Politechnika Warszawska

Wydział Matematyki i Nauk Informacyjnych

Rozwiazywanie układów równań różniczkowych zwyczajnych

Autor: Grzegorz Prasek

Prowadzacy: Profesor Paweł Mazurek

Warszawa 25.11.2023

Lista symboli matematycznych i akronimów, użytych w sprawozdaniu:

- `plotuj(x, y)` - Funkcja przyjmujaca jako parametry dwa wektory (*argumenty*, *wartości*) i generujaca wykres na podstawie tych wektorów.

- `y_values` - Zmienna przechowujaca macierz wyników danej metody, uzyskanych w wyniku próbkowania zgodnego z funkcja podana jako argument. Macierz ta ma wymiary $2 \times n$, gdzie n to ilość punktów, dla których przeprowadzono proces różniczkowania.

- `dt` oraz `h` - Wartości określajace wielkość kroku próbkowania w kontekście różniczkowania. Parametr `dt` to dokładny krok czasowy, natomiast `h` określa wielkość kroku dla metody numerycznej. Oba parametry sa istotne dla procesu badania zmian wartości funkcji w kolejnych chwilach czasu.

- `f` - Definicja równań różniczkowych

- $\delta 1(h)$ `i` $\delta 2(h)$ - błedy wzgledne

- `metoda nr. ...` - (np. 2.1) to zadanie, algorytm, motoda implementujaca metode z punktu podanego jako numer.

- `I` - Macierz jednostkowa.

- `URRZ` - Układ rozwiazań równań różniczkowych

Spis treści

Wprowadzenie

Projekt ten poświecony jest problematyce rozwiazywania układów równań różniczkowych zwyczajnych (URRZ). URRZ to rodzaj zagadnienia matematycznego, w którym poszukujemy funkcji, spełniajacej jednocześnie zestaw równań różniczkowych. W ramach projektu skupimy sie na rozwiazaniach numerycznych tego rodzaju problemów, korzystajac z narzedzi dostepnych w jezyku programowania MATLAB.

Cel Projektu

Celem tego projektu jest zgłebienie metod rozwiazywania URRZ, a nastepnie zastosowanie ich do konkretnego zestawu równań. Projekt obejmuje wykorzystanie narzedzi MATLAB, takich jak procedura `ode45` oraz różnych metod numerycznych, w tym tych opartych na wzorach iteracyjnych.

Struktura Projektu

Projekt podzielony jest na trzy główne zadania. Pierwsze z nich polega na wyznaczeniu dokładnych rozwiazań URRZ za pomoca procedury `dsolve` z wykorzystaniem MATLAB Symbolic Toolbox. Drugie zadanie obejmuje rozwiazanie URRZ przy użyciu różnych metod numerycznych, takich jak procedura `ode45` oraz metody zdefiniowane wzorami. Ostatnie zadanie skupia sie na analizie zależności dokładności rozwiazań numerycznych od długości kroku całkowania.

Narzedzia i Technologie

Do realizacji projektu wykorzystamy jezyk programowania MATLAB ze szczególnym uwzglednieniem funkcji dostepnych w Symbolic Toolbox oraz różnych metod numerycznych wbudowanych w to narzedzie.

Tematyka Zadania Projektowego

Zadanie projektowe skupia sie na rozwiazaniu układu równań różniczkowych zwyczajnych (URRZ) o postaci:

$$\frac{dy_1}{dt} = -\frac{19}{3}y_1 + \frac{8}{3}y_2 + x(t)$$
$$\frac{dy_2}{dt} = -\frac{8}{3}y_1 + \frac{1}{3}y_2 + x(t)$$

gdzie $x(t) = \exp(-t) \cdot \sin(t)$, a $t \in [0, 8]$.

Metody i Algorytmy

Do rozwiazania zadania projektowego wykorzystano różne metody numeryczne, w tym:

- **Procedura dsolve (MATLAB Symbolic Toolbox):** Wyznaczenie dokładnego rozwiazania URRZ dla zerowych warunków poczatkowych.
- **Procedura ode45:** Rozwiazanie URRZ przy użyciu funkcji ode45 w MATLAB.
- **Metoda wzoru:** $y_n = y_{n-1} + \frac{h}{2}(3f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$.
- **Metoda wzoru:** $y_n = y_{n-1} + \frac{h}{12}(5f(t_n, y_n) + 8f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$.
- **Metoda wzoru:** $y_n = y_{n-1} + h\sum_{k=1}^{3} w_k f_k$, gdzie $f_k = f(t_{n-1} + c_k h, y_{n-1} + h\sum_{k=1}^{3} a_{k,k} f_k)$.

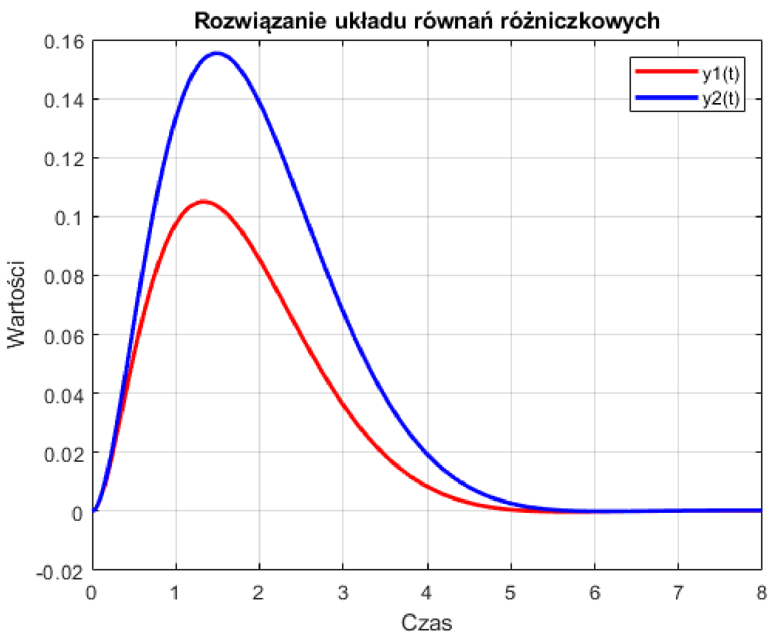
Oraz porównamy ich skuteczność. Wprowadzenie do tematyki zadania projektowego oraz opis zastosowanych metod i algorytmów stanowi wstep do bardziej szczegółowego omówienia każdej z tych cześci w kolejnych rozdziałach raportu.

Metodyka i wyniki doświadczeń

Metoda 1

Zadanie 1: Wyznacz dokładne rozwiazanie URRZ, $\tilde{y1}$ i $\tilde{y2}$, dla zerowych warunków poczatkowych za pomoca procedury dsolve (MATLAB Symbolic Toolbox).

W ramach zadania 1, celem jest wyznaczenie dokładnego rozwiazania układu równań różniczkowych zwyczajnych (URRZ) oraz funkcji wymuszajacej. W tym celu korzystamy z narzedzi MATLAB Symbolic Toolbox, które umożliwiaja symboliczne rozwiazywanie równań. Zacznijmy od zdefiniowania parametrów układu równań różniczkowych. Wartości $a = \frac{19}{3}$, $b = \frac{8}{3}$, $c = \frac{8}{3}$, $d = \frac{1}{3}$ zostały określone na podstawie specyfikacji problemu. Nastepnie definiujemy układ równań różniczkowych dy_1/dt i dy_2/dt przy użyciu narzedzi symbolicznych, uwzgledniajac zmienne t, $y_1(t)$, $y_2(t)$ oraz $x(t)$. Określamy warunki poczatkowe dla y_1 i y_2 w chwili $t = 0$, co stanowi punkt wyjścia dla procesu rozwiazywania równań. Nastepnie definiujemy funkcje wymuszajaca $x(t) = e^{-t}\sin(t)$, która reprezentuje wpływ zewnetrzny na układ. Dokonujemy podstawienia funkcji wymuszajacej do układu równań różniczkowych, uwzgledniajac wpływ czynnika zewnetrznego w procesie ewolucji zmiennych y_1 i y_2. Rozwiazujemy uzyskane równania różniczkowe symbolicznie przy użyciu funkcji `dsolve`, co pozwala na otrzymanie analitycznych rozwiazań układu przy określonych warunkach poczatkowych i funkcji wymuszajacej. Nastepnie przypisujemy uzyskane rozwiazania do zmiennych $y1\_sol$ i $y2\_sol$, co umożliwia dalsza analize wyników. W celu ilustracji uzyskanych rozwiazań generujemy wykresy funkcji $y_1(t)$ oraz $y_2(t)$ dla zadanego zakresu czasowego od $t = 0$ do $t = 8$, uwzgledniajac zdefiniowany krok czasowy dt. Przy użyciu `plotuj` ukażmy otrzymane wyniki(Rysunek 1).



Rysunek 1: Wykres przedstawiajacy rozwiazanie układu równań różniczkowych za pomoca funkcji dsolve

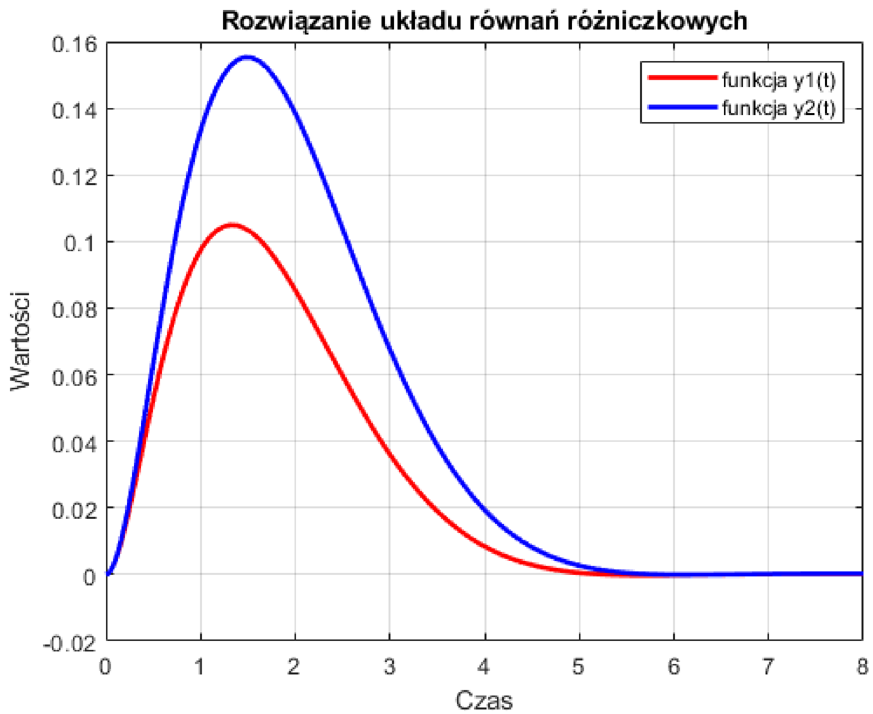
Metoda 2.1

Rozwiazanie URRZ za pomoca Procedury ode45.

W celu rozwiazania układu równań różniczkowych zwyczajnych (URRZ) zdefiniowano funkcje f zgodnie z warunkami zadania. Równania różniczkowe opisujace dynamike układu zostały zaimplementowane przy użyciu funkcji anonimowej $f(t, y)$, gdzie t to czas, a y to wektor zmiennych stanu $[y_1, y_2]$.

Określono warunki poczatkowe y_0 oraz przedział czasowy $tspan$, co stanowiło podstawe do zastosowania numerycznej metody rozwiazania układu równań ode45 w środowisku MATLAB. Wyniki uzyskano w postaci macierzy t i y, reprezentujacych odpowiednio dyskretny punkt czasowy i odpowiadajace mu wartości zmiennych stanu.

Otrzymane rozwiazania zostały zilustrowane na wykresie, gdzie zmienna t oznacza czas, a y_1 oraz y_2 przedstawiaja ewolucje odpowiednich zmiennych stanu. Wpływ funkcji wymuszajacej $x(t) = e^{-t}\sin(t)$ na rozwiazanie układu został uwzgledniony, co pozwala na pełniejsze zrozumienie dynamiki systemu. Efekty zadania zamieszczono na rysunku numer 2.



Rysunek 2: Wykres przedstawiajacy rozwiazanie układu równań różniczkowych za pomoca funkcji ode45

Metoda 2.2

Metoda zdefiniowana wzorem: $y_n = y_{n-1} + \frac{h}{2}(3f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$

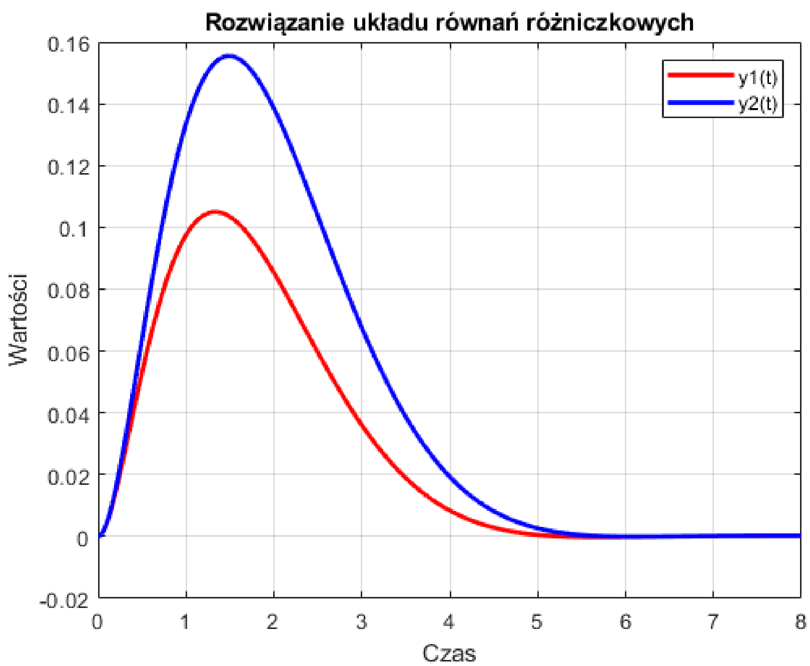
Rozwiazanie układu równań różniczkowych zwyczajnych (URRZ) jest kluczowym zagadnieniem w analizie dynamicznych procesów. W tym kontekście skonstruowano numeryczna implementacje algorytmu rozwiazujacego zadanie oznaczone jako `zadanie2_2` w środowisku MATLAB.

Pierwszym etapem było zdefiniowanie układu równań opisujacych ewolucje zmiennych $y_1(t)$ i $y_2(t)$. W tym celu skorzystano z funkcji $f(t, y) = Ay + Bx(t)$ wyznaczonej na wykładzie w której, A stanowi macierz parametrów, $y = [y_1, y_2]$, $B = [1, 1]^T$, oraz w której uwzgledniono wpływ funkcji wymuszajacej $x(t) = e^{-t}\sin(t)$. Parametry równań, takie jak $\frac{-19}{3}$, $\frac{-8}{3}$, $\frac{8}{3}$, $\frac{1}{3}$, zostały zainicjowane w macierzy A.

Nastepnie określono warunki poczatkowe $y_1(0) = 0$ i $y_2(0) = 0$, przedział czasowy $tspan$ od 0 do 8 oraz krok czasowy h. Wektor wynikowy y_values został zainicjowany wartościami poczatkowymi.

Algorytm numeryczny zaimplementowano jako petle iteracyjna, w której dla każdego kroku obliczeniowego i wyznaczano nowe wartości y_1 i y_2. Metoda przyjeta w implementacji opierała sie na schemacie przybliżonym, gdzie kolejne pochodne $f1$ i $f2$ były wykorzystywane do wyznaczenia nowych wartości zgodnie z równaniem $y\_values(:, i) = y\_values(:, i-1) + \frac{h}{2}(3f_1 - f_2)$.

Otrzymane wyniki zostały zebrane w macierzy y_values, reprezentujacej trajektorie rozwiazań układu równań w kolejnych punktach czasowych. Również tutaj wartości zostały dostosowane do formatu zgodnego z konwencja MATLAB. Finalnie wygenerowano wykres z otrzymanymi wynikami powyższej metody, który prezentujemy na rysunku numer 3.



Rysunek 3: Wykres przedstawiajacy rozwiazanie układu równań różniczkowych za pomoca metody 2.2

Metoda 2.3

Metoda zdefiniowana wzorem $y_n = y_{n-1} + \frac{h}{12}(5f(t_n, y_n) + 8f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$

Rozważajac zagadnienie rozwiazania układu równań różniczkowych zwyczajnych (URRZ), zdecydowano sie skorzystać z numerycznej metody, wykorzystujacej zdefiniowana formułe która nastepnie przekształciliśmy w poniższy sposób wykorzystujac wzór $f(t, y) = Ay + Bx(t)$:

$$y_n = y_{n-1} + \frac{h}{12}(5f(t_n, y_n) + 8f(t_{n-1}, y_{n-1}) - f(t_{n-2}, y_{n-2}))$$
$$y_n = y_{n-1} + \frac{h}{12}5Ay_n + \frac{h}{12}5Bx(t_n) + \frac{h}{12}8f(t_{n-1}, y_{n-1}) - \frac{h}{12}f(t_{n-2}, y_{n-2}))$$
$$(I - \frac{h}{12}5A)y_n = y_{n-1} + \frac{h}{12}5Bx(t_n) + \frac{h}{12}8f(t_{n-1}, y_{n-1}) - \frac{h}{12}f(t_{n-2}, y_{n-2}))$$
$$y_n = (I - \frac{h}{12}5A)^{-1}(y_{n-1} + \frac{h}{12}5Bx(t_n) + \frac{h}{12}8f(t_{n-1}, y_{n-1}) - \frac{h}{12}f(t_{n-2}, y_{n-2})))$$

Przypisano do zmiennej s wartość $\frac{h}{12}$ oraz otrzymano końcowe przekształcenie:

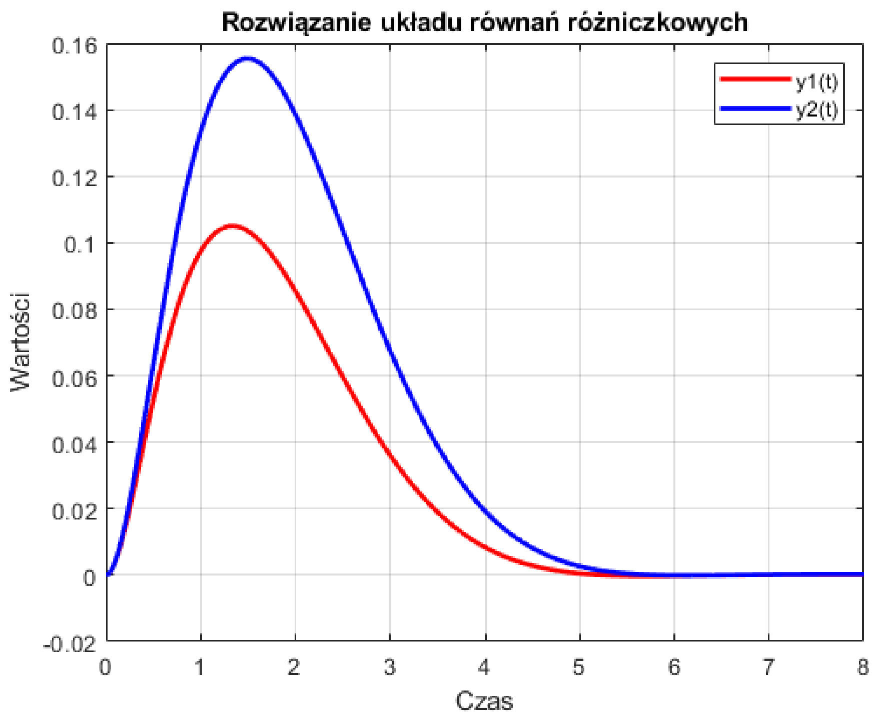
$$y_n = (I - 5sA)^{-1}(y_{n-1} + 5sBx(t_n) + 8sf_{n-1} - sf_{n-2})$$

gdzie s oznacza $\frac{h}{12}$ długości kroku całkowania , I to macierz jednostkowa, a $x(t)$ jest funkcja wymuszajaca. Przedział czasowy przyjeto jako $t \in [0, 8]$, a krok czasowy równy h.

W pierwszym etapie procesu rozwiazania, zainicjowano warunki poczatkowe y_0 i przygotowano strukture równań różniczkowych $f(t, y)$ z uwzglednieniem funkcji wymuszajacej. Nastepnie, w ramach petli numerycznej, dla każdego kroku czasowego t_n, obliczano wartości $x(t_n)$, f_{n-1}, f_{n-2}, a nastepnie dokonywano obliczeń zgodnie ze zdefiniowana metoda. Otrzymane wyniki y_n zapisywano do macierzy wynikowej y.

Podczas implementacji metody uwzgledniono różne aspekty numeryczne, takie jak formatowanie danych, optymalizacja obliczeń, a także zadbanie o efektywność i precyzje numeryki. Dla kompleksowej analizy wyników, możliwe jest porównanie uzyskanych rezultatów z wynikami innych metod rozwiazania URRZ, co pozwala ocenić skuteczność oraz stabilność metody co rozważamy zadaniu 3.

Dodatkowo, eksperymenty numeryczne moga być przeprowadzane poprzez dostosowywanie wartości kroku s w celu zbadania wpływu na stabilność numeryczna oraz dokładność wyników. Warto zauważyć, że przy zbyt dużych wartościach s może wystapić zjawisko niestabilności numerycznej. Jednakże przyjmujac optymistyczna wartość $h = 0.01$ otrzymujemy wykres prezentowany na rysunku numer 4.



Rysunek 4: Wykres przedstawiajacy rozwiazanie układu równań różniczkowych za pomoca metody 2.3

Metoda 2.4

Metoda zdefiniowana wzorem $y_n = y_{n-1} + h\sum_{k=1}^{3} w_k f_k, gdzie f_k = f(t_{n-1} + c_k h, y_{n-1} + h\sum_{k=1}^{3} a_{k,k} f_k$, gdzie dana jest tabela Butchera, $y_n = [y_1(t_n) y_2(t_n)]^T$ a funkcja $f(t_n, y_n)$ określona jest przez URRZ $\frac{dy(t)}{dt}|_{t=t_{t_n}} = f(t_n, y_n)$.

Przedstawmy teraz Tabele Butchera (Rysunek 5), która wykorzystamy w naszej obecnej metodzie.

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

Rysunek 5: Rysunek przedstawiajacy Tabele Butchera wykorzystywana w metodzie 2.4

A zatem rozpoczeto program od zdefiniowania macierzy A (macierz parametrów)

$$A = \begin{bmatrix} \frac{-19}{3} & \frac{8}{3} \\ \frac{-8}{3} & \frac{1}{3} \end{bmatrix}$$

Macierzy B (współczynników przy funkcji $x(t)$), $B = [1, 1]^T$, oraz funkcji $x(t) = e^{-t}\sin(t)$.

Nastepnie zdefiniowano zmienne a_matrix, w_vector, oraz C_vector, które zgodnie z rysunkiem przyjmuja nastepujaca postać:

$$w\_vector = \left[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right], C\_vector = [0, \frac{1}{2}, 1]$$

oraz

$$a\_matrix = \begin{bmatrix} \frac{1}{6} & 0 & -\frac{1}{6} \\ \frac{1}{12} & \frac{5}{12} & 0 \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{6} \end{bmatrix}$$

Warunki poczatkowe y_0 oraz przedział czasowy $t \in [0, 8]$ zostały zdefiniowane, a krok czasowy h ustalono na wartość przekazana jako argument funkcji. Zainicjowano wektor czasów t_values w zakresie od 0 do 8 z zadanym krokiem czasowym. Wektor y_values został zainicjowany jako macierz zerowa o rozmiarze odpowiadajacym liczbie kroków czasowych.

W pierwszej kolejności przekształćmy równania, abyśmy mogli zbudować układ równań:

$$f_1 = A(y_{n-1} + ha_{1,1}f_1 + ha_{1,2}f_2 + ha_{1,3}f_3 = Bx(t_{n-1} + c_1h)$$
$$f_2 = A(y_{n-1} + ha_{2,1}f_1 + ha_{2,2}f_2 + ha_{2,3}f_3 = Bx(t_{n-1} + c_2h)$$
$$f_3 = A(y_{n-1} + ha_{3,1}f_1 + ha_{3,2}f_2 + ha_{3,3}f_3 = Bx(t_{n-1} + c_3h)$$

Nastepnie przekształćmy go:

$$(I - ha_{1,1}A)f_1 - ha_{1,2}Af_2 + ha_{1,3}Af_3 = Ay_{n-1} + Bx(t_{n-1} + c_1h)$$
$$ha_{2,1}Af_1 - (I - ha_{2,2})Af_2 + ha_{1,3}Af_3 = Ay_{n-1} + Bx(t_{n-1} + c_2h)$$
$$ha_{2,1}Af_1 - ha_{1,2}Af_2 + (I - ha_{3,3})Af_3 = Ay_{n-1} + Bx(t_{n-1} + c_3h)$$

Otrzymaliśmy równanie typu $L \cdot g = p$, gdzie $g = (f_1, f_2, f_3)^T$. Zatem przejdźmy do implementacji powyższych własności.

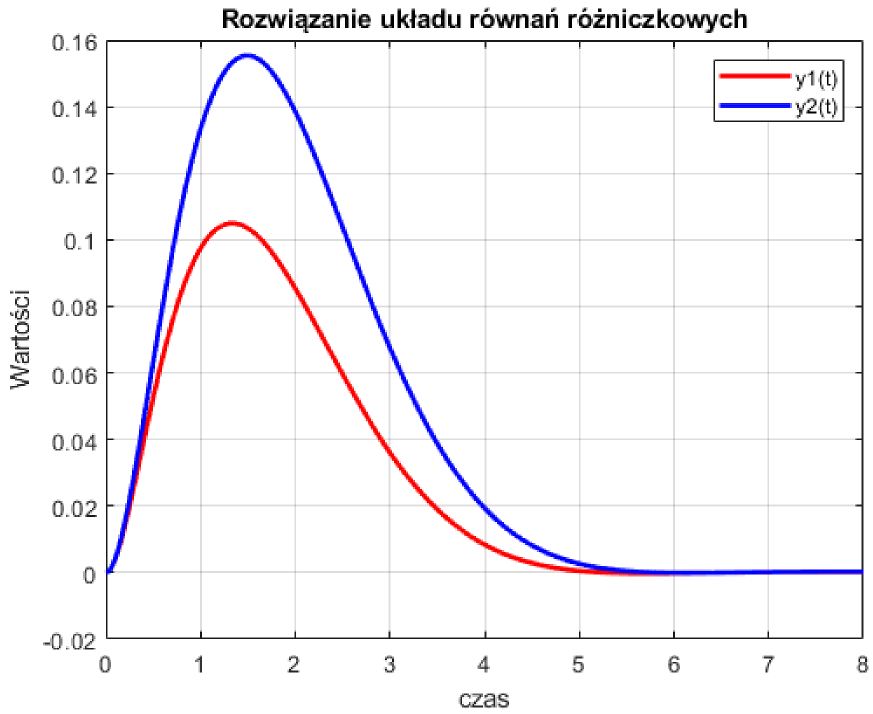
W pierwszej kolejności tworzona jest macierz równań L gdzie tworzymy macierz taka która po przemnożeniu przez wektor$g = [f_1, f_2, f_3]'$ da nam lewa strone równania za pomoca operacji na macierzach. Nastepnie, w petli for, dla kolejnych wartości czasu, obliczana jest prawa strona tego równania. Robimy to zgodnie ze wzorem za pomoca polecenia $Ay(:, i-1) + bx(t(i-1) + C(k)dt$ gdzie i to kolejnym obieg petli a k jest z zakresu 1-3. Ostatnim krokiem jest wykonanie operacji $g = \frac{P}{L}$, i tak otrzymujemy zmienne typu f_k.

Zatem ostatnim krokiem bedzie wykonanie operacji:

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

gdzie w_k to wartość z pod indeksu k w w_vector, oraz $f_k = fs[k, k+1]$ gdzie k zgodnie z suma jest z zakresu 1-3.

Podsumowujac, kod ten implementuje skomplikowany proces numerycznego rozwiazania układu równań różniczkowych, korzystajac z określonych metod numerycznych. Poniżej przedstawiamy wykres wartości uzyskanych ta metoda przy kroku próbkowania $h = 0.01$(Rysunek 6).



Rysunek 6: Wykres przedstawiajacy rozwiazanie układu równań różniczkowych za pomoca metody 2.4

Dyskusja wyników eksperymentów numerycznych

Zbadajmy zależność dokładności rozwiazań numerycznych, uzyskanych za pomoca trzech ostatnich metod zdefiniowanych w zadaniu 2, od długości kroku całkowania $h \in [h_{min}, h_{max}]$. Dobierzmy zakres zmienności $h \in [h_{min}, h_{max}]$ w taki sposób, aby zaobserwować zjawisko niestabilności numerycznej dla zbyt dużego kroku h. Jako kryterium dokładności rozwiazań przyjmij zagregowane błedy wzgledne: $\delta_1(h)$ i $\delta_2(h)$.

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

gdzie $\dot{y}_1$ i $\dot{y}_2$ to wartość funkcji uzyskanych w zadaniu 1, a $\hat{y}_1$ i $\hat{y}_2$ to ich estymaty uzyskane dla kroku całkowania h. N(h) oznacza zależna od kroku całkowania liczbe punktów rozwiazania.

Rozważmy algorytm, który ma na celu ocene dokładności różnych metod numerycznych w przybliżaniu rozwiazań równań różniczkowych w zależności od kroku czasowego.

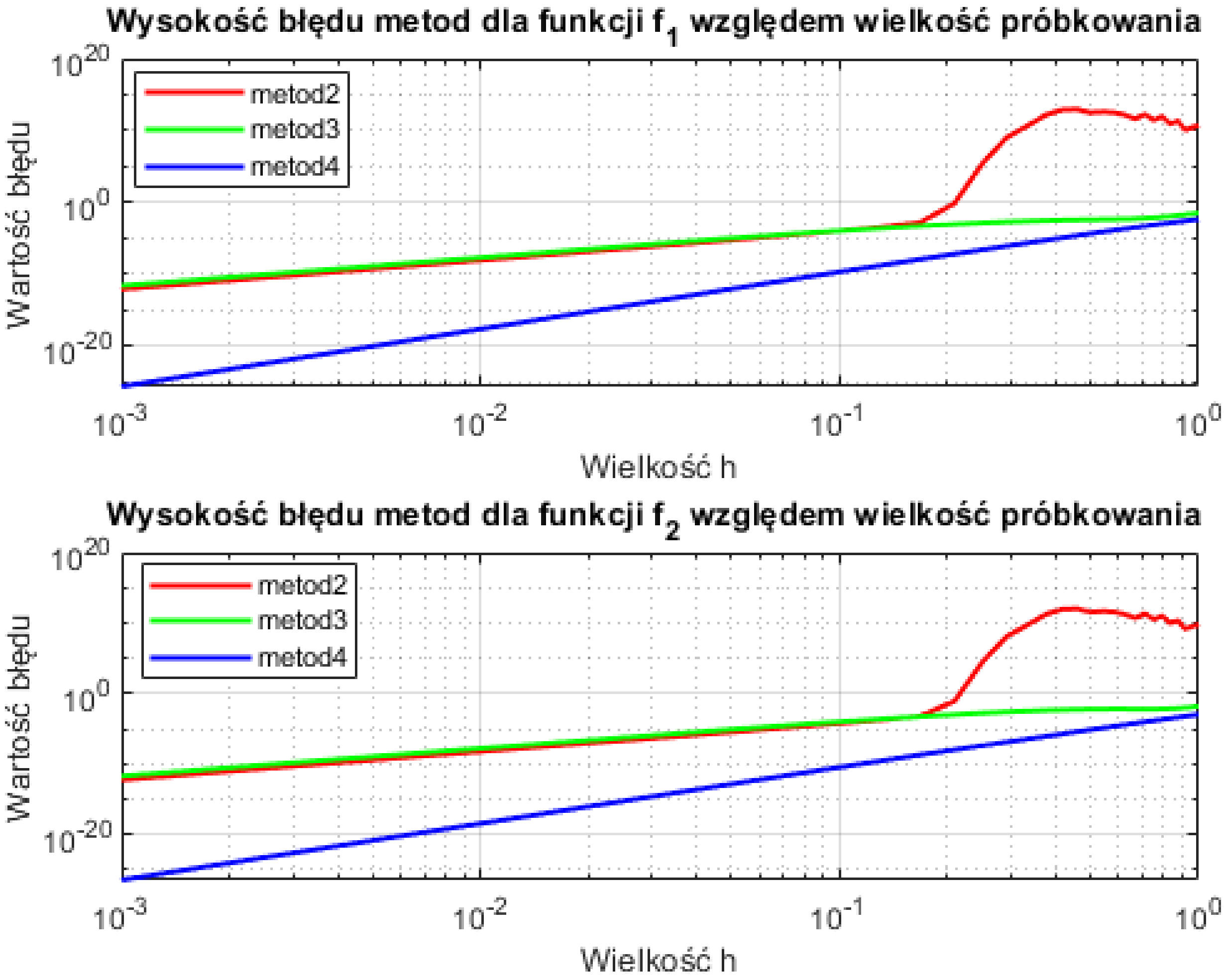
Na poczatku zdefiniowano przedział czasowy, gdzie krok czasowy zmienia sie od dt_min = 0.001 do dt_max = 0.1 z num_steps = 10 równomiernie rozmieszczonymi krokami.

Nastepnie zainicjowano wektory błedów dla różnych metod (blad1_m_2, blad2_m_2, blad1_m_3, blad2_m_3, blad1_m_4, blad2_m_4) oraz wektor osi x (osx).

W petli for iteracyjnie przeprowadzono analize dla różnych kroków czasowych. Obliczono wartość kroku czasowego dt w danej iteracji. Przeprowadzono symulacje dla różnych metod, w tym zadanie1, zadanie2_3, zadanie2_2, zadanie2_4.Nastepnie wyliczono błedy pomiedzy wynikami otrzymanymi z różnych metod, używajac funkcji esytymat. Finalnie zapisujac uzyskane błedy do odpowiednich wektorów.

Funkcja esytymat służy do pomiaru błedów miedzy dwoma rozwiazaniami numerycznymi. Mierzy kwadrat różnicy pomiedzy tymi rozwiazaniami, a nastepnie normalizuje ten bład przez kwadrat wartości referencyjnej. Takie podejście pozwala porównać efektywność różnych metod numerycznych w przybliżaniu rozwiazań równań różniczkowych.

Zapisane wartości błedów dla poszczególnych metod oraz odpowiadajace im wartości kroków czasowych (osx) stanowia podstawe analizy i porównań, co umożliwia wybór optymalnej metody numerycznej dla danego problemu. Zaprezentujmy teraz wykresy błedów poszczególnych metod (Rysunek 7). $h < 0,8$



Rysunek 7: Wykresy przedstawiajace błedy rozwiazań równań różniczkowych za pomoca metod 2.2,2.3 i 2.4

W analizie wyników eksperymentów numerycznych dotyczacych rozwiazywania URRZ, możemy jednoznacznie stwierdzić, że spośród testowanych metod, czwarta wykazuje najwyższa skuteczność w precyzyjnym oszacowywaniu wyników. Jest to szczególnie widoczne w porównaniu z innymi metodami w różnych przedziałach kroków czasowych.

W przypadku drugiej i trzeciej metody, obserwujemy podobna skuteczność dla wartości $h < 0,8$. Jednakże, dla wartości $h > 0,8$ zauważamy, że skuteczność metody drugiej znacznie spada, stajac sie istotnie gorsza od trzeciej metody.

W rezultacie, wybór optymalnej metody rozwiazania układów równań różniczkowych może zależeć od zakresu wartości kroku czasowego. Należy rozważyć charakterystyke danego problemu i dostosować metode numeryczna do jego specyfiki, biorac pod uwage optymalna precyzje wyników oraz efektywność obliczeniowa.

Wnioski

Wnioski z projektu sa kluczowym elementem podsumowujacym przeprowadzone badania i eksperymenty numeryczne. Projekt porusza różne metody numeryczne do rozwiazania układów równań różniczkowych zwyczajnych (URRZ). Porównanie wyników uzyskanych różnymi metodami pozwala zauważyć, że metoda ode45 dostepna w MATLAB czesto daje wyniki bardzo zbliżone do rozwiazania dokładnego. Jednakże, różnice te moga sie pojawić w bardziej skomplikowanych przypadkach.

Eksperymenty z różnymi wartościami kroku czasowego (h) ukazuja, że wybór tej wartości ma wpływ na dokładność uzyskiwanych wyników. Zbyt duży krok czasowy może prowadzić do utraty dokładności numerycznej, podczas gdy zbyt mały może zwiekszyć obciażenie obliczeniowe. Optymalna wartość h zależy od charakterystyki rozważanego układu równań różniczkowych.

Metoda dsolve dostepna w MATLAB Symbolic Toolbox pozwala na uzyskanie dokładnych rozwiazań URRZ. Jest to szczególnie użyteczne w przypadku prostych układów, gdzie możemy porównać wyniki numeryczne z wynikami analitycznymi. Jednakże, dla bardziej złożonych układów, ta metoda może okazać sie niewystarczajaca.

Metody numeryczne, zwłaszcza te oparte na wzorach iteracyjnych, moga być podatne na niestabilności numeryczne. Warto zwracać uwage na ten aspekt podczas wyboru metody numerycznej, zwłaszcza dla bardziej skomplikowanych układów równań różniczkowych. W naszym projekcie skupiliśmy sie na 3 takich iteracyjnych metodach gdzie po analizie błedów możemy zauważyć, że metody 2.2 i 2.3 maja podobna skuteczność w kontekście przybliżania wartości wyników URRZ natomiast metoda 2.4 mimo najcieższej implementacji wykazuje sie znacznie lepsza skutecznościa co możemy dostrzec na wykresach przedstawiajacych analize błedów.

Analiza wyników numerycznych powinna być przeprowadzona z uwzglednieniem kontekstu problemu matematycznego. Warto zastanowić sie nad sensownościa otrzymanych wyników w kontekście rzeczywistych zjawisk fizycznych, które układ równań różniczkowych może modelować.

Eksperymenty numeryczne pozwalaja na lepsze zrozumienie właściwości rozważanego układu równań różniczkowych. Poprzez badanie wpływu różnych parametrów, takich jak krok czasowy czy metoda numeryczna, można doprecyzować proces rozwiazywania problemów matematycznych.

W kontekście bardziej zaawansowanych problemów numerycznych, istnieje potrzeba optymalizacji kodu szczególnie w przypadku metody 2.4, zwłaszcza jeśli obliczenia sa intensywne. Optymalizacje moga obejmować zoptymalizowane struktury danych, unikanie zbednych obliczeń czy wykorzystanie równoległości.

Podsumowujac, projekt dostarcza wgladu w proces rozwiazywania układów równań różniczkowych zwyczajnych z wykorzystaniem różnych metod, co pozwala na lepsze zrozumienie ich zalet, ograniczeń i wpływu parametrów na wyniki. Analiza numeryczna stanowi kluczowy element w praktycznym podejściu do rozwiazywania problemów matematycznych, zwłaszcza tych zwiazanych z modelowaniem procesów dynamicznych.

Listing programów

```
function [y1_values, y2_values] =  zadanie1(dt)
% Autor: Grzegorz Prasek
% Funkcja pomocnicza reprezentujaca zadanie 1

% Parametry równań rózniczkowych
a = 19/3;
b = 8/3;
c = 8/3;
d = 1/3;

% Definicja równań rózniczkowych
syms t y1(t) y2(t) x(t)
eq1 = diff(y1) == -a*y1 + b*y2 + x;
eq2 = diff(y2) == -c*y1 + d*y2 + x;

% Warunki poczatkowe
initialConditions = [y1(0) == 0, y2(0) == 0];

% Funkcja wymuszajaca
x_t = exp(-t) * sin(t);

% Podstawienie x(t) do równań rózniczkowych
eq1 = subs(eq1, x, x_t);
eq2 = subs(eq2, x, x_t);

% Rozwiazanie równań rózniczkowych
sol = dsolve([eq1, eq2], initialConditions, 't');

% Wyświetlenie rozwiazań
y1_sol = sol.y1;
y2_sol = sol.y2;
h = dt;
% Rysowanie wykresów
t_values = 0:h:8;
y1_values = double(subs(y1_sol, t, t_values));
y2_values = double(subs(y2_sol, t, t_values));
```

Listing 1: Zadanie 1

```
function zadanie2_1()
% Autor: Grzegorz Prasek
% Funkcja pomocnicza reprezentujaca metode 2.1

% Parametry równań rózniczkowych
a = 19/3;
b = 8/3;
c = 8/3;
d = 1/3;

% Definicja równań rózniczkowych
f = @(t, y) [-a*y(1) + b*y(2) + exp(-t)*sin(t); -c*y(1) + d*y(2) + exp(-t)*sin(t)];

% Warunki poczatkowe
y0 = [0; 0];

% Przedzia␣l czasowy
tspan = [0, 8];

% Rozwiazanie ukladu równań rózniczkowych za pomoca ode45
[t, y] = ode45(f, tspan, y0);

% Wykres
figure;

plot(t, y(:, 1), 'r', 'LineWidth', 2, 'DisplayName', 'funkcja y1(t)');
hold on;
plot(t, y(:, 2), 'b', 'LineWidth', 2, 'DisplayName', 'funkcja y2(t)');
xlabel('Czas');
ylabel('Wartości');
title('Rozwiazanie ukladu równań rózniczkowych');
legend('show');
grid on;
```

Listing 2: Zadanie 2.1

```
function y_values = zadanie2_2(dt)
% Autor: Grzegorz Prasek
% Funkcja pomocnicza reprezentujaca metode 2.2

% Definicja ukladu równań rózniczkowych
x = @(t) exp(-t) * sin(t);
A = [-19/3,8/3;-8/3,1/3];
B = [1;1];
f = @(t, y) ( A*y + B*x(t) );

% Warunki poczatkowe
initial_conditions = [0; 0];

% Przedzial czasu
tspan = [0 8];

% Krok czasowy
h = dt;

% Inicjalizacja wektorów wynikowych
t = tspan(1):h:tspan(2);
y_values = zeros(2,length(t));
y_values(:,1)= initial_conditions';

% Implementacja metody 2
for i = 3:length(t)
    f1 = f(t(i-1), y_values(:, i-1));
    f2 = f(t(i-2), y_values(:, i-2));

    y_values(:, i) = y_values(:, i-1) + h/2*( 3*f1 - f2 );
end
y_values = y_values';
```

Listing 3: Zadanie 2.2

```
function y_values = zadanie2_3(dt)
% Autor: Grzegorz Prasek
% Funkcja pomocnicza reprezentujaca metode 2.3

A = [-19/3,8/3;-8/3,1/3];
B = [1;1];
% Warunki poczatkowe
initial_conditions = [0; 0];
x_t = @(t) exp(-t) * sin(t);
ode_system = @(t, y) A*y + B*x_t(t);
% Przedzial czasu
tspan = [0 8];

% Krok czasowy
h = dt;
s = h/12;

% Inicjalizacja wektorów wynikowych
t_values = tspan(1):h:tspan(2);
y_values = zeros(length(t_values), length(initial_conditions));
y_values(1,:)= initial_conditions';
y_values(2,:) = (eye(2)-h*A)\(y_values(1,:)'+h*B*x_t(t_values(2)));

% Implementacja metody 3
for n = 3:length(t_values)
    t_n = t_values(n);
    y_n_minus_1 = y_values(n-1, :)';
    y_n_minus_2 = y_values(n-2, :)';
    f_n_minus_1 = ode_system(t_n - h, y_n_minus_1);
    f_n_minus_2 = ode_system(t_n - 2*h, y_n_minus_2);

    % Wzór metody
    y_n = (eye(2) - 5*s*A)\(y_n_minus_1 + 5*s*B*x_t(t_n) + 8*s*f_n_minus_1 - s*
   f_n_minus_2);
    % Zapisanie wyniku
    y_values(n, :) = y_n';
end
```

Listing 4: Zadanie 2.3

```
function y = zadanie2_4(dt)
% Autor: Grzegorz Prasek
% Funkcja pomocnicza reprezentujaca metode 2.4

tspan = [0, 8];
y0 = [0, 0];
A = [-19/3,8/3;-8/3,1/3];
b = [ 1 ; 1];
x = @(t) ( exp(-t)*sin(t) );
h= dt;

% Inicjalizacja wektorów wynikowych
t = tspan(1):h:tspan(2);
y = zeros(2,length(t));
y(:, 1) = y0';

% Współczynniki Butchera
B = [1/6, 0, -1/6; 1/12, 5/12, 0; 1/2, 1/3, 1/6];
W = [1/6, 2/3, 1/6];
C = [ 0,  1/2, 1 ];

% Obliczanie lewej strony równań
L = [(eye(2)-h*B(1,1)*A),-h*B(1,2)*A, -h*B(1,3)*A;
    -h*B(2,1)*A,(eye(2)-h*B(2,2)*A), -h*B(2,3)*A;
    -h*B(3,1)*A,-h*B(3,2)*A, (eye(2)-h*B(3,3)*A);];

for i = 2:length(t)

    % Obliczanie prawej strony równań
    P = [A*y(:,i-1) + b*x(t(i-1) + C(1)*dt);
         A*y(:,i-1) + b*x(t(i-1) + C(2)*dt);
         A*y(:,i-1) + b*x(t(i-1) + C(3)*dt)];

    % Rozwiazywanie ukladu równań
    g = L \ P;
    % Obliczanie ostatecznej sumy
    fSum = W(1)*g([1 2]) + W(2)*g([3 4]) + W(3)*g([5 6]);
    y(:, i) = y(:, i-1) + dt*fSum;
end

y = y';
```

Listing 5: Zadanie 2.4

```
function main()
% Autor: Grzegorz Prasek
% Uwaga: By uruchomić wszytskie zadania trzeba urchomić jedynie funkcje
% main bez parametrów.
% funkcja ta zwraca wszystkie wykresy do wszystkich zadań.

% uruchoiemnie zadania 1 i zadania 2
[x_1,y_1] = zadanie1(0.01);
zad1 = [x_1',y_1'];

plotuj(0:0.01:8,zad1);
zadanie2_1();
plotuj(0:0.01:8,zadanie2_2(0.01));
plotuj(0:0.01:8,zadanie2_3(0.01));
plotuj(0:0.01:8,zadanie2_4(0.01));

% zadanie 3
% zdefiniowanie przedzialu
dt_min = 0.001;
dt_max = 1;
num_steps = 25;

blad1_m_2 = zeros(1,num_steps);
blad2_m_2 = zeros(1,num_steps);
blad1_m_3 = zeros(1,num_steps);
blad2_m_3 = zeros(1,num_steps);
blad1_m_4 = zeros(1,num_steps);
blad2_m_4 = zeros(1,num_steps);
osx = zeros(1,num_steps);

 for i = 1:num_steps
        dt = dt_min + (dt_max - dt_min) * (i - 1) / (num_steps - 1);
        osx(i) = dt;
        [zad1_1_values, zad1_2_values] = zadanie1(dt);
        zad2_3 = zadanie2_3(dt);
        zad2_2 = zadanie2_2(dt);
        zad2_4 = zadanie2_4(dt);
        blad1_m_2(i) = esytymat(zad1_1_values,zad2_2(:,1)');
        blad2_m_2(i) = esytymat(zad1_2_values,zad2_2(:,2)');
        blad1_m_3(i) = esytymat(zad1_1_values,zad2_3(:,1)');
        blad2_m_3(i) = esytymat(zad1_2_values,zad2_3(:,2)');
        blad1_m_4(i) = esytymat(zad1_1_values,zad2_4(:,1)');
        blad2_m_4(i) = esytymat(zad1_2_values,zad2_4(:,2)');
```

```
 end

 figure;

subplot(2, 1, 1);
loglog(osx, blad1_m_2, 'r', 'LineWidth', 1.5, 'DisplayName', 'metod2');
hold on;
loglog(osx, blad1_m_3, 'g', 'LineWidth', 1.5, 'DisplayName', 'metod3');
loglog(osx, blad1_m_4, 'b', 'LineWidth', 1.5, 'DisplayName', 'metod4');
xlabel('Wielkość h');
ylabel('Wartość b␣ledu');
title('Wysokość b␣ledu metod dla funkcji f_1 wzgledem wielkość próbkowania ');
legend('show');
grid on;
hold off;

subplot(2, 1, 2);
loglog(osx, blad2_m_2, 'r', 'LineWidth', 1.5, 'DisplayName', 'metod2');
hold on;
loglog(osx, blad2_m_3, 'g','LineWidth',1.5, 'DisplayName', 'metod3');
loglog(osx, blad2_m_4, 'b', 'LineWidth',1.5, 'DisplayName', 'metod4');
xlabel('Wielkość h');
ylabel('Wartość b␣ledu');
title('Wysokość b␣ledu metod dla funkcji f_2 wzgledem wielkość próbkowania ');
legend('show');
grid on;
hold off;

end

function wynik =  esytymat(solution1, solution2_numer)
   wynik = sum((solution2_numer - solution1).^2);
   wynik = wynik/sum(solution1.^2);
end
```

Listing 6: Zadanie 3 oraz uruchomienie pozostałych funkcji

```
function plotuj(t_values, y_values)
% Autor: Grzegorz Prasek
% Funkcja pomocnicza przedsawiajajaca uzyskane wartości na wykresie

figure;
plot(t_values, y_values(:, 1), 'r', 'LineWidth', 2, 'DisplayName','funkcja y1(t)');
hold on;
plot(t_values, y_values(:, 2), 'b', 'LineWidth', 2, 'DisplayName','funkcja y2(t)');
xlabel('czas');
legend('y1(t)', 'y2(t)');
ylabel('Wartości');
title('Rozwiazanie uk␣ladu równań róz̨niczkowych');
legend('show');
grid on;
end
```

Listing 7: Funkcja do ukazywania wykresów