Kaczmarek Kacper
MNUM - Sprawozdanie
Zadanie 4.08

Zadanie 1.

Ruch punktu jest opisany równaniami:

$$x_1' = x_2 + x_1 * (0.2 - x_1^2 - x_2^2)$$

$$x_2' = -x_1 + x_2 * (0.2 - x_1^2 - x_2^2)$$

Należy obliczyć przebieg trajektorii ruchu na przedziale [0, 20] dla następujących warunków początkowych:

a) $x_1(0) = 8$, $x_2(0) = 7$

b) $x_1(0) = 0$, $x_2(0) = 0.4$

c) $x_1(0) = 5$, $x_2(0) = 0$

d) $x_1(0) = 0.01$, $x_2(0) = 0.001$

Do rozwiązania należy użyć zaimplementowanych przez siebie metod:
1. Rungego-Kutty czwartego rzędu (RK4) ze stałym krokiem. Proszę przy tym tykonać tyle prób (kilka – kilkanaście), ile będzie potrzebnych do znalezienia takiego kroku, którego zmniejszenie nie wpływa znacząco na rozwiązanie, podczas gdy zwiększenie – już wpływa;
2. Wielokrokowej predyktor – korektor Adamsa czwartego rzędu ze stałym krokiem, który należy dobrać w sposób podany dla metody z punktu 1;
3. Rungego-Kutty czwartego rzędu (RK4) ze zmiennym krokiem. W każdym kroku należy szacować błąd aproksymacji.

Równania różniczkowe

Równania różniczkowe są powszechnie stosowane do modelowania matematycznych układów dynamicznych. Układy równań różniczkowych opisujące rzeczywiste układy dynamiczne są zazwyczaj nieliniowe i z reguły nie są znane metody wyznaczania ich rozwiązań analitycznych i jedynym sposobem na znalezienie rozwiązania są metody numeryczne.

Poszukujemy rozwiązania układu równań postaci

$$\frac{dy_i(x)}{dx} = f_i\big(x, y_1(x), \dots, y_m(x)\big), \qquad i = 1, \dots, m,$$

Na danym odcinku $< a; b >$, przy warunkach początkowych $y_i = y_{ia}, i = 1, \dots, m$.

Metody jednokrokowe

Ogólny wzór określający pojedynczy krok metody jednokrokowej przy stałej długości kroku h można zdefiniować następująco:

$$y_{n+1} = y_n + h\Phi_f(x_n, y_n; h),$$

Gdzie

$$y(x_0) = y_0 = y_a, \qquad x_n = x_0 + nh, \qquad n = 0, 1, \dots,$$

Zaś $\Phi_n(x_n, y_n; h)$ to funkcja definiująca metodę. Jeśli $h \neq 0$, to wzór powyższy można zapisać w postaci

$$\Phi_f(x_n, y_n; h) = \frac{y_{n+1} - y_n}{h},$$

Zdefiniujemy

$$\Delta_f(x_n, y_n; h) = \frac{y(x_n + h) - y(x_n)}{h}$$

Mówimy, że metoda jest zbieżna, gdy

$$h \to 0 \Rightarrow y(x_n; h) \to y(x),$$

tzn, gdy

$$h \to 0 \Rightarrow y_n \to y(x_n), y_{n+1} \to y(x_n + h)$$

co implikuje

$$\Phi_f(x_n, y_n; h) \underset{h \to 0}{\longrightarrow} \Delta_f(x_n, y_n; h)$$

Z kolei mamy

$$\Delta_f(x_n, y_n; h) \underset{h \to 0}{\longrightarrow} y'(x_n) = f(x_n, y_n)$$

Stąd warunkiem aproksymacji (warunkiem zgodności metody z równaniem) nazywamy warunek

$$\Phi_f(x, y; 0) = f(x, y)$$

Błąd aproksymacji

Lokalny błąd metody będziemy oznaczać $r_n(h)$. Jest to błąd powstały w jednym kroku (tj. przy założeniu, że startujemy w tym kroku z dokładnej wartości rozwiązania $y_n = y(x_n)$). Definiujemy go następująca:

$$r_n(h) \overset{\text{def}}{=} y(x_n + h) - [\, y(x_n) + h\Phi_f(x_n, y_n; h)\,]$$

Gdzie $y(x_n + h)$ jest rozwiązaniem, dla $x = x_n + h$ układu równań

$$y' = f(x, y(x))$$

$$y(x_n) = y_n, \qquad x \in [\, x_n, b]$$

Tzn rozwiązaniem przechodzącym przez punkt $y(x_n) = y_n$.

Metody Rungego-Kutty (RK)

Metody te można zdefiniować następującym wzorem:

$$y_{n+1} = y_n + h * \sum_{i=1}^{m} w_i k_i$$

Gdzie

$$k_1 = f(x_n, y_n),$$

$$k_i = f\left(x_n + c_i h, y_n + h * \sum_{j=1}^{i-1} a_{ij} k_j\right), \quad i = 2, 3, \dots, m,$$

Przy czym

$$\sum_{j=1}^{i-1} a_{ij} = c_i, i = 2, 3, \dots, m.$$

Do wykonania jednego kroku metody należy obliczyć wartości prawych stron dokładnie m razy, stąd metody są m-etapowe. Największe znaczenie praktyczne mają metody m = 4 i rzędu 4 - kompromis między dokładnością, a nakładem obliczeń na jedną iterację.

Metoda Rungego-Kutty rzędu 4. (RK4)

$$y_{n+1} = y_n + \frac{1}{6} * h * (k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(x_n, y_n)$$

$$k_2 = f(x_n + \frac{1}{2} * h, y_n + \frac{1}{2} * h * k_1)$$

$$k_3 = f(x_n + \frac{1}{2} * h, y_n + \frac{1}{2} * h * k_2)$$

$$k_4 = f(x_n + h, y_n + h * k_3)$$

Wybór długości kroku

Podstawowym zagadnieniem przy praktycznej implementacji metod rozwiązywania równań różniczkowych jest kwestia doboru długości kroku całkowanie h_n.Przy wyznaczaniu długości kroku występują dwie przeciwstawne sobie zagadnienia:

- Jeśli krok h_n maleje, to maleje błąd metody, dla metody zbieżnej błąd maleje do zera przy h dążącym do zera
- Jeśli krok h_n maleje, to zwiększa się liczba iteracji potrzebnych do wyznaczenia rozwiązania na zadanym odcinku $< a, b >$, a stąd liczba obliczeń i związanych z nimi błędów numerycznych.

Z powyższych punktów wynika, że powinien istnieć optymalny krok, dla którego jednocześnie błędy metody i numeryczne nie będą zbyt duże.

Szacowanie wartości błędu według zasady podwójnego kroku

W celu szacowania błędu, oprócz kroku o długości h wykonujemy dodatkowo równoległe i dokładnie tą samą metodą dwa dodatkowe kroki o długości $\frac{1}{2}h$ każdy.

Wychodząc ze wzorów

$y(x_n + h) = y_n^{(1)} + \frac{r_n^{(p+1)}(0)}{(p+1)!} * h^{p+1} + O(h^{p+2})$ - po kroku pojedynczym

$y(x_n + h) \approx y_n^{(2)} + 2 * \frac{r_n^{(p+1)}(0)}{(p+1)!} * (\frac{h}{2})^{p+1} + O(h^{p+2})$ - po kroku podwójnym

Po przekształceniach dochodzimy do wzoru

$\sigma_n(h) = \frac{2^p}{2^p-1}(y_n^{(2)} - y_n^{(1)}) \overset{p=4}{\longrightarrow} \frac{16}{15}(y_n^{(2)} - y_n^{(1)})$ – oszacowanie błędu po pojedynczym kroku

$\sigma_n\left(2\ x\ \frac{h}{2}\right) = \frac{y_n^{(2)} - y_n^{(1)}}{2^p-1} \overset{p=4}{\longrightarrow} \frac{y_n^{(2)} - y_n^{(1)}}{15}$ – oszacowanie błędu po podwójnym kroku

Gdzie $y_n^{(1)}$ jest nowym punktem uzyskanym w kroku o długości h, $y_n^{(2)}$ jest nowym punktem uzyskanym po dwóch krokach o długości.

Kod programu realizującego RK4 ze stałym krokiem:

```
function [xvalues, errors] = RK4const( x1, x2, h)
xvalues = zeros( 20/h, 2);
x1values = zeros( 20/h, 1);
x2values = zeros( 20/h, 1);
errors = zeros( 20/h, 2);
i = 0;
k = zeros(4,2);
a = 0;

while ( a < 20)
    vectora(i+1) = a;
    x1values(i+1) = x1;
    x2values(i+1) = x2;
    % new points
    k(1,1) = dx1(x1, x2);
    k(1,2) = dx2(x1, x2);

    k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
    k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );

    k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
    k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

    k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
    k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

    tempx1 = x1;
    tempx2 = x2;

    x1 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
    x2 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

    newx1 = x1;
```

```
    newx2 = x2;
    % errors
    % first half-step
    h = 0.5*h;
    x1 = tempx1;
    x2 = tempx2;

    k(1,1) = dx1(x1, x2);
    k(1,2) = dx2(x1, x2);

    k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
    k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );

    k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
    k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

    k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
    k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

    x1 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
    x2 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

    %second half-step

    k(1,1) = dx1(x1, x2);
    k(1,2) = dx2(x1, x2);

    k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
    k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );

    k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
    k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

    k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
    k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

    tempx12 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
    tempx22 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

    errors(i+1,1) = (tempx12 - x1)/15;
    errors(i+1,2) = (tempx22 - x2)/15;
    x1 = newx1;
    x2 = newx2;

    h = 2*h;

    a = a + h;
    i = i+1;
end
plot(x1values, x2values, 'b');
for i=1:1:20/h
    xvalues(i, 1) = x1values(i, 1);
    xvalues(i, 2) = x2values(i, 1);
end
```

Krok wyznaczyłem wpisując kolejne wartości kroku i patrzyć na błędy metody sumując moduły wartości wektora errors. Oto tabelka dla subiektywnie wybranych kroków:

a) ($x_1 = 8, x_2 = 7$)

Krok	Błąd x1	Błąd x2
0.1	NaN	NaN
0.025	NaN	NaN
0.02	0.2945	0.2483
0.01	0.2630	0.3144
0.008	0.3049	0.3416
0.005	0.3618	0.3776
0.002	0.4099	0.4100
0.001	0.4239	0.4201
0.0001	0.4359	0.4291
0.00005	0.4365	0.4295

Dla przykładu a) idealnym krokiem wydaje się krok 0.02.

b) ($x_1 = 0, x_2 = 0.4$)

Krok	Błąd x1	Błąd x2
0.1	0.2075	0.2087
0.025	0.1949	0.1911
0.020	0.1940	0.1898
0.010	0.1919	0.1871
0.008	0.1916	0.1868
0.005	0.1909	0.1859
0.002	0.1903	0.1852
0.001	0.1901	0.1850
0.0001	0.1899	0.1848
0.00005	0.1899	0.1847

Dla przykładu b) idealnym krokiem wydaje się krok 0.0001 lub 0.00005, ale przyjmiemy większą wartość.

c) ($x_1 = 5, x_2 = 0$)

Krok	Błąd x1	Błąd x2
0.1	NaN	NaN
0.025	0.2691	0.2125
0.020	0.2930	0.2130
0.010	0.3251	0.2121
0.008	0.3301	0.2120
0.005	0.3367	0.2116
0.002	0.3429	0.2113
0.001	0.3448	0.2111
0.0001	0.3465	0.2110
0.00005	0.3466	0.2110

Dla przykładu c) idealnym krokiem wydaje się krok 0.025, ponieważ zmiany błędu x2 nie są bardzo znaczenie natomiast x1 zmiennie się znacznie.

d) ($x_1 = 0.01, x_2 = 0.001$)

Krok	Błąd x1	Błąd x2
0.1	0.1420	0.1438
0.025	0.0825	0.0869
0.020	0.0764	0.0809
0.010	0.0625	0.0672
0.008	0.0596	0.0642
0.005	0.0548	0.0594
0.002	0.0499	0.0544
0.001	0.0482	0.0527
0.0001	0.0466	0.0511
0.00005	0.0465	0.0510

Dla przykładu c) idealnym krokiem wydaje się krok 0.0001 lub 0.00005, ale przyjmiemy większą wartość.

Wyznaczanie zmienionej długości kroku

Ogólny wzór na część główną błędu metody rzędu p dla kroku h jest w postaci

$\sigma_n(h) = \gamma * h^{p+1}$, gdzie $\gamma = \frac{r_n^{p+1}(0)}{(p+1)!}$

Jest pierwszym niezerowym współczynnikiem w rozwinięciu błędu aproksymacji w szereg Taylora. Gdy zmieniamy krok z h na αh, to

$$\sigma_n(\alpha h) = \gamma * (\alpha h)^{p+1}$$

skąd

$$\sigma_n(\alpha h) = \alpha^{p+1} * \sigma_n(h)$$

Założenie dokładności obliczeń na wartości ε oznacza

$$|\sigma_n(\alpha h)| = \varepsilon$$

Z tych wzorów uzyskujemy

$$\alpha = \left(\frac{\varepsilon}{|\sigma_n(h)|}\right)^{\frac{1}{p+1}}$$

Na podstawie danego α możemy obliczyć kolejny krok ze wzoru

$$h_{n+1} = s * \alpha * h_n$$

Gdzie s = 0.9

Ponadto parametr dokładności obliczeń ε określa się na ogół następująco:

$$\varepsilon = |y_n| * \varepsilon_w + \varepsilon_b$$

Gdzie

ε_w – dokładność względna

ε_b – dokładność bezwzględna

Dla układu układów równań liczymy α dla każdego z równań i wybieramy najmniejszą z obliczonych α.

Aby za bardzo nie zwiększać kroku wybieramy kolejny krok

$$h_{n+1} = \min(h_{n+1}^{*}, 5h_n, b - x_n)$$

Program rozwiązujący równania różniczkowe RK4 ze zmiennym krokiem:

```
function [xvalues, errors] = RK4variable( x1, x2, h, epsr, epsa)
% x1 - first value of x1
% x2 - first value of x2
% h - step
% epsr - relative epsilon
% epsa - absolute epsilon
i = 0;
k = zeros(4,2);
a = 0;
while ( a < 20 )
    vectora(i+1) = a;
    x1values(i+1) = x1;
    x2values(i+1) = x2;

    % new points
    k(1,1) = dx1(x1, x2);
    k(1,2) = dx2(x1, x2);

    k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
    k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );

    k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
    k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

    k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
    k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

    tempx1 = x1;
    tempx2 = x2;

    x1 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
    x2 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

    newx1 = x1;
    newx2 = x2;
    % errors
    % first half-step
    h = 0.5*h;
    x1 = tempx1;
    x2 = tempx2;

    k(1,1) = dx1(x1, x2);
    k(1,2) = dx2(x1, x2);

    k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
    k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );
```

```
k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

x1 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
x2 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

%second half-step

k(1,1) = dx1(x1, x2);
k(1,2) = dx2(x1, x2);

k(2,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,1)) );
k(2,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(1,2)) );

k(3,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,1)) );
k(3,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(2,2)) );

k(4,1) = dx1( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,1)) );
k(4,2) = dx2( (x1 + 0.5 * h), (x2 + 0.5 * h * k(3,2)) );

tempx12 = x1 + 1/6 * h * ( k(1,1) + 2*k(2,1) + 2*k(3,1) + k(4,1) );
tempx22 = x2 + 1/6 * h * ( k(1,2) + 2*k(2,2) + 2*k(3,2) + k(4,2) );

errors(i+1,1) = (tempx12 - x1)/15;
errors(i+1,2) = (tempx22 - x2)/15;

x1 = newx1;
x2 = newx2;

h = 2*h;

eps(1) = abs(x1) * epsr + epsa;
eps(2) = abs(x2) * epsr + epsa;

alpha1 = ( eps(1) / abs(errors(i+1, 1)*(h^5)) )^(1/5);
alpha2 = ( eps(2) / abs(errors(i+1, 2)*(h^5)) )^(1/5);

alpha = min(alpha1, alpha2);

hnew = 0.9 * alpha * h;

if ( 0.9 * alpha >= 1 )
    if ( a + hnew >= 20 )
        break;
    else
        a = a + h;
        h = min([hnew, 5*h, 20-a]);
        i = i+1;
    end
else
    if ( hnew < h )
        h = hnew;
    else
        error('Cant solve with this epsilon');
```

```
        end
    end
end
plot(x1values, x2values, 'b');
for j = 1:(i-1)
    xvalues(j, 1) = x1values(j);
    xvalues(j, 2) = x2values(j);
end
```

Tabela przedstawiająca porównanie błędów najlepszego kroku dla metody stałego kroku i zmiennego kroku

	X1 stały	X1 zmienny	X2 stały	X2 zmienny
Przykład a	0.2945	0.2056	0.2483	0.2073
Przykład b	0.1899	0.2096	0.1848	0.2104
Przykład c	0.2691	0.2699	0.2125	0.2202
Przykład d	0.0466	0.1833	0.0511	0.1850

Wnioski dotyczące różnicy między metodą korku stałego a zmiennego

Błędy niekoniecznie są mniejsze, kiedy zastosujemy krok zmienny.

Metody wielokrokowe

Ogólna postać wzoru definiującego krok (iterację) metody k-krokowej liniowej ze stałą długością kroku h, jest następująca:

$$y_n = \sum_{j=1}^{k} \alpha_j * y_{n-j} + h * \sum_{j=0}^{k} \beta_j * f(x_{n-j}, y_{n-j})$$

Metoda ta różni się od metody jednokrokowej tym, że kolejne punkty są wyliczane na podstawie poprzednich punktów.

Metody Adamsa

Równanie różniczkowe

$$y'(x) = f\big(x, y(x)\big)$$
$$y(a) = y_a, \qquad a \in [a, b]$$

Równoważne jest równaniu całkowemu

$$y(x) = y(a) + \int_a^x f\big(t, y(t)\big)dt$$

Metody Adamsa dostajemy rozważając to równanie na przedziale $[x_{n-1}, x_n]$

$$y(x_n) = y(x_n - 1) + \int_{x_{n-1}}^{x_n} f\big(t, y(t)\big)dt$$

Metody jawne (Adamsa – Bashfortha)

Funkcję podcałkową f przybliżamy wielomianem interpolacyjnym W(x) stopnia co najwyżej k-1 opartym na węzłach $x_{n-1}, \dots, x_{n-k}$. Przyjmując przybliżenie $y(x_n - 1) \approx y_{n-j}$ i stosując wzór interpolacyjny Lagrange'a mamy

$$f\big(x,y(x)\big) \approx W(x) = \sum_{j=1}^{k} f(x_{n-j}, y_{n-j}) L_j(x)$$

$$y_n = y_{n-1} + \sum_{j=1}^{k} f(x_{n-j}, y_{n-j}) \int_{x_{n-1}}^{x_n} L_j(t)dt$$

Gdzie $L_j(x)$ to wielomiany Lagrange'a,

$$L_j(x) = \prod_{m=1, m\neq j}^{k} \frac{x - x_{n-m}}{x_{n-j} - x_{n-m}}$$

Stąd po scałkowaniu przy założeniu $x_{n-j} = x_n - jh$ dla j = 1, 2, … k otrzymamy

$$y_n = y_{n-1} + h\sum_{j=1}^{k} \beta_j f(x_{n-j}, y_{n-j})$$

Gdzie wartości dla kolejnych β odczytujemy z tabeli podanej w książce.

Metody niejawne (Adamsa-Moultona)

Funkcję podcałkową przybliżamy wielomianem interpolacyjnym stopnia co najwyżej k opartym na węzłach $x_n, x_{n-1}, \dots, x_{n-k}$ z wartościami rozwiązania $y\big(x_{n-j}\big) \approx y_{n-j}$. Następnie postępując tak jak w przypadku poprzednim metod Adamsa-Bashfortha otrzymamy

$$y_n = y_{n-1} + h\sum_{j=0}^{k} \beta_j^* f(x_{n-j}, y_{n-j}) = y_{n-1} + h * \beta_0^* * f(x_n, y_n) + h\sum_{j=1}^{k} \beta_j^* f(x_{n-j}, y_{n-j})$$

Wartości dla parametrów β_j^* odczytamy z tabeli w książce.

Metody predyktor- korektor

Metoda ta wzięła się z określania wymagań idealnej metody. Zawiera ona zalety metody jawnej i również zalety metody niejawnej
Zalety metody niejawnej:
1. wysoki rząd i mała stała obliczeń
2. możliwie duży obszar absolutnej stabilności
Zaleta metody jawnej to:
1. możliwie mała liczba obliczeń na iteracje
Po połączeniu obydwu metod otrzymujemy tzw. predyktor-korektor
W kolejnych iteracjach wykonuje ona działania w skrócie nazwane PEKE

P: $y_n^{[0]} = \sum_{j=1}^{k} \alpha_j * y_{n-j} + h * \sum_{j=0}^{k} \beta_j * f_{n-j}$ (krok metody jawnej) – predykcja
E: $f_n^{[0]} = f(x_n, y_n^{[0]})$ - ewaluacja
K: $y_n = \sum_{j=1}^{k} \alpha_j^* y_{n-j} + h * \beta_0^* * f_n^{[0]} + h\sum_{j=1}^{k} \beta_j^* f_n^{[0]}$ (krok metody niejawnej) – korekcja
E: $f_n = f(x_n, y_n)$ - ewaluacja

Dla metod Adamsa algorytm PK ma postać:

P: $y_n^{[0]} = y_{n-1} + h * \sum_{j=0}^{k} \beta_j * f_{n-j}$ (krok metody jawnej) – predykcja
E: $f_n^{[0]} = f(x_n, y_n^{[0]})$ - ewaluacja
K: $y_n = y_{n-1} + h * \beta_0^* * f_n^{[0]} + h\sum_{j=1}^{k} \beta_j^* f_n^{[0]}$ (krok metody niejawnej) – korekcja
E: $f_n = f(x_n, y_n)$ - ewaluacja

Liczenie błędu aproksymacji

Błędem aproksymacji nazywamy różnice

$$r_n(h) \stackrel{\text{def}}{=} \left[\sum_{j=1}^{k} \alpha_j y(x_{n-j}) + h\sum_{j=0}^{k} \beta_j f\left(x_{n-j}, y(x_{n-j})\right)\right] - y\,(x_n)$$

Jest to dokładnie błąd metody, który informuje o rtm, jaki błąd wnosi metoda w danym kroku (n-tym), a więc po przyjęciu w równaniu metody (wyrażenie w nawiasie kwadratowym) punktów dokładnych jest to definicja taka sama jak w przypadku metod jednokrokowych.
Kiedy połączymy równanie metody i równanie błędu dostajemy:

$$y_n - y(x_n) = h\beta_o\left[f(x_n, y_n) - f\left(x_n, y(x_n)\right)\right] + r_n(h)$$

Gdzie $r_n(h)$ wynosi

$$r_n(h) = \; c_{p+1}h^{p+1}y^{(p+1)}(x_n) + O(h^{p+2})$$

Dla metody rzędu p

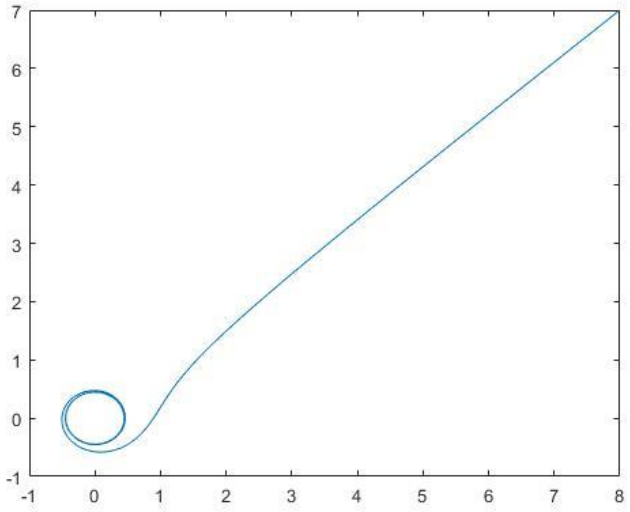
Ogólnie błąd możemy wyliczać z równania

$$y_n - y(x_n) = h * \frac{251}{720} * \left[f(x_n, y_n) - f\left(x_n, y(x_n)\right)\right] + \frac{19}{720}h^5 y^{(5)}(x_n)$$

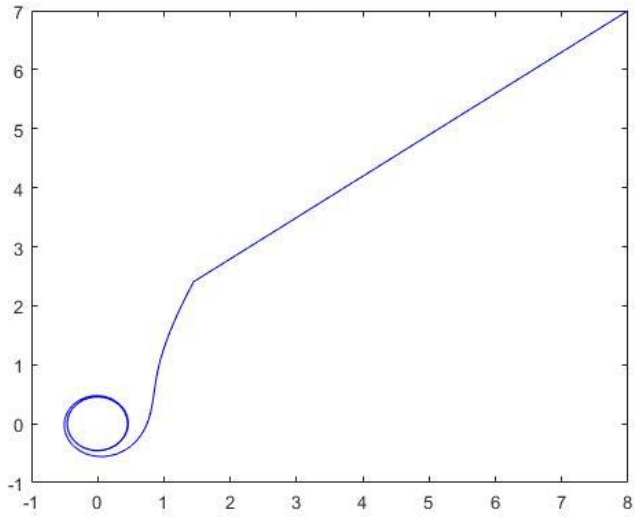
Obliczanie piątej pochodnej jest nieco skomplikowane z tego względu przy wyznaczaniu idealnego kroku dla metody predyktor-korektor posłużyłem się metodą „na oko”.

Przykład a)

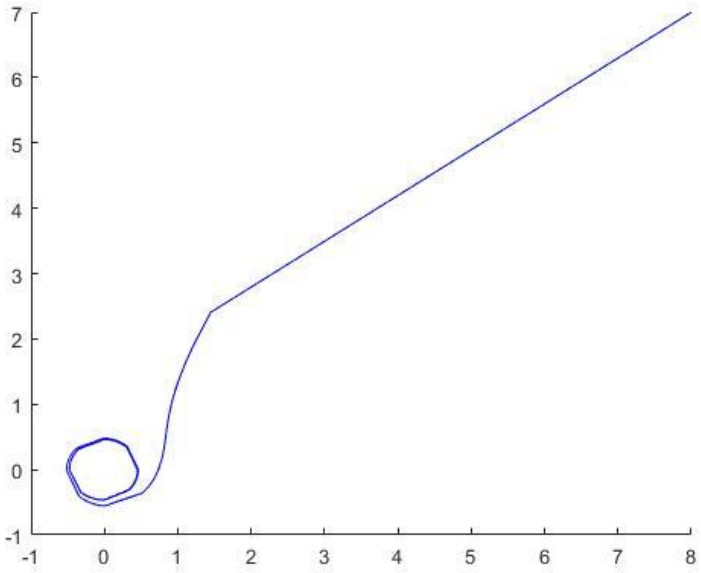
1. Wykres stworzony przed funkcje ode45



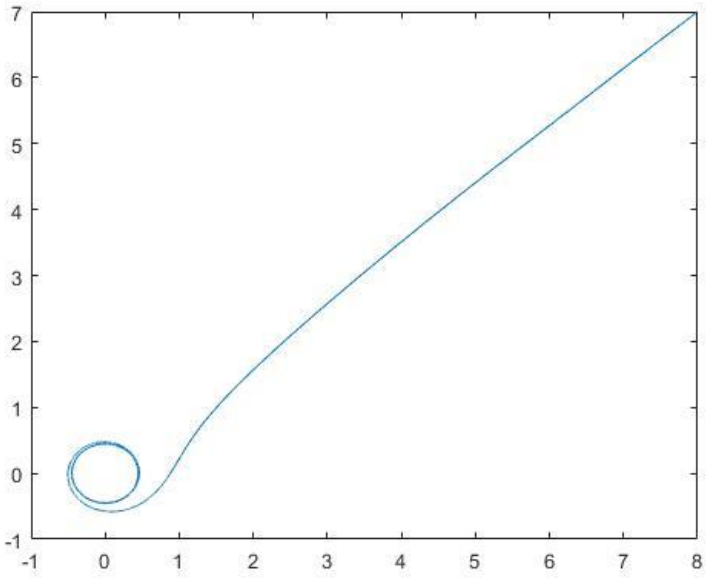
2. Metoda RK4 ze stałym krokiem dla najlepszego kroku h = 0.02



3. Metoda RK4 ze zmiennym krokiem (krok startu h = 0.02 epsilony to 1e-11)

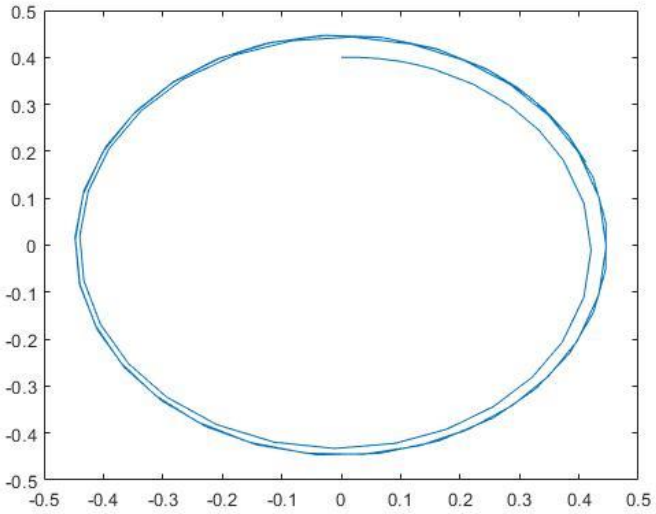


4. Metoda predykor-korektor ze stałym krokiem dla najlepszego kroku h = 0.001

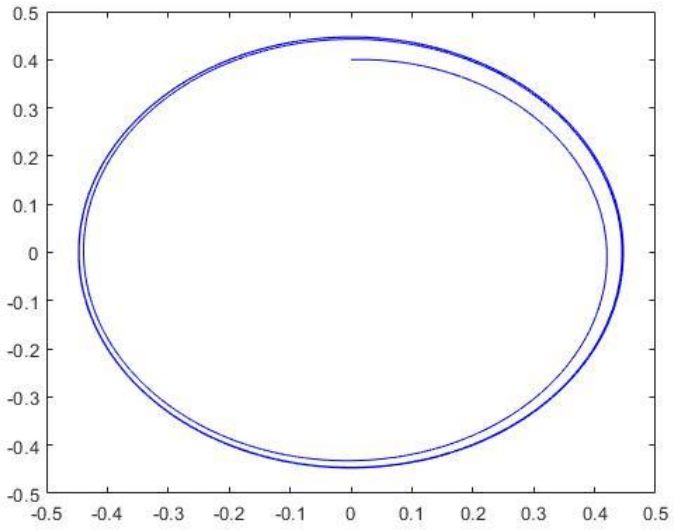


Przykład b)

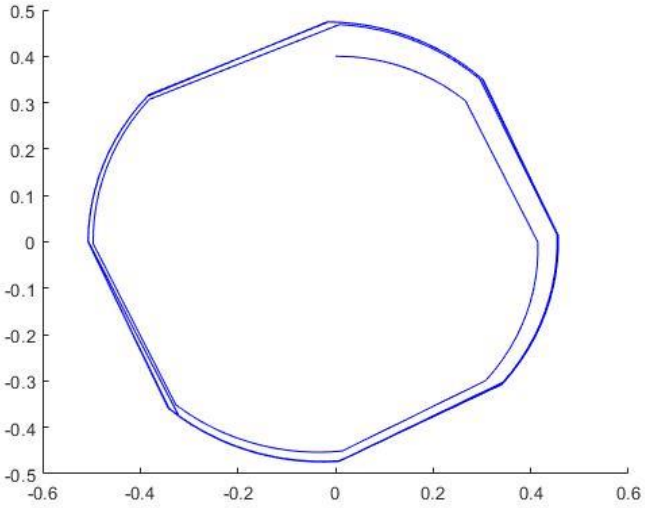
1. Wykres stworzony przed funkcje ode45



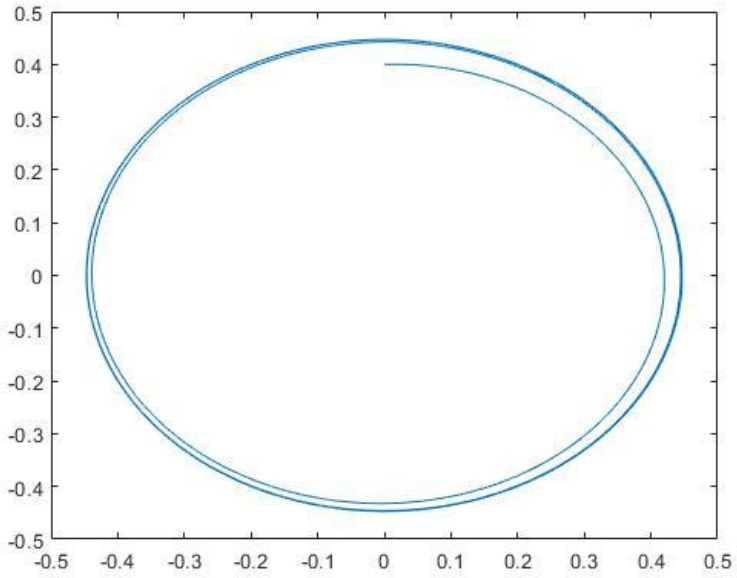
2. Metoda RK4 ze stałym krokiem dla najlepszego kroku h = 0.0001



3. Metoda RK4 ze zmiennym krokiem dla h = 0.0005, obydwa epsilony 10e-9

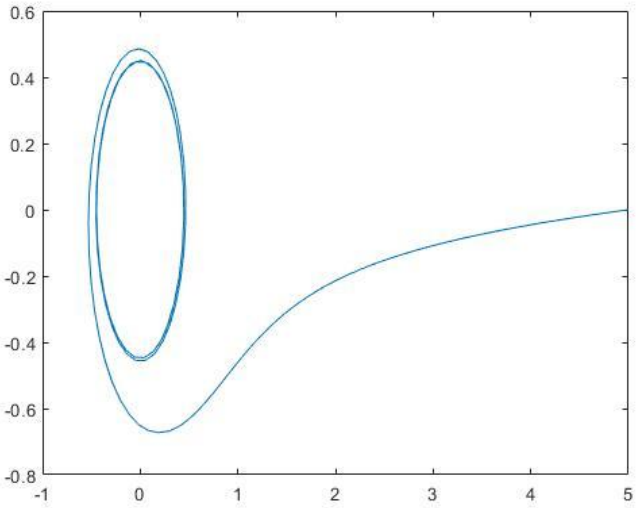


4. Metoda predykor-korektor ze stałym krokiem dla najlepszego kroku h = 0.0001

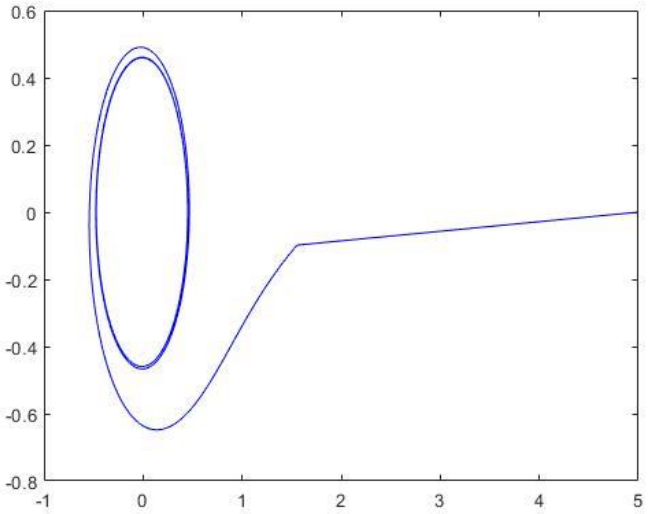


Przykład c)

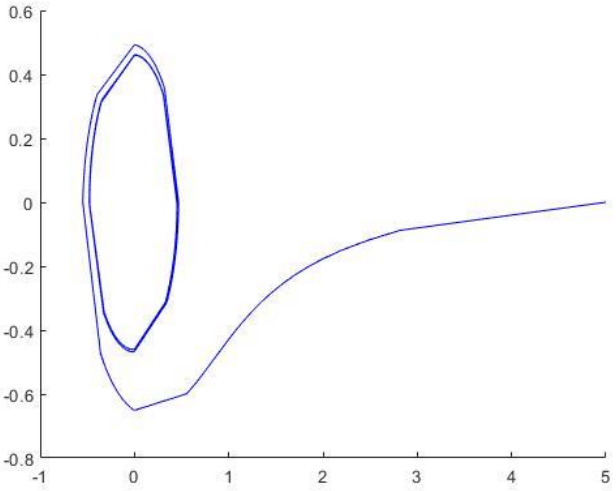
1. Wykres stworzony przed funkcje ode45



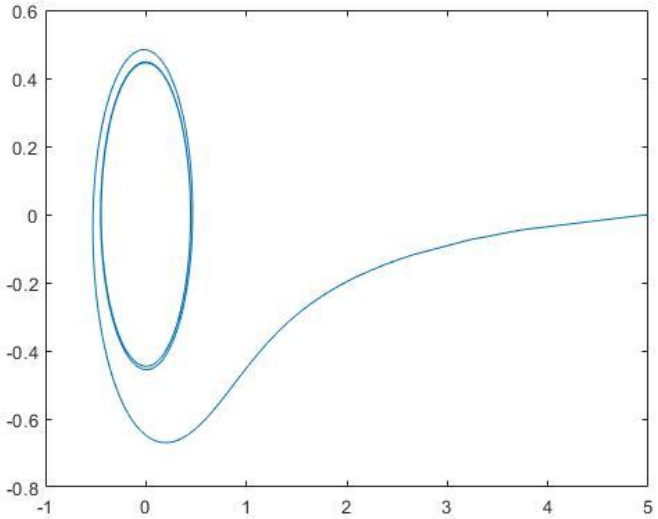
2. Metoda RK4 ze stałym krokiem dla najlepszego kroku h = 0.0001



3. Metoda RK4 ze zmiennym krokiem h = 0.001, epsilony to 1e-11

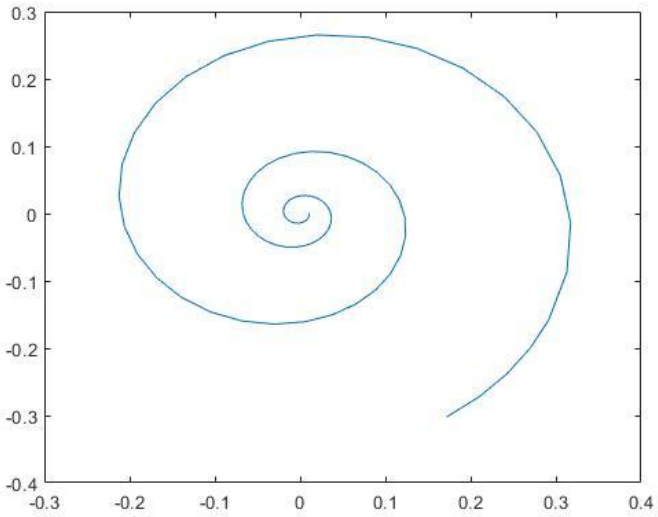


4. Metoda predykor-korektor ze stałym krokiem dla najlepszego kroku h = 0.01

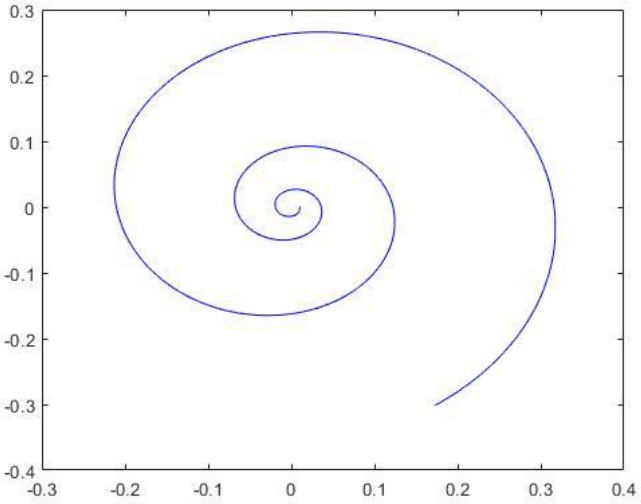


Przykład d)

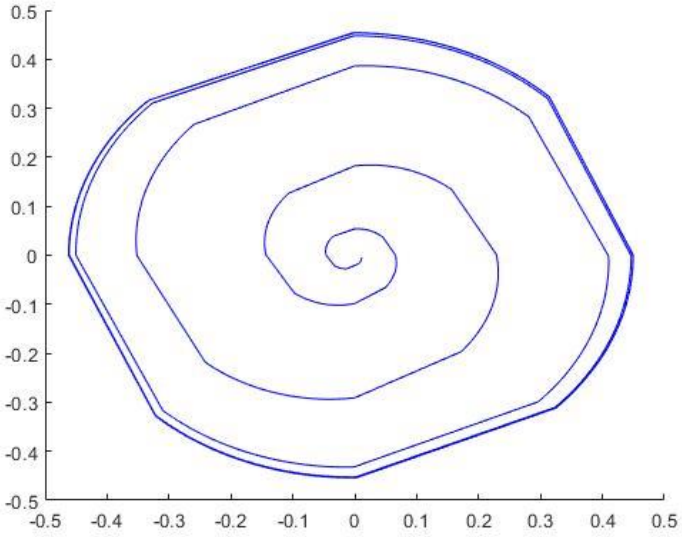
1. Wykres stworzony przed funkcje ode45



2. Metoda RK4 ze stałym krokiem dla najlepszego kroku h = 0.02



3. Metoda RK4 ze zmiennym krokiem h = 0.1, epsilony to 1e-12



4. Metoda predykor-korektor ze stałym krokiem dla najlepszego kroku h = 0.001