do funkcji, która jako wynik zwraca ścieżkę o stałym kroku 10 cm, możemy tu tylko sprawdzać, czy wynik kinematyki prostej zachowuje się podobnie do przebiegu zadanego z początku. Przebiegi różniące się o obrót lub przesunięcie są brane jako identyczne ze względu na różne położenia początkowe pojazdów.

3.2.1 Jazda po okręgu w prawo

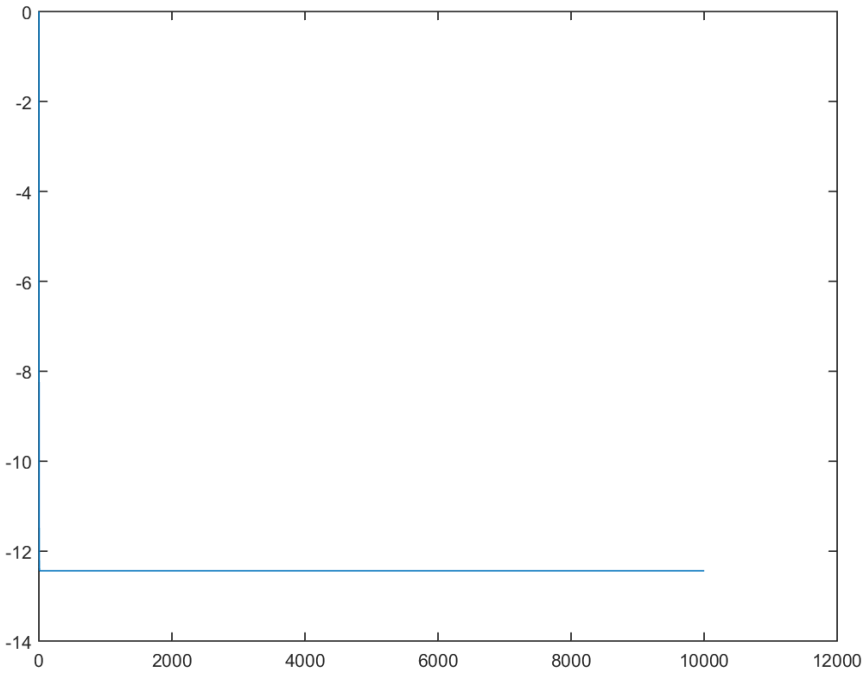
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki zgadzają się. Ze względu na to, że okrąg jest bardzo prostą figurą i został tu zatoczony wielokrotnie, wyniki przebiegu uzyskanego z kinematyki prostej pokrywają się z wynikami uzyskanymi w kinematyce odwrotnej.



Rysunek 3.28: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.29: Ścieżka którą podążał pojazd

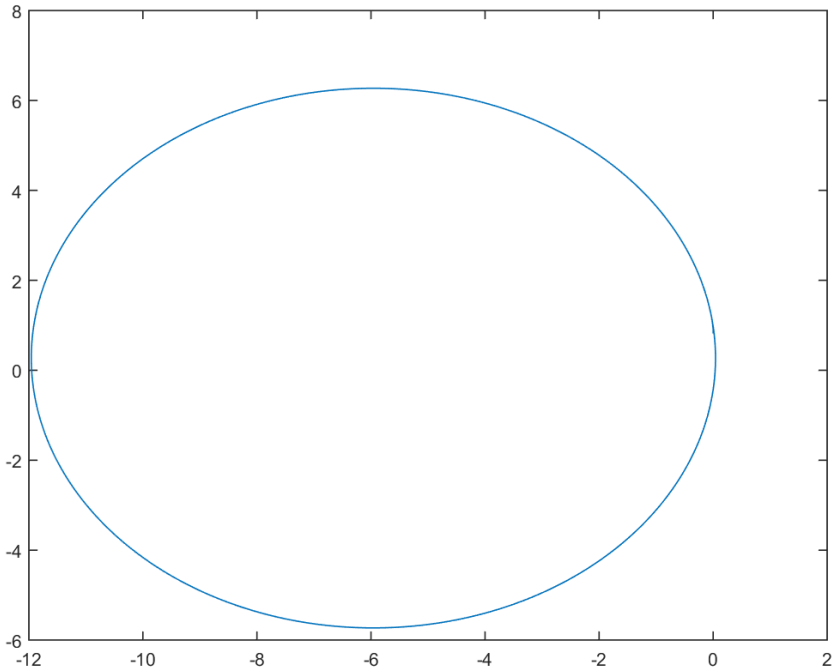


Rysunek 3.30: Kąt wyznaczony przez kinematykę odwrotną

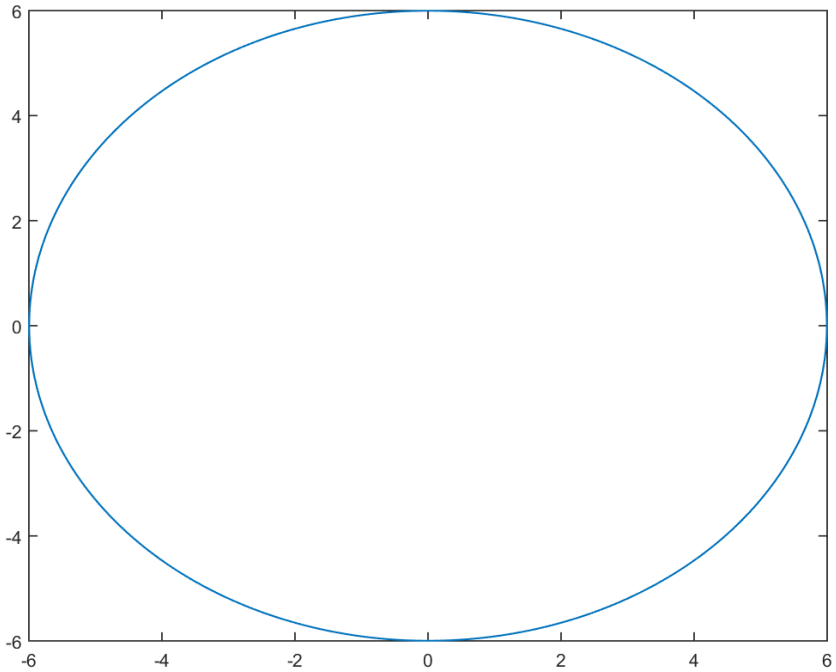
Zgodnie z oczekiwaniami wartość kąta jest ujemna oraz jest linią prostą. Te same wyniki ruchu środka masy w kinematyce prostej i odwrotnej sprawiają, że test można uznać za poprawny.

3.2.2 Jazda po okręgu w lewo

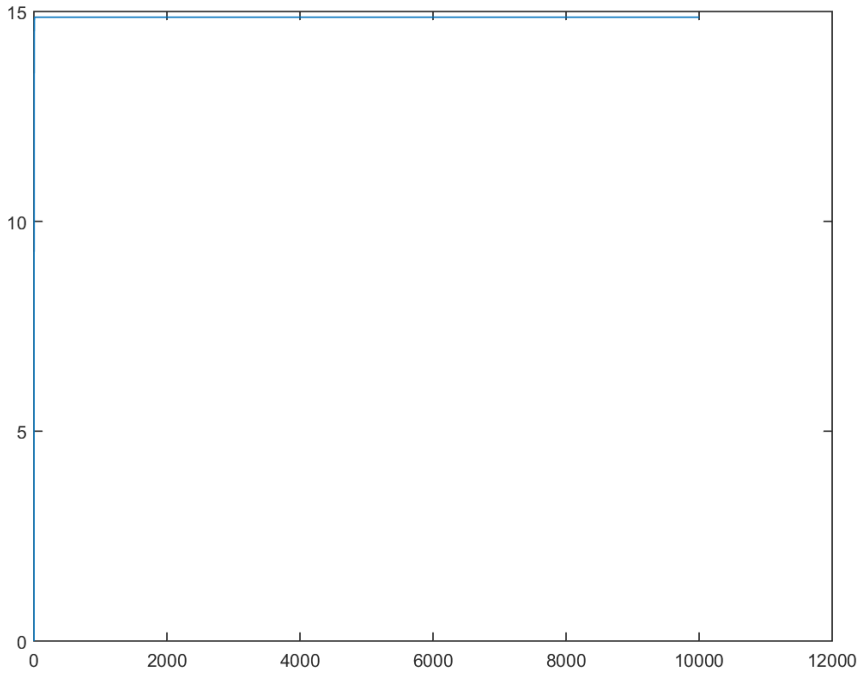
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki zgadzają się, środek ciężkości pojazdu przemieszcza się tak samo w przypadku kinematyki prostej, jak i odwrotnej.



Rysunek 3.31: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.32: Ścieżka którą podążał pojazd

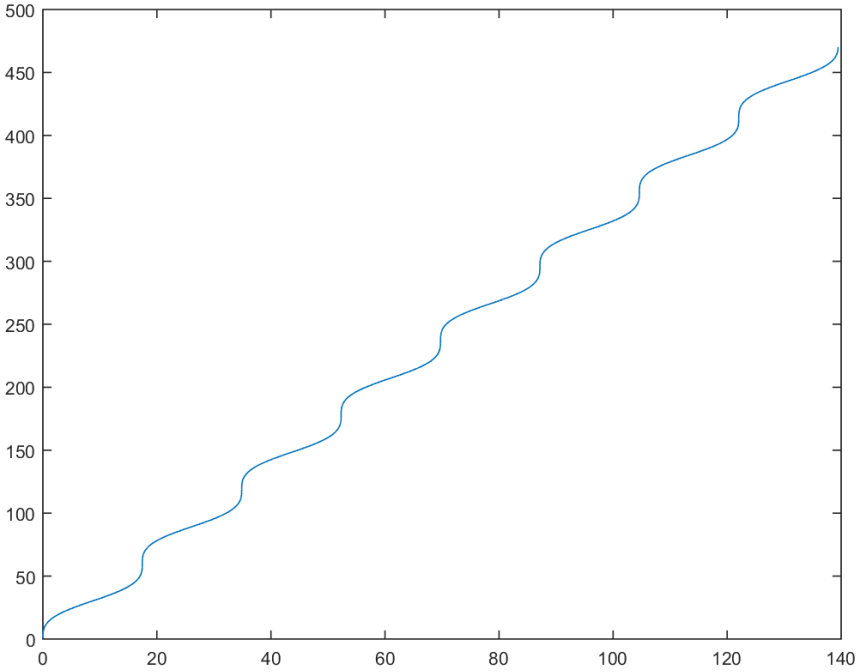


Rysunek 3.33: Kąt wyznaczony przez kinematykę odwrotną

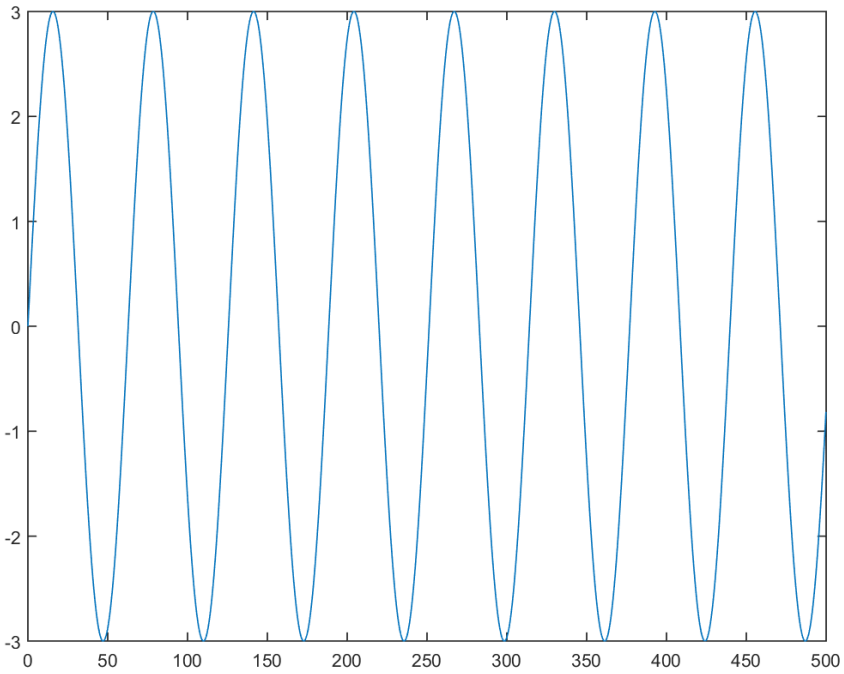
Kąt utrzymuje stałą, dodatnią wartość – tak jak powinien dla dodatniego kierunku trygonometrycznego. Jest to wartość zgodna z obliczeniami.

3.2.3 Jazda po sinusoidzie

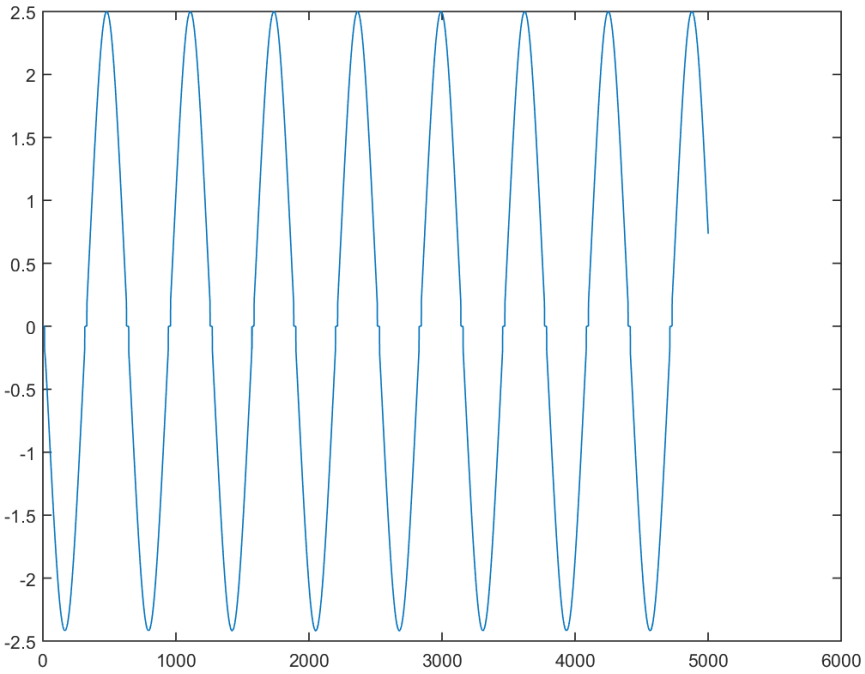
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki zachowują się podobnie. Trzeba dla tego testu wziąć pod uwagę różnicę między stałym i zmiennym krokiem oraz różnice w położeniu startowym pojazdu.



Rysunek 3.34: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.35: Ścieżka którą podążał pojazd

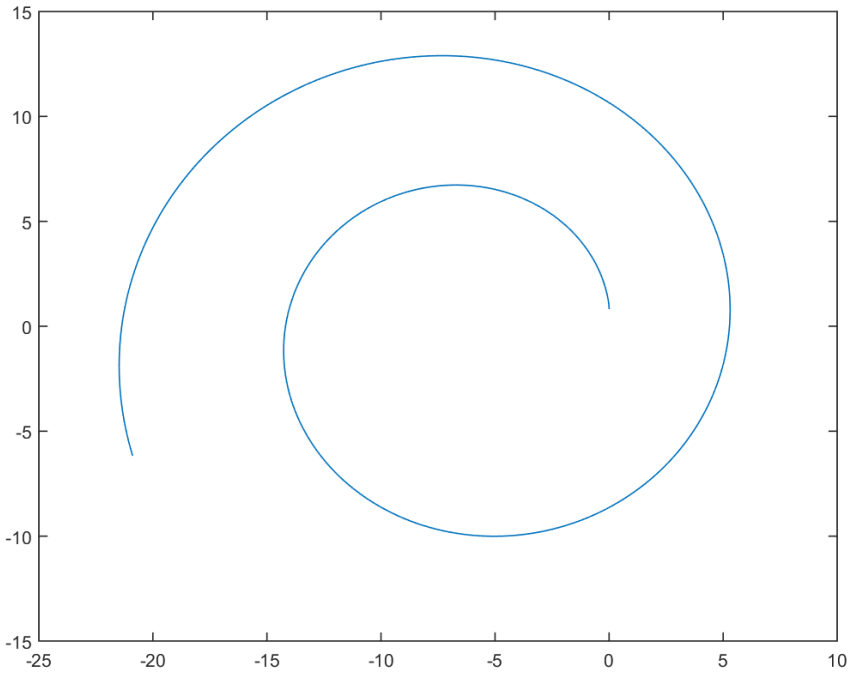


Rysunek 3.36: Kąt wyznaczony przez kinematykę odwrotną

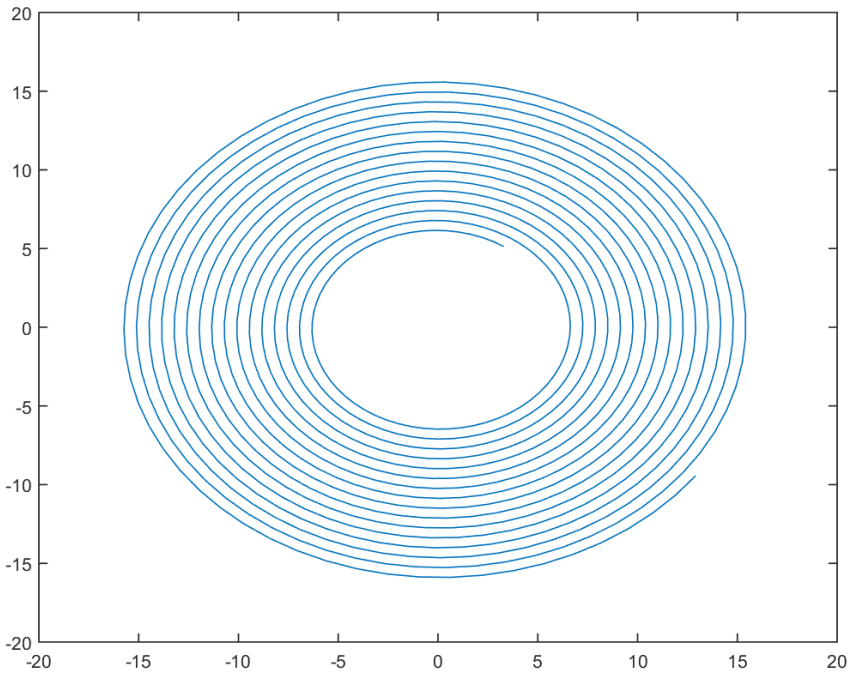
Kąt zachowuje się zgodnie z przewidywaniami - można zauważyć przejścia przez zero, które następują tyle razy ile pojazd zaczyna zmieniać kierunek skrętu. Początkowo pojazd skręca w prawo (ujemny kierunek trygonometryczny), co jest zgodne z przebiegami.

3.2.4 Jazda po spirali]

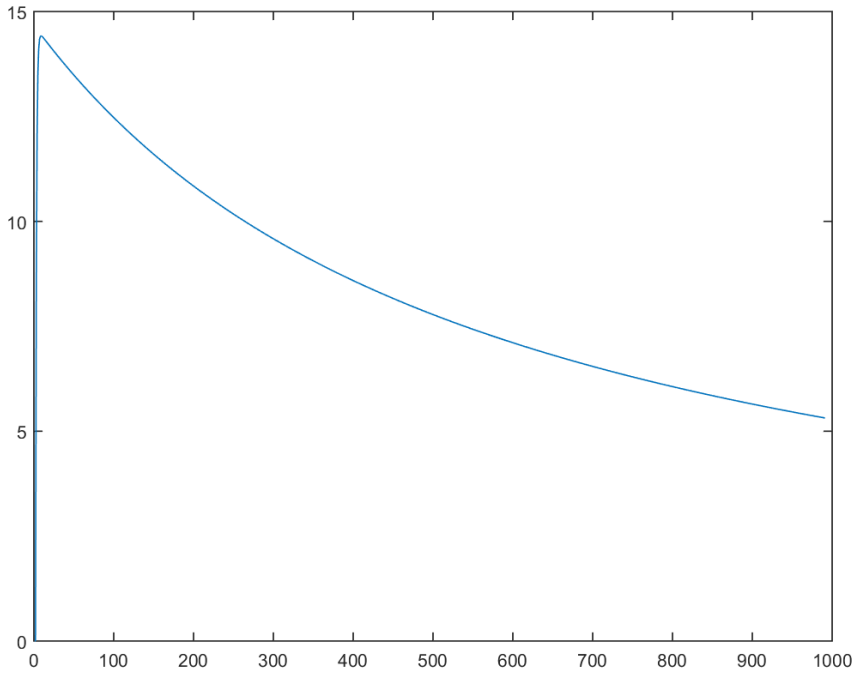
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki zachowują się podobnie.



Rysunek 3.37: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.38: Ścieżka którą podążał pojazd

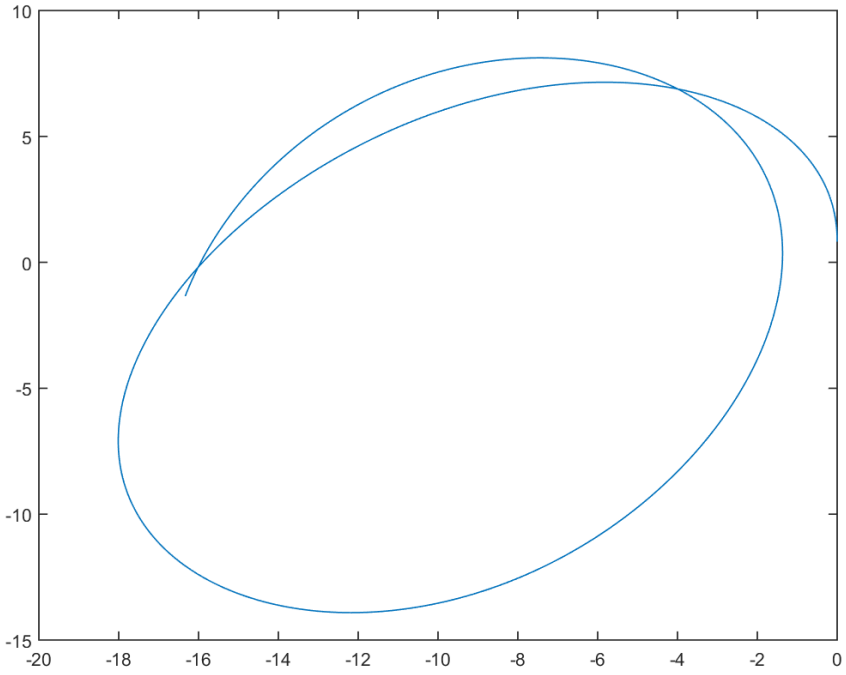


Rysunek 3.39: Kąt wyznaczony przez kinematykę odwrotną

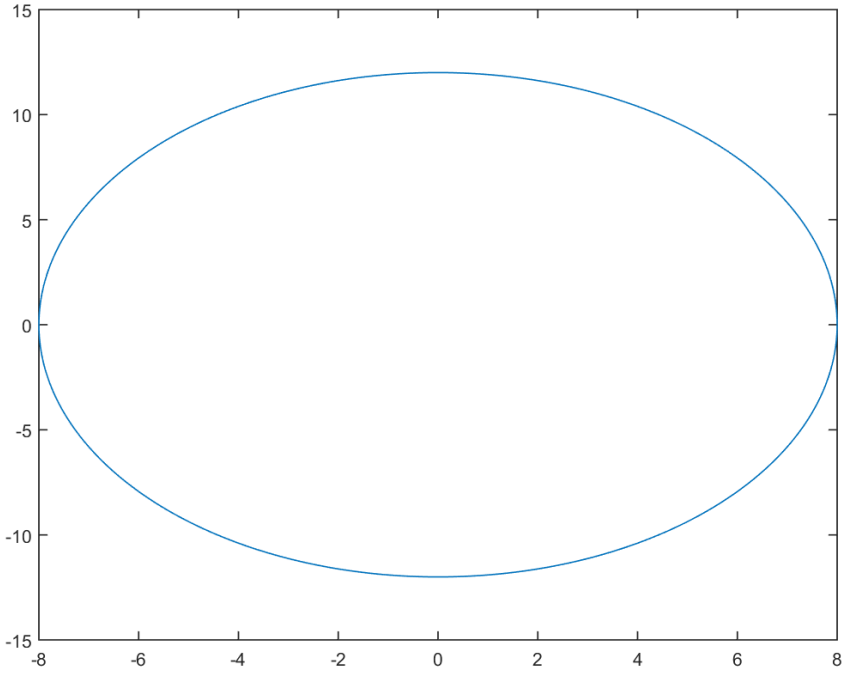
Z początku widać jak pojazd próbuje wjechać na spiralę, a następnie zgodnie z oczekiwaniami wartość kąta maleje, gdyż oddalając się od środka samochód zatacza coraz to większe okręgi.

3.2.5 Jazda po elipsie

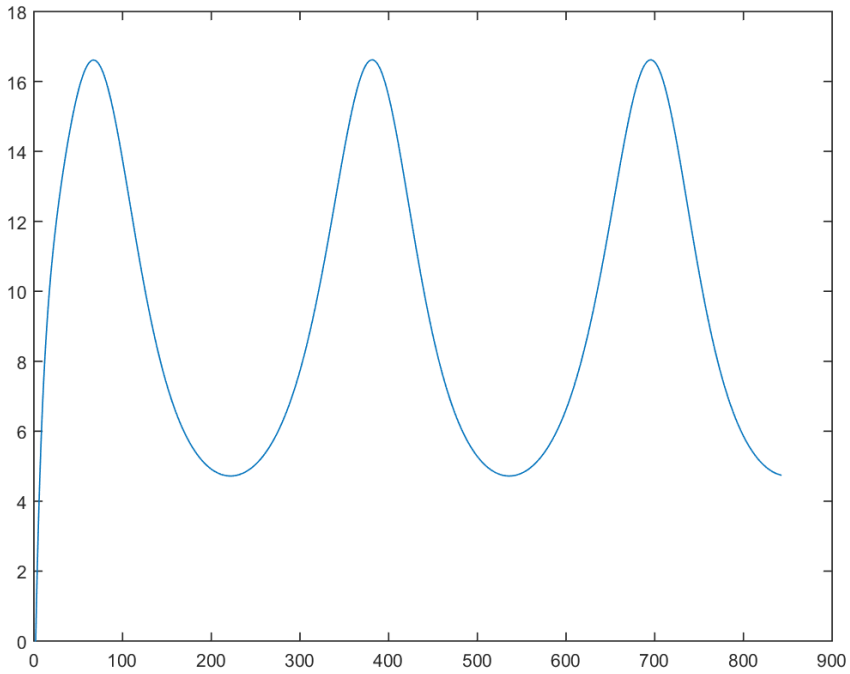
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki zachowują się podobnie, jednak stały krok powoduje rozjechanie się elipsy – elipsa się nie zamyka.



Rysunek 3.40: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.41: Ścieżka którą podążał pojazd

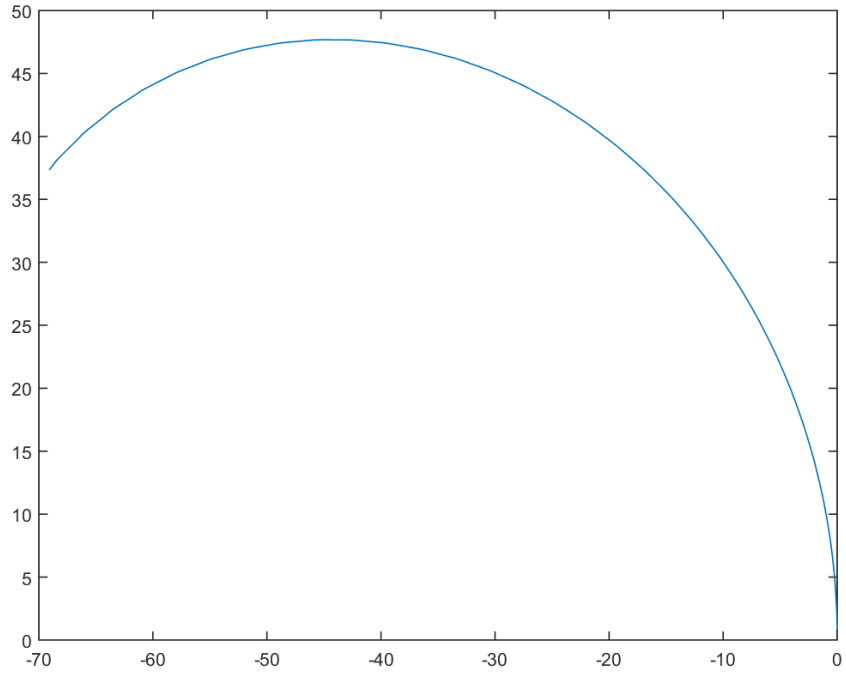


Rysunek 3.42: Kąt wyznaczony przez kinematykę odwrotną

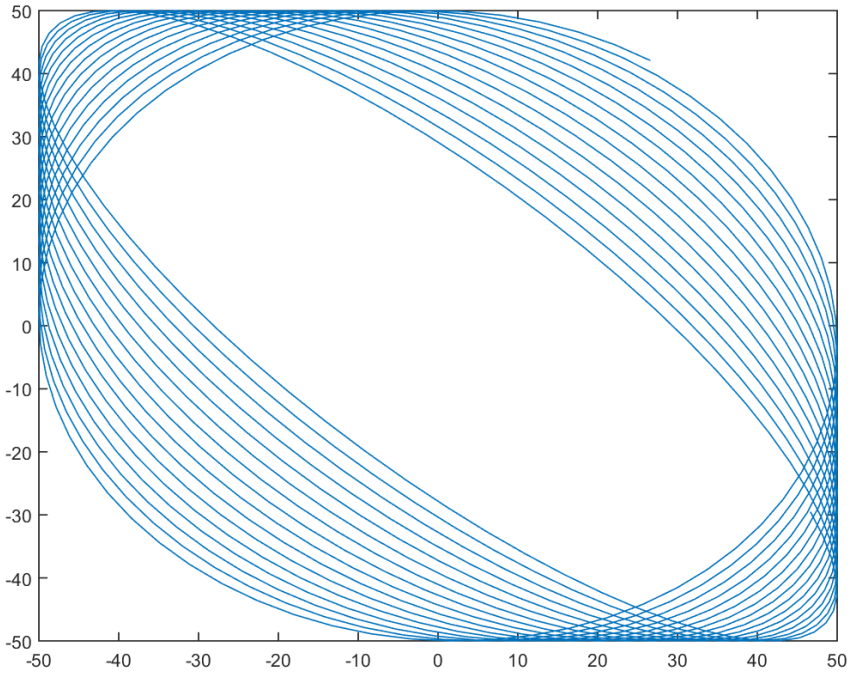
Przebieg kąta jest zgodny z oczekiwaniami – zmienia się pulsując, gdyż tymczasowy środek obrotu zbliża się i oddala od pojazdu w trakcie przemieszczania się po elipsie. Można zauważyć, że przekroczona została maksymalna wartość kąta możliwego do osiągnięcia przez samochód EVE, jednak nie tego, który przyjęliśmy dla symulacji.

3.2.6 Krzywa Lissajous o niewielkiej różnicy współczynników

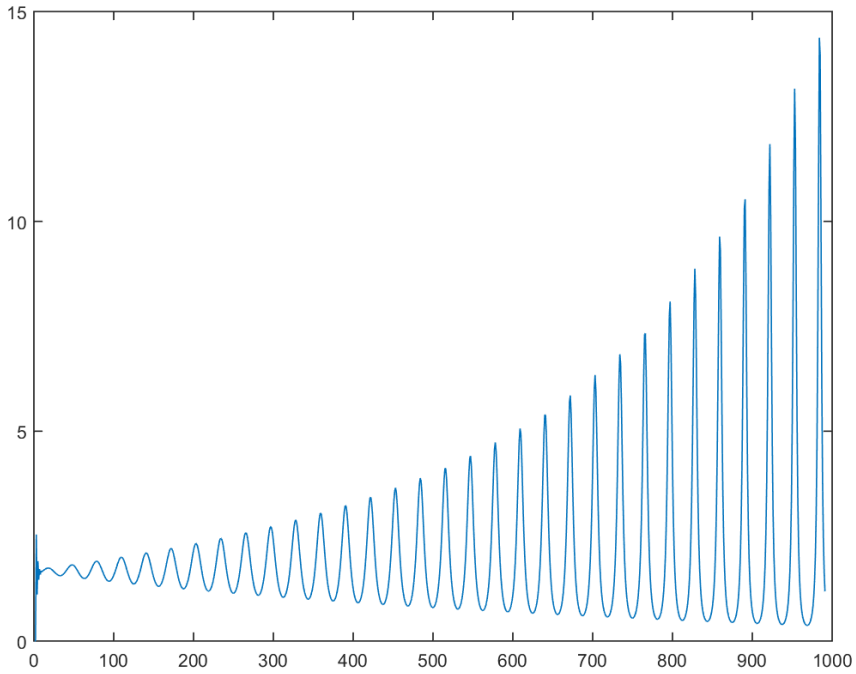
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki różnią się. Różnica między ścieżkami z kinematyki prostej i odwrotnej wynika tu ze stałego kroku w modelu kinematyki prostej.



Rysunek 3.43: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.44: Ścieżka którą podążał pojazd

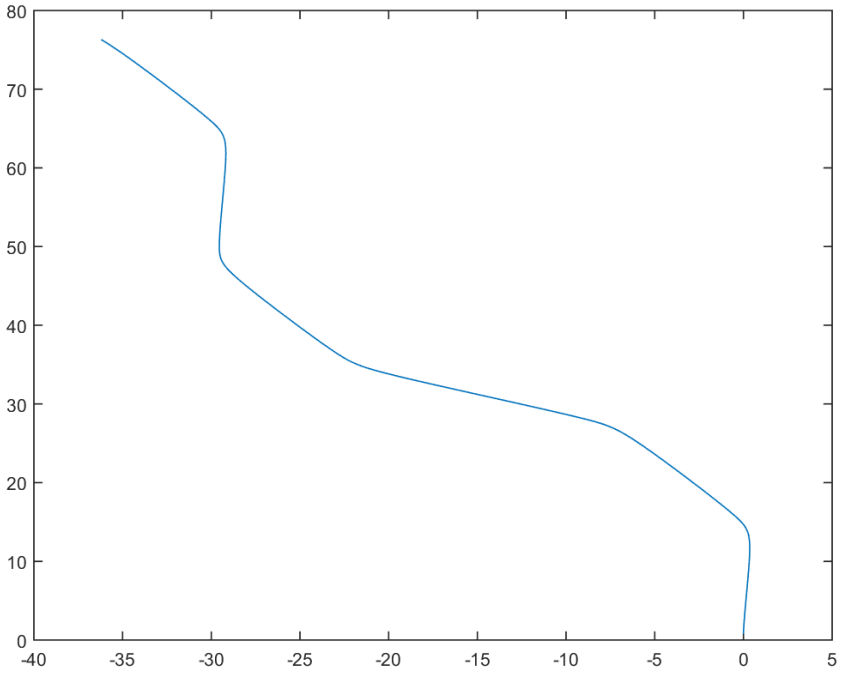


Rysunek 3.45: Kąt wyznaczony przez kinematykę odwrotną

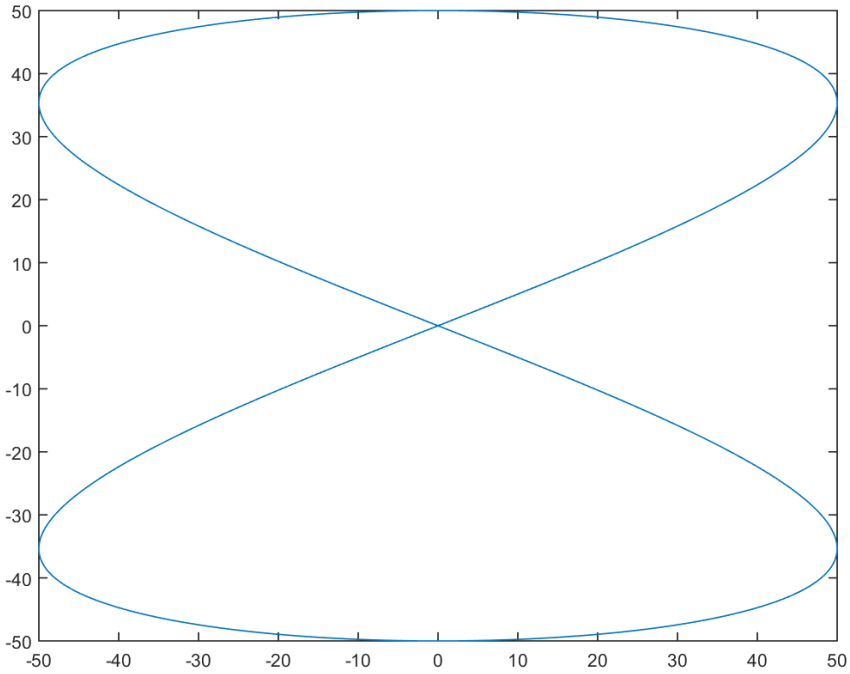
Zgodnie z przewidywaniami przebieg kąta mieści się między dwiema krzywymi o ciągle narastającym module. Dla dłuższej symulacji program wyrzuca komunikat błędu związany z przekroczeniem maksymalnego kąta. Dzieje się tak, gdyż krzywa ta zmierza od krzywej bliskiej kształtem okręgowi do krzywej bliskiej kształtem prostej – brak możliwości zakrętu na końcach.

3.2.7 Krzywa Lissajous o współczynnikach o stosunku 2

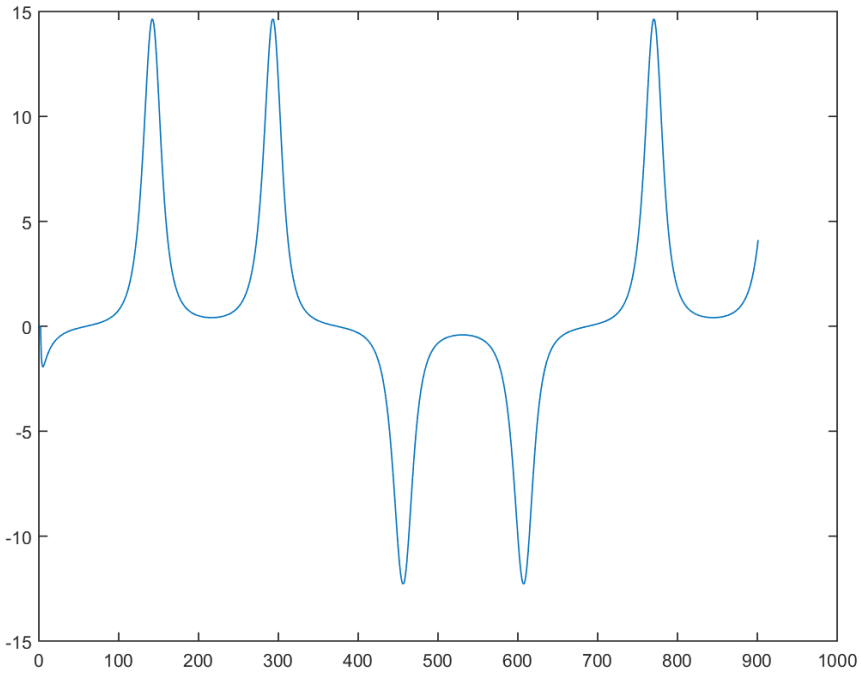
Trasa środka ciężkości jest zgodna z trasą zadaną. Ścieżki różnią się. Przyczyną różnicy jest stały krok w symulacji.



Rysunek 3.46: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.47: Ścieżka którą podążał pojazd

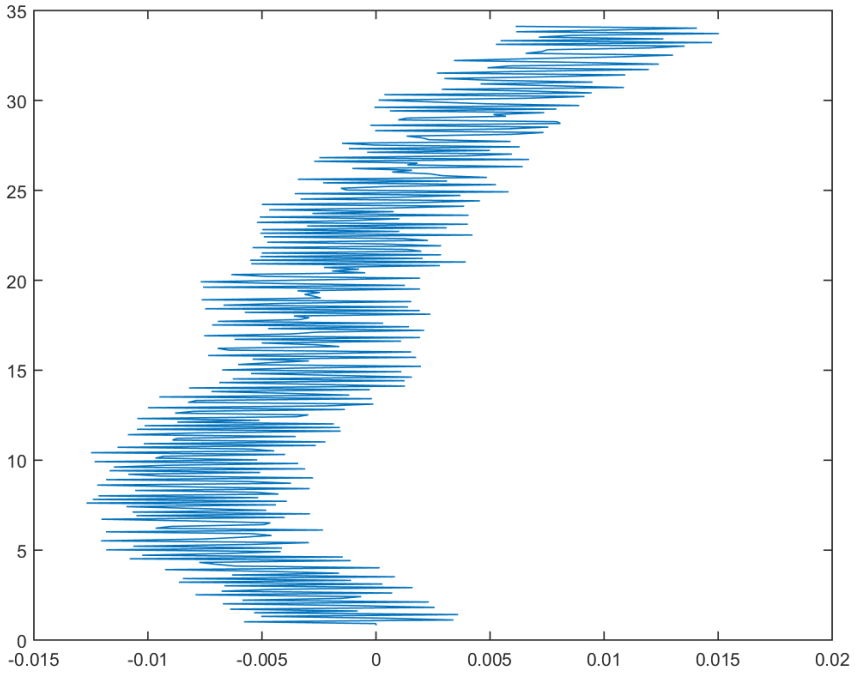


Rysunek 3.48: Kąt wyznaczony przez kinematykę odwrotną

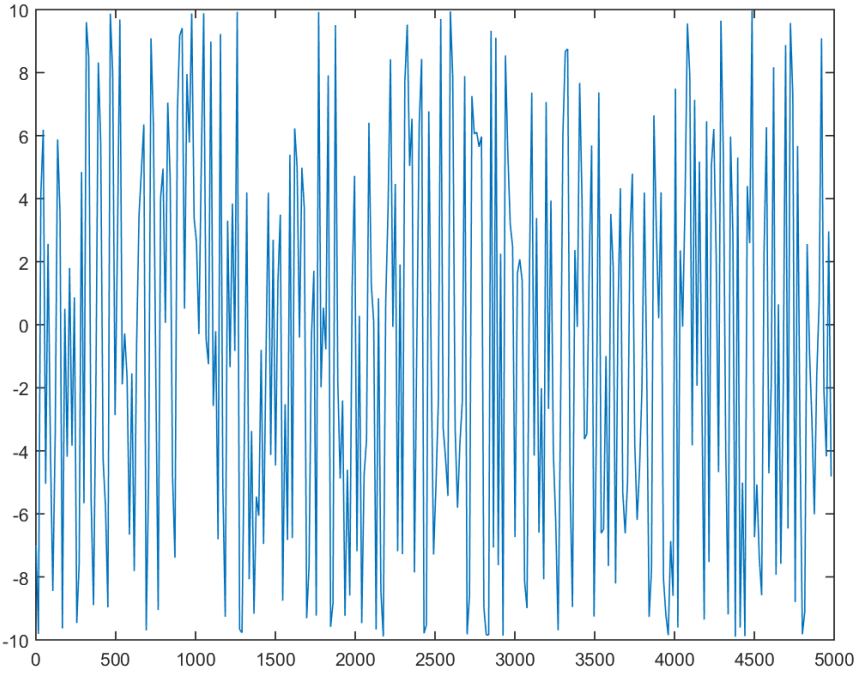
Przebieg kąta zachowuje się zgodnie z przewidywaniami – zmienia się cyklicznie, posiadając po cztery ekstrema na okres – cztery brzegi „klepsydry" zakreślanej przez pojazd. Ponieważ jest to krzywa ciągła, która się nakłada na siebie, wartości kąta nie rozjeżdżają się jak w poprzednim teście.

3.2.8 Jazda po krzywej losowej

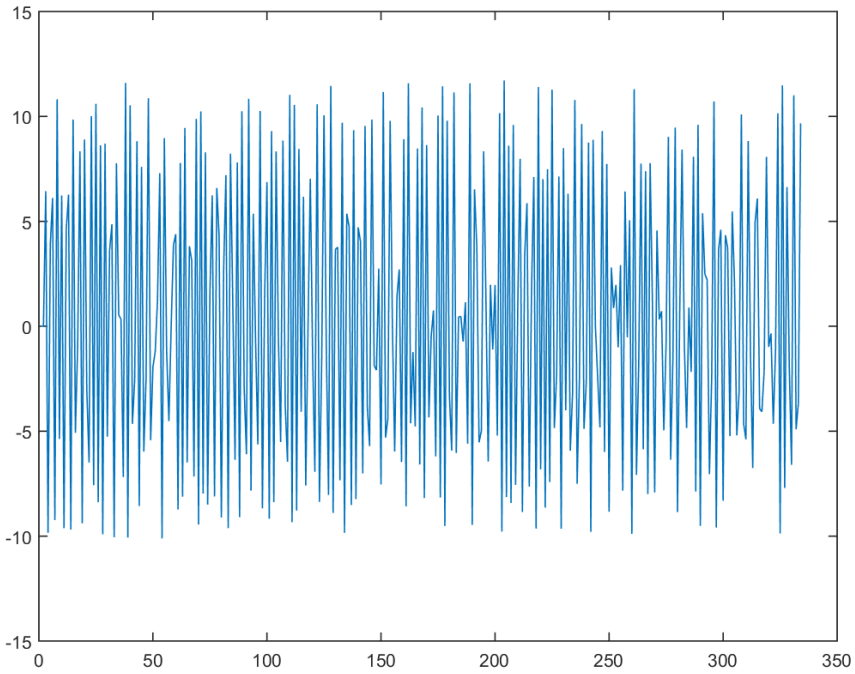
To jest test, który miał za zadanie sprawdzić, czy nasz model jest w stanie poprawnie wychwycić ograniczenia związane ze skrętnością kół, został wywołany dla różnych parametrów. Za każdym razem jak błąd występował – był wykrywany. Poniżej przykładowe przebiegi, gdzie wartość nie wykraczała poza maksymalny kąt.



Rysunek 3.49: Ścieżka wyznaczona przez kinematykę prostą



Rysunek 3.50: Ścieżka którą podążał pojazd



Rysunek 3.51: Kąt wyznaczony przez kinematykę odwrotną

Rozdział 4

Zakończenie

Projekt uważamy za zakończony. Wszystkie jego cele zostały zrealizowane, model kinematyki prostej i odwrotnej pojazdu autonomicznego EVE został wykonany i przetestowany ze względu na swoją poprawność z wynikiem pozytywnym. Testy którym poddaliśmy już w pełni gotowy model pokazały ograniczenia samochodu EVE oraz pozwolił na sprawdzenie czy jest w stanie pokonać trasę z ograniczeniami.

Skrypt testujący wraz ze wszystkimi funkcjami, przebiegami jest załączony na githubie: `https://github.com/tastypenguinbacon/EVE_kinematics-`. Zawarty tam jest też kod źródłowy tego raportu w formacie LaTeX.