Rozwiązywanie układów równań różniczkowych zwyczajnych

Modelowanie Matematyczne 2023/2024 semestr zimowy

Zadanie Projektowe nr 1

Autor: Mikołaj Karbowski
Prowadzący: dr inż. Paweł Mazurek

Politechnika Warszawska,
Wydział Matematyki i Nauk Informacyjnych,
Warszawa, 30 listopada 2023

Oświadczam, że niniejsza praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu Modelowanie Matematyczne została wykonana przeze mnie samodzielnie

Spis treści

Lista symboli matematycznych

δ	- zagregowany błąd względny
$\mathbf{A}$	- macierze współczynników w URRZ
$\mathbf{b}$	- wektory wsółczynników przy funkcji x(t)
$\mathbf{I}$	- macierz jednostkowa
$\mathbf{t}$	- wektor czasu
$\mathbf{y}$	- wektor rozwiązań
h	- krok całkowy

1 Wprowadzenie

Celem projektu jest zastosowanie różnych metod rozwiązywania układów równań różniczkowych zwyczajnych (URRZ). Metody te pozwalają uzyskać przybliżone wartości funkcji, które opisują zmiany zachodzące w czasie. Wybór metody rozwiązania URRZ ma wpływ na dokładność, oraz efektywność uzyskanych wyników. Poświęcimy uwagę czterem metodom rozwiązywania URRZ i przetestujemy je na podanym układzie: (7):

$$\begin{cases} \dfrac{dy_1}{dt} = -\dfrac{8}{3}y_1 + \dfrac{2}{3}y_2 + x(t) \\ \dfrac{dy_2}{dt} = -\dfrac{2}{3}y_1 - \dfrac{13}{3}y_2 + x(t) \end{cases} \tag{1}$$

dla $t \in [0, 8]$, gdzie $x(t) = exp(-t)sin(t)$ oraz zerowych warunków początkowych: $y_1(0) = 0$ oraz $y_2(0) = 0$.

1.1 Przedstawienie zadania

Układ równań różniczkowych zwyczajnych (URRZ), to zbiór równań różniczkowych zwyczajnych, które opisują zależności między funkcjami jednej lub więcej zmiennych niezależnych (najczęściej czasu). W odróżnieniu od pojedynczego równania różniczkowego, układ równań różniczkowych zwyczajnych składa się z kilku równań, z których każde opisuje zmiany w czasie jednej z funkcji.

Ogólna postać układu równań różniczkowych zwyczajnych pierwszego rzędu może być zapisana jako: (2):

$$\begin{cases} \dfrac{dx_1}{dt} = f_1(x_1, x_2, \ldots, x_n, t) \\ \dfrac{dx_2}{dt} = f_2(x_1, x_2, \ldots, x_n, t) \\ \quad\vdots \\ \dfrac{dx_n}{dt} = f_n(x_1, x_2, \ldots, x_n, t) \end{cases} \tag{2}$$

Gdzie:
t to zmienna zależna(najczęściej czasu)
$x_1, x_2, \ldots, x_n$ to funkcje zależne od t
$\frac{dx_i}{dt}$ to pierwsze pochodne funkcji x_i względem t
$f_1, f_2, \ldots, f_n$ to funkcje określające zależności między t a $x_1, x_2, \ldots, x_n$

Rozwiązywanie URRZ metodami analitycznymi jest zadaniem bardzo czasochłonnym, a czasami nawet niewykonalnym. W większość praktycznych zastosowań wystarczy przybliżone z pewną określoną dokładnością rozwiązanie numeryczne układów równań różniczkowych.

1.2 Przedstawienie użytych metod rozwiązywania URRZ

1.2.1 dsolve

Funkcja dsolve jest elementem pakietu Matlab Symolic Toolbox. Używana jest do rozwiązywania symbolicznych równań różniczkowych. Głównym zadaniem dsolve jest znalezienie rozwiązania równania różniczkowego (lub układu równań) za pomocą metod analitycznych.

Funkcja ta jako argumenty przyjmuje równania symboliczne lub układy równań oraz warunki początkowe. Funkcja próbuje znaleźć ogólne rozwiązanie tego równania.

Dsolve w rzypadku powodzenia zwraca funkcję symboliczną, będącą rozwiązaniem wprowadzonego równania.

1.2.2 ode45

Metoda ode45 jest popularnym narzędziem w pakiecie oprogramowania do rozwiązywania równań różniczkowych zwyczajnych. Jest to skrót od "ordinary differential equations - 4th order, 5th stage,"co odnosi się do jej charakterystyki numerycznej. Metoda ta jest często używana ze względu na swoją skuteczność w obszarze rozwiązywania różnorodnych równań różniczkowych, zarówno prostych, jak i bardziej złożonych.

Główną zaletą ode45 jest to, że jest metodą adaptacyjną, co oznacza, że dostosowuje kroki czasowe w trakcie obliczeń w zależności od zmienności funkcji. W praktyce oznacza to, że może być stosowana do efektywnego rozwiązywania zarówno stabilnych, jak i niestabilnych równań różniczkowych.

Metoda ta opiera się na kombinacji algorytmu Rungego-Kutty (czwartego rzędu) oraz piątego etapu interpolacyjnego, co przekłada się na wysoką dokładność numeryczną. Działa dobrze dla problemów, gdzie istnieje potrzeba precyzyjnego odwzorowania dynamiki układu w czasie.

Ode45 zwraca dwa wyjścia: **t**, czyli wektor czasu, w którym uzyskano rozwiązanie równań różniczkowych, oraz **y** - macierz, której kolumny zawierają przybliżone wartości funkcji dla odpowiadających punktów czasowych.

1.2.3 Zmodyfikowana metoda Eulera

Przybliżona wartość wektora funkcji $\mathbf{y}_n$ w punkcie czasowym t_n opisana jest równaniem

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\mathbf{f}(t_{n-1} + \frac{h}{2}, \mathbf{y}_{n-1} + \frac{h}{2}\mathbf{f}(t_{n-1}, \mathbf{y}_{n-1})) \tag{3}$$

Gdzie h to krok czasowy a $\mathbf{y}_{n-1}$ to wartość wektora funkcji w poprzednim punkcie czasowym t_{n-1}.

Jest to zmodyfikowana metoda Eulera, znana również jako metoda punktu środkowego. To szczególny przypadek metody Rungego-Kutty. Metoda ta polega na podziale przedziału czasowego na kroki od długości h. Następnie w sposób iteracyjny wyznaczaniu kolejnych wartości przybliżonego rozwiązania w chwilach t_n.

Metoda punktu środkowego przybliża zmiany wartości funkcji w połowie kroku czasowego, co stanowi korzyść w porównaniu do metody Eulera, która przybliża zmianę na początku kroku.

1.2.4 Metoda dwukrokowa

Przybliżoną wartości funkcji w chwili czasowej t_n opisuje równanie.

$$\mathbf{y}_n = \frac{4\mathbf{y}_{n-1} - \mathbf{y}_{n-2}}{3} + \frac{2h}{3}\mathbf{f}(t_n, \mathbf{y}_n) \tag{4}$$

Metoda ta należy do rodzinym metod wielokrokowych. Do wyznaczenia wartości funkcji w danym kroku, używa ona wartości obliczonych w dwóch poprzednich krokach.

Z uwagi na to, że wyrażenie dla $\mathbf{y}_n$ nie jest udzielone w sposób jawny, konieczne jest odpowiednie przekształcenie wzoru przed jego implementacją.

Zaimplementowanie tej metody sprowadza się do rozwiązywania układu równań w każdej iteracji. Z racji tego że algorytm korzysta z dwóch poprzednich kroków, a dane są tylko warunki początkowe, wartości w pierwszym kroku należy wyznaczy używając metody jednokrokowej.

1.2.5 Metoda Rungego-Kutty

Wzór opisujący tę metodę przedstawiają równania (5) i (6).

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{3} w_i\mathbf{f}_i \tag{5}$$

gdzie:

$$\mathbf{f}_i = \mathbf{f}\Big(t_{n-1} + c_k h, \mathbf{y}_{n-1} + h\sum_{j=1}^{3} a_{i,j}\mathbf{f}_j\Big) \tag{6}$$

A współczynniki podane są w tablicy Butchera:

$$\begin{array}{c|ccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline & w_1 & w_2 & w_3 \end{array} = \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Współczynniki w tabeli Butchear są dobrane w taki sposób, aby uzyskać jak największą dokładność wyniku.

Metoda ta jest jedną z metod Rungego-Kutty. Jej zaletą jest lepsza dokładność, w porównaniu do poprzednich. Jednak sposób w jaki zdefiniowane są f_1, f_2 i $f3$ powoduje, że implementacja tej metody jest trudniejsza. Podobnie jak w poprzedniej metodzie, implementacja sprowadza się do rozwiązywania układu równań w każdej iteracji, aby wyznaczyć f_1, f_2 i $f3$.

2 Metodyka i wyniki doświadczeń

W tym rozdziale pokażemy, jak w praktyce rozwiązywać układy równań różniczkowych zwyczajnych, na przykładzie podanego wcześniej układu i z zastosowaniem przedstawionych metod.

Podany układ wygląda następująco:

$$\begin{cases} \frac{dy_1}{dt} = -\frac{8}{3}y_1 + \frac{2}{3}y_2 + x(t) \\ \frac{dy_2}{dt} = -\frac{2}{3}y_1 - \frac{13}{3}y_2 + x(t) \end{cases} \tag{7}$$

dla $t \in [0, 8]$, gdzie $x(t) = exp(-t)sin(t)$ oraz zerowych warunków początkowych: $y_1(0) = 0$ oraz $y_2(0) = 0$.

Aby wygodnie operować na podanym układzie w środowisku matlab, przedstawimy go w postaci macierzowej:

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}x \tag{8}$$

gdzie:

$$\mathbf{A} = \begin{bmatrix} -\frac{8}{3} & \frac{2}{3} \\ -\frac{2}{3} & -\frac{13}{3} \end{bmatrix}$$

$$\mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\mathbf{A}$ — macierz współczynników przy zmiennych y_1 oraz y_2,
$\mathbf{y}'$ — wektor pochodnych funkcji y_1 oraz y_2
$\mathbf{y}$ — wektor zmiennych,
$\mathbf{b}$ — wektor współczynników przy funkcji $x(t)$.

Zdefiniujmy funkcję, która ułatwi nam implementację algorytmów:

$$\mathbf{f}(t, \mathbf{y}) = \mathbf{A}\mathbf{y}(t) + \mathbf{b}x(t) \tag{9}$$

2.1 Rozwiązanie analityczne z użyciem dsolve

Zaczniemy od wyznaczenia rozwiązania układu funkcją dsolve. Posłuży nam ono jako rozwiązanie wzorcowe, do którego będziemy mogli porównać wyniki pozostałych metod i zbadać ich dokładność.

Znalezienie rozwiązania analitycznego przy użyciu dsolve jest bardzo proste. Wystarczy zdefiniować równania symbliczne i warunki początkowe, następnie podać je jako argumenty funkcji. Dokładna implementacja została przedstawiona w listingu (1)

2.2 Rozwiązanie numeryczny z użyciem ode45

Użycie wbudowanej funkcji ode45 do rozwiązania URRZ jest równie proste, co użycie dsolve. Jako argumenty musimy podać jej uchwyt do pomocniczej funkcji opisującej podany układ równań. Funkcja ta jako argumenty przyjmuje wektory $\mathbf{t}$ oraz $\mathbf{y}$, a zwraca wektor odpowiadający wartościom pochodnych. Zauważmy że jest to dokładnie ta sama funkcja, co zdefiniowana przez nas wcześniej (9). Kolejne argumenty ode45 to przedział na jakim będziemy wyznaczać rozwiązania i warunki początkowe.

2.3 Rozwiązanie numeryczne zmodyfikowaną metodą Eulera

Iteracyjną implementację przedstawia listing (4). W implementacji wykorzystano funkcję odefun odpowiadającą przedstawionej funkcji (9). Która również została przedstawiona w listingu (2).

2.4 Rozwiązanie numeryczne metodą dwukrokową

Aby móc zaimplementować tę metodę musimy przekształcić podany wzór:

$$\mathbf{y}_n = \frac{4\mathbf{y}_{n-1} - \mathbf{y}_{n-2}}{3} + \frac{2h}{3}\mathbf{f}(t_n, \mathbf{y}_n) \tag{10}$$

Najpierw przemnożymy obie strony przez 3 i rozpiszemy $\mathbf{f}(t_n, \mathbf{y}_n)$ korzystają ze wzoru (9), otrzymujemy następujące równanie:

$$3\mathbf{y_n} = 4\mathbf{y_{n-1}} - \mathbf{y_{n-2}} + 2h\mathbf{A}\mathbf{y_n} + 2h\mathbf{b}x(t) \tag{11}$$

Po przeniesieniu y_n na lewą stronę:

$$(3\mathbf{I} - 2h\mathbf{A})\mathbf{y_n} = 4\mathbf{y_{n-1}} - \mathbf{y_{n-2}} + 2h\mathbf{b}x(t) \tag{12}$$

Wyznaczamy y_n:

$$\mathbf{y_n} = (3\mathbf{I} - 2h\mathbf{A})^{-1}(4\mathbf{y_{n-1}} - \mathbf{y_{n-2}} + 2h\mathbf{b}x(t)) \tag{13}$$

Otrzymujemy w ten sposób wzór jawny y_n. Pozostaje jedynie kwestia skąd wziąć $\mathbf{y_2}$. My posłużymy się przybliżeniem otrzymanym przy użyciu poprzedniej metody. Pełny kod tej metody znajduje się w listingu(5).

2.5 Metoda Rungego-Kutty

Przypomnijmy, że wzór tej metody ma postać:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{3} w_i\mathbf{f}_i \tag{14}$$

gdzie:

$$\mathbf{f}_i = \mathbf{f}\Big(t_{n-1} + c_k h, \mathbf{y}_{n-1} + h\sum_{j=1}^{3} a_{i,j}\mathbf{f}_j\Big) \tag{15}$$

Jako współczynniki przyjmiemy wartości z podanej wcześniej tablicy Butchear:

$$\begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Współczynniki $\mathbf{f}_i$ nie są opisane w sposób jawny, zatem w każdej itreacji będziemy musieli wyznaczać rozwiązanie układ równań liniowych, powstały poprzez przkształcenie wzorów na $\mathbf{f}_1$, $\mathbf{f}_2$ i $\mathbf{f}_3$:

$$\begin{cases} \mathbf{f}_1 = \mathbf{f}\Big(t_{n-1} + c_1 h, \mathbf{y_{n-1}} + h\sum_{j=1}^{3} a_{1,j}\mathbf{f}_j\Big) \\ \mathbf{f}_2 = \mathbf{f}\Big(t_{n-1} + c_2 h, \mathbf{y_{n-1}} + h\sum_{j=1}^{3} a_{2,j}\mathbf{f}_j\Big) \\ \mathbf{f}_3 = \mathbf{f}\Big(t_{n-1} + c_3 h, \mathbf{y_{n-1}} + h\sum_{j=1}^{3} a_{3,j}\mathbf{f}_j\Big) \end{cases} \tag{16}$$

Do przekształceń wykorzystujemy wzór funkcji $\mathbf{f}$ (9):

$$\begin{cases} \mathbf{f}_1 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{1,1}\mathbf{f}_1 + ha_{1,2}\mathbf{f}_2 + ha_{1,3}\mathbf{f}_3\Big) + \mathbf{b}x(t_{n-1} + c_1 h) \\ \mathbf{f}_2 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{2,1}\mathbf{f}_1 + ha_{2,2}\mathbf{f}_2 + ha_{2,3}\mathbf{f}_3\Big) + \mathbf{b}x(t_{n-1} + c_2 h) \\ \mathbf{f}_3 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{3,1}\mathbf{f}_1 + ha_{3,2}\mathbf{f}_2 + ha_{3,3}\mathbf{f}_3\Big) + \mathbf{b}x(t_{n-1} + c_3 h) \end{cases} \tag{17}$$

$$\begin{cases} (\mathbf{I} - ha_{1,1}\mathbf{A})\mathbf{f}_1 + (-ha_{1,2}\mathbf{A})\mathbf{f}_2 + (-ha_{1,3}\mathbf{A})\mathbf{f}_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_1 h) \\ (-ha_{2,1}\mathbf{A})\mathbf{f}_1 + (\mathbf{I} - ha_{2,2}\mathbf{A})\mathbf{f}_2 + (-ha_{2,3}\mathbf{A})\mathbf{f}_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_2 h) \\ (-ha_{3,1}\mathbf{A})\mathbf{f}_1 + (-ha_{3,2}\mathbf{A})\mathbf{f}_2 + (\mathbf{I} - ha_{3,3}\mathbf{A})\mathbf{f}_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_3 h) \end{cases} \tag{18}$$

Otrzymany układ przekształcamy do postaci macierzowej wprowadzając oznaczenia (19).

$$\mathbf{Lg} = \mathbf{p} \tag{19}$$

$$\begin{aligned} \mathbf{L} &= \begin{bmatrix} (\mathbf{I} - ha_{1,1}\mathbf{A}) & (-ha_{1,2}\mathbf{A}) & (-ha_{1,3}\mathbf{A}) \\ (-ha_{2,1}\mathbf{A}) & (\mathbf{I} - ha_{2,2}\mathbf{A}) & (-ha_{2,3}\mathbf{A}) \\ (-ha_{3,1}\mathbf{A}) & (-ha_{3,2}\mathbf{A}) & (\mathbf{I} - ha_{3,3}\mathbf{A}) \end{bmatrix}, \\ \mathbf{g} &= \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix}, \\ \mathbf{p} &= \begin{bmatrix} \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_1 h) \\ \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_2 h) \\ \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_3 h) \end{bmatrix} \end{aligned} \tag{20}$$

L — macierz współczyników przy f_1, f_2 i f_3

g — szukany wektor,

p — wektor reprezentujący prawą stronę równania,

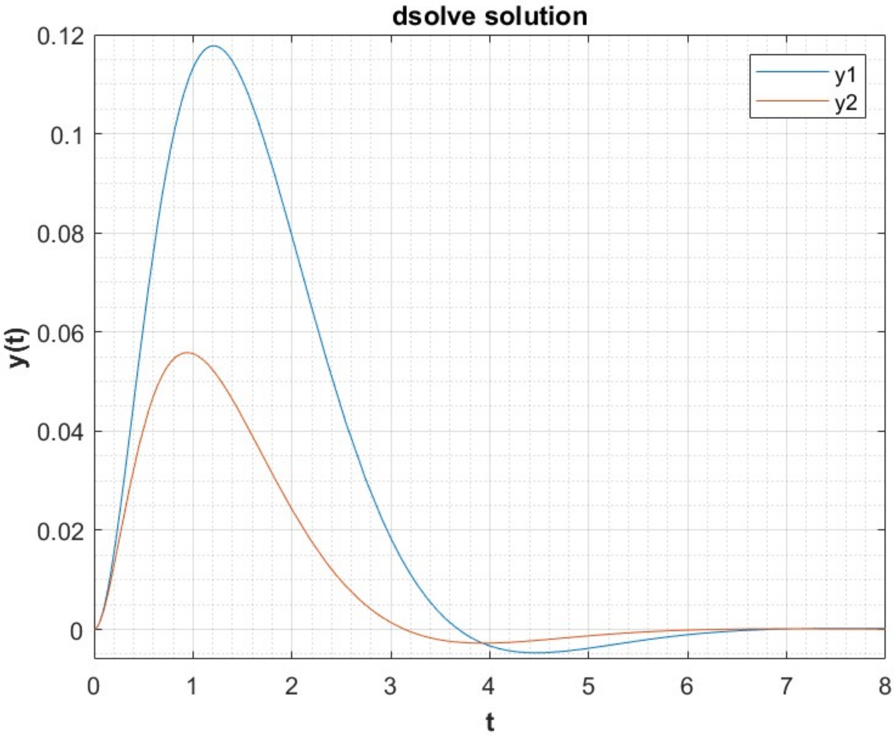
Macierze **A** i **I** są wymiaru 2x2, zatem macierz **L** będzie macierzą 6x6,

Kod odpowiedzialny za rozwiązanie URRZ za pomocą metody Rungego-Kutty, znajduje się w listingu (9)

3 Dyskusja wyników eksperymentów numerycznych

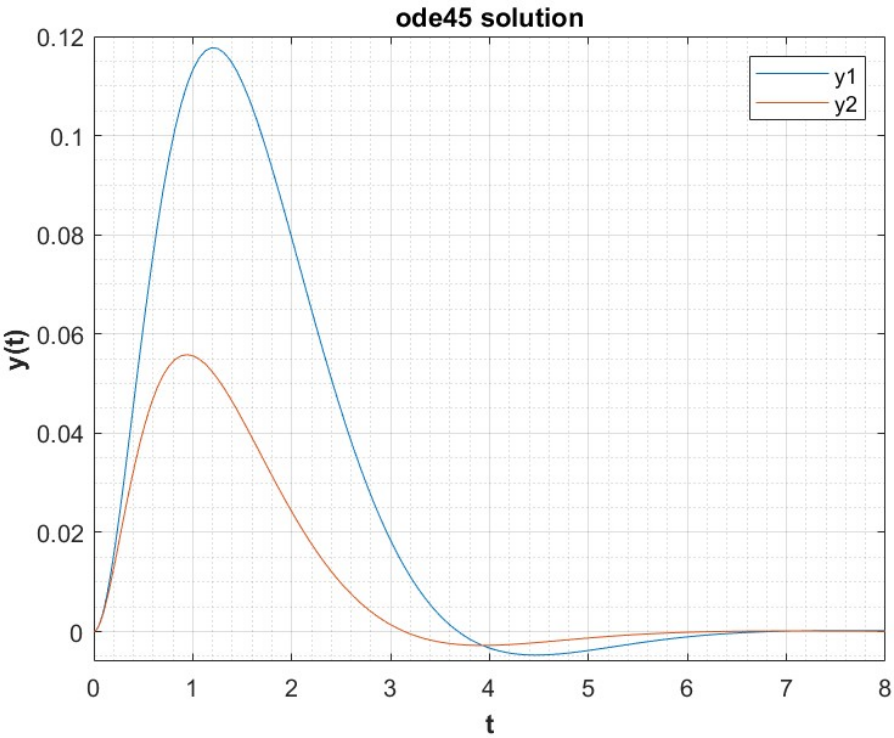
W tym rozdziale przedstawimy wyniki otrzymane dzięki wykorzystaniu wymienionych metod. Porównamy także ich dokładność względem wzorcowego rozwiązania.

Zaimplementowane metody dały nam następujące rezultaty:



Rysunek 1: Rozwiązanie analityczne funkcji dsolve

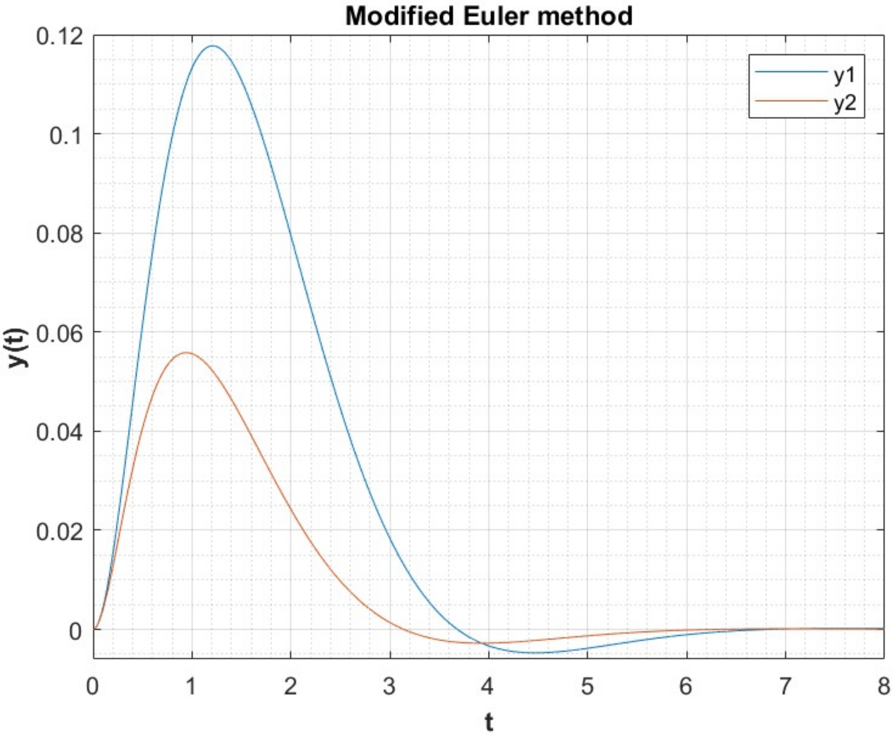
Ten diagram ukazuje precyzyjne rozwiązanie symboliczne podanego układu równań różniczkowych. Będzie ono nam służyć jako rozwiązanie referencyjne.



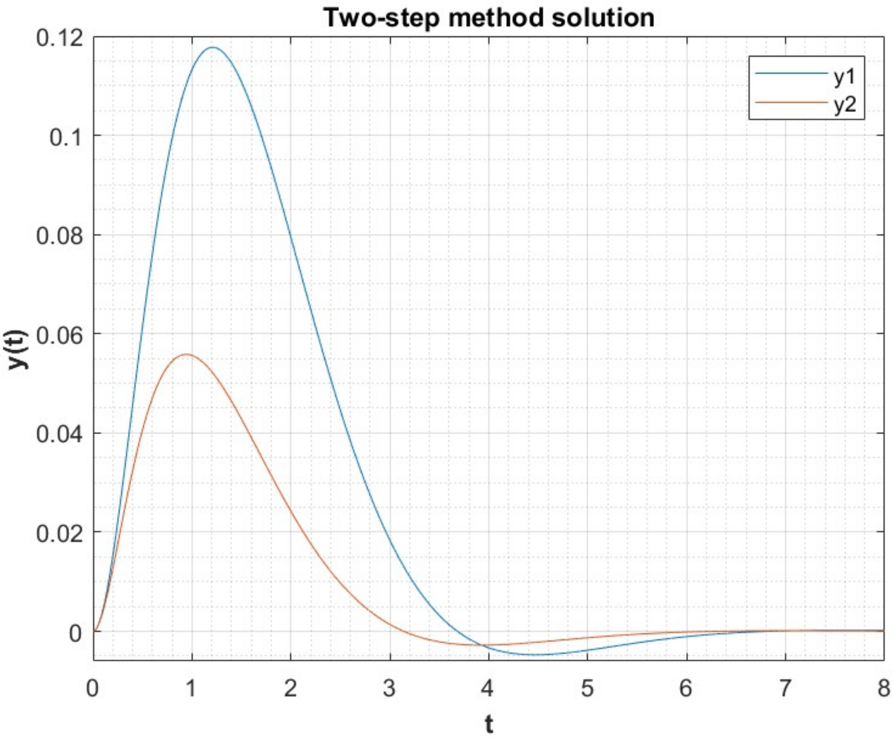
Rysunek 2: Rozwiązania ode45

Patrząc jedynie na wykres nie jesteśmy w stanie znaleźć żadnych odstępstw od rozwiązania wzorcowego.

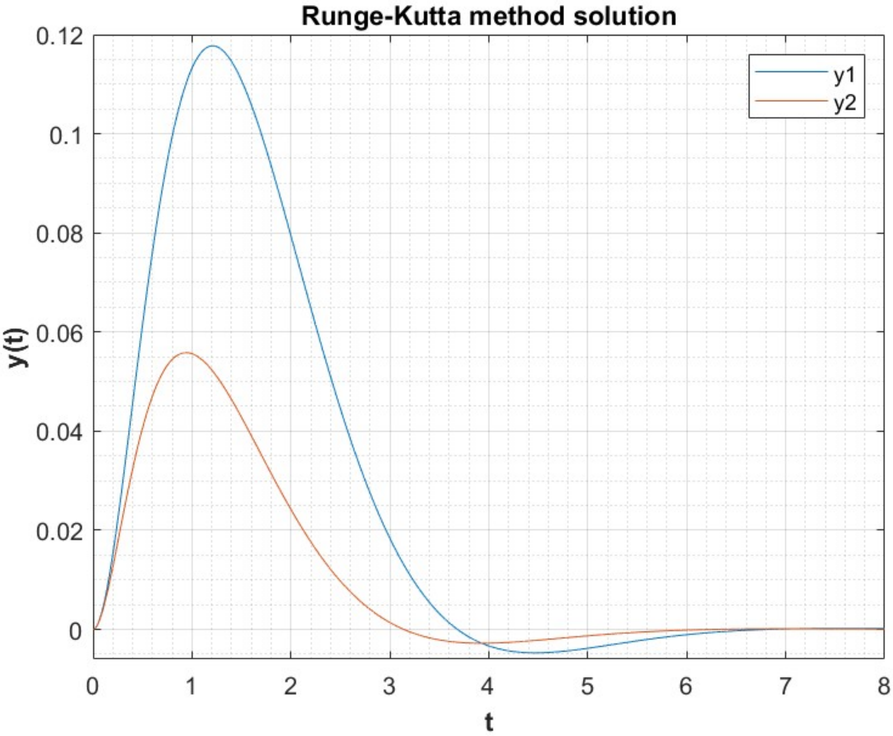
Poniżej przedstawiamy wykresy rezoltatów pozostałych metod. Dla wszystkich tych metod rozmiar kroku całkowania został przyjęty $h = 0.001$.



Rysunek 3: Rozwiązanie numeryczne zmodyfikowaną metodą Eulera



Rysunek 4: Rozwiązanie numeryczne metodą dwukrokową



Rysunek 5: Rozwiązanie numeryczne metodą Rungego-Kutty

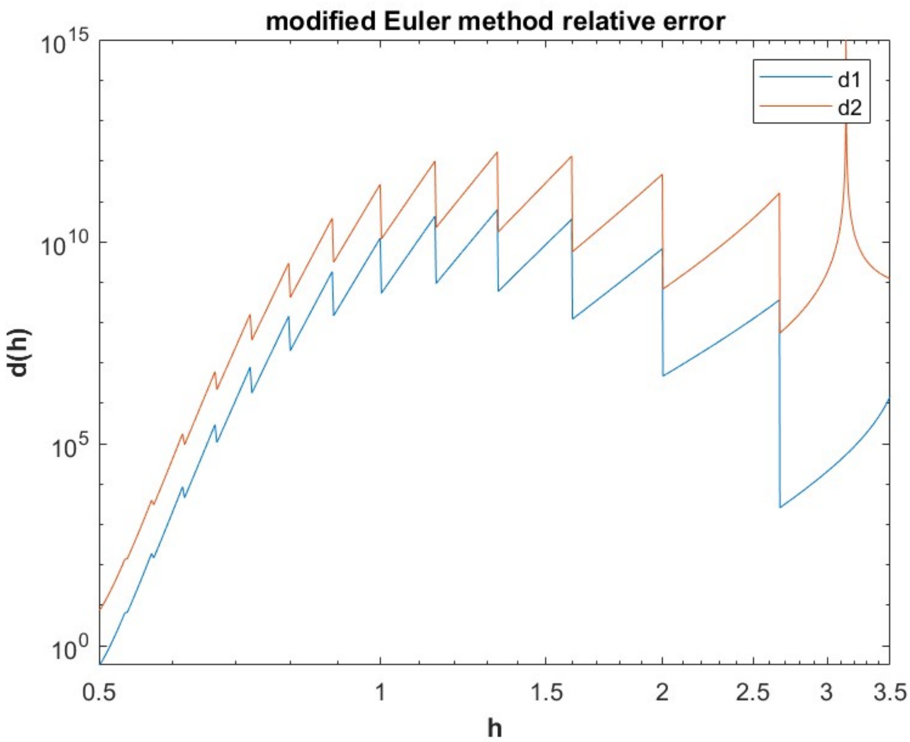
Jak widać dla tak małego kroku, nie jesteśmy w stanie znaleźć żadnej optycznej różnicy względem rozwiązania referencyjnego. Nie jest to jednak wystarczające aby ocenić skuteczność tych metod. W celu określenia jakości tych metod, jako kryterium posłużymy się wartościami zagregowanych błędów względnych. Błędy te określone są następującymi wzorami:

$$\delta_1(h) = \frac{\sum_{i=1}^{N(h)} \left(\hat{y}_1(t_n, h) - \dot{y}_1(t_n)\right)^2}{\sum_{i=1}^{n} (\dot{y}_1(t_n))^2} \tag{21}$$

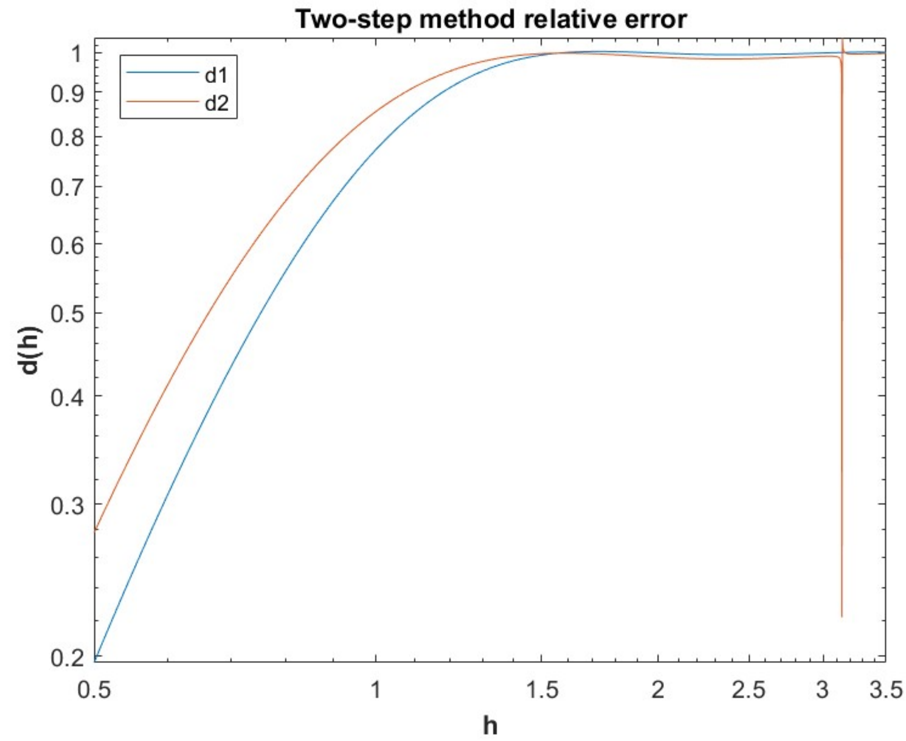
$$\delta_2(h) = \frac{\sum_{i=1}^{N(h)} \left(\hat{y}_2(t_n, h) - \dot{y}_2(t_n)\right)^2}{\sum_{i=1}^{n} (\dot{y}_2(t_n))^2} \tag{22}$$

gdzie $\dot{y}_1(t_n)$ i $\dot{y}_2(t_n)$ to wartości funkcji uzyskane dzięki metodzie dsolve, a $\hat{y}_1(t_n, h)$ i $\hat{y}_2(t_n, h)$ to ich estymaty uzyskane dla kroku całkowanie h. $N(h)$ oznacza zależną od kroku całkowania liczbę punktów rozwiązania.

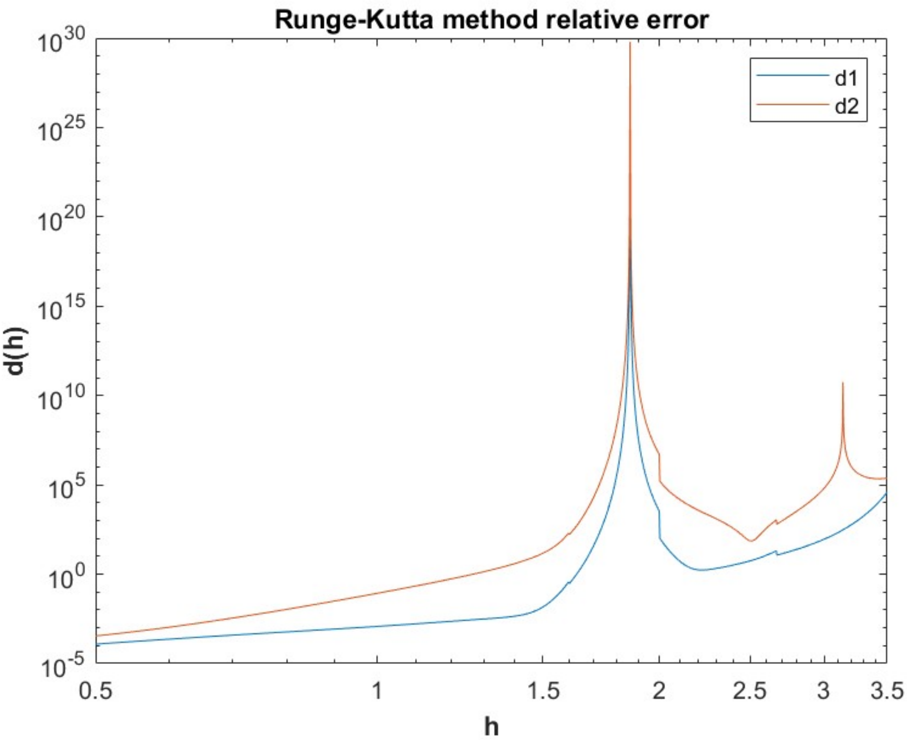
Przedstawimy wykresy wartości błędów w zależności od kroku całkowania $h \in [h_{min}, h_{max}]$, przedział $[h_{min}, h_{max}]$ dobierzemy dla każdej metody w taki sposób, aby dało się zaobserwować zjawisko niestabilności numerycznej.



Rysunek 6: Błąd dla metody Eulera



Rysunek 7: Błąd dla metody dwukrokowej



Rysunek 8: Błąd dla metody Rungego-Kutty

Jak widać na przedziale $[0.5, 3.5]$ obserwujemy zjawisko niestabilności numerycznej dla każdej z metod. Objawia się ono nagłymi i nieprzewidywalnymi zmianami dokładności dla niewielkiej zmiany kroku całkowania.

4 Wnioski

Celem niniejszego projektu było sprawdzenie różnych metod rozwiązywania URRZ i porównanie ich do rozwiązania wzorcowego. Numeryczne metody rozwiązywania równań różniczkowych są bardzo użytecznym narzędziem dla inżynierów, gdyż nie zawsze możliwe jest znalezienie rozwiązania analitycznego. Każda z tych metod ma swoje wady i zalety, a wybranie konkretnej zależy od celu do jakiego chcemy jej użyć.

Dużą zaletą pierwszej metody jest zdecydowanie prostota implementacji i szybkość obliczeń. Dla naszego przykładu zachowuje ona bardzo dobre wartości błędu dla kroku h mniejszego od 0.4, jednak później błąd drastycznie rośnie a metoda staje się niestabilna numerycznie.

Druga metoda w cechuje się bardzo dobrą stabilnością numeryczną. Uzyskujemy takie zachowanie dzięki wyznaczaniu kolejnych przybliżeń na podstawie wartości uzyskanych w dwóch poprzednich krokach a nie jednym, tak jak ma to miejsce w przypadku pozostałych metod.

Ostania metoda jest najdokładniejsza ze wszystkich dla kroku całkowania $h < 1$. Jest zatem najlepszym wyborem jeśli zależy nam na dokładności. Jej wadą jest natomiast, skomplikowana implementacja.

5 Listing

Listing 1: Rozwiązanie dsolve

```
function [Sol] = zad1_dsolve(A)
% Wejscie:
%   A - macierz wspolczynnikow
% Wyjscie:
%   Sol - struktura z rozwiazaniem symbolicznym
%
syms y1(t) y2(t)
x(t) = exp(-t)*sin(t);

% Rownania
eqn1 = diff(y1,t) == A(1,1)*y1 + A(1,2)*y2 + x;
eqn2 = diff(y2,t) == A(2,1)*y1 + A(2,2)*y2 + x;

% Warunki poczatkowe
ic1 = y1(0) == 0;
ic2 = y2(0) == 0;

Sol = dsolve([eqn1,eqn2],[ic1,ic2]);

end % function
```

Listing 2: odefun

```
function [out] = odefun(t,y)
% Wejscie:
% t - wektor kolejnych krokow
% y - wektor wartosci funkcji
% Wyjscie:
% out - wektor wartosci pochodnych
%
A=[-8/3,2/3; -2/3,-13/3];
b=[1;1];

out = A*y + b*exp(-t)*sin(t);
end % function
```

Listing 3: Implementacja ode45

```
function [t_ode,sol_ode] = zad2_1(tspan)
% Wejscie:
%   tspan - przedzial calkowania
```

```
% Wyjscie:
%   t_ode - wektor czasu, w ktorym zostaly obliczone wartosci
   rozwiazania,
%   sold_ode - macierz, gdzie kazda kolumna odpowiada jednemu z
%   elementow rozwiazania rownania rozniczkowego.
%
[t_ode,sol_ode] = ode45(@odefun,tspan,[0,0]);

end % function
```

Listing 4: Implementacja zmodyfikowanej metody Eulera

```
function [t,y] = zad2_2(tspan,h)
% Wejscie:
%   tspan - przedzial calkowania
%   h     - rozmiar kroku calkowania
% Wyjscie:
%   t - wektor czasu, w ktorym zostaly obliczone wartosci
   rozwiazania,
%   y - macierz z rozwiazaniem
%

% podzial przedzialu calkowania
t=tspan(1):h:tspan(end);

y=zeros(2,length(t));
for i=2:length(t)
    y(:,i) = y(:,i-1) + h*odefun(t(i-1)+h/2,y(:,i-1)+h/2*...
        odefun(t(i-1),y(:,i-1)));
end
end % function
```

Listing 5: Implementacja metody dwukrokowej

```
function [t,y] = zad2_3(A,b,x,tspan,h,init_vals)
% Wejscie:
%   A         - macierz wspolczynnikow
%   b         - macierz wspolczynnikow przy funkcji x
%   x         - uchwyt do funkcji
%   tspan     - przedzial calkowania
%   h         - rozmiar kroku calkowania
%   init_vals - macierz wartosci poczatkowych
% Wyjscie:
%   t - wektor czasu, w ktorym zostaly obliczone wartosci
   rozwiazania,
%   y - macierz z rozwiazaniem
%

% podzial przedzialu calkowania
t=tspan(1):h:tspan(end);

y=zeros(2,length(t));
B=3*eye(2) - 2*h*A;

% ustawienie wartosci poczatkowych dla metody z podwojnym krokiem
y(:,1:2) = init_vals;

for i=3:length(t)
    y(:,i) = linsolve(B, (4*y(:,i-1)-y(:,i-2)+2*h*b*x(t(i))));
end

end % function
```

Listing 6: Implementacja metody Rungego-Kutty

```
function [t,y] = zad2_4(A,b,x,tspan,h)
% Wejscie:
%   A         - macierz wspolczynnikow
%   b         - macierz wspolczynnikow przy funkcji x
%   x         - uchwyt do funkcji
%   tspan     - przedzial calkowania
%   h         - rozmiar kroku calkowania
% Wyjscie:
%   t - wektor czasu, w ktorym zostaly obliczone wartosci
  rozwiazania,
%   y - macierz z rozwiazaniem
%

% podzial przedzialu calkowania
t = tspan(1):h:tspan(end);
y = zeros(2,length(t));

I = eye(2); % macierz jednostkowa

% wspolczynniki z tabeli Butchera
a = [1/6, -1/6, 0; 1/6, 1/3, 0; 1/6, 5/6, 0];
c = [0; 1/2; 1];
w = [1/6, 2/3, 1/6];

% macierz wpolczynnikow przy f1, f2 i f3
L = [I-h*a(1,1), -h*a(1,1)*A, -h*a(1,3)*A;...
    -h*a(2,1)*A, I-h*a(2,2)*A, -h*a(2,3)*A;...
    -h*a(3,1)*A, -h*a(3,2)*A, I-h*a(3,3)*A];


for i=2:length(t)
    % macierz wyrazow wolnych
    p = [A*y(:,i-1) + b*x(t(i-1)+c(1)*h); ...
        A*y(:,i-1) + b*x(t(i-1)+c(2)*h); ...
        A*y(:,i-1) + b*x(t(i-1)+c(3)*h)];
    f = linsolve(L,p);
    s = zeros(2,1);
    for j=1:3
        s = s+w(j)*f(j*2-1:j*2);
    end
    y(:,i) = y(:,i-1) + h*s;
end

end % function
```

Listing 7: Sprawdzenie dokładności metody Eulera

```
function [delta] = zad3_1(tspan,h,fun)
% Wejscie:
%   tspan - przedzial calkowania
%   h     - krok calkowania
%   fun   - uchwyt do funkcji wzorcowej
% Wyjscie:
%   delta - macierz z zagregowanymi bledami wzglednymi
%
delta=zeros(2,length(h));
for i=1:length(h)
    [t,y] = zad2_2(tspan,h(i));
    y_dok = fun(t);
    delta(:,i) = sum((y - y_dok).^2,2) ./ sum(y_dok.^2,2);
```

```
end

end % function
```

Listing 8: Sprawdzenie dokładności metody dwukrokowej

```
function [delta] = zad3_2(A,b,x,tspan,h,init,fun)
% Wejscie:
%   A     - macierz wspolczynnikow
%   b     - macierz wspolczynnikow przy x
%   x     - uchwyt do funkcji
%   tspan - przedzial calkowania
%   h     - krok calkowania
%   init  - wartosci poczatkowe
%   fun   - uchwyt do funkcji wzorcowej
% Wyjscie:
%   delta - macierz z zagregowanymi bledami wzglednymi
%
delta=zeros(2,length(h));
for i=1:length(h)
    [t,y] = zad2_3(A,b,x,tspan,h(i),init);
    y_dok = fun(t);
    delta(:,i) = sum((y - y_dok).^2,2) ./ sum(y_dok.^2,2);
end

end % function
```

Listing 9: Sprawdzenie dokładności metody Rungego-Kutty

```
function [delta] = zad3_3(A,b,x,tspan,h,fun)
% Wejscie:
%   A     - macierz wspolczynnikow
%   b     - macierz wspolczynnikow przy x
%   x     - uchwyt do funkcji
%   tspan - przedzial calkowania
%   h     - krok calkowania
%   fun   - uchwyt do funkcji wzorcowej
% Wyjscie:
%   delta - macierz z zagregowanymi bledami wzglednymi
%
delta=zeros(2,length(h));
for i=1:length(h)
    [t,y] = zad2_4(A,b,x,tspan,h(i));
    y_dok = fun(t);
    delta(:,i) = sum((y - y_dok).^2,2) ./ sum(y_dok.^2,2);
end

end % function
```

Źródła

[1] https://www.mathworks.com/help/symbolic/dsolve.html

[2] https://www.mathworks.com/help/matlab/ref/ode45.html

[3] https://pl.wikipedia.org/wiki/Metoda_Eulera

[4] https://pl.wikipedia.org/wiki/Algorytm_Rungego-Kutty

[5] Iwona Wróbel, Notatki do wykładu Metody Numeryczne 2