Wydział Matematyki i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Rozwiązywanie układów równań różniczkowych zwyczajnych

Modelowanie Matematyczne 2023/2024 semestr zimowy

Zadanie Projektowe nr 1

Autor: Adam Grącikowski
Prowadzący: dr inż. Jakub Wagner

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

Warszawa, 30 listopada 2023

Spis treści

Przyjęte oznaczenia i akronimy

$[a, b]$	przedział obustronnie domknięty o końcach a i b
$\delta(h)$	zagregowany błąd względny w funkcji kroku całkowania h
$\frac{dy}{dt}$	pochodna funkcji y po zmiennej t
$\mathbf{A}$	wielkość macierzowa (pogrubiona czcionka)
$\mathbf{b}$	wielkość wektorowa (pogrubiona czcionka)
$\mathbf{I}$	macierz jednostkowa
h	krok całkowania
$N(h)$	zależna od kroku całkowania h liczba punktów rozwiązania
ARE	zagregowany błąd względny (aggregate relative error)
RRZ	równanie różniczkowe zwyczajne
URRZ	układ równań różniczkowych zwyczajnych

1 Wprowadzenie

W niniejszej pracy omówione zostaną wybrane metody numeryczne i algorytmy związane z rozwiązywaniem URRZ oraz możliwości, jakie oferuje w tym temacie środowisko programistyczne MATLAB.

Sposób działania poszczególnych metod i algorytmów zostanie przedstawiony na przykładzie układu równań różniczkowych, który dany jest równaniem (1):

$$\begin{cases} \frac{dy_1}{dt} = -2y_1 - 2y_2 + x \\ \frac{dy_2}{dt} = 2y_1 + 7y_2 + x \end{cases} \tag{1}$$

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

1.1 Zdefiniowanie problemu

Układem równań różniczkowych zwyczajnych (URRZ) nazywamy układ składający się z n, $n > 1$ równań różniczkowych zwyczajnych (RRZ). Ogólną postać układu n równań różniczkowych zwyczajnych przedstawia (2):

$$\begin{cases} \frac{dy_1}{dt} = f_1(y_1, y_2, \dots, y_n, t) \\ \frac{dy_2}{dt} = f_2(y_1, y_2, \dots, y_n, t) \\ \frac{dy_3}{dt} = f_3(y_1, y_2, \dots, y_n, t) \\ \quad \vdots \\ \frac{dy_n}{dt} = f_n(y_1, y_2, \dots, y_n, t) \end{cases} \tag{2}$$

Rozwiązaniem układu (2) są zmienne niezależne, czyli funkcje: $y_1, y_2, \dots, y_n$.

Znalezienie dokładnego rozwiązania URRZ przy pomocy narzędzi analizy analitycznej dla wielu rzeczywistych przypadków okazuje się być zadaniem czasochłonnym lub wręcz niemożliwym, dlatego w praktycznych zastosowaniach inżynierskich wyznacza się przybliżone rozwiązania numeryczne takich układów z pewną określoną dokładnością.

Numeryczne metody rozwiązywania RRZ i URRZ można podzielić na dwie główne kategorie: wielokrokowe metody liniowe oraz metody Rungeggo-Kutty. Dalszy podział obejmuje m.in. metody jawne oraz niejawne.

1.2 Znaczenie problemu

Zagadnienie równań różniczkowych występuje w różnych dziedzinach nauki, takich jak fizyka, chemia, biologia, czy ekonomia i stanowi istotny czynnik warunkujący rozwój tych dziedzin. Układy równań różniczkowych opisują ewolucję wielu zjawisk dynamicznych, począwszy od ruchu planet aż po reakcje chemiczne i procesy biologiczne.

1.3 Opis rozważanych metod numerycznych i algorytmów

1.3.1 Procedura dsolve (MATLAB Symbolic Toolbox)

Funkcja dsolve jest używana do symbolicznego rozwiązywania równań różniczkowych oraz układów takich równań. Rozwiązanie symboliczne polega na obliczeniach wykonywanych na wyrażeniach matematycznych, a nie na liczbach (jak w przypadku standardowego rozwiązania numerycznego), w wyniku czego dostajemy również wyrażenie matematyczne.

Równanie różniczkowe dowolnego rzędu, zapisane przy pomocy zmiennych symbolicznych, można wprowadzić jako argument funkcji dsolve, a ona spróbuje znaleźć ogólne rozwiązanie tego rówania.

Wartością zwracaną funkcji dsolve jest funkcja symboliczna będąca rozwiązaniem wprowadzonego równania symbolicznego, lub struktura zawierającą rozwiązanie w postaci funkcji symbolicznych w przypadku, gdy wprowadzony został układ równań.

1.3.2 Procedura ode45

Jeżeli chcemy rozwiązać URRZ w sposób numeryczny, możemy skorzystać z wbudowanej funkcji środowiska MATLAB, która wykorzystuje parę metod Rungego-Kutty rzędu czwartego oraz piątego.

Funkcja ode45 może być stosowana w sytuacjach, gdzie rozwiązania symboliczne są trudne do uzyskania lub niepraktyczne. Jest bardzo często stosowana w analizie systemów oraz projektowaniu symulacji numerycznych.

Wartością zwracaną funkcji ode45 jest macierz zawierająca umieszczone wierszowo wektory reprezentujące wartości rozwiązania w punktach określonych odpowiednimi elementami wektora kolumnowego **t**, który jest jedną z wartości funkcji ode45.

1.3.3 Udoskonalona metoda Eulera (metoda Heuna)

Udoskonalona metoda Eulera to metoda, która w porównaniu do zwykłej metody Eulera wykorzystuje dodatkowo współczynnik nachylenia stycznej Δy obliczany za pomocą średniej arytmetycznej (3).

$$\Delta y = \frac{h}{2} \cdot \Big[f(t_{n-1}, y_{n-1}) + f(t_{n-1} + h, y_{n-1} + h \cdot f(t_{n-1}, y_{n-1})) \Big] \tag{3}$$

Kolejne kroki tej metody można zinterpretowanć następująco:

1. Obliczamy współczynnik nachylenia w punkcie (t_{n-1}, y_{n-1}): $k_1 = f(t_{n-1}, y_{n-1})$
2. Współczynnik ten wykorzystujemy do oszacowania wartości funkcji w punkcie (t_n, y_n): $k_2 = f(t_{n-1}, y_{n-1} + h \cdot k_1)$
3. Uaktualniamy wartość funkcji na podstawie średniej arytmetycznej współczynników nachylenia: $y_n = y_{n-1} + \frac{h}{2}(k_1 + k_2)$

Średnia arytmetyczna współczynników nachylenia poprawia dokładność metody w porównaniu ze standardową metodą Eulera.

Pełny wzór stosowany w iteracyjnej implementacji przedstawia równanie (4).

$$y_n = y_{n-1} + \frac{h}{2} \cdot \Big[f(t_{n-1}, y_{n-1}) + f(t_{n-1} + h, y_{n-1} + h \cdot f(t_{n-1}, y_{n-1})) \Big] \tag{4}$$

1.3.4 Metoda Adamsa-Bashfortha trzeciego rzędu

Metoda ta jest jedną z niejawnych metod wykorzystywanych w rozwiązywaniu RRZ. Wzór ten pochodzi z rodziny metod wielokrokowych, co oznacza, że wartość y_n jest obliczana na podstawie wartości funkcji w kilku poprzednich krokach czasowych.

Kolejne kroki tej metody można zinterpretowanć następująco:

1. Obliczamy pochodne w trzech punktach czasowych: $f_n = f(t_n, y_n)$, $f_{n-1} = f(t_{n-1}, y_{n-1})$, $f_{n-2} = f(t_{n-2}, y_{n-2})$
2. Używamy obliczonych w pierwszym kroku wartości do oszacowania wartości funkcji w chwili t_n w następujący sposób: $y_n = y_{n-1} + \frac{h}{12} \cdot (5f_n + 8f_{n-1} - f_{n-2})$

Pełny wzór stosowany w iteracyjnej implementacji przedstawia równanie (5).

$$y_n = y_{n-1} + \frac{h}{12} \cdot \Big[5f(t_n, y_n) + 8f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2})) \Big] \tag{5}$$

Zaletą tej metody jest trzeci rząd dokładności, co oznacza, że jest bardziej precyzyjna niż przykładowo metoda Adamsa-Bashfortha drugiego rzędu. Jest ona również jawna, co oznacza, że nie wymaga rozwiązywania równań nieliniowych w trakcie poszczególnych iteracji.

1.3.5 Metoda Rungego-Kutty z trzema etapami

Metoda ta należy do rodziny metod Rungego-Kutty, które są szeroko stosowane w praktyce. Zaletą tej metody jest większa dokładność w porównaniu z metodą Eulera oraz szeroka stosowalność, ponieważ metoda ta sprawdza się zarówno dla równań jednorodnych, jak i niejednorodnych. Jako wadę można uznać nieco bardziej skomplikowaną implementację niż w przypadku metody Eulera oraz konieczność rozwiązywania układu równań przy każdej iteracji.

Pełny wzór stosowany w iteracyjnej implementacji przedstawia równanie (6) oraz (7).

$$y_n = y_{n-1} + h \cdot \sum_{i=1}^{3} w_i f_i \tag{6}$$

gdzie:

$$f_i = f\Big(t_{n-1} + c_i h, y_{n-1} + h \cdot \sum_{j=1}^{3} a_{i,j} f_j\Big) \tag{7}$$

To właśnie sposób zdefiniowania współczynników f_1, f_2 oraz f_3 powoduje, że w każdej iteracji algorytmu musimy rozwiązywać dodatkowy układ równań.

Aby zaimplementować metodę Rungego-Kutty z trzema etapami potrzebujemy tablicy współczynników zawartych w tabeli Butchera (1.3.5):

$$\begin{array}{c|ccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline & w_1 & w_2 & w_3 \end{array}$$

Wartości w tej tabeli decydują o tym, jakie obliczenia są wykonywane na poszczególnych etapach metody Rungego-Kutty. Współczynniki te są dostosowywane w taki sposób, aby uzyskać oczekiwany rząd dokładności. Istnieją różne tabele Butchera dla różnych metod Rungego-Kutty, a ich dobór może wpływać na stabilność, dokładność i efektywność numeryczną metody.

Współczynniki c_i to współczynniki czasowe, określające, w jakich chwilach należy obliczyć f_i. Współczynniki $a_{i,j}$ definiują, jakie wartości f_j należy uwzględnić przy obliczaniu f_i, natomiast b_i to wagi, które wpływają na przyczynę poszczególnych f_i.

2 Metodyka i wyniki doświadczeń

W tym rozdziale zobaczymy w jaki sposób możemy zastosować w praktyce, czyli na przykładzie prostego URRZ (8), metody i algorytmy poznane w rozdziale 1.

Przypomnijmy, że rozważany przez nas przykładowy URRZ ma postać (8)

$$\begin{cases} \dfrac{dy_1}{dt} = -2y_1 - 2y_2 + x \\ \dfrac{dy_2}{dt} = 2y_1 + 7y_2 + x \end{cases} \tag{8}$$

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

Zacznijmy od przekształcenia układu do postaci macierzowej i wprowadzenia odpowiednich oznaczeń:

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{b}x \tag{9}$$

gdzie:

$$\mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} -2 & -2 \\ 2 & -7 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\mathbf{y}'$ — wektor pochodnych funkcji y_1 oraz y_2

$\mathbf{A}$ — macierz współczynników przy zmiennych y_1 oraz y_2,

$\mathbf{y}$ — wektor zmiennych,

$\mathbf{b}$ — wektor pionowy, przez który mnożymy funkcję x.

Dodatkowo, na podstawie wprowadzonych oznaczeń zdefiniujmy funkcję (10), która posłuży nam do zapisania metod iteracyjnych w dalszej części rozdziału.

$$f(t, \mathbf{y}) = \mathbf{A}\mathbf{y}(t) + \mathbf{b}x(t) \tag{10}$$

2.1 Dokładne rozwiązanie przy pomocy dsolve

Aby w dalszej części pracy móc mówić o dokładności poszczególnych metod numerycznych i algorytmów, potrzebujemy dokładnego rozwiązania rozważanego URRZ (8). W tym celu posłużymy się omówioną w 1.3.1 funkcją dsolve z MATLAB Symbolic Toolbox.

Znalezienie dokładnego rozwiązania URRZ przy pomocy dsolve jest bardzo proste - wystarczy zdefiniować odpowiednie zmienne i równania symboliczne, określić warunki początkowe i wywołać funkcję. Pełny kod odpowiedzialny za rozwiązanie URRZ (8) przy pomocy dsolve przedstawia Listing 1.

2.2 Numeryczne rozwiązanie przy pomocy ode45

Aby zaprezentować w praktyce alternatywny, wbudowany w środowisko MATLAB, sposób rozwiązywania URRZ, przedstawimy jeszcze numeryczne rozwiązanie rozważanego URRZ (8) przy pomocy funkcji ode45 omówionej w 1.3.2. Pełny kod odpowiedzialny za rozwiązanie URRZ (8) przy pomocy ode45 przedstawia Listing 2.

2.3 Udoskonalona metoda Eulera (metoda Heuna)

Implementację iteracyjnej wersji udoskonalonej metody Eulera w środowisku MATLAB przedstawia Listing 3. W implementacji wykorzystano zapis URRZ (8) w postaci 9 oraz wzór 4 znany z podrozdziału 1.3.3.

2.4 Metoda Adamsa-Bashfortha trzeciego rzędu

W implementacji tej metody intensywniej skorzystamy z faktu, że rozważany przez na URRZ (8) jest układem liniowym ze względu na zmienne y_1 oraz y_2. W tym celu przekształcimy wzór 5 znany z podrozdziału 1.3.4 do postaci jawnej tak, aby pojawiły się w nim wprowadzone wcześniej w bieżącym rozdziale oznaczenia 9:

$$\mathbf{y_n} = \mathbf{y_{n-1}} + \frac{h}{12} \cdot \Big(5f(t_n, \mathbf{y_n}) + 8f(t_{n-1}, \mathbf{y_{n-1}}) - f(t_{n-2}, \mathbf{y_{n-2}}))\Big) \tag{11}$$

$$\mathbf{y_n} = \mathbf{y_{n-1}} + \frac{h}{12} \cdot \Big(5\mathbf{A}\mathbf{y_n} + 5x(t_n) + 8(\mathbf{A}\mathbf{y_{n-1}} + x(t_{n-1})) - \mathbf{A}\mathbf{y_{n-2}} - x(t_{n-2}))\Big) \tag{12}$$

$$\mathbf{y_n}(\mathbf{I} - \frac{5}{12}h\mathbf{A}) = \mathbf{y_{n-1}} + \frac{h}{12} \cdot \Big(5x(t_n) + 8(\mathbf{A}\mathbf{y_{n-1}} + x(t_{n-1})) - \mathbf{A}\mathbf{y_{n-2}} - x(t_{n-2}))\Big) \tag{13}$$

$$\mathbf{y_n} = (\mathbf{I} - \frac{5}{12}h\mathbf{A})^{-1}\Big[\mathbf{y_{n-1}} + \frac{h}{12} \cdot \Big(5x(t_n) + 8(\mathbf{A}\mathbf{y_{n-1}} + x(t_{n-1})) - \mathbf{A}\mathbf{y_{n-2}} - x(t_{n-2}))\Big)\Big] \tag{14}$$

Ostatecznym wzorem, który bezpośrednio zaimplementujemy jest 14. Dodatkowo, aby wyznaczyć potrzebne wartości w chwili t_2, skorzystamy z jawnej metody Eulera, czyli ze wzoru 15, który po przekształceniu przyjmuje postać 16.

$$\mathbf{y_2} = \mathbf{y_1} + h \cdot f(t_1, \mathbf{y_1}) \tag{15}$$

$$\mathbf{y_2} = h\mathbf{A}\mathbf{y_1} + hx(t_1) \tag{16}$$

Pełny kod odpowiedzialny za rozwiązanie URRZ (8) przy pomocy metody Adamsa-Bashfortha trzeciego rzędu przedstawia Listing 4.

2.5 Metoda Rungego-Kutty z trzema etapami

W implementacji metody Rungego-Kutty z trzema etapami, przyjmiemy współczynniki zebrane w tabeli Butchera:

$$\begin{array}{c|ccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline & w_1 & w_2 & w_3 \end{array} = \begin{array}{c|ccc} 0 & \frac{1}{6} & 0 & -\frac{1}{6} \\ \frac{1}{2} & \frac{1}{12} & \frac{5}{12} & 0 \\ 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

W implementacji tej metody ponownie intensywniej skorzystamy z faktu, że rozważany przez nas URRZ (8) jest układem liniowym ze względu na zmienne y_1 oraz y_2.

Zauważmy, że występujące we wzore 6 współczynniki f_i są zdefiniowane wzorem niejawnym. W każdej itreacji będziemy zatem musieli rozwiązać pewien układ równań liniowych. Układ ten otrzymamy przekształcając wzory na f_1, f_2 oraz f_3:

$$\begin{cases} f_1 = f\Big(t_{n-1} + c_1 h, \mathbf{y_{n-1}} + h \cdot \sum_{j=1}^{3} a_{1,j} f_j\Big) \\ f_2 = f\Big(t_{n-1} + c_2 h, \mathbf{y_{n-1}} + h \cdot \sum_{j=1}^{3} a_{2,j} f_j\Big) \\ f_3 = f\Big(t_{n-1} + c_3 h, \mathbf{y_{n-1}} + h \cdot \sum_{j=1}^{3} a_{3,j} f_j\Big) \end{cases} \tag{17}$$

Do przekształceń wykorzystujemy wzór funkcji f, czyli równanie 10:

$$\begin{cases} f_1 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{1,1}f_1 + ha_{1,2}f_2 + ha_{1,3}f_3\Big) + \mathbf{b}x(t_{n-1} + c_1 h) \\ f_2 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{2,1}f_1 + ha_{2,2}f_2 + ha_{2,3}f_3\Big) + \mathbf{b}x(t_{n-1} + c_2 h) \\ f_3 = \mathbf{A}\Big(\mathbf{y_{n-1}} + ha_{3,1}f_1 + ha_{3,2}f_2 + ha_{3,3}f_3\Big) + \mathbf{b}x(t_{n-1} + c_3 h) \end{cases} \tag{18}$$

$$\begin{cases} (\mathbf{I} - ha_{1,1}\mathbf{A})f_1 + (-ha_{1,2}\mathbf{A})f_2 + (-ha_{1,3}\mathbf{A})f_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_1 h) \\ (-ha_{2,1}\mathbf{A})f_1 + (\mathbf{I} - ha_{2,2}\mathbf{A})f_2 + (-ha_{2,3}\mathbf{A})f_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_2 h) \\ (-ha_{3,1}\mathbf{A})f_1 + (-ha_{3,2}\mathbf{A})f_2 + (\mathbf{I} - ha_{3,3}\mathbf{A})f_3 = \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_3 h) \end{cases} \tag{19}$$

Otrzymany układ przekształcamy do postaci macierzowej wprowadzając oznaczenia 20.

$$\mathbf{Lg} = \mathbf{p} \tag{20}$$

$$\begin{aligned} \mathbf{L} &= \begin{bmatrix} (\mathbf{I} - ha_{1,1}\mathbf{A}) & (-ha_{1,2}\mathbf{A}) & (-ha_{1,3}\mathbf{A}) \\ (-ha_{2,1}\mathbf{A}) & (\mathbf{I} - ha_{2,2}\mathbf{A}) & (-ha_{2,3}\mathbf{A}) \\ (-ha_{3,1}\mathbf{A}) & (-ha_{3,2}\mathbf{A}) & (\mathbf{I} - ha_{3,3}\mathbf{A}) \end{bmatrix}, \\ \mathbf{g} &= \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, \\ \mathbf{p} &= \begin{bmatrix} \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_1 h) \\ \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_2 h) \\ \mathbf{A}\mathbf{y_{n-1}} + \mathbf{b}x(t_{n-1} + c_3 h) \end{bmatrix} \end{aligned} \tag{21}$$

$\mathbf{L}$ — macierz współczyników przy niewiadomych f_1, f_2 i f_3

$\mathbf{g}$ — wektor niewiadomych,

$\mathbf{p}$ — wektor prawych stron,

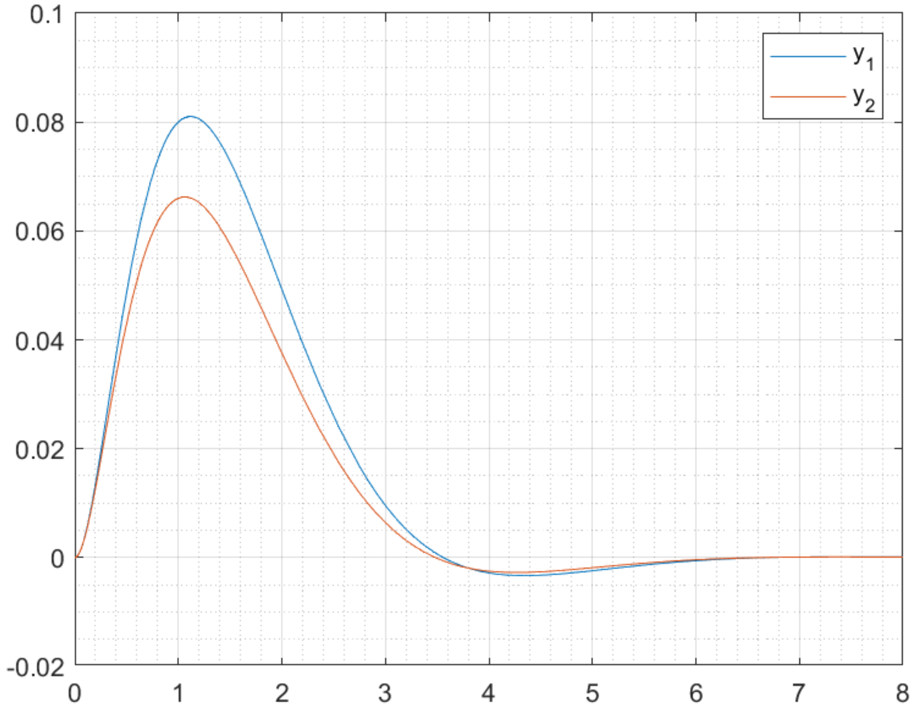
Pamiętajmy, że macierz $\mathbf{L}$ jest macierzą o 6 wierszach i 6 kolumnach, ponieważ macierze $\mathbf{A}$ oraz $\mathbf{I}$ są macierzami kwadratowymi o 2 wierszach i 2 kolumnach. Składnia MATLAB-a pozwala jednak na swobodne łączenie macierzy, więc wszystkie zapisane przez nas równania będzie można bezpośrednio zaimplementować w odpowiedniej funkcji.

Pełny kod odpowiedzialny za rozwiązanie URRZ (8) przy pomocy metody Rungego-Kutty z trzema etapami przedstawia Listing 5.

3 Dyskusja wyników eksperymentów numerycznych

W tej części pracy zaprezentujemy wyniki otrzymane za pomocą każdej z metod omówionych w rozdziale 2 i porównamy dokładność otrzymanych rozwiązań z analitycznym rozwiązaniem wyznaczonym w podrozdziale 2.1 przy pomocy procedury dsolve.

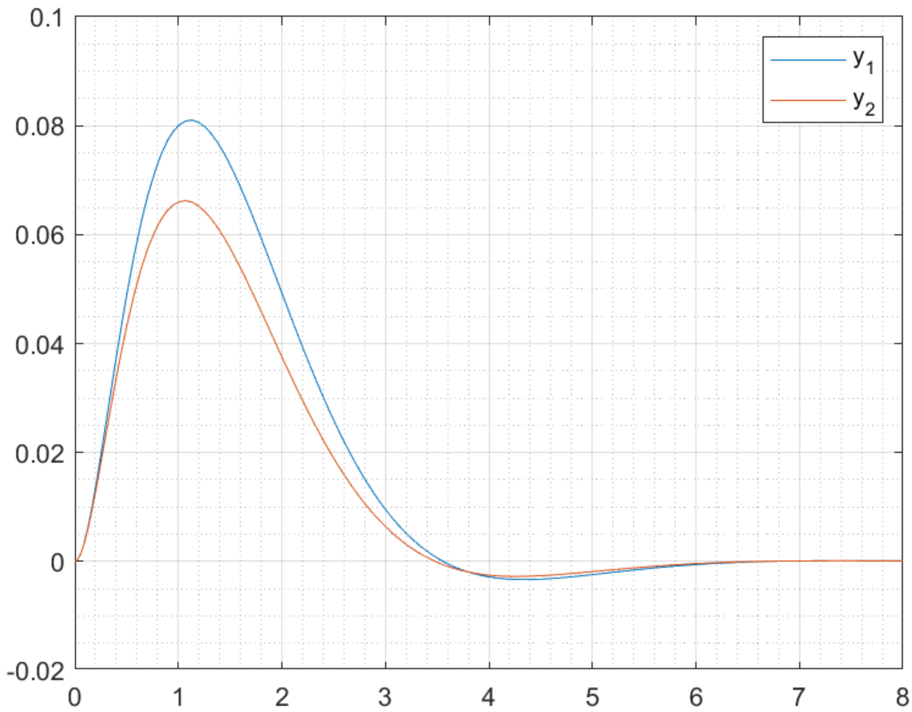
Na początek przedstawimy wykresy otrzymanych rozwiązań rozważanego URRZ (8) dla każdej z zaimplementowanych w rozdziale 2 metod.



Rysunek 1: Dokładne rozwiązanie otrzymane za pomocą funkcji dsolve

Wykres 1 przedstawia dokładne, symboliczne rozwiązanie URRZ (8) otrzymane przy pomocy procedury dsolve.

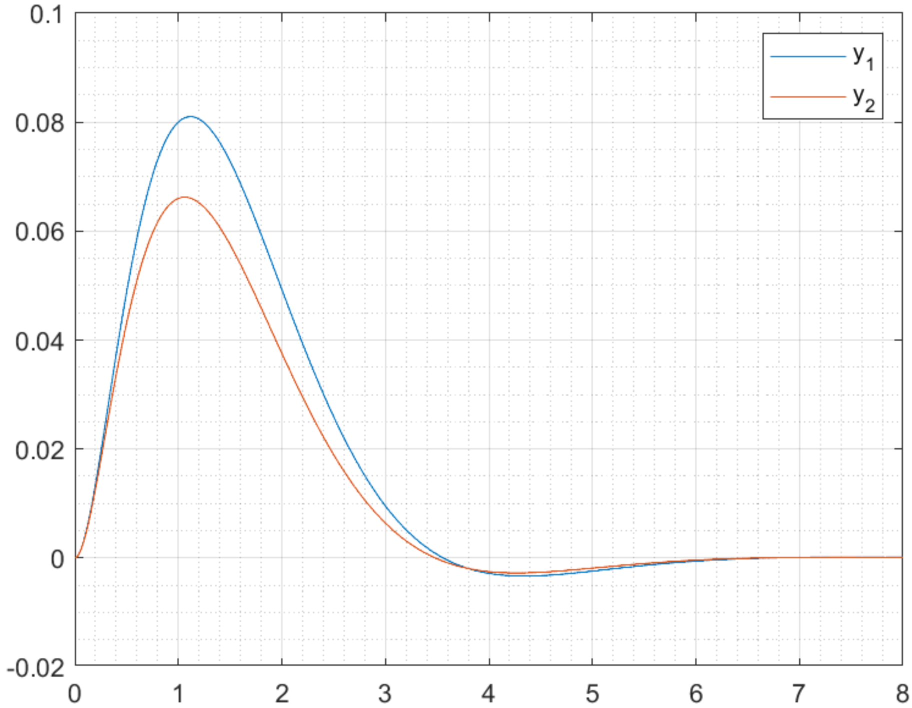
Procedura ode45, której sposób wykorzystania został przedstwiony w 2.2 zwraca numeryczne rozwiązanie URRZ (8) przedstawione na wykresie 2.



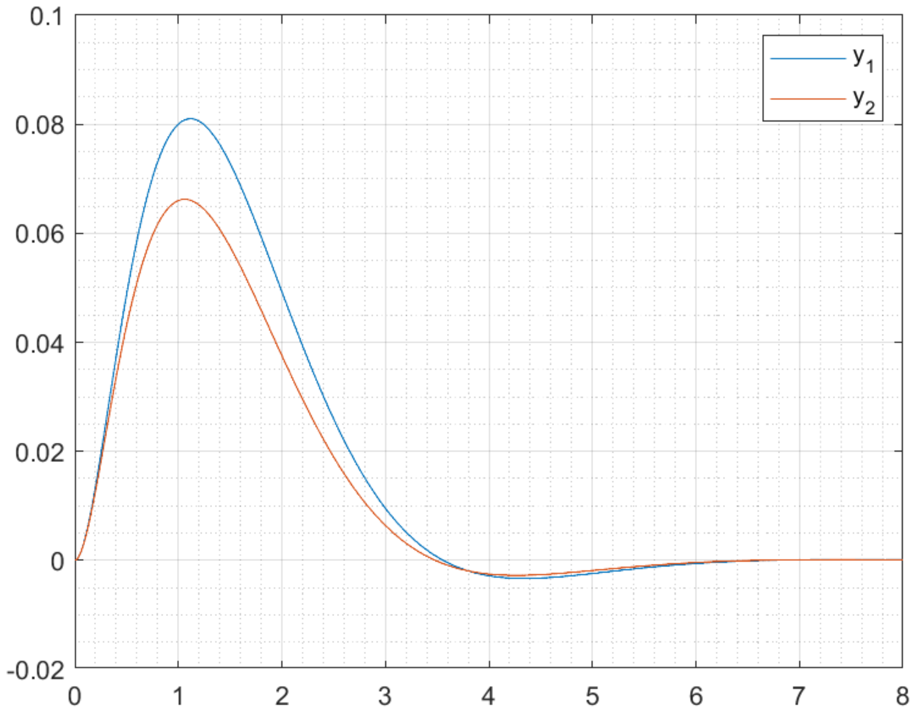
Rysunek 2: Numeryczne rozwiązanie otrzymane za pomocą funkcji ode45

Pomimo ustawienie podziałki na osi rzędnych z dużą dokładnością, nie jesteśmy w stanie zaobserwować istotnych zmian w przebiegu rozwiązania.

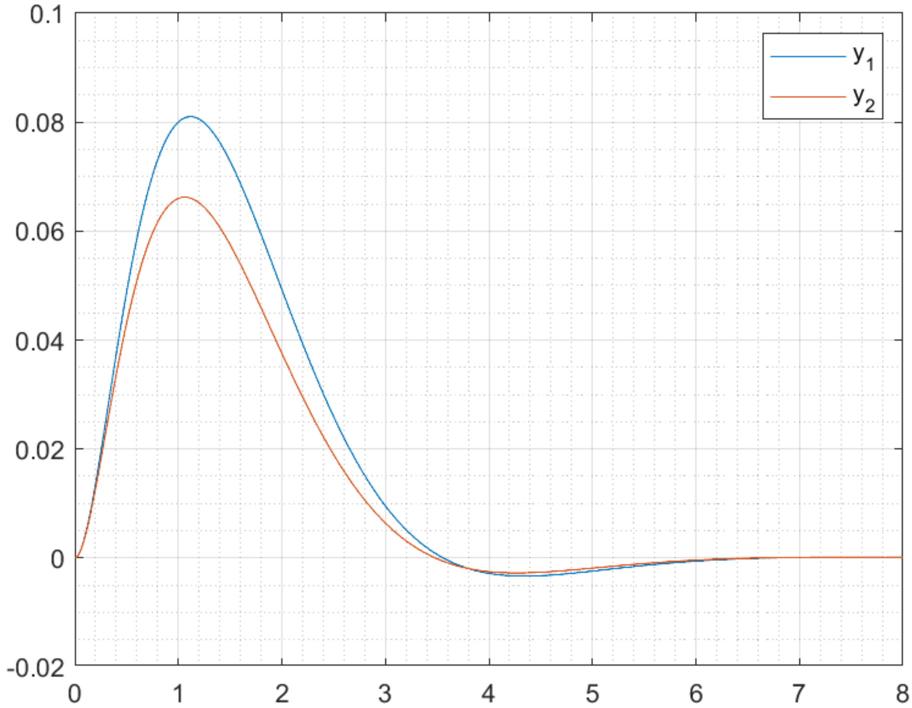
Jeżeli krok całkowania h ustawimy na bardzo małą wartość: $h = 0.001$, to podobna sytuacja ma miejsce dla udoskonalonej metody Eulera (2.3) - wykres 3, metody Adamsa-Bashfortha (2.4) - wykres 4 oraz metody Rungego-Kutty (2.5) - wykres 5.



Rysunek 3: Rozwiązanie otrzymane za pomocą udoskonalonej metody Eulera



Rysunek 4: Rozwiązanie otrzymane za pomocą metody Adamsa-Bashfortha trzeciego rzędu



Rysunek 5: Rozwiązanie otrzymane za pomocą metody Rungego-Kutty z trzema etapami

Z jednej strony jesteśmy zadowoleni z faktu, że rozwiązania przedstawione na wykresach na pierwszy rzut oka niczym się nie różnią - świadczą one o tym, że metody zostały zaimplementowane poprawnie. Z drugiej jednak strony, są one dla nas mało informatywne, ponieważ chcielibyśmy dowiedzieć się więcej o dokładności poszczególnych metod i ich stabilności numerycznej.

Satysfakcjonującym dla nas rozwiązaniem będzie sporządzenie wykresów dokładności rozwiązań otrzymanych przy pomocy zaimplementowanych metod w zależności od długości kroku całkowania $h \in [h_{min}, h_{max}]$, gdzie przedział $[h_{min}, h_{max}]$ zostanie dla każdej z trzech zaimplementowanych metod dobrany w taki sposób, aby zaobserwować zjawisko niestabilności numerycznej rozważanych metod. Aby zjawisko to było łatwiej obserwowalne, osie tych wykresów zostaną ustawione na logarytmiczne o podstawie 10.

Jako kryterium dokładności rozwiązań przyjmiemy zagregowane błędy względne określone wzorami 22 oraz 23:

$$\delta_1(h) = \frac{\sum_{i=1}^{N(h)} \left(\hat{y}_1(t_n, h) - \dot{y}_1(t_n)\right)^2}{\sum_{i=1}^{n} (\dot{y}_1(t_n))^2} \tag{22}$$

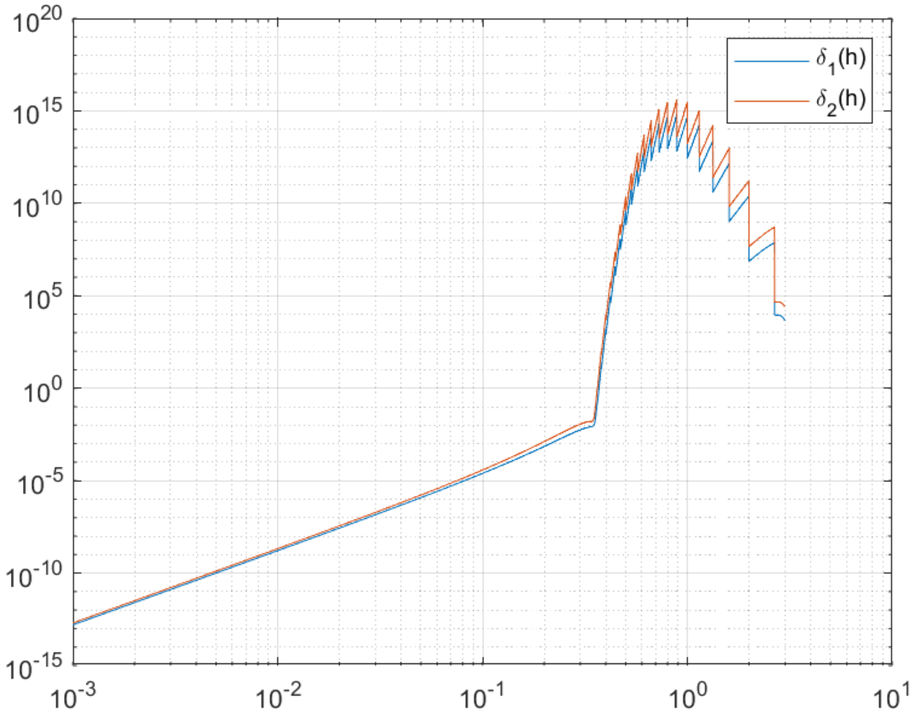
$$\delta_2(h) = \frac{\sum_{i=1}^{N(h)} \left(\hat{y}_2(t_n, h) - \dot{y}_2(t_n)\right)^2}{\sum_{i=1}^{n} (\dot{y}_2(t_n))^2} \tag{23}$$

gdzie $\dot{y}_1(t_n)$ i $\dot{y}_2(t_n)$ to wartości funkcji uzyskane w 2.1, 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.

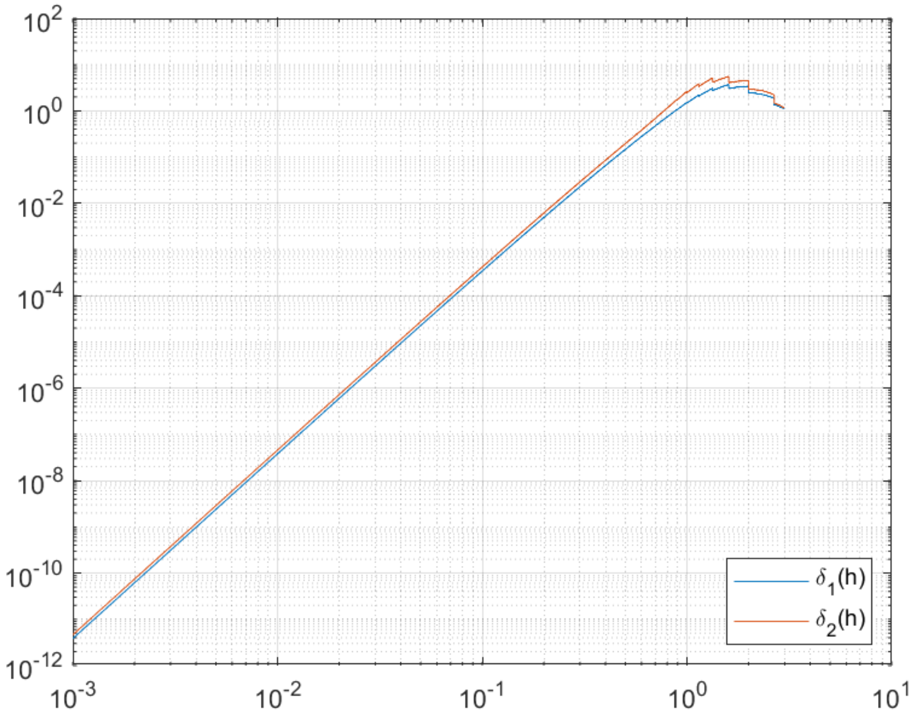
Takie kryterium dokładności, zwane zagregowanym błędem względnym (ARE), daje nam informację o jakości dopasowania danych do wartości referencyjnych. Błąd względny agregowany jest mierzony jako stosunek sumy kwadratów różnic między wartościami rzeczywistymi a referencyjnymi do sumy kwadratów wartości referencyjnych. Jest to miara, która uwzględnia zarówno wielkość, jak i rozkład błędów na przestrzeni całego zbioru danych. Im mniejsza wartość błędu, tym lepsze dopasowanie między rzeczywistymi danymi a wartościami referencyjnymi. Wynik równy zero oznacza idealne dopasowanie.

Zarówno dla udoskonalonej metody Eulera, jak i metody Adamsa-Bashfortha trzeciego rzędu wybraliśmy przedział $[h_{min}, h_{max}] = [0.001, 3]$ próbkowany z krokiem $dh = 0.001$.

Wykresy 6 oraz 7 przedstawiają zależność wielkości określonych wzorami 22 oraz 23 w funkcji kroku całkowania h.



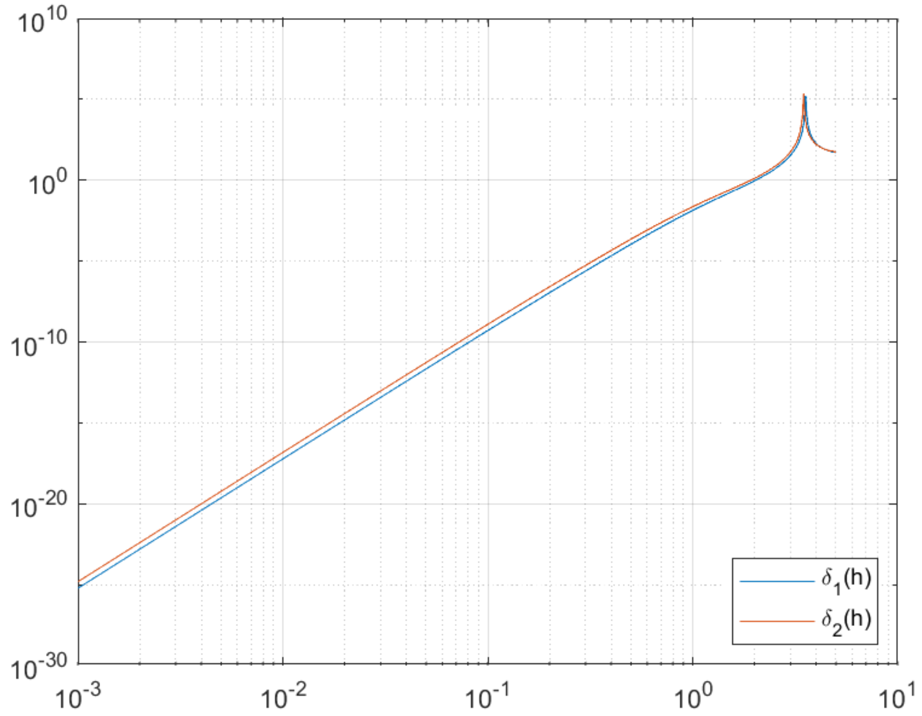
Rysunek 6: ARE dla udoskonalonej metody Eulera



Rysunek 7: ARE dla udoskonalonej metody Adamsa-Bashfortha trzeciego rzędu

Na wykresie 6 obserwujemy nagły wzrost δ_1 oraz δ_2, a także bardzo chaotyczne zachowanie obu parametrów - skokowe wzrosty i spadki wartości. Zmiany te świadczą o niestabilności numerycznej algorytmu dla wartości kroku całkowania h bliskich prawemu końcu badanego przedziału. Warto zwrócić również uwagę na bardzo duże wartości parametrów δ_1 oraz δ_2, które świadczą o bardzo dużej rozbieżności otrzymywanych rozwiązań względem rozwiązania dokładnego.

Podobny charakter do wykresu 6 ma wykres 7, jednak w przypadku metody Adamsa-Bashfortha trzeciego rzędu obserwujemy mniejszą chaotyczność zmian wartości parametrów δ_1 oraz δ_2, a także mniejszy rząd wartości błędów otrzymywanych dla poszczególnych wartości kroku całkowania h.



Rysunek 8: ARE dla udoskonalonej metody Rungego-Kutty z trzema etapami

Dla metody Rungego-Kutty z trzema etapami postanowiliśmy zbadać nieco szerszy przedział $[h_{min}, h_{max}] = [0.001, 5]$.

Charakterystyczną cechą wykresu 8 jest bardzo mała wartość parametów δ_1 oraz δ_2 dla wartości h zbliżonych do lewego końca rozważanego przedziału wartości. Świadczy to o bardzo niewielkiej rozbieżność rozwiązań URRZ (8) otrzymywanych przy pomocy metody Rungego-Kutty z rozwiązaniem dokładnym.

Dodatkowo na wykresie 8 możemy zaobserwować obszar niemożliwy do rozwiązania, który objawia się nagłym, skokowym wzrostem wartości parametrów δ_1 oraz δ_2.

4 Wnioski

W tej części pracy podsumujemy otrzymane wyniki oraz wyciągniemy wnioski dotyczące zaimplementowanych metod numerycznych rozwiązywania URRZ.

Udoskonalona metoda Eulera (opisana w 1.3.3) jest najprostsza i najbardziej zrozumiała w implementacji, jednak z powodu ograniczonej dokładności powinna być używana do celów edukacyjnych i problemów niewymagających znacznej precyzji.

Metoda Adamsa-Bashfortha (opisana w 1.3.4) ma globalny błąd rzędu h^3, zatem oferuje wyższy stopień dokładności w porównaniu do metody Eulera. Jest ona jednak nieco bardziej skomplikowana w implementacji ponieważ jest metodą niejawną.

Najwyższą dokładność oferuje metoda Rungego-Kutty (opisana w 1.3.5), która ma globalny błąd rzędu h^4. Z tą poprawą dokładności wiąże się jednak wzrost złożoności implementacji - metoda jest niejawna, a zatem w każdej iteracji algorytmu musimy rozwiązywać dodatkowy układ równań.

5 Listing

```
function [y1, y2] = problem1dsolve()
% Projekt 1, Zadanie 1: dsolve
% Adam Gracikowski, 327350
%
% OUTPUT:
% y1        - funkcja symboliczna zmiennej y1 rozwazanego URRZ
% y2        - funkcja symboliczna zmiennej y2 rozwazanego URRZ

tspan = [0, 8];
ic = [0; 0];
A = [-2, -2; 2, -7];

syms t x(t) y1(t) y2(t)

x(t) = exp(-t)*sin(t);

dy1 = diff(y1, t, 1);
dy2 = diff(y2, t, 1);

% zdefiniowanie rownan ukladu:
eqn1 = dy1 == A(1,1)*y1 + A(1,2)*y2 + x;
eqn2 = dy2 == A(2,1)*y1 + A(2,2)*y2 + x;

% okreslenie warunkow poczatkowych:
ic1 = y1(tspan(1)) == ic(1);
ic2 = y2(tspan(1)) == ic(2);

[y1, y2] = dsolve([eqn1 eqn2], [ic1, ic2]);

end % function
```

Listing 1: Dokładne rozwiązanie przy pomocy dsolve

```
function [tode45, y1ode45, y2ode45] = problem2ode45()
% Projekt 1, Zadanie 2: a) ode45
% Adam Gracikowski, 327350
%
% OUTPUT:
% t          - wektor punktow czasowych
% y1         - wektor wartosci funkcji y1 w punktach czasowych t
% y2         - wektor wartosci funkcji y2 w punktach czasowych t

tspan = [0, 8];
ic = [0, 0];
A = [-2, -2; 2, -7];

x = @(t) exp(-t).*sin(t);
f = @(t, y) A*y + x(t);

[tode45, v] = ode45(f, tspan, ic);
y1ode45 = v(:, 1);
y2ode45 = v(:, 2);

end % function
```

Listing 2: Numeryczne rozwiązanie URRZ przy pomocy ode45

```
function [t, y1, y2] = problem2method1(h)
% Projekt 1, Zadanie 2: b) method1
% Adam Gracikowski, 327350
%
% INPUT:
% h          - krok calkowania
% OUTPUT:
% t          - wektor punktow czasowych
% y1         - wektor wartosci funkcji y1 w punktach czasowych t
% y2         - wektor wartosci funkcji y2 w punktach czasowych t

tspan = [0 8];
ic = [0; 0];
A = [-2 -2; 2 -7];

x = @(t) exp(-t).*sin(t);
f = @(t, y) A*y + x(t);

t = tspan(1):h:tspan(2);

n = size(t, 2);
y = zeros(2, n);
y(:, 1) = ic;

for i = 2:n
    y(:, i) = y(:, i-1) + h/2 * ...
        (f(t(i-1), y(:, i-1)) + ...
        f(t(i-1) + h, y(:, i-1) + h*f(t(i-1), y(:, i-1))));
end % for

y1 = y(1, :);
y2 = y(2, :);

end % function
```

Listing 3: Numeryczne rozwiązanie URRZ przy pomocy udoskonalonej metody Eulera

```
function [t, y1, y2] = problem2method2(h)
% Projekt 1, Zadanie 2: b) method2
% Adam Gracikowski, 327350
%
% INPUT:
% h          - krok calkowania
% OUTPUT:
% t          - wektor punktow czasowych
% y1         - wektor wartosci funkcji y1 w punktach czasowych t
% y2         - wektor wartosci funkcji y2 w punktach czasowych t

tspan = [0, 8];
ic = [0; 0];
A = [-2, -2; 2, -7];

x = @(t) exp(-t).*sin(t);

t = tspan(1):h:tspan(2);

n = size(t, 2);
y = zeros(2, n);
y(:, 1) = ic;
I = eye(2);

% jawna metoda Eulera w chwili t2
if n >= 2
    y(:, 2) = h*A*y(:, 1) + h*x(t(1));
end % if

for i = 3:n
    y(:, i) = (I - 5/12*h*A) \ (y(:, i-1) + h/12 * ...
        (5*x(t(i)) + 8*(A*y(:, i-1) + x(t(i-1))) - ...
        (A*y(:, i-2) + x(t(i-2)))));
end % for

y1 = y(1, :);
y2 = y(2, :);

end % function
```

Listing 4: Numeryczne rozwiązanie URRZ przy pomocy metody Adamsa-Bashfortha trzeciego rzędu

```
function [t, y1, y2] = problem2method3(h)
% Projekt 1, Zadanie 2: c) method3
% Adam Gracikowski, 327350
%
% INPUT:
% h          - krok calkowania
% OUTPUT:
% t          - wektor punktow czasowych
% y1         - wektor wartosci funkcji y1 w punktach czasowych t
% y2         - wektor wartosci funkcji y2 w punktach czasowych t

tspan = [0, 8];
ic = [0; 0];
A = [-2, -2; 2, -7];

x = @(t) exp(-t).*sin(t);

c = [0, 1/2, 1];
w = [1/6, 2/3, 1/6];
a = [1/6, 0, -1/6; 1/12, 5/12, 0; 1/2, 1/3, 1/6];


t = tspan(1):h:tspan(2);

n = size(t, 2);
y = zeros(2, n);
y(:, 1) = ic;
I = eye(2);

hA = h*A;

L = [(I - a(1,1)*hA) (-a(1,2)*hA) (-a(1,3)*hA); ...
     (-a(2,1)*hA) (I - a(2,2)*hA) (-a(2,3)*hA); ...
     (-a(3,1)*hA) (-a(3,2)*hA) (I - a(3,3)*hA)];

for i = 2:n
    p = [A*y(:, i-1) + x(t(i-1) + c(1)*h); ...
         A*y(:, i-1) + x(t(i-1) + c(2)*h);
         A*y(:, i-1) + x(t(i-1) + c(3)*h)];
    g = L\p;
    g = reshape(g, 2, []);

    y(:, i) = y(:, i-1) + h*sum(g.*w, 2);
end % for

y1 = y(1, :);
y2 = y(2, :);

end % function
```
Listing 5: Numeryczne rozwiązanie URRZ przy pomocy metody Rungego-Kutty z trzema etapami

```
function [are] = aggrelerr(t, yd, y)
% Projekt 1, Zadanie 3: aggregate relative error
% Adam Gracikowski, 327350
%
% INPUT:
% t          - wektor punktow czasowych
% yd        - dokladne rozwiazanie y zwrocone przez dsolve
% OUTPUT:
% are1       - zagregowany blad wzgledny dla y

are = sum((y - yd(t)).^2)/sum(yd(t).^2);

end % function
```

Listing 6: Funkcja obliczająca zagregowany błąd względny

```
function [are1, are2] = problem3(vh, y1d, y2d, intFunc)
% Projekt 1, Zadanie 3: numerical stability
% Adam Gracikowski, 327350
%
% INPUT:
% vh         - wektor krokow calkowania
% yd1        - dokladne rozwiazanie y1 zwrocone przez dsolve
% yd2        - dokladne rozwiazanie y2 zwrocone przez dsolve
% intFunc    - funkcja implementujaca metode rozwiazywania URRZ
% OUTPUT:
% are1       - zagregowany blad wzgledny dla y1
% are2       - zagregowany blad wzgledny dla y2

are1 = 0*vh;
are2 = 0*vh;

i = 1;
for h = vh
    [t, y1, y2] = intFunc(h);
    are1(i) = aggrelerr(t, y1d, y1);
    are2(i) = aggrelerr(t, y2d, y2);
    i = i + 1;
end

end % function
```

Listing 7: Funkcja wykorzystana do tworzenia wykresów zagregowanych błędów względnych

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modelowanie Matematyczne 2023/2024 - semestr zimowy
% Projekt 1: Rozwiazywanie ukladow rownan rozniczkowych zwyczajnych
% Autor: Adam Gracikowski, 327350
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clearvars; close all;
tspan = [0 8];

% zadanie 1:
[y1dsolve, y2dsolve] = problem1dsolve();

% zadanie 2:

% a) ode45:
[tode45, y1ode45, y2ode45] = problem2ode45();

% b) metoda 1:
h1 = 0.001;
[t1, y1m1, y2m1] = problem2method1(h1);

% b) metoda 2:
h2 = 0.001;
[t2, y1m2, y2m2] = problem2method2(h2);

% c) metoda 3:
h3 = 0.001;
[t3, y1m3, y2m3] = problem2method3(h3);

% zadanie 3:

y1d = matlabFunction(y1dsolve);
y2d = matlabFunction(y2dsolve);

% a) metoda 1:

h_min1 = 0.001;
h_max1 = 3;
dh1 = 0.001;
vh1 = h_min1:dh1:h_max1;

[are1m1, are2m1] = problem3(vh1, y1d, y2d, @problem2method1);

% b) metoda 2:

h_min2 = 0.001;
h_max2 = 3;
dh2 = 0.001;
vh2 = h_min2:dh2:h_max2;

[are1m2, are2m2] = problem3(vh2, y1d, y2d, @problem2method2);

% c) metoda 3:

h_min3 = 0.001;
h_max3 = 5;
dh3 = 0.001;
vh3 = h_min3:dh3:h_max3;

[are1m3, are2m3] = problem3(vh3, y1d, y2d, @problem2method3);
```

```
% tworzenie wykresow y1, y2 dla roznych metod:

subplot(2, 3, 1);
fplot(y1dsolve, tspan);
adjustAxes(gca, 1);
hold on;
fplot(y2dsolve, tspan);
title('dsolve');
legend('y_1', 'y_2');
hold off;

subplot(2, 3, 2);
plot(tode45, y1ode45, tode45, y2ode45);
adjustAxes(gca, 1);
title('ode45');
legend('y_1', 'y_2');

subplot(2, 3, 3);
plot(t1, y1m1, t1, y2m1);
adjustAxes(gca, 1);
title('metoda␣1:␣udoskonalona␣Eulera');
legend('y_1', 'y_2');

subplot(2, 3, 4);
plot(t2, y1m2, t2, y2m2);
adjustAxes(gca, 1);
title('metoda␣2:␣Adamsa-Bashfortha␣3-ego␣rzedu');
legend('y_1', 'y_2');

subplot(2, 3, 5);
plot(t3, y1m3, t3, y2m3);
adjustAxes(gca, 1);
title('metoda␣3:␣Rungego-Kutty␣z␣3␣etapami');
legend('y_1', 'y_2');

% tworzenie wykresow zagregowanych bledow dla roznych metod:

figure;
subplot(1, 3, 1);
loglog(vh1, are1m1, vh1, are2m1);
adjustAxes(gca, 0);
title('metoda␣1:␣udoskonalona␣Eulera');
legend('\delta_1(h)', '\delta_2(h)', 'Location','southeast');

subplot(1, 3, 2);
loglog(vh2, are1m2, vh2, are2m2);
adjustAxes(gca, 0);
title('metoda␣2:␣Adamsa-Bashfortha␣3-ego␣rzedu');
legend('\delta_1(h)', '\delta_2(h)', 'Location','southeast');

subplot(1, 3, 3);
loglog(vh3, are1m3, vh3, are2m3);
adjustAxes(gca, 0);
title('metoda␣3:␣Rungego-Kutty␣z␣3␣etapami');
legend('\delta_1(h)', '\delta_2(h)', 'Location','southeast');
```

```
function [] = adjustAxes(ax, p)
    if p
        ax.XLim = [0 8];
        ax.YLim = [-0.02 0.1];
    end
    ax.XGrid = "on";
    ax.XMinorGrid = "on";
    ax.YGrid = "on";
    ax.YMinorGrid = "on";
end
```

Listing 8: Główny skrypt programu

Źródła

[1] https://en.wikipedia.org/wiki/Ordinary_differential_equation

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

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

[4] https://en.wikipedia.org/wiki/Linear_multistep_method

[5] https://en.wikipedia.org/wiki/Euler_method

[6] https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

[7] https://en.wikipedia.org/wiki/Numerical_methods_for_ordinary_differential_equations