Rozwiązywanie układów równań różniczkowych zwyczajnych

W ramach kierunku Informatyka i Systemy Informacyjne

Na Wydziale Matematyki i Nauk Informacyjnych
Politechniki Warszawskiej

W ramach przedmiotu: Modelowanie Matematyczne

Nadzorowane przez: dr inż. Jakub Wagner

Autor: Kamila Wachulec

Baszkówka, 26 listopada 2024r.

Spis treści

1 Lista symboli i oznaczeń

1.1 Akronimy

- **URRZ** – Układ Równań Różniczkowych Zwyczajnych (strona 1)
- **KR** – Szczególny Przypadek Metody Rungego-Kutty (strona 5)

1.2 Funkcje

- **dsolve** – Wbudowana funkcja analitycznego rozwiązywania równań różniczkowych
- **ode45** – Wbudowana funkcja numerycznego rozwiązywania równań różniczkowych
- **plot** – Wbudowana procedura tworząca wykresy rozwiązań
- **delta** – Funkcja wyznaczająca zagregowane błędy względne

1.3 Symbole

- t – Dobrane punkty w czasie, punkty pomiaru
- $y_1(t), y_2(t)$ – Składowe rozwiązania układu równań różniczkowych
- h – Krok całkowania w metodach numerycznych
- A – Macierz współczynników w układzie równań różniczkowych
- $b, x(t)$ – Wektory współczynników i funkcji wymuszających w układzie równań
- c_k – Współczynniki tabeli Butchera dla metody Rungego-Kutty
- $a_{k,\kappa}$ – Elementy tabeli Butchera, współczynniki wagowe etapów metody Rungego-Kutty
- w_k – Wagi w metodzie Rungego-Kutty stosowane do obliczania wyniku końcowego
- g – Wektor wartości pośrednich w metodzie Rungego-Kutty, składający się z f_1, f_2, f_3
- L, P – Macierze układu równań w metodzie Rungego-Kutty
- $\delta_1(h), \delta_2(h)$ – Zagregowane błędy względne dla $y_1(t)$ i $y_2(t)$ odpowiednio
- $\hat{y}_1(t_n, h), \hat{y}_2(t_n, h)$ – Numeryczne estymaty rozwiązań dla kroku całkowania h

- $\dot{y}_1(t_n), \dot{y}_2(t_n)$ – Rozwiązania analityczne dla y_1 i y_2, uzyskane metodą symboliczną
- $N(h)$ – Liczba punktów rozwiązania dla kroku całkowania h
- $tspan$ – Przedział czasowy, na którym rozwiązanie jest wyznaczane, np. $[0, 8]$
- $conds$ – Warunki początkowe dla układu równań
- $hspan$ – Wektor wartości kroków całkowania w analizie numerycznej
- D – Macierz przechowująca błędy względne $\delta_1(h)$ i $\delta_2(h)$ dla różnych metod

2 Wprowadzenie

2.1 Tematyka Zadania

Zadanie "Rozwiązywanie układów równań różniczkowych" ma na celu zaimplementowanie różnych metod rozwiązywania równań różniczkowych zwyczajnych przedstawionych na wykładzie i przećwiczonych przez studentów podczas przedmiotu Modelowanie Matematyczne przy pomocy funkcji i procedur dostępnych w aplikacji MATLAB. Dla pewnego układu równań różniczkowych zwyczajnych (dalej używa się sformułowania URRZ) należy efektywnie przedstawić metody które korzystają z: **dsolve**, **ode45**, **jawnej metody Eulera**, **niejawnej metody Eulera** oraz **szczególnego przypadku metody Rungego-Kutty**. Następnie należy zbadać zależność dokładności rozwiązań numerycznych uzyskanych z metod: **jawnej metody Eulera**, **niejawnej metody Eulera** oraz **szczególnego przypadku metody Rungego-Kutty** od długości kroku całkowania dobranego w każdej metodzie, gdzie dla porównania rozważa się metodę **dsolve**.

Powyższe obserwacje przeprowadzone zostaną dla URRZ i założeń (1,2):

$$\begin{cases} \dfrac{\mathrm{d}y_1(t)}{\mathrm{d}t} = -\dfrac{20}{3}y_1(t) - \dfrac{4}{3}y_2(t) + x(t), \\ \dfrac{\mathrm{d}y_2(t)}{\mathrm{d}t} = \dfrac{4}{3}y_1(t) - \dfrac{10}{3}y_2(t) + x(t) \end{cases} \tag{1}$$

$$\text{dla} \quad t \in [0, 8], \text{ w którym} \quad x(t) = exp(-t)sin(t) \tag{2}$$

2.2 Oczekiwania

Po wykonaniu algorytmów oczekuje się możliwości stwierdzenia, która metoda jest miarodajna i odpowiada wynikom uzyskanym z **dsolve** oraz dla jakiego kroku całkowania osiąga się najlepszą dokładność wyników. Po końcowych obliczeniach powinno dać się zauważyć również dla jakich wartości kroku całkowania obserwuje się zjawisko niestabilności numerycznej.

2.3 Metodyka

2.3.1 Metoda dsolve

Procedura **dsolve** zwracająca symboliczne sformułowania będące dokładnymi wyrażeniami matematycznymi odnosi najlepsze wyniki wśród powyższych metod - najlepiej przybliża rozwiązanie URRZ. Nie zależy również od innych paramterów numerycznych, tolerancji błędu ani kroku czasowego. Dlatego też do tej metody należy porównywać wyniki innych metod przy badaniu dokładności. W powyższym przypadku należy przyjąć zerowe warunki początkowe: $y_1(0) = 0$, $y_2(0) = 0$.

2.3.2 Metoda ode45

Procedura **ode45** jest, tak jak **dsolve**, wbudowaną metodą MATLAB'a rozwiązywania UZZR. Jest ona szybsza od **doslve** ze względu na to, że stosuje rozwiązywanie numeryczne, a nie symboliczne, co przy złożonych układach jest mniej czasochłonne i wymaga mniejszej mocy obliczeniowej. Stosując **ode45** zatracona zostaje jednak dokładność - ma średni rząd dokładności i może prowadzić do zakłamanych w mniejszym lub większym stopniu wyników.

2.3.3 Metoda jawna Eulera

Metoda **jawna Eulera** jest pierwszą metodą tu omawianą, której idea opiera się na dobieraniu kroku czasowego, który zarazem definiuje dokładność wyników. Tym samym wraz z mniejszym krokiem rośnie dokładność obliczeniowa. W zadaniu rozważamy metodę zdefiniowaną wzorem:

$$y_n = y_{n-1} + hf(t_{n-1} + \frac{h}{2}, y_{n-1} + \frac{h}{2} f(t_{n-1}, y_{n-1})) \tag{3}$$

Metoda ta wylicza y_n na podstawie y_{n-1}, dlatego też wymagane jest wyliczenie pierwszego kroku inną metodą, bądź przyjęcie go w warunkach początkowych (tutaj: $y_1(0) = 0$, $y_2(0) = 0$). Funkcja $f(t, y)$ jest postaci:

$$f(t, y) = Ay + bx(t) \tag{4}$$

gdzie A jest macierzą zawierającą współczynniki z rozważanego URRZ, b jest wektorem $[1; 1]$, a $y = [y_1 \quad y_2]$. To znaczy:

$$A = \begin{bmatrix} -\frac{20}{3} & -\frac{4}{3} \\ \frac{4}{3} & -\frac{10}{3} \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5}$$

2.3.4 Metoda niejawna Eulera

Niejawna metoda Eulera jest podobna do **jawnej metody Eulera** - również wymaga dobierania kroku czasowego, który definiuje dokładność wyników. Wylicza ona jednak y_n korzystając z y_{n-1}, y_{n-2}, ale i y_n. Dlatego też nie tylko należy wyznaczyć inną metodą pierwsze dwie wartości y dla danego kroku, ale też nie można z niej korzystać w oryginalnym stanie - wymaga przekształceń, by dało się wyznaczyć y_n. W zadaniu rozważana jest metoda zdefiniowaną wzorem:

$$y_n = \frac{4y_{n-1} - y_{n-2}}{3} + \frac{2h}{3} f(t_n, y_n) \tag{6}$$

Wartości A oraz b dla funkcji f należy przyjąć takie same jak w poprzedniej metodzie (Równania 4,5).

2.3.5 Metoda szczególnego przypadku Rungego-Kutty

Ostatnia metoda jest **szczególnym przypadkiem metody Rungego-Kutty**. Jest ona bardziej złożona, gdyż nie tylko y_n zależy od y_{n-1}, ale również jest zależna od dodatkowych zmiennych. W zadaniu przyjmuje ona postać:

$$y_n = y_{n-1} + h \sum_{k=1}^{3} w_k f_k \tag{7}$$

$$\text{gdzie: } f_k = f(t_{n-1} + hc_k, y_{n-1} + h \sum_{\kappa=1}^{3} a_{k,\kappa} f_\kappa) \tag{8}$$

Współczynniki przyjmują wartości przedstawione w poniższej tabeli Butchera:

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

Tabela 1: Tabela Butchera

gdzie $y_n = [y_1(t_n) \quad y_2(t_n)]^T$, a funkcja $f(t_n, y_n)$ określona jest przez URRZ: $\left.\frac{\mathrm{d}y(t)}{\mathrm{d}t}\right|_{t=t_n} = f(t_n, y_n)$.

By móc wygodnie skorzystać z tej metody wymaga się przeprowadzenia przekształceń i wyprowadzenia odpowiednio wzorów na f_1, f_2, f_3 oraz wyznaczenia ich z podanych zmiennych i kroku całkowania, a następnie podstawienia do ogólnego wzoru na y_n.

3 Algorytmy i wyniki doświadczeń

Ta część raportu poświęcona zostaje omówieniu algorytmów stosowanych w rozwiązaniach w aplikacji MATLAB oraz wynikom doświadczeń przeprowadzonych dla każdej metody. Zostaną zamieszczone przekształcenia, schematy myślowe i rozwiązania dla konkretnych pytań badawczych pojawiających się w zadaniu.

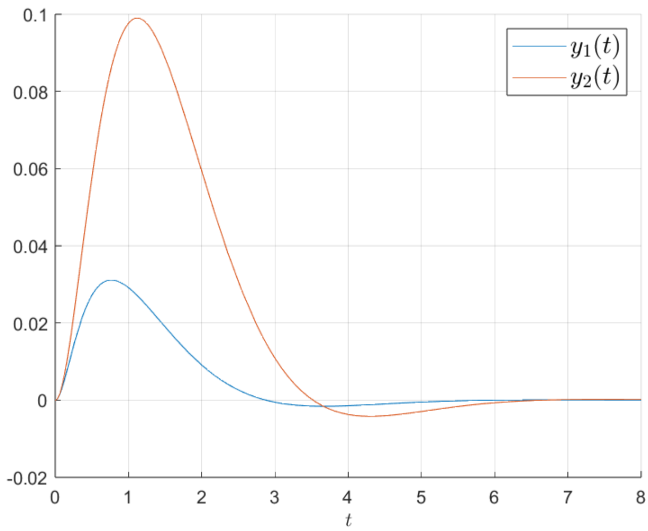
3.1 Algorytm metody dsolve

Stosując **dsolve** należy stworzyć zmienne symboliczne $x(t)$, $y_1(t)$ oraz $y_2(t)$ i do $x(t)$ przypisać $x(t) = exp(-t)sin(t)$. Następnie niech $tspan = [0, 8]$ będzie przedziałem całkowania, a $conds = [y_1(0) = 0; \quad y_2(0) = 0]$ warunkami początkowymi jak podano w treści zadania. Wtedy można utworzyć równanie różniczkowe postaci

$$eqns = [diff(y_1, t) == a_{1,1}y_1 + a_{1,2}y_2 + x, diff(y_2, t) == a_{2,1}y_1 + a_{2,2}y_2 + x]$$

gdzie $a_{i,j}$ są wartościami z macierzy A (Równanie 5). Stosując procedurę **dsolve** do $eqns$ i $conds$ uzyskuje się wyrażenia symboliczne na y_1 oraz y_2. Za pomocą **fplot** można następnie zaobserwować wyniki na Rysunku 1:

wyniki $y_1(t)$, $y_2(t)$ dla $t \in [0, 8]$



Rysunek 1: Wyniki URRZ z **dsolve** dla $t \in [0, 8]$

Wyniki te, będąc najbardziej miarodajnymi, będą w późniejszej części raportu wyznacznikiem dokładności pozostałych metod.

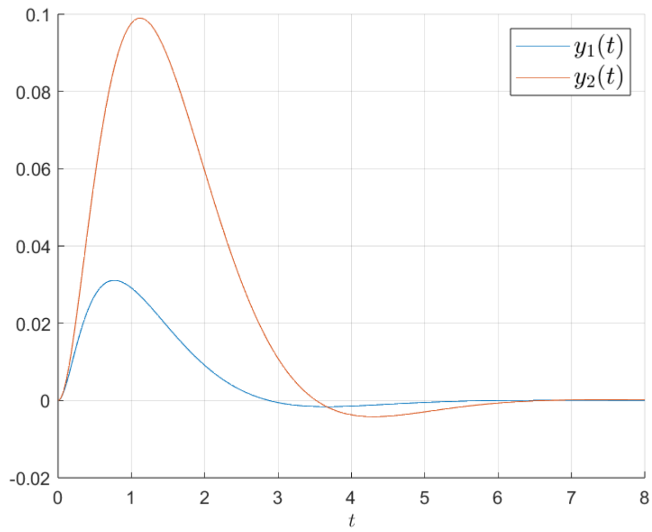
3.2 Algorytm metody ode45

Procedura **ode45** jest pierwszą numeryczną metodą która pojawia się z tym zadaniu. By wyznaczyć wyniki ukladu równań różniczkowych deklaruje się macierz A jak w rówaniu (5), albo zapisuje współczynniki w formie odosobnionych zmiennych. Następnie delkaruje się $tspan = [0, 8]$ oraz $conds = [0; 0]$, czyli przedział całkowania i warunki początkowe na y_1 i y_2. Następnie należy stworzyć **uchwyt do funkcji**

$$dydt = @(t, y) \quad [a_{1,1}y(1)+a_{1,2}y(2)+exp(-t)sin(t); \quad a_{2,1}y(1)+a_{2,2}y(2)+exp(-t)sin(t)],$$

gdzie $a_{i,j}$ są wartościami z macierzy A. Następnie stosując procedurę **ode45** do $dydt$, $tspan$ i $conds$ uzyskuje się macierz $y = [y_1 \quad y_2]$. Za pomocą **plot** można zaobserwować wyniki na Rysunku 2:

wyniki $y_1(t)$, $y_2(t)$ dla $t \in [0, 8]$



Rysunek 2: Wyniki URRZ z **ode45** dla $t \in [0, 8]$

3.3 Algorytm metody jawnej Eulera

Jest to pierwsza metoda korzystająca z **kroku całkowania** h. Krok może być dowolnie mały i maksymalnie równy 8 - równy przedziałowi całkowania, ale wraz z mniejszym krokiem całkowania dokładność obliczeniowa rośnie. Dlatego też w tej metodzie, metodzie **niejawnej Eulera** i metodzie **przypominającej Rungego-Kutty** przyjęty zostaje krok całkowanie $h = \frac{1}{1000}$. Definuje się macierz A oraz wektor b jak w Równaniu 5. Następnie tworzymy wektor $t = 0 : h : 8$, który przechowuje kolejne etapy skoku. Ponieważ metoda ta dla y_n zależy od y_{n-1}, to przyjmujemy zgodnie z założeniami zadania $y_1(0) = 0$ oraz $y_2(0) = 0$, gdzie $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$.

Następnie dla każdej wartości z t, $\quad (t > 0)$ należy wyznaczyć:

$$y_n = y_{n-1} + hf(t_{n-1} + \frac{h}{2}, y_{n-1} + \frac{h}{2}f(t_{n-1}, y_{n-1})) \tag{9}$$

wiedząc że: $f(t, y) = Ay + bx(t)$

Problem możemy podzielić na etapy. Dla każdej rozważanej wartości i, $\quad (i \in [2, length(t)])$ obliczamy:

$$f_{wew} = f(t_{n-1}, y_{n-1}) = Ay_{i-1} + bx(t_{i-1}) \tag{10}$$

$$y_{zew} = y_{i-1} + \frac{h}{2} f_{wew} \tag{11}$$

$$t_{zew} = t_{i-1} + \frac{h}{2} \tag{12}$$

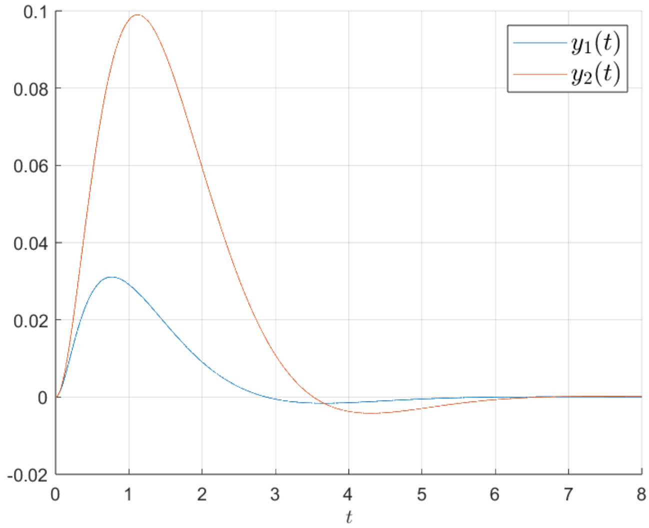
$$f_{zew} = Ay_{zew} + bx(t_{zew}) \tag{13}$$

Rozbijając problem można teraz łatwo skorzystać z Równania 9 po podstawieniach:

$$y_i = y_{i-1} + hf_{zew} \tag{14}$$

Za pomocą **plot** można następnie zaobserwować wyniki na Rysunku 3:

wyniki $y_1(t)$, $y_2(t)$ dla $t \in [0, 8]$



Rysunek 3: Wyniki URRZ z **jawnej metody Eulera** dla $t \in [0, 8]$

3.4 Algorytm metody niejawnej Eulera

Metoda ta jest podobna do **jawnej metody Eulera**, natomiast tutaj y_n jest w oficjalnym wzorze zależne od samego siebie. Dlatego też dokonuje się przekształceń, by wyodrębnić y_n na jedną stronę równania. Przyjęty zostaje krok całkowanie $h = \frac{1}{1000}$. Definujemy macierz A oraz wektor b jak w Równaniu 5. Następnie tworzymy wektor $t = 0 : h : 8$, który przechowuje kolejne etapy skoku. Metoda ta w zadaniu jest opisana wzorem:

$$y_n = \frac{4y_{n-1} - y_{n-2}}{3} + \frac{2h}{3} f(t_n, y_n), \tag{15}$$

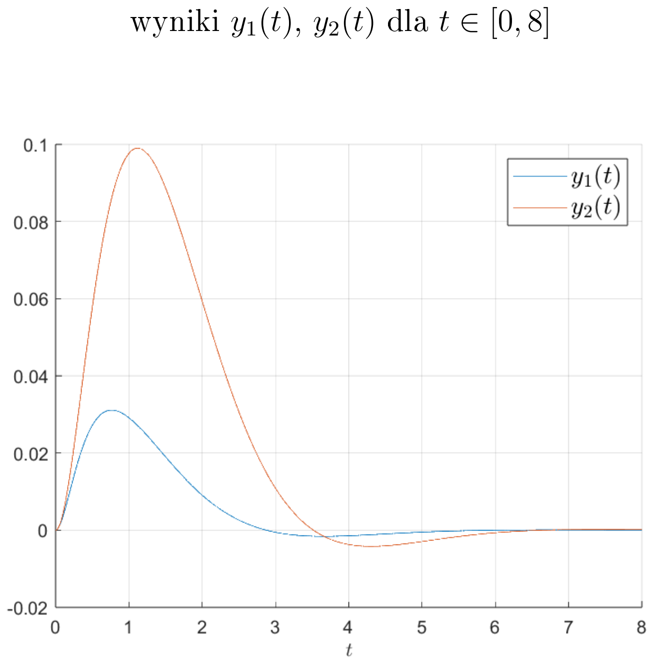
co z Równania (3) da się przekształcić do:

$$y_n = \frac{4}{3} y_{n-1} - \frac{1}{3} y_{n-2} + \frac{2}{3} hAy_n + \frac{2}{3} hbx(t_n) \tag{16}$$

$$(I - \frac{2}{3}hA)y_n = \frac{4}{3}y_{n-1} - \frac{1}{3}y_{n-2} + \frac{2}{3}hbx(t_n), \tag{17}$$

gdzie I jest macierzą jednostkową postaci: $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Ta postać ułatwia wykonywanie dalszych obliczeń. Wzór ten jest natomiast zależny nie tylko od y_{n-1}, ale też y_{n-2}. Dlatego też o ile można przyjąć, że warunki początkowe są zerowe, czyli $y_1(0) = 0$ i $y_2(0) = 0$, to pierwszy krok całkowania należy wyliczyć inną metodą. Dlatego też dla y_2 zastosowana została **jawn metodę Eulera** przedstawioną w Równaniach 10-14.

Za pomocą procedury **plot** można następnie zaobserwować wyniki na Rysunku 4:



Rysunek 4: Wyniki URRZ z **niejawnej metody Eulera** dla $t \in [0, 8]$

3.5 Algorytm metody Rungego-Kutty

Jest to szczególny przypadek metody Rungego-Kutty - rodziny technik rozwiązywania URRZ, które ewaluują wartości uzyskane dla danych kroków całkowania - tu ponownie zastosowany będzie mały krok $h = \frac{1}{1000}$ - później łącząc je średnimi ważonymi. Korzysta się z tabeli Butchera przedstawionej w Tabeli 1, która przechowuje wagi wykorzystywane do połączenia etapów całkowania. By ułatwić zadanie dokonuje się przekształceń, a dokładnie wyznaczenia jak największej liczby zmiennych. By uzyskać y_n przy pomocy Równania 7, należy poznać funkcje f_1, f_2 oraz f_3, gdzie f_k z Równania 8 to:

$$f_k = f(t_{n-1} + hc_k, y_{n-1} + h\sum_{\kappa=1}^{3} a_{k,\kappa}f_\kappa)$$

. Podstawiając Równanie 4, mamy że:

$$\begin{cases} f_1 = A(y_{n-1} + ha_{1,1}f_1 + ha_{1,2}f_2 + ha_{1,3}f_3) + bx(t_{n-1} + c_1h), \\ f_2 = A(y_{n-1} + ha_{2,1}f_1 + ha_{2,2}f_2 + ha_{2,3}f_3) + bx(t_{n-1} + c_2h), \\ f_3 = A(y_{n-1} + ha_{3,1}f_1 + ha_{3,2}f_2 + ha_{3,3}f_3) + bx(t_{n-1} + c_3h) \end{cases} \tag{18}$$

Można zauważyć, że następujący układ równań można zapisać w postaci działania na macierzach:

$$Lg = P, \tag{19}$$

gdzie $g = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$, A jest macierzą niezależnych od $t = 0 : h : 8$ i zależnych od f_1, f_2, f_3 wartości, a P resztą. Przekształcając układ Równań 18 uzyskuje się:

$$\begin{cases} (I - Aha_{1,1})f_1 - Aha_{1,2}f_2 - Aha_{1,3}f_3 = Ay_{n-1} + bx(t_{n-1} + c_1h), \\ -Aha_{2,1}f_1 + (I - Aha_{2,2})f_2 - Aha_{2,3}f_3 = Ay_{n-1} + bx(t_{n-1} + c_2h), \\ -Aha_{3,1}f_1 - Aha_{3,2}f_2 + (I - Aha_{3,3})f_3 = Ay_{n-1} + bx(t_{n-1} + c_3h) \end{cases} \tag{20}$$

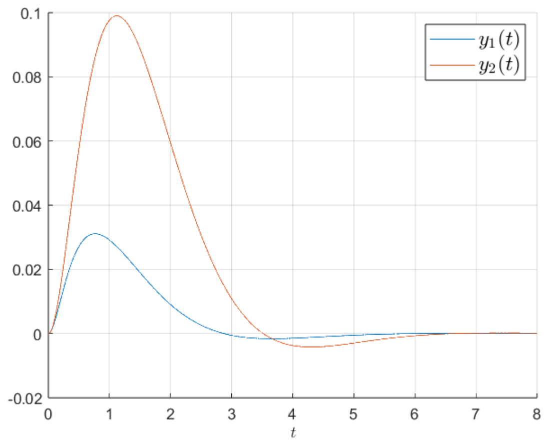
Wtedy $L = \begin{bmatrix} I - Aha_{1,1} & -Aha_{1,2} & -Aha_{1,3} \\ -Aha_{2,1} & I - Aha_{2,2} & -Aha_{2,3} \\ -Aha_{3,1} & -Aha_{3,2} & I - Aha_{3,3} \end{bmatrix}$, $P = \begin{bmatrix} Ay_{n-1} + bx(t_{n-1} + c_1h) \\ Ay_{n-1} + bx(t_{n-1} + c_2h) \\ Ay_{n-1} + bx(t_{n-1} + c_3h) \end{bmatrix}$

Następnie dzieleniem macierzowym z Równania 19 można uzyskać g. W ten sposób otrzymane f_1, f_2 oraz f_3 należy podstwić do Równania 7:

$$y_n = y_{n-1} + h\sum_{k=1}^{3} w_k f_k$$

Za pomocą procedury **plot** można następnie zaobserwować wyniki na Rysunku 5:

wyniki $y_1(t)$, $y_2(t)$ dla $t \in [0, 8]$



Rysunek 5: Wyniki URRZ z **szczególnego przypadku RK** dla $t \in [0, 8]$

4 Dyskusja wyników i eksperymentów numerycznych

Jak widać na Rysunkach 1-5 wyniki układu równań różniczkowych zwyczajnych są podobne. By jednak określić, która metoda odnosi największy sukces i dla jakiego kroku h daje najbardziej miarodajne wyniki w porównaniu z procedurą **dsolve** można porównać dokładności tych metod. W Zadaniu należy zbadać zależność dokładności rozwiązań numerycznych, uzyskanych za pomocą metod: **jawnej Eulera**, **niejawnej Eulera** i **szczególnego przypadku Rugnego-Kutty** od długości kroku całkowania $h \in [h_{min}, h_{max}]$. Zakres zmienności h należy dobrać w taki sposób, aby zaobserwować zjawisko niestabilności numerycznej dla zbyt dużego kroku h. Jako kryterium dokładności rozwiązań należy przyjąć zagregowane błędy względne:

$$\delta_1(h) = \frac{\sum_{n=1}^{N(h)}(\hat{y}_1(t_n,h) - \dot{y}_1(t_n))^2}{\sum_{n=1}^{N(h)}(\dot{y}_1(t_n))^2} \text{ i } \delta_2(h) = \frac{\sum_{n=1}^{N(h)}(\hat{y}_2(t_n,h) - \dot{y}_2(t_n))^2}{\sum_{n=1}^{N(h)}(\dot{y}_2(t_n))^2} \tag{21}$$

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

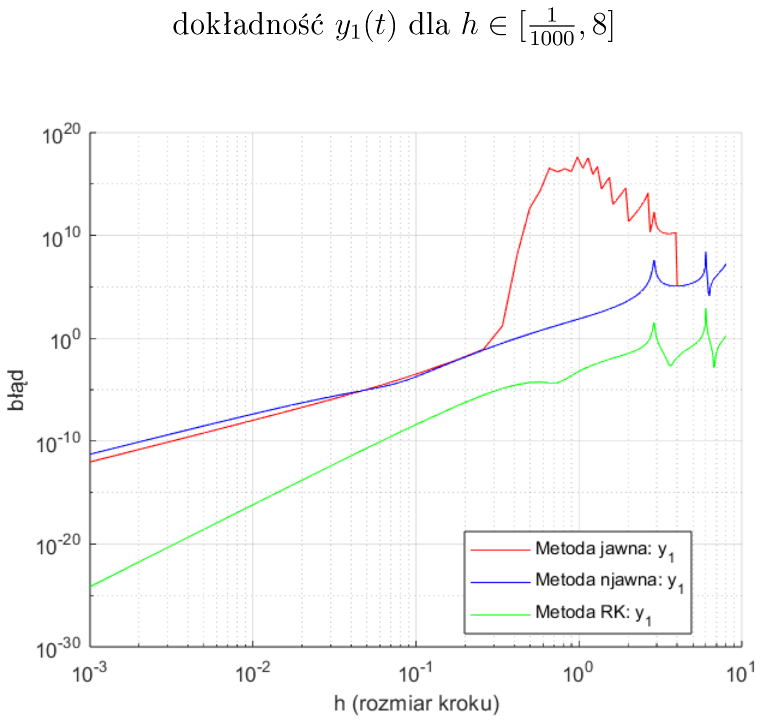
By móc dobrze zaobserwować zjawisko niestabilności numerycznej dla zbyt dużego kroku h w tym rozwiązaniu zostają dobrane kroki:

$$hspan = [linspace(\frac{1}{1000}, \frac{1}{10}, 50), linspace(\frac{1}{10}, 8, 100)] \tag{22}$$

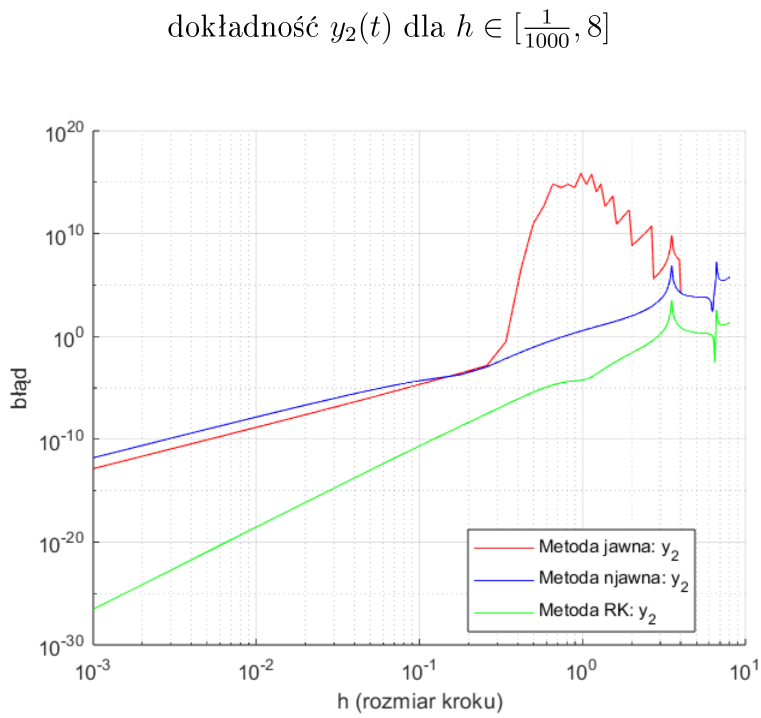
czyli rozważa się zakres kroków od bardzo małego ($h = \frac{1}{1000}$) do jednego obrotu funkcji ($h = 8$). Dlatego też więcej h dobiera się dla większych wartości, by dobrze przybliżyż niestabilność numeryczną.

Następnie tworzy się macierz D, która na razie jest wypełniona zerami, ale później będzie przechowywać w osobnych wierszach błędy względne δ_1 oraz δ_2 dla każdej metody w kolejnych dwóch wierszach. Zatem $size(D) = 6 \quad 150$, gdzie procedura $size$ bada rozmiar macierzy, 6 to liczba wierszy, a 150 liczba kolumn (badanych kroków). W kolejnym kroku dla każdego h z $hspan$ wyznaczane są macierze przechowujące wyniki y_1 oraz y_2 dla każdej metody z: **jawnej Eulera**, **niejawnej Eulera** oraz **szczególnego przypadku Rungego-Kutty**. Następnie wywołuje się funkcję **delta**, która zwraca wartości δ_1 oraz δ_2 dla macierzy wartości danej metody.

Funkcja **delta** musi zawierać pętlę od 1 do $N(h)$, gdzie w każdym obrocie pętli zbierane jest: sum_{y1} (suma $\dot{y}_1(t_n)^2$), sum_{y2} (suma $\dot{y}_2(t_n)^2$), $sumRoz_{y1}$ (suma $(\hat{y}_1(t_n,h) - \dot{y}_1(t_n))^2$) i $sumRoz_{y2}$ (suma $(\hat{y}_2(t_n,h) - \dot{y}_2(t_n))^2$). Na końcu po wyjściu z pętli funkcja zwraca $\frac{sumRoz_y1}{sum_{y1}}$ oraz $\frac{sumRoz_y2}{sum_{y2}}$. Punkty te zostają przypisane do odpowiedniej metody i wartości kroku h, a następnie za pomocą procedury **plot** można zaobserwować wyniki na Rysunkach 6 i 7:



Rysunek 6: Błędy względne rozwiązań dla metody **jawnej** i **niejawnej Eulera** oraz **KR** dla y_1 gdy $h \in [\frac{1}{1000}, 8]$



Rysunek 7: Błędy względne rozwiązań dla metody **jawnej** i **niejawnej Eulera** oraz **KR** dla y_2 gdy $h \in [\frac{1}{1000}, 8]$

Ze względu na wartości, jakie przyjmują błędy, należy zastosować skalę logarytmiczną do pokazywania danych na wykresie. Jak widać na Rysunkach 6, 7 krok wynosi od 10^{-3}

do 8, a zagregowane błędy względne przyjmują wartości mniejsz niż 10^{-15} i większe niż 10^{15}. Skala logarytmiczna ułatwia tutaj porównanie danych.

Wcześniej można było stwierdzić, że wszystkie metody są stanowczo podobne ze względu na podobieństwo w wykresach. Teraz natomiast widać, że niektóre metody radzą sobie lepiej, a niektóre gorzej, z większymi krokami całkowania.

4.1 Metoda jawna Eulera - omówienie

Metoda jawna wykazuje szybki wzrost błędu wraz ze zwiększaniem kroku h - już dla $h \approx 10^{-5}$ niestabilność objawia się nagłymi oscylacjami oraz wykładniczym wzrostem błędu, co sprawia, że metoda ta staje się zawodna dla większych kroków. Przy małych h metoda jawna średnio osiąga błędy rzędu 10^{-5} jednak dla większych kroków błędy szybko przekraczają wartość 10^{15}. Ze względu na brak stabilności metoda ta nadaje się głównie do rozwiązywania równań różniczkowych z bardzo drobnymi krokami.

4.2 Metoda niejawna Eulera - omówienie

Metoda **niejawna** wykazuje większą stabilność w porównaniu z metodą **jawną**, a błędy rosną stopniowo wraz z większymi krokami h. Jednak niestabilność zaczyna być zauważalna przy $10^{0.2}$, gdzie pojawiają się rozbieżności błędu. Mimo lepszej stabilności niż metoda jawna, metoda niejawna ma wciąż wysokie błędy względne dla większych wartości h.

4.3 Metoda Rungego-Kutty - omówienie

Metoda będąca **szczególnym przypadkiem metody Rungego-Kutty** uzyskuje najmniejsze wskaźniki błędu, tym samym jest najbardziej miarodajną metodą - najbardziej przypomina rozwiązania uzyskane dla zastosowania procedury **dsolve**. Dla małych kroków h błędy są średnio rzędu 10^{-15}, a nawet dla dużych kroków nie przekraczają wartości 10^{0}. Dzięki swojej odporności na niestabilności oraz precyzyjnemu odwzorowaniu wyników, metoda Rungego-Kutty jest najlepiej dostosowana do rozwiązywania równań różniczkowych zwyczajnych, oferując niezawodność zarówno dla małych, jak i dużych kroków.

5 Spis tabel i rysunków

Spis tabel

Spis rysunków

6 Spis programów

plik: Zadanie1.m

```
function [y1n, y2n] = Zadanie1(show)

if ~exist('show', 'var')
    show = false;
end

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

a = -20/3;
b = -4/3;
c = 4/3;
d = -10/3;

tspan = [0, 8];
x(t) = exp(-t)*sin(t);

eqns = [diff(y1,t) == a*y1 + b*y2 + x, diff(y2,t) == c*y1 + d*y2 + x];
conds = [y1(0) == 0; y2(0) == 0];
[y1_s, y2_s] = dsolve(eqns, conds);
y1n = matlabFunction(y1_s);
y2n = matlabFunction(y2_s);

if show
figure('Name','Zadanie1rozwy1y2','NumberTitle','off');
hold on
fplot(y1_s, tspan)
fplot(y2_s, tspan)
xlabel('$t$', 'Interpreter','latex')
legend({'$y_1(t)$', '$y_2(t)$'}, 'Interpreter', 'latex','FontSize',14)
grid on
hold off
end
end
```

plik: Zadanie2ode45.m

```
function [] = Zadanie2ode45()

clearvars

a = -20/3;
b = -4/3;
c = 4/3;
d = -10/3;

dydt = @(t,y) [a*y(1) + b*y(2) + exp(-t)*sin(t); c*y(1) + d*y(2) + exp(-t)*sin(t)];
tspan = [0, 8];
conds = [0; 0];

[t, y] = ode45(dydt,tspan, conds);

disp(y(2,:))

figure('Name','Zadanie2ode45Rozwy1y2','NumberTitle','off');
hold on
plot(t, y(:,1))
plot(t, y(:,2))
xlabel('$t$', 'Interpreter','latex')
legend({'$y_1(t)$', '$y_2(t)$'}, 'Interpreter', 'latex','FontSize',14)
grid on
hold off

end
```

plik: Zadanie2RK.m

```
function [y] = function [y] = Zadanie2RK(show, h)

if ~exist('show', 'var')
    show = false;
end

if ~exist('h', 'var')
    h = 1/1000;
end

A = [-20/3, -4/3; 4/3, -10/3];
c = [0, 1/2, 1];
w = [1/6, 2/3, 1/6];
a = [1/6, -1/3, 1/6; 1/6, 5/12, -1/12; 1/6, 2/3, 1/6];
I = [1 0; 0 1];
b = [1; 1];
t = 0:h:8;
y = zeros(2, length(t));

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

H = h*A;
L = [I-a(1,1)*H, -H*a(1,2), -H*a(1,3); -H*a(2,1), I-H*a(2,2), -H*a(2,3); -H*a(3,1), -H*a(3,2), I-H*a(3,3)];

for i = 2:length(t)
    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)];
    g = L\P;

    suma = 0;
    for j = 1:3
        suma = suma + w(j)*g([j*2-1,j*2]);
    end

    y(:, i) = y(:,i-1) + h*suma;
end

if show
    figure('Name','Zadanie2RKrozw','NumberTitle','off');
    hold on
    plot(t, y(1, :));
    plot(t, y(2, :));
    xlabel('$t$', 'Interpreter','latex')
    legend({'$y_1(t)$', '$y_2(t)$'}, 'Interpreter', 'latex','FontSize',14)
    grid on;
end

end
```

plik: Zadanie2niejawna.m

```
function [y] = function [y] = Zadanie2niejawna(show, h)

if ~exist('show', 'var')
    show = false;
end
if ~exist('h', 'var')
    h = 1/1000;
end
A = [-20/3 -4/3; 4/3 -10/3];
I = [1 0; 0 1];
b = [1; 1];
t = 0:h:8;
y = zeros(2, length(t));

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

f_wew = A*y(:, 1) + b*x(t(1));
y_zew = y(:, 1) + (h/2)*f_wew;
t_zew = t(1) + h/2;
f_zew = A*y_zew + b*x(t_zew);
y(:, 2) = y(:, 1) + h*f_zew;

for i = 3:length(t)
    syf = I-(2/3)*h*A;
    y(:, i) = syf\((4/3)*y(:,i-1)-(1/3)*y(:,i-2)+(2/3)*h*b*x(t(i)));
end

if show
figure('Name','Zadanie2niejawnarozw','NumberTitle','off');
hold on
plot(t, y(1, :));
plot(t, y(2, :));
xlabel('$t$', 'Interpreter','latex')
legend({'$y_1(t)$', '$y_2(t)$'}, 'Interpreter', 'latex','FontSize',14)
grid on;
end
end
```

plik: Zadanie2jawna.m

```
function [y] = function [y] = Zadanie2jawna(show, h)

if ~exist('show', 'var')
    show = false;
end
if ~exist('h', 'var')
    h = 1/1000;
end
A = [-20/3 -4/3; 4/3 -10/3];
b = [1; 1];
t = 0:h:8;
y = zeros(2, length(t));

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

for i = 2:length(t)
    f_wew = A*y(:, i-1) + b*x(t(i-1));
    y_zew = y(:, i-1) + (h/2)*f_wew;
    t_zew = t(i-1) + h/2;

    f_zew = A*y_zew + b*x(t_zew);
    y(:, i) = y(:, i-1) + h*f_zew;
end
if show
figure('Name','Zadanie2JawnaRozw','NumberTitle','off');
hold on
plot(t, y(1, :));
plot(t, y(2, :));
xlabel('$t$', 'Interpreter', 'latex')
legend({'$y_1(t)$', '$y_2(t)$'}, 'Interpreter', 'latex','FontSize',14)
grid on;
end
end
```

plik: Zadanie3.m

```
function [] = Zadanie3()

hspan1 = linspace(1/1000, 1/10, 50);
hspan2 = linspace(1/10, 8, 100);
hspan = [hspan1, hspan2];
D = zeros(6, length(hspan));

for i = 1:length(hspan)
    h = hspan(i);
    y_jawna_h = Zadanie2jawna(false, h);
    y_njawna_h = Zadanie2niejawna(false, h);
    y_RK_h = Zadanie2RK(false, h);

    [D(1,i), D(2,i)] = delta(h, y_jawna_h);
    [D(3,i), D(4,i)] = delta(h, y_njawna_h);
    [D(5,i), D(6,i)] = delta(h, y_RK_h);
end

figure('Name', 'Zadanie3y1', 'NumberTitle', 'off');
hold on
grid on
styles = {'r', 'b', 'g'};
labels = {
    'Metoda␣jawna:␣y_1', 'Metoda␣njawna:␣y_1', 'Metoda␣RK:␣y_1', 'Interpreter','latex'};
for i = 1:3
    plot(hspan, D(2*i-1, :), styles{i}, 'DisplayName', labels{i});
end
xlabel('h␣(rozmiar␣kroku)');
ylabel('blad');
xscale log
yscale log
legend('Location', 'best');
hold off

figure('Name', 'Zadanie3y2', 'NumberTitle', 'off');
hold on
grid on
styles = {'r', 'b', 'g'};
labels = {
    'Metoda␣jawna:␣y_2', 'Metoda␣njawna:␣y_2', 'Metoda␣RK:␣y_2', 'Interpreter','latex'};

for i = 1:3
    plot(hspan, D(2*i, :), styles{i}, 'DisplayName', labels{i});
end
xlabel('h␣(rozmiar␣kroku)');
ylabel('blad');
xscale log
yscale log
legend('Location', 'best');
hold off
end
```

plik: delta.m

```
function [d1, d2] = delta(h, y_dasz)

t = 0:h:8;

sum_y1 = 0;
sum_y2 = 0;
sum_roz_y1 = 0;
sum_roz_y2 = 0;
[y1n, y2n] = Zadanie1(false);

for i = 1:length(y_dasz)
    sum_y1 = sum_y1 + y1n(t(i))*y1n(t(i));
    sum_y2 = sum_y2 + y2n(t(i))*y2n(t(i));
    sum_roz_y1 = sum_roz_y1 + (y_dasz(1,i)-y1n(t(i)))^2;
    sum_roz_y2 = sum_roz_y2 + (y_dasz(2,i)-y2n(t(i)))^2;
end
d1 = sum_roz_y1/sum_y1;
d2 = sum_roz_y2/sum_y2;

end
```

Bibliografia

[1] MATLAB Help Center, url: https://www.mathworks.com/help/matlab/

[2] dr hab. inż. Kajetana Marta Snopek, ogólnodostępne materiały wykładowe

[3] dr inż. Jakub Wagner, wsparcie w walce z głupimi błędami