Politechnika Krakowska
im. Tadeusza Kościuszki
Wydział Fizyki, Matematyki i Informatyki

Agnieszka Rabiej
Nr albumu: 108921

Rozwój oprogramowania do szukania osobliwości rozwiązań równań różniczkowych zwyczajnych na płaszczyźnie zespolonej

Development of software for searching singularities of ODEs in the complex plane

Praca magisterska
na kierunku FIZYKA TECHNICZNA

Praca wykonana pod kierunkiem
dr Radosław Kycia
Instytut Fizyki

Uzgodniona ocena: .....

.....
podpisy promotora i recenzenta

Kraków 2018

Spis treści

Abstrakt

Równania różniczkowe zwyczajne posiadają osobliwości dwóch typów. Równania liniowe posiadają jedynie osobliwości ustalone, które są osobliwościami współczynników w równaniu. Równania nieliniowe posiadają dodatkowo osobliwości ruchome, które są osobliwościami rozwiązań. W pracy zostały omówione oba typy osobliwości i metody ich analizy, a także opis metod Rungego-Kutty I-V rzędu, metod adaptacyjnych polegających na zmianie rozmiaru kroku całkowania oraz zagadnień programowania równoległego. Metody te zostały dołączone do oprogramowania szukającego osobliwości ruchomych na płaszczyźnie zespolonej. Implementacja została wykonana w języku C++.

Abstract

Ordinary differential equations have singularities of two types. Linear equations have only fixed singularities, which are singularities of the coefficients of the equation. Nonlinear equations, in addition, have also movable singularities, which are singularities of the solutions. In the thesis both types of singularities and methods of their analysis were outlined, as well as, the description of the Runge-Kutta I-V rank methods, adaptive methods that change the step size, and parallel programming were presented. These methods were included in the software that search for movable singularities in the complex plane. Implementation was done in C++ language.

Rozdział 1

Wstep

Większość procesów fizycznych jest opisana różnego rodzaju równaniami różniczkowymi. Najlepszym przykładem układu równań różniczkowych zwyczajnych drugiego rzędu są równania ruchu Newtona.

Rozwiązanie nawet najprostszych równań różniczkowych zwyczajnych może stanowić problem, gdyż występujące całki mogą nie wyrażać się przez funkcje elementarne. Równania liniowe drugiego rzędu posiadają najogólniejsze rozwiązania w postaci szeregów potęgowych, których promienie zbieżności na płaszczyźnie zespolonej zadane są przez osobliwości współczynników równań, tzw. ustalone osobliwości (ang. fixed singularities). Konstrukcje takich rozwiązań podaje Twierdzenie Frobeniusa (omówione w rozdziale 4.4.2). Osobliwości te mogą być trzech typów:

- bieguny
- punkty rozgałęzienia
- punkty istotnie osobliwe

W przypadku nieliniowych równań różniczkowych zwyczajnych sytuacja jest jeszcze bardziej skomplikowana, gdyż oprócz osobliwości współczynników równania pojawia się nowy typ osobliwości nazywany osobliwościami ruchomymi (ang. movable singularities). Ich położenie na płaszczyźnie zespolonej możemy określić analizując rozwiązanie, które musimy znać w skończonej formie (tzn. nie w postaci szeregu potęgowego). Nazwa 'ruchome' pochodzi od faktu, iż położenie tych osobliwości zmienia się, gdy zmieniane są dane początkowe oraz parametry równania.

Jedno z podejść do problemu rozwiązania nieliniowych równań różniczkowych zwyczajnych zaprezentowane zostało przez Paula Painlevé - wszechstronnego matematyka francuskiego i dwukrotnego ministra (4.3.2). Zgodnie z tym podejściem rozwiązanie powinno być (jednowartościową) funkcją, a dodatkowo powinna to być skończona funkcja (nie nieskończony szereg). Zatem rozpatrując rozwiązanie na płaszczyźnie zespolonej należy wiedzieć jakiego typu osobliwości występują, żeby w razie potrzeby wykonać uniformizację - z 'funkcji wielowartościowej' zrobić prawdziwą funkcję jednowartościową. Możemy to zrobić na dwa możliwe sposoby:

- wykonać cięcia dla punktów rozgałęzień i usunąć istotne osobliwości [12];
- zdefiniować powierzchnię Riemanna [12] będącą zbiorem wartości dla rozwiązania;

Dzięki takiemu zabiegowi otrzymujemy prawdziwą funkcję.

Kolejnym elementem definicji rozwiązań przez P. Painlevé jest zdefiniowanie nowych funkcji specjalnych dla równań, które nie mają rozwiązań wyrażanych przez znane nam funkcje lub rozwiązania wyrażają się w postaci nieskończonego szeregu potęgowego. Jest to podejście różne od popularnego

podejścia Cauchy'ego, gdzie istotną rolę gra lokalne rozwiązanie najczęściej w postaci szeregu potęgowego. Podejście Painlevé jest podejściem globalnym. Sytuację możemy zilustrować następująco: gdybyśmy nie zdefiniowali funkcji „specjalnych" *sin* i *cos* to wówczas rozwiązanie równania oscylatora harmonicznego wyrażone byłoby przez nieskończone szeregi, będące właśnie szeregami zbieżnymi do wspomnianych funkcji (przy założeniu, że nie używamy *exp* z wykładnikiem zespolonym, która również jest funkcją specjalną zdefiniowaną przez równanie różniczkowe zwyczajne).

Według Painlevé, najlepiej zachowujące się równania, które posiadają ruchome osobliwości w postaci biegunów, gdyż wówczas nie musimy wykonywać uniformizacji, a rozwiązanie jest funkcją meromorficzną [2]. Taka własność nazywa się własnością Painlevé. Autor tej własności rozpoczął analizę równań różniczkowych definiujące najwyższą pochodną jako wyrażenie wymierne niższych pochodnych (dokładny opis zostanie zaprezentowany poniżej) i odkrył następujące klasy:

- równania I rzędu - równanie Ricattiego oraz Weierstrassa
- równania II rzędu - 53 kanoniczne równania z których 47 jest całkowalne przy pomocy znanych funkcji, a 6 definiuje nowe funkcje transcendentalne Painlevé;

Obecnie klasyfikowane są równania w postaci wymiernej IV rzędu. Te równania są w rzeczywistości klasami równoważności równań, które różnią się od siebie transformacją Möbiusa z holomorficznymi współczynnikami, gdyż taka transformacja nie zmienia charakteru osobliwości.

W tym podejściu bardzo istotne jest położenie i typ osobliwości, aby taką uniformizację wykonać. Niestety obecne metody analityczne nie pozwalają na ogólną analizę położenia i typu ruchomych osobliwości. Analiza tych cech przebiega od równania do równania, a metody opracowane dla jednego równania nie zawsze dają się zastosować do innych równań. Dlatego tak ważne jest opracowanie metod numerycznych, które pozwalają znaleźć położenie ruchomych osobliwości rozwiązań na płaszczyźnie zespolonej, a dodatkowo pokazać ich charakter. Takie informacje często pozwalają odgadnąć właściwą metodę analityczną, umożliwiającą opisanie osobliwości w zadowalający sposób.

Rozdział 2

Cel, zakres pracy i metodyka

Celem pracy jest rozwój oprogramowania do analizy rozwiązań równań różniczkowych zwyczajnych na płaszczyźnie zespolonej opisany w [1]. Oprogramowanie to może być użyte do analizy nieznanych równań i wizualizacji znanych rozwiązań.

Praca składa się z dwóch części. Część teoretyczna opisuje w szczegółach Twierdzenie Frobeniusa o rozwiązywaniu liniowych równań różniczkowych zwyczajnych, a następnie problemy związane z pojawieniem się ruchomych osobliwości równań nieliniowych. Następnie opisane są metody numeryczne całkowania równań różniczkowych. Te wiadomości zostały wykorzystane w części drugiej do wzbogacenia programu z pracy [1] o nowe metody całkowania.

Praca w głównej mierze opisuje rozwiązania programistyczne napisane w języku C/C++. Zastosowano programowanie równoległe w celu przyśpieszenia analizy rozwiązań.

Część I

Część teoretyczna

Rozdział 3

Szereg Laurenta

Niektórych funkcji nie da się rozwinąć w szereg Taylora, w takich właśnie przypadkach z pomocą przychodzi nam rozwinięcie w szereg Laurenta. Zostało ono opublikowane w roku 1843 przez francuskiego matematyka Pierre Alphonse Laurenta, od którego nazwiska pochodzi nazwa szeregu. Specyfika rozwinięcia polega na tym, że posiada ono składniki o wykładniku ujemnym. Szeregi te wykorzystujemy do klasyfikacji ruchomych osobliwości przedstawionych w rozdziale 4. Definicje, z których korzystamy zostały zaczerpnięte z [13].

Spójrzmy na poniższe szeregi:

$$\sum_{n=0}^{\infty} c_n(z-z_0)^n \quad \text{i} \quad \sum_{n=1}^{\infty} c_{-n}(z-z_0)^{-n}. \tag{3.0.0.1}$$

Pierwszy z nich jest zbieżny w kole $K(z_0,R)$, gdzie $R = \frac{1}{limsup_{n\to\infty}\sqrt[n]{|c_n|}}$, zaś drugi jest zbieżny na zewnątrz koła $\overline{K(z_0,r)} = \{x\epsilon C : |z = z_0| > r\}$, gdzie $r = limsup_{n\to\infty}\sqrt[n]{|C_{-n}|}$.

Definicja 1 *Szeregiem Laurenta o środku w punkcie z_0 nazywamy sumę szeregów zdefiniowanych w (3.0.0.1) oraz zapisujemy jako:*

$$\sum_{n=-\infty}^{\infty} c_n(z-z_0)^n \tag{3.0.0.2}$$

Definicja 2 *Szereg Laurenta jest zbieżny, wtedy gdy każdy z szeregów (3.0.0.1) jest zbieżny .*

Z powyższej definicji wynika, że szereg Laurenta jest zbieżny w pierścieniu kołowym będącym częścią wspólną obszarów zbieżności składników sumy szeregu Laurenta.

Rozdział 4

Równania różniczkowe zwyczajne

4.1 Definicja równań różniczkowych zwyczajnych

Podrozdział został opracowany na podstawie książki [3]. Stanowi wprowadzenie do metody rozwiązywania liniowych równań różniczkowych zwyczajnych metodą Frobeniusa. Zawiera podstawowe definicje równań różniczkowych zwyczajnych liniowych oraz przykłady ich zastosowań w fizyce.

4.1.1 Równania różniczkowe zwyczajne pierwszego rzędu

Definicja 3 *Równaniem różniczkowym zwyczajnym rzędu n nazywamy równanie w postaci:*

$$F(t, x, \dot{x}, \ddot{x}, ..., x^{(n)}) = 0. \tag{4.1.1.1}$$

Gdzie t jest zmienną niezależną, a x jest zmienną zależną. Funkcję F oraz zmienną zależną $x(t)$ traktujemy jako funkcje wektorowe o wartościach w przestrzeni $\mathbb{R}^m$. Funkcje $\varphi(t)$ co najmniej klasy C^n, podstawioną odpowiednio do równania za x, w miejsce $\dot{x}$ - φ', a w miejsce $x^{(n)}$ - $\varphi(n)$, zmieniającą równanie w tożsamość, nazywamy rozwiązaniem równania (4.1.1.1).

Definicja 4 *Wykres funkcji $\varphi(t)$ w przestrzeni $\mathbb{R}^{m+1}$ zmiennych (t, x) nazywany jest krzywą całkową równania (4.1.1.1).*

W praktyce nie posługujemy się równaniami różniczkowymi w postaci (4.1.1.1), ponieważ jest to dość niewygodne. Zakładając, że najwyższa pochodna $x^{(n)}$ w sposób nietrywialny wchodzi do funkcji F, to z reguły spełnione są założenia twierdzenia o funkcji uwikłanej, a równanie (4.1.1.1) możemy rozwikłać względem najwyższej pochodnej:

$$x^{(n)}(t) = f(t, x, \dot{x}, ..., x(n-1)). \tag{4.1.1.2}$$

Równanie n-tego rzędu w powyższej postaci (4.1.1.2) możemy z łatwością sprowadzić do postaci równania pierwszego rzędu. W tym celu wprowadzamy następujące oznaczenia:

$$x_0(t) = x(t), x_1(t) = \dot{x}(t), ..., x_{n-1}(t) = x^{(n-1)}(t). \tag{4.1.1.3}$$

Z tych zmiennych utworzymy wektor:

$$\bar{x}(t) = \begin{pmatrix} x_0(t) \\ x_1(t) \\ \vdots \\ x_{n-1}(t) \end{pmatrix}. \tag{4.1.1.4}$$

Wówczas równanie (4.1.1.2) możemy przedstawić w postaci równania pierwszego rzędu:

$$\dot{\bar{x}} = g(t, \bar{x}), \tag{4.1.1.5}$$

gdzie $g(t, \bar{x})$ jest wektorem

$$\bar{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ f(t, x_0, x_1, ..., x_{n-1}) \end{pmatrix}. \tag{4.1.1.6}$$

Nas interesować będą równania pierwszego rzędu w postaci:

$$\dot{x} = f(t, x). \tag{4.1.1.7}$$

4.1.2 Równania różniczkowe zwyczajne drugiego rzędu

Wiele zagadnień fizyki opisywanych jest przez równania różniczkowe liniowe zwyczajne drugiego rzędu. Przyjmują one następującą postać:

$$\alpha(z)\omega''(z) + \beta(z)\omega'(z) + \gamma(z)\omega(z) = 0, \tag{4.1.2.1}$$

gdzie $\omega(z)$ jest poszukiwaną przez nas funkcją zmiennej zespolonej z, funkcje $\alpha(z), \beta(z)$ oraz $\gamma(z)$ są znane. Upraszczając otrzymujemy:

$$\omega''(z) + p(z)\omega'(z) + q(z)\omega(z) = 0, \tag{4.1.2.2}$$

gdzie

$$p(z) = \frac{\beta(z)}{\alpha(z)}, q(z) = \frac{\gamma(z)}{\alpha(z)}. \tag{4.1.2.3}$$

Z uwagi na zero, które występuje po prawej stronie równania (4.1.2.2), jest to równania jednorodne. Jak wiadomo, równania jednorodne posiadają dwa rozwiązania szczególne $\omega_1(z)$ i $\omega_2(z)$, które są liniowo niezależne. Rozwiązaniem ogólnym $\omega(z)$ nazywamy kombinacje rozwiązań szczególnych:

$$\omega(z) = c_1\omega_1(z) + c_2\omega_2(z), \tag{4.1.2.4}$$

gdzie c_1 i c_2 są liczbami zespolonymi ustalanymi na podstawie warunków początkowych.

4.1.3 Przykładowe zastosowania równań różniczkowych w fizyce

Równania różniczkowe znalazły wiele zastosowań zarówno w opisie zjawisk fizycznych jak i w technice [3]. Reprezentatywnymi przykładami tych zastosowań są między innymi:

- **Druga zasada dynamiki Newtona**

  Niech p_0 będzie punktem materialnym o ustalonej masie m, który porusza się w przestrzeni fizycznej $\mathbb{R}^3$. Powiązanie przyspieszenia tego punktu z siłą na niego działającą, przedstawione jest wzorem:

  $$m\ddot{P}(t) = f(.), \tag{4.1.3.1}$$

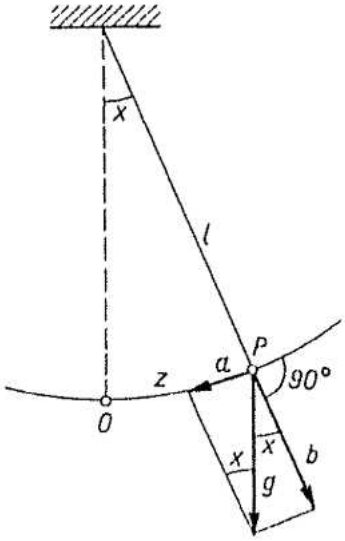
  gdzie siła f(.) jest zależna od położenia - siła grawitacyjna, prędkości - siła elektromagnetyczna, a w ogólności zależy od czasu, położenia i prędkości.

  $$f = f(t, P(t), \dot{P}(t)) \tag{4.1.3.2}$$

  Powyższe równanie (4.1.3.2) jest opisem procesu ewolucyjnego ruchu punktu materialnego p_0. Nazywamy je równaniem różniczkowym, wiąże ze sobą zmienną niezależną t oraz zmienne zależne $P(t)$ i ich pochodne $\dot{P}(t)$, $\ddot{P}(t)$.

- **Wahadło matematyczne**

  Wahadłem matematycznym nazywamy układ składający się z punktu materialnego o ustalonej masie m, zawieszonego na długiej nierozciągliwej, cienkiej nici. Punkt wykonuje wahania wokół najniżej położonego punktu O, który nazywamy środkiem wahań. Schemat przedstawiony został na rysunku (4.1)



Rysunek 4.1: Schemat na rysunku przedstawia wahadło matematyczne [3].

Oznaczmy jako z odchylenie od środka wahań oraz załóżmy, że wahadło wykonuje jedynie małe wahania. Aby znaleźć przyspieszenie wahadła a w wybranym punkcie P należy zauważyć, że siłę, która działa na punkt materialny o masie m reprezentuje składowa siły ciążenia, która jest styczna do toru wahań. Wtedy:

$$ma = -mgsinx. \tag{4.1.3.3}$$

Jest to szczególny przykład równań ruchu Newtona. W powyższym wzorze występuje znak minus, ponieważ wychylenie oraz siła mają przeciwne kierunki, g jest przyspieszeniem ziemskim, a x kątem wychylenia wahadła od pionu. Przyspieszenie wahadła możemy wyrazić poprzez drugą pochodną odchylenia z, w efekcie otrzymujemy:

$$m\ddot{z} = -mgsinx. \tag{4.1.3.4}$$

Punkt materialny porusza się po łuku, zauważmy, że z mierzymy w jego wzdłuż, dlatego

$$z = xl, \tag{4.1.3.5}$$

gdzie l jest długością wahadła. Powyższe rozważania prowadzą do otrzymania równania wahadła:

$$\ddot{x} = -\frac{g}{l}sinx. \tag{4.1.3.6}$$

Jeśli rozważamy jedynie małe drgania wahadła, to zadowalającym nas przybliżeniem jest $sinx \approx x$, zatem małe drgania dają się opisać równaniem:

$$\ddot{x} = -kx, \tag{4.1.3.7}$$

gdzie $k = l/g$, a postać zagadnień początkowych dla tego równania jest następująca:

$$\begin{aligned} \ddot{x} &= -kx, \\ x(t_0) &= x_0, \\ \dot{x}(t_0) &= x_1. \end{aligned} \tag{4.1.3.8}$$

- **Teoria obwodów elektrycznych**

  Podstawowymi prawami rządzącymi przepływem prądu w układach elektrycznych są prawa Kirchoffa. Pierwsze z nich sformułowane dla oczek sieci mówi, że suma różnic potencjałów w oczku musi być równa zero. Drugie zaś, zostało sformułowane dla węzłów sieci i mówi, że suma prądów wpływających do danego węzła jest równa sumie prądów wypływających z niego. Prawa te uzupełnimy o pewne równania, a mianowicie takie, które wiążą przepływ prądu przez indukcyjność, pojemność oraz oporność z odpowiednimi różnicami potencjałów:

$$L\frac{dj}{dt} = v, \tag{4.1.3.9}$$

$$C\frac{dv}{dt} = j, \tag{4.1.3.10}$$

$$Rj = v. \tag{4.1.3.11}$$

  Poprzez połączenie powyższych zależności z równaniem Kirchoffa dla oczka sieci, otrzymujemy równanie obwodu elektrycznego w postaci równania oscylatora harmonicznego:

$$\ddot{x} + 2k\dot{v} + \omega_0^2 v = 0, \tag{4.1.3.12}$$

  gdzie $2k = \frac{R}{L}$, a $\omega_0^2 = \frac{1}{LC}$.

W przypadku kiedy zamiast prostego obwodu elektrycznego składającego się z trzech oczek rozważamy obwód elektryczny, na który działa zewnętrzna siła elektromotoryczna $E(t)$, to równanie przyjmuje postać:

$$\ddot{v} + 2k\dot{v} + \omega_0^2 v = \omega_0^2 E(t). \tag{4.1.3.13}$$

Dla przykładu, w przypadku, gdy nie ma oporności w obwodzie, mamy do czynienia z drganiami swobodnymi, wtedy nasze równanie redukuje się do postaci równania oscylatora harmonicznego bez tłumienia:

$$\ddot{v} + \omega_0^2 v = 0. \tag{4.1.3.14}$$

Rozwiązaniem ogólnym takiego równania jest:

$$v(t) = c_1 cos\omega_0 t + c_2 sin\omega_0 t. \tag{4.1.3.15}$$

Przekształcając to równanie do nieco innej postaci, poprzez wprowadzenie nowych stałych dowolnych:

$$\begin{aligned} c_1 &= Acos\delta, \\ c_2 &= Asin\delta. \end{aligned} \tag{4.1.3.16}$$

Wtedy dostajemy wzór opisujący swobodne drgania elektryczne z częstotliwością ω_0, o amplitudzie A i przesunięciu fazowym δ:

$$v(t) = Acos(\omega_0 t - \delta). \tag{4.1.3.17}$$

4.2 Równania różniczkowe zwyczajne liniowe z osobliwościami i metoda Frobeniusa

4.2.1 Klasyfikacja punktów osobliwych

Definicja 5 *Funkcja $f(x)$ jest analityczna w punkcie x_0, tylko w przypadku, jeśli w otoczeniu tego punktu jest równa sumie swojego szeregu Taylora [5]*

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} f^{(k)}(x_0)(x - x_0)^k \tag{4.2.1.1}$$

dla $|x - x_r| < R$, gdzie R jest nieujemną liczbą rzeczywistą .

Definicja 6 *Punktem nieosobliwym [5] równania w postaci $y'' + p(x)y' + g(x)y = 0$ nazywamy taki punkt x_0, w którego otoczeniu funkcje $p(x)$ i $g(x)$ są analityczne .*

Definicja 7 *Regularnym punktem osobliwym [5] równania $y'' + p(x)y' + g(x)y = 0$ nazywamy taki punkt x_0, w którego otoczeniu funkcje $p(t)$ oraz $q(t)$ nie są analityczne, ale funkcje $(x - x_0)p(x)$ oraz $(x - x_0)^2 q(x)$ są analityczne.*

Definicja 8 ***Twierdzenie Fuchsa** [4]. Dla punktu regularnie osobliwego lub punktu regularnego możemy zawsze znaleźć niezerową liczbę rozwiązań szczególnych równania różniczkowego liniowego, poprzez rozwinięcie w szereg wokół tego punktu:*

$$\omega(z) = \sum_{n=0}^{\infty} a_n (z - z_0)^{n+c}, \tag{4.2.1.2}$$

gdzie c jesz liczbą zespoloną, którą chcemy wyznaczyć, a promień zbieżności jest określony przez odległość do najbliższego punktu osobliwego tego równania.

W tabeli poniżej przedstawione zostały przykłady rozwiązań jednorodnych równań różniczkowych zwyczajnych drugiego rzędu wraz z punktami osobliwymi regularnymi (RPO) oraz nieregularnymi (NPO).

Równanie	RPO	NPO
Oscylatora harmonicznego $\omega" + \omega^2\omega = 0$	brak	∞
Hermite'a - kwantowy oscylator harmoniczny $\omega" + 2z\omega' + 2\alpha\omega = 0$	brak	∞
Laguerre'a - atom wodoru $z\omega" + (1-z)\omega' + a\omega = 0$	0	∞
Bessela $z^2\omega" + z\omega' + (z^2 - v^2)\omega = 0$	0	∞
Konfluentne $z\omega" + (\gamma - z)\omega' + \alpha\omega = 0$	0	∞
Czebyszewa $(1-z^2)\omega - z\omega' + n^2\omega = 0$	-1,1,∞	brak
Legendre'a - kwantowy kręt orbitalny $(1-z^2)\omega - 2z\omega' + [l(l+1) - \frac{m^2}{1-z^2}\omega = 0$	-1,1,∞	brak
Gaussa (hipergeometryczne) $z(z-1)\omega" + [(1+\alpha+\beta)z - \gamma]\omega' + \alpha\beta\omega = 0$	0,1,∞	brak

Rysunek 4.2: Przykłady rozwiązań jednorodnych równań różniczkowych zwyczajnych drugiego rzędu wraz z punktami osobliwymi regularnymi (RPO) oraz nieregularnymi (NPO) [4].

4.2.2 Metoda Frobeniusa

Metoda Frobeniusa pozwala nam na rozwiązywanie równań różniczkowych zwyczajnych poprzez rozwinięcie w szeregi potęgowe. Metoda została opracowana na podstawie [6].

Główną ideą metody Frobeniusa jest poszukiwanie rozwiązań w postaci:

$$(x - x_0)^r \sum_{n=0}^{\infty} a_n (x - x_0)^n. \tag{4.2.2.1}$$

Rozwiążmy równanie:

$$y'' + p(x)y' + g(x)y = 0. \tag{4.2.2.2}$$

Niech x_0 będzie pojedynczym punktem osobliwym równania. Otrzymujemy:

$$p(x)(x - x_0) = \sum_{n=0}^{\infty} p_n (x - x_0)^n, \tag{4.2.2.3}$$

oraz

$$q(x)(x - x_0)^2 = \sum_{n=0}^{\infty} q_n (x - x_0)^n, \tag{4.2.2.4}$$

z pewnymi promieniami zbieżności.

Aby obliczenia były bardziej czytelne, ustalmy, że $x_0 = 0$, po podstawieniu x_0 do równania (4.2.2.1) otrzymujemy:

$$y = x^r \sum_{n=0}^{\infty} a_n x^n. \tag{4.2.2.5}$$

Podstawiamy y do równania (4.2.2.2), co w rezultacie daje:

$$(x^r \sum_{n=0}^{\infty} a_n x^n)'' + p(x)(x^r \sum_{n=0}^{\infty} a_n x^n)' + q(x)(x^r \sum_{n=0}^{\infty} a_n x^n) = 0. \tag{4.2.2.6}$$

Wyliczamy pochodne poszczególnych składników równania (4.2.2.6):

$$\begin{aligned}(x^r \sum_{n=0}^{\infty} a_n x^n)'' &= (\sum_{n=0}^{\infty} a_n x^{n+r})'' = \\ &= \sum_{n=0}^{\infty}(n+r)(n+r-1)a_n x^{n+r-2},\end{aligned} \tag{4.2.2.7}$$

$$\begin{aligned}p(x)(x^r \sum_{n=0}^{\infty} a_n x^n)' &= p(x)(\sum_{n=0}^{\infty} a_n x^{n+r})' = \\ = p(x)(\sum_{n=0}^{\infty}(n+r)a_n x^{n+r-1}) &= (p(x)x)(\sum_{n=0}^{\infty}(n+r)a_n x^{n+r-2}) = \\ = (\sum_{n=0}^{\infty} p_n x^n)(\sum_{n=0}^{\infty}(n+r)a_n x^{n+r-2}) &= \sum_{n=0}^{\infty}[\sum_{m=0}^{\infty} p_{n-m}(m+r)a_m]x^{n+r-2},\end{aligned} \tag{4.2.2.8}$$

$$\begin{aligned}q(x)(x^r \sum_{n=0}^{\infty} a_n x^n) &= x^(r-2)(\sum_{n=0}^{\infty} q_n x^n)(\sum_{n=0}^{\infty} a_n x^n) = \\ &= \sum_{n=0}^{\infty}[\sum_{m=0}^{n} q_{n-m}a_m]x^{n+r-2}).\end{aligned} \tag{4.2.2.9}$$

Z powyższych obliczeń otrzymujemy równanie w postaci:

$$\sum_{n=0}^{\infty}\{(n+r)(n+r-1)a_n + \sum_{n=0}^{\infty}[(m+r)p_{n-m} + q_{n-m}]a_m\}x^{n+r-2} = 0 \tag{4.2.2.10}$$

lub w postaci równoważnej

$$\sum_{n=0}^{\infty}\{[(n+r)(n+r-1) + (n+r)p_0 + q_0]a_n + \sum_{m=0}^{n-1}[(m+r)p_{n-m} + q_{n-m}]a_m\}x^{n+r-2} = 0. \tag{4.2.2.11}$$

Rozwiązaniem dla powyższych równań są:
$(n = 0)$:

$$[r(r-1) + p_0 r + q_0]a_0 = 0, \tag{4.2.2.12}$$

$(n \geq 0)$:

$$[(n+r)(n+r-1) + (n+r)p_0 + q_0]a_n + \sum_{m=0}^{n-1}[(m+r)p_{n-m} + q_{n-m}]a_m = 0, \tag{4.2.2.13}$$

wyróżniamy $n = 0$, ponieważ chcemy, aby $a_n \neq 0$ (naturalnie przy $a_n = 0$ dostaniemy $y = x^{r+1}\sum_{m=0}^{\infty} b_m x^m$, gdzie $b_m = a_{m+1}$) dostajemy warunek na r:

$$r(r+1) + p_0 r + q_0 = 0. \tag{4.2.2.14}$$

Powyższe równanie nazywamy wskaźnikowym i daję nam ono dwa rozwiązania: r_1 i r_2, zwane wykładnikami pojedynczego punktu osobliwego dla $x = 0$.

Rozważmy teraz trzy przypadki rozwiązań:

- Jeśli $r_1 \neq r_2$ i $r_1 - r_2$ nie daje nam liczby naturalnej, to istnieją dwa liniowo niezależne rozwiązania:

$$y_1(x) = |x - x_0|^{r_1} \sum_{n=0}^{\infty} a_n (x - x_0)^n, \tag{4.2.2.15}$$

$$y_2(x) = |x - x_0|^{r_2} \sum_{n=0}^{\infty} \overline{a}_n (x - x_0)^n. \tag{4.2.2.16}$$

  Współczynniki a_n i $\overline{a}_n$ określamy przez następującą relacje rekurencyjną:

$$[(n+r)(n+r+1) + (n+r)p_0 + q_0]a_n + \sum_{m=0}^{n-1}[(m+r)p_{n-m} + q_{n-m}]a_m = 0. \tag{4.2.2.17}$$

- W przypadku, kiedy $r_1 = r_2$ to y_1 określone jest tak samo jak we wcześniejszym przypadku, a y_2 określone jest wzorem:

$$y_2(x) = y_1(x) ln|x - x_0|^{r_1} \sum_{n=1}^{\infty} d_n (x - x_0)^n. \tag{4.2.2.18}$$

- Kiedy $r_1 - r_2$ jest liczbą naturalną, to zakładamy, że r_1 jest 'większym korzeniem', innymi słowy stanowi większą część rozwiązania, a $Re(r_1) \geq Re(r_2)$, wtedy y_1 jest takie samo jak we wcześniejszych przypadkach, a y_2 wynosi:

$$y_2(x) = c y_1(x) ln|x - x_0| + |x - x_0|^{r_2} \sum_{n=0}^{\infty} e_n (x - x_0)^n. \tag{4.2.2.19}$$

Zauważmy, że c może być równe 0.

We wszystkich wyżej wymienionych przypadkach rozwiązania są zbieżne dla $0 < |x - x_0| < q$.

4.3 Nieliniowe równania różniczkowe zwyczajne - ruchome osobliwości

Nieliniowe równania różniczkowe poza osobliwościami ustalonymi znanymi z Twierdzenia Frobeniusa z podrozdziału 4.2.1 posiadają również ruchome osobliwości (ang. movable singularities), które są osobliwościami rozwiązań. Te osobliwości poruszają się po płaszczyźnie zespolonej, gdy zmieniamy warunki początkowe, a także parametry równania, stąd nazwa ruchome.

Z analizy zespolonej wiemy [2], że funkcje na płaszczyźnie zespolonej mogą posiadać trzy typy osobliwości w których zachowanie funkcji jest następujące (C to liczba zespolona):

- osobliwości usuwalne, które nie są osobliwościami, jak w przypadku funkcji $\frac{\sin(x)}{x}$ w $x = 0$;
- bieguny dla funkcji holomorficznych, w otoczeniu których główny wkład ma postać $\frac{1}{(x-C)^n}$, gdzie $n \in \mathbb{N}$; Wówczas w szeregu Laurenta funkcji w okolicy osobliwego punktu mamy skończoną liczbę wyrazów o ujemnych potęgach.
- punkty rozgałęzienia postaci $(x-C)^\alpha$, gdzie α jest liczbą wymierną, niecałkowitą; W otoczeniu punktu rozgałęzienia mamy zdefiniowaną 'funkcję wielowartościową', która nie jest funkcją w klasycznym tego słowa znaczeniu.
- punkty istotnie osobliwe dla funkcji holomorficznych, w otoczeniu których szereg Laurent funkcji analitycznej ma nieskończoną liczbę wyrazów o ujemnych potęgach;

Powyższe osobliwości stanową klasę osobliwości izolowanych, gdyż są odseparowane od siebie. Jednak rozwiązania mogą również posiadać osobliwości nieizolowane, jak np. naturalne granice lub punkty skupienia osobliwości. Osobliwości nieizolowanych nie będziemy w pracy analizować, gdyż ich teoria nie jest dostatecznie ugruntowana w matematyce.

Najprostszym przykładem nieliniowego równania generującego osobliwość ruchomą jest

$$\frac{dy(x)}{dx} + y(x)^2 = 0, \tag{4.3.0.1}$$

które posiada rozwiązanie

$$y(x) = \frac{-1}{x-C}. \tag{4.3.0.2}$$

Stała C jest określona przez warunek początkowy $y(0) = \frac{1}{C}$ dla $C \neq 0$. Łatwo zauważyć, że dla $x = C$ rozwianie (4.3.0.2) posiada biegun. Jest to osobliwość ruchoma, gdyż zmieniając wartość warunku początkowego osobliwość zmienia położenie na płaszczyźnie zespolonej, więc jest to osobliwość ruchoma.

4.3.1 Klasyfikacja ruchomych osobliwości

Interesować nas będzie nie samo istnienie osobliwości, ale jej rodzaj, ponieważ to daje nam prawdziwy wgląd w globalną strukturę rozwiązań. W przypadku kiedy napotykamy na ruchomą osobliwość możemy spodziewać się trzech odmiennych typów tej osobliwości. Podrozdział stanowi opis rodzajów osobliwości ruchomych oraz przykłady równań różniczkowych zwyczajnych nieliniowych, w których występują.

- **Biegun** (ang. pole) [8], [9], [10]. Z biegunem funkcji meromorficznej $f(z)$ mamy do czynienia, kiedy funkcja nie jest ograniczona w otoczeniu tego punktu osobliwego $z = a$, a ponadto:

$$\lim_{z \to a} f(z) = \infty. \tag{4.3.1.1}$$

Możemy również określić, którego rzędu jest biegun. Sprowadza się to do określenia z ilu wyrazów składa się szereg Laurenta powstały z rozwinięcia części osobliwej wokół punktu a. Zauważmy, że jeśli punkt a jest ruchomą osobliwością w postaci bieguna m-krotnego funkcji $f(z)$ to funkcja

$$g(z) = \frac{1}{f(z)} \tag{4.3.1.2}$$

jest meromorficzna i w punkcie osobliwym a posiada zero m-krotne. Podobnie w sytuacji, gdy funkcja $f(z)$ posiada zero m-krotne w punkcie a to funkcja $g(z)$ posiada osobliwość w postaci bieguna m-krotnego w punkcie a.

Przykład: Rozważmy funkcje:

$$f(z) = \frac{z+2}{(z-5)^2(z+7)^3}. \tag{4.3.1.3}$$

Funkcja jest meromorficzna w całej przestrzeni sfery Riemanna. Posiada biegun rzędu 2 w punkcie $z = 5$, biegun rzędu 3 w punkcie $z = -7$ oraz pojedyncze zero w punkcie $z = -2$ i poczwórne zero w nieskończoności.

- **Rozgałęzienia** (ang. branch point) [11]. Punkt rozgałęzienia, to taki punkt z_0 funkcji analitycznej, że jeśli rozszerzymy dziedzinę tej funkcji dookoła punktu z_0 tworząc łańcuch kół $K_0, K_1, ...K_n$, spełniający następujące warunki:
  - każde z kół zawiera punkt z_0,
  - każde jest przedłużeniem analitycznym - rozszerzeniem dziedziny poprzedniego,
  - każde n-te koło posiada część wspólną z kołem K_0 nie będącą punktem z_0.

  Otrzymamy w kole K_n funkcję o wartościach innych niż w części wspólnej $K_n \bigcap K_0$. Uściślając, jeśli przemieszczamy punkt z dookoła punktu z_0 po krzywej zamkniętej, to wartości funkcji $f(z)$ będą się zmieniały w sposób ciągły, aczkolwiek na końcu tej pętli wartość funkcji $f(z)$ będzie się różniła od wartości na początku pętli mierzonej w tym samym punkcie.

  Zauważmy, że mamy do czynienia z funkcją, która nie jest holomorficzna na przestrzeni pierścienia otaczającego punkt rozgałęzienia, dlatego nie możemy rozwinąć jej w szereg Laurenta w tym pierścieniu. Możemy jednak określić jej gałąź, która jest jednoznaczna w jednospójnym obszarze (takim, w którym dwa dowolne punkty możemy połączyć drogą oraz każde dwie drogi łączące dwa punkty należące do przestrzeni są homotopijne) niezawierającym punktu rozgałęzienia.

  Przykład:

  W funkcji:

  $$log(z - z_0) \tag{4.3.1.4}$$

  punktem rozgałęzienia jest punkt z_0. Podobnie zachowuje się funkcja $\sqrt{z - z_0}$.

- **Osobliwość istotna** (ang. essential singularity)[7]. Trzeci rodzaj osobliwości reprezentują osobliwości istotne. Funkcje posiadające takie osobliwości mają w szeregu Laurenta nieskończenie wiele ujemnych potęg. Punkt a nazywany jest istotną osobliwością funkcji f, jeśli osobliwość nie jest ani biegunem, ani usuwalną osobliwością(eng. removable singularity).

  Przykład: Funkcja:

  $$f(z) = e^{1/z} \tag{4.3.1.5}$$

  posiada istotną osobliwość w punkcie $z = 0$.

4.3.2 Własność Painlevé

Osobliwości rozwiązań równań różniczkowych zwyczajnych stały się przedmiotem zainteresowania francuskiego matematyka Paula Painlevé. Wprowadził on własność Painlevé, którą posiadają równania, których osobliwości rozwiązań to jedynie bieguny. Ze względu na fakt, iż typ osobliwości nie zmienia się poprzez transformację Möbiusa:

$$f(z) = \frac{az+b}{cz+d}, \tag{4.3.2.1}$$

gdzie a, b, c, d są funkcjami holomorficznymi oraz $ad - cb \neq 0$, to klasyfikację osobliwości można wykonać na klasach abstrakcji równań różniczkowych indukowanych przez tą transformację.

Painlevé przeprowadził klasyfikację równań z prawą stroną będącą wyrażeniem wymiernym i otrzymał kilka równań posiadających wspomnianą własność. Do równań pierwszego rzędu posiadających własność Painlevé zaliczył równanie Weierstrassa oraz równanie Ricattiego. Wśród równań II rzędu znalazł on 53 równania posiadające tę własność. Okazało się ze 47 z nich można opisać przez istniejące już funkcje, zaś 6 pozostałych nie daje się sprawdzić do znanych równań i rozwiązać w postaci znanych nam funkcji. Painlevé zdefiniował więc nowe funkcje rozwiązujące te właśnie równania [12]. Poniżej przedstawiam 6 nieredukowalnych równań znalezionych przez Paula Painlevé. Funkcje, które są rozwiązaniami tych równań nazywamy funkcjami transcendentalnymi Painlevé.

- **I (Painlevé)**

$$\frac{d^2y}{dt^2} = 6y^2 + t, \tag{4.3.2.2}$$

- **II (Painlevé)**

$$\frac{d^2y}{dt^2} = 2y^3 + ty + \alpha, \tag{4.3.2.3}$$

- **III (Painlevé)**

$$ty\frac{d^2y}{dt^2} = t(\frac{dy}{dt})^2 - y\frac{dy}{dt} + \delta t + \beta y + \alpha y^3 + \gamma t y^4, \tag{4.3.2.4}$$

- **IV Gambier**

$$y\frac{d^2y}{dt^2} = \frac{1}{2}(\frac{dy}{dt})^2 + \beta + 2(t^2 - \alpha)y^2 + 4ty^3 + \frac{3}{2}y^4, \tag{4.3.2.5}$$

- **V Gambier**

$$\frac{d^2y}{dt^2} = (\frac{1}{2y} + \frac{1}{y-1})(\frac{dy}{dt})^2 - \frac{1}{t}\frac{dy}{dt} + \frac{(y-1)^2}{t^2}(\alpha y + \frac{\beta}{y}) + \gamma\frac{y}{t} + \delta\frac{y(y+1)}{y-1}, \tag{4.3.2.6}$$

- **VI R.Fuchs**

$$\begin{aligned}\frac{d^2y}{dt^2} = \frac{1}{2}(\frac{1}{y} + \frac{1}{y-1} + \frac{1}{y-t})(\frac{dy}{dt})^2 - (\frac{1}{t} + \frac{1}{t-1} + \frac{1}{y-t})\frac{dy}{dt} \\ + \frac{y(y-1)(y-t)}{t^2(t-1)^2}(\alpha + \beta\frac{t}{y^2} + \gamma\frac{t-1}{(y-1)^2} + \delta\frac{t(t-1)}{(y-t)^2})\end{aligned}. \tag{4.3.2.7}$$

W powyższych wzorach α, β, γ i δ są stałymi zespolonymi. Przez przeskalowanie y oraz t możemy wybrać dwa parametry dla III typu równania i jeden parametr dla typu V, dlatego wnioskujemy, że te typy mają tak naprawdę odpowiednio 2 i 3 niezależne parametry. Osobliwości dla typów III, V i VI znajdujemy w ∞ i w 0 oraz w 1 dla VI typu. Dla pierwszego typu osobliwości występują w postaci podwójnego bieguna. Dla takich osobliwości szereg Laurenta ma postać:

$$(z-z_0)^{-2} - \frac{z_0}{10}(z-z_0)^2 - \frac{1}{6}(z-z_0)^3 + h(z-z_0)^4 + \frac{z_0^2}{300}(z-z_0)^6 + ... \tag{4.3.2.8}$$

Dla II typu wszystkie osobliwości mają postać pojedynczych biegunów.

Rozdział 5

Numeryczne metody rozwiązywania równań różniczkowych zwyczajnych

Częściowe informacje na temat lokalizacji ruchomych osobliwości możemy uzyskać za pomocą metod numerycznych [19]. Najprostszym, a zarazem najbardziej intuicyjnym sposobem znajdowania osobliwości jest całkowanie równania wzdłuż pewnej ścieżki leżącej na płaszczyźnie zespolonej i kończenie całkowania w momencie, gdy jakiś stan sugeruje sąsiedztwo osobliwości. Jeśli ścieżki całkowania są wystarczająco gęste, metoda ta może dać nam pewne prognozy na temat zachowywania się rozwiązania i struktury osobliwości.

5.1 Metoda Eulera

Dane jest równanie postaci $y' = f(x, y)$ z warunkami początkowymi $(x_0, y_0) : y_0 = y(x_0)$, poruszamy się z krokiem h zatem kolejne punkty na osi x to:

$$x_{n+1} = x_n + h, \tag{5.1.0.1}$$

z definicji pochodnej $y' = \frac{\Delta y}{h}$ zatem:

$$f(x_n, y_n) = y' = \frac{\Delta y}{h}, \tag{5.1.0.2}$$

po przekształceniu powyższego wzoru otrzymujemy:

$$\Delta y = hf(x_n, y_n). \tag{5.1.0.3}$$

Chcemy otrzymać wzór na y_{n+1}, dlatego do wzoru $y_{n+1} = y_n + \Delta y$ podstawiamy wyliczone Δy (5.1.0.3). Tym sposobem dochodzimy do postaci:

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{5.1.0.4}$$

Porównujemy wynik z rozwinięciem w szereg Taylora:

$$y_{n+1} = y(x_{n+1}) = y(x_n + h) = y_n + hf(x_n, y_n) + \frac{f^{(2)}(\xi)}{2!}h^2, \tag{5.1.0.5}$$

gdzie $x_n < \xi < x_{n+1}$.

Błąd przybliżenia jest rzędu $O(h^2)$. Większą dokładność przybliżenia otrzymalibyśmy przez zmniejszenie kroku iteracji. W praktyce, metoda ta nie jest zalecana, ze względu na swoją niedokładność w porównaniu z innymi metodami tego samego stopnia.

5.2 Metody RK

Metoda Rungego-Kutty II rzędu

W tym przypadku wzór na całkowanie równania różniczkowego wygląda następująco:

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h, \tag{5.2.0.1}$$

gdzie:

$$k_1 = f(x_i, y_i), \tag{5.2.0.2}$$

$$k_2 = f(x_i + p_1 h; y_i + q_{11} k_1 h), \tag{5.2.0.3}$$

a zależności pomiędzy poszczególnymi współczynnikami przyjmują postać:

$$\begin{aligned} a_1 + a_2 &= 1, \\ a_2 p_1 &= \frac{1}{2}, \\ a_2 q_{11} &= \frac{1}{2}. \end{aligned} \tag{5.2.0.4}$$

Aby wyznaczyć współczynniki a_1, a_2, p oraz q posłużmy się rozwinięciami w szeregi Taylora.
Zacznijmy od rozważenia funkcji $f(x, y)$.
Otrzymujemy:

$$y_{i+1} = y_1 + f(x_i, y_i)h + \frac{f'(x_i, y_i)h^2}{2!} + O(h^3), \tag{5.2.0.5}$$

$$f'(x_i, y_i) = \frac{\partial f(x_i, y_i)}{\partial x} + \frac{\partial f(x_i, y_i)}{\partial y}\frac{dy}{dx}, \tag{5.2.0.6}$$

$$y_{i+1} = y_i + f(x_i, y_i)h + (\frac{\partial f(x_i, y_i)}{\partial x} + \frac{\partial f(x_i, y_i)}{\partial y}\frac{dy}{dx})\frac{h^2}{2!} + O(h^3). \tag{5.2.0.7}$$

Weźmy teraz funkcje $g(x, y)$, jeśli jest ona ciągła to możemy przedstawić ją w postaci:

$$g(x + r, y + s) = g(x, y) + r\frac{\partial g}{\partial x} + s\frac{\partial g}{\partial y} + ... \tag{5.2.0.8}$$

zatem, k_2 możemy zapisać w postaci:

$$k_2 = f(x_i + p_1 h; y_i + q_{11} k_1 h) = f(x_i, y_i) + p_1 h\frac{\partial f}{\partial x} + q_{11} k_1 h\frac{\partial f}{\partial y} + O(h^2), \tag{5.2.0.9}$$

gdzie $O(h^2)$ jest wyrazem wyższego rzędu.
Po podstawieniu, otrzymujemy:

$$y_{i+1} = y_i + a_1 h f(x_i, y_i) + a_2 h f(x_i, Y_i) + a_2 p_1 h^2\frac{\partial f}{\partial x} + a_2 q_{11} h^2\frac{\partial f}{\partial y} f(x_i, y_i) + O(h^3). \tag{5.2.0.10}$$

Uporządkujmy powyższy wzór, w efekcie dostajemy:

$$y_{i+1} = y_i + hf(x_i, y_i)[a_1 + a_2] + [a_2 p_1 \frac{\partial f}{\partial x} + a_2 p_{11} \frac{\partial f}{\partial y} f(x_i, y_i)]h^2 + O(h^3). \tag{5.2.0.11}$$

Powyższe wyrażenie powinno być równoważne z rozwinięciem funkcji w szereg Taylora:

$$y_{i+1} = y_i + f(x_i, y_i)h + (\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx})\frac{h^2}{2!}, \tag{5.2.0.12}$$

stąd właśnie dochodzimy do podanych wcześniej relacji (5.2.0.4):

$$\begin{aligned} a_1 + a_2 &= 1, \\ a_2 p_1 &= \frac{1}{2}, \\ a_2 q_{11} &= \frac{1}{2}. \end{aligned} \tag{5.2.0.13}$$

bazując na powyższych relacjach, możemy wyprowadzić wzory dla metod R-K drugiego rzędu:

- Metoda Heun'a.

  Bazuje na założeniu: $a_2 = \frac{1}{2}$, korzystając z zależności (5.2.0.13) $a_1 = 1/2$, $p_1 = q_{11} = 1$,

$$y_{i+1} = y_i + (\frac{1}{2}k_1 + \frac{1}{2}k_2)h, \tag{5.2.0.14}$$

$$k_1 = f(x_i, y_i), \tag{5.2.0.15}$$

$$k_2 = f(x_i + h; y_i + k_1 h). \tag{5.2.0.16}$$

- Metoda punktu środkowego.

  W tym przypadku $a_2 = 1$, z zależności (5.2.0.13) wynika, że $a_1 = 0$, $p_1 = q_{11} = 1/2$,

$$y_{i+1} = y_i + k_2 h, \tag{5.2.0.17}$$

$$k_1 = f(x_i, y_i), \tag{5.2.0.18}$$

$$k_2 = f(x_i + \frac{1}{2}h; y_i + \frac{1}{2}k_1 h). \tag{5.2.0.19}$$

- Metoda Ralstona'a.

  Gwarantuje najmniejszy możliwy dla metod R-K II-go rzędu błąd odcięcia, tutaj $a_2 = 2/3$, a z zależności (5.2.0.13) wynika, że $a_1 = 1/3$ $p_1 = q_{11} = 3/4$,

$$y_{i+1} = y_i + (\frac{1}{3}k_1 + \frac{2}{3}k_2)h, \tag{5.2.0.20}$$

$$k_1 = f(x_i, y_i), \tag{5.2.0.21}$$

$$k_2 = f(x_i + \frac{3}{4}h; y_i + \frac{3}{4}k_1 h). \tag{5.2.0.22}$$

Metoda Rungego-Kutty III rzędu

W tej metodzie wzór na całkowanie równania różniczkowego przybiera następującą postać:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_2 + k_3)h, , \tag{5.2.0.23}$$

$$k_1 = f(x_i, y_i), \tag{5.2.0.24}$$

$$k_2 = f(x_i + \frac{1}{2}h; y_i + \frac{1}{2}k_i h), \tag{5.2.0.25}$$

$$k_3 = f(x_i + h; y_i - k_1 h + 2k_2 h). \tag{5.2.0.26}$$

W tym przypadku błąd aproksymacji jest rzędu $E_a \sim O(h^4)$, a błąd globalny wynosi $E_t \sim O(h^3)$.

Metoda Rungego-Kutty IV rzędu

Metoda IV rzędu jest najbardziej popularna, zarówno przez łatwość implementacji jak i przez zadowalające przybliżenie.

Wzór na całkowanie ma postać:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h, \tag{5.2.0.27}$$

$$k_1 = f(x_i, y_i), \tag{5.2.0.28}$$

$$k_2 = f(x_i + \frac{1}{2}h; y_i + \frac{1}{2}k_1 h), \tag{5.2.0.29}$$

$$k_3 = f(x_i + \frac{1}{2}h; y_i + \frac{1}{2}k_2 h), \tag{5.2.0.30}$$

$$k_4 = f(x_i + h; y_i + k_3 h), \tag{5.2.0.31}$$

$$\phi = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{5.2.0.32}$$

Błąd aproksymacji tej metody wynosi $E_a \sim O(h^5)$, a błąd globalny to $E_t \sim O(h^4)$.

Metoda Rungego-Kutty V rzędu

W tej metodzie wzór na całkowanie ma postać:

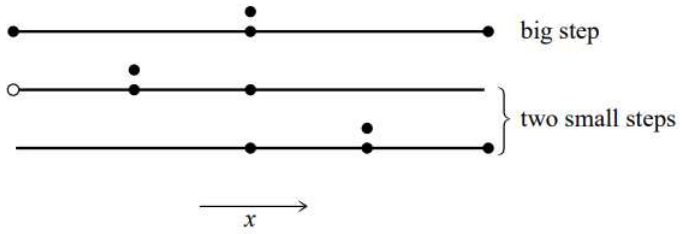
$$y_{i+1} = y_i + \frac{1}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6)h. \tag{5.2.0.33}$$

5.3 Metody adaptacyjne

Dobry integrator powinien sprawować swego rodzaju kontrole nad własnym rozwojem, wprowadzając częste zmiany rozmiaru wykonywanego kroku [15]. W miejscach gdzie najbardziej ciekawi nas struktura równania powinniśmy poruszać się małymi krokami, aby dokładnie zbadać rozwiązanie. Za to w mniej ciekawych okolicach, oczekujemy dużych kroków.

5.3.1 Metoda podwojonego kroku

Dla metody Rungego-Kutty IV rzędu, najprostszą techniką jest podwojenie kroku. W metodzie tej każdy z kroków wykonujemy dwukrotnie. Za pierwszym razem jest to pełny krok, a za drugim składa się on z dwóch kroków połówkowych. Idea została przedstawiona na rysunku (5.1).



Rysunek 5.1: Rysunek przedstawia metodę podwojonego kroku [15].

Oznaczmy dokładne rozwiązanie dla rozwinięcia od x do $x+2h$ przez $y(x+2h)$ oraz dwa przybliżone rozwiązania przez y_1 - dla jednego kroku oraz y_2 dla dwóch kroków o rozmiarze h. Biorąc pod uwagę metodę Runnego- Kutty IV rzędu, rozwiązania oraz przybliżenia są ze sobą powiązane przez następujące zależności:

$$y(x+2h) = y_1 + (2h)^5\Phi + O(h^6) + ... \tag{5.3.1.1}$$

$$y(x+2h) = y_2 + 2(h^5)\Phi + O(h^6) + ... \tag{5.3.1.2}$$

Różnica pomiędzy y_2, a y_1 jest wygodnym wskaźnikiem błędu przybliżenia. Stanowi ją poniższy wzór:

$$\Delta \equiv y_2 - y_1. \tag{5.3.1.3}$$

Jest to zadowalająca różnica, którą staramy się zachować dla oczekiwanego stopnia dokładności, aby to zapewnić dostosowujemy odpowiednio h. Przy odpowiednich założeniach [15] równania (5.3.1.1) i (5.3.1.2) możemy połączyć w następujący sposób:

$$y(x+2h) = y_2 + \frac{\Delta}{15} + O(h^6) \tag{5.3.1.4}$$

Do piątego rzędu jest to dość dokładne oszacowanie, niekoniecznie użycie wyższych rzędów da nam lepszą dokładność przybliżenia. Ponadto metoda ta nie daje nam możliwości monitorowania błędu przybliżenia. Dlatego użycie jej przeważnie nie przyniesie nam szkód, ale nie mamy możliwości bezpośredniego dowiedzenia się czy przyniosło nam to jakiekolwiek korzyści.

5.3.2 Metoda Rungego-Kutty Fehlberga

Alternatywnym algorytmem dostosowującym rozmiar kroku jest algorytm bazujący na metodach Rungego-Kutty opracowany przez Fehlberga. Fehlberg odkrył metodę piątego rzędu z sześcioma funkcjami, w przypadku gdy inna kombinacja sześciu funkcji daje nam metodę czwartego rzędu. Różnica pomiędzy tymi dwoma oszacowaniami może być zastosowana do oszacowania błędu, a w następstwie do oszacowania kroku.

Ogólna formuła Rungego-Kutty V rzędu przedstawiona jest przez następujące wzory:

$$k_1 = hf(x_n, y_n), \tag{5.3.2.1}$$

$$k_2 = hf(x_n + a_2 h, y_n + b_{21} k_1), \tag{5.3.2.2}$$

$$\dots$$

$$k_6 = hf(x_n + a_6 h, y_n + b_{61} k_1 + \dots + b_{65} k_5), \tag{5.3.2.3}$$

$$y_{n+1} = y_n + c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4 + c_5 k_5 + c_6 k_6 + O(h^6), \tag{5.3.2.4}$$

a rozwinięta metoda IV rzędu to:

$$y^*_{n+1} = y_n + c^*_1 k_1 + c^*_2 k_2 + c^*_3 k_3 + c^*_4 k_4 + c^*_5 k_5 + c^*_6 k_6 + O(h^5). \tag{5.3.2.5}$$

W tym przypadku błąd wynosi:

$$\Delta \equiv y_{n+1} - y^*_{n+1} = \sum_{i=1}^{6} (c_i - c^*_i) k_i. \tag{5.3.2.6}$$

Konkretne wartości wykorzystywanych stałych prezentuje poniższa tabela. Opracowane zostały przez Cash-Karp i są one stosunkowo lepsze od tych pierwotnie podanych przez Fehlberga, ponieważ pozwalają na uzyskanie lepszego błędu przybliżenia.

i	a_i	b_{ij}					c_i	c^*_i
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
$j =$		1	2	3	4	5		

Rysunek 5.2: Tabela przedstawia stałe Cash-Karp stosowane w metodach R-K [15].

Wiemy już w przybliżeniu ile wynosi nasz błąd. Musimy zastanowić się jak otrzymać go tak, aby mieścił się w pożądanych granicach. Relację pomiędzy Δ, a h możemy wyliczyć z następującego

rozumowania. Zgodnie ze wzorami na V i IV rząd R-K, Δ skaluje się na h^5. Dlatego jeśli weźmiemy h_1 i dostaniemy błąd Δ_1 wówczas możemy szacować h_0 jako:

$$h_0 = h_1 |\frac{\Delta_0}{\Delta 1}|^{0.2} \tag{5.3.2.7}$$

Powyższe równanie ma dwa zastosowania. Jeśli Δ_1 jest większe od Δ_0 to równanie daje nam informacje, aby zmniejszyć wielkość 'nieudanego' kroku przy jego ponownym wykonywaniu. W przypadku kiedy Δ_1 jest mniejsze od Δ_0, równanie mówi nam, o ile możemy bezpiecznie zwiększyć rozmiar kroku dla następnej iteracji. Lokalna ekstrapolacja polega na akceptacji rozmiaru piątego rzędu y_{n+1} pomimo tego, że oszacowanie błędu ma zastosowanie dla rzędu czwartego y^*_{n+1}.

Jeśli Δ_0 oszacujemy bezpośrednio przez h, to wykładnik 0.20 nie jest poprawny. W przypadku kiedy rozmiar kroku zostanie zmniejszony ze zbyt dużej wartości, to nowa przewidywana wartość h_1 nie spełnia wymaganej dokładności. Rozwiązaniem problemu jest skalowanie zamiast na $0.20 = \frac{1}{5}$ to na $0.25 = \frac{1}{4}$. Wykładniki 0.20 i 0.25 nie różnią się od siebie w znacznym stopniu, dlatego przyjmujemy, że jeśli zwiększamy rozmiar kroku, to używamy wyższej wartości wykładnika, nie zastanawiając się nad tym czy go potrzebujemy czy nie, a w przypadku zmniejszaniu wartości kroku, stosujemy wykładnik o mniejszej wartości. Ponieważ nasze oszacowania błędu nie są zbyt dokładne, zaleca się wprowadzenie współczynnika bezpieczeństwa, spełniającego warunki $S < 1$ i $S >>> 0$. Równanie 5.3.2.7 zostaje zastąpione przez empiryczny wzór [15]:

$$h_0 = \begin{cases} Sh_1|\frac{\Delta_0}{\Delta_1}|^{0.20}, gdzie \Delta_0 \geq \Delta_1 \\ Sh_1|\frac{\Delta_0}{\Delta_1}|^{0.25}, gdzie \Delta_0 \leq \Delta_1 \end{cases} \tag{5.3.2.8}$$

Rozdział 6

Programowanie równoległe

6.1 Wprowadzenie

Oprogramowanie służące do szukania osobliwości rozwiązań równań różniczkowych zostało stworzone w oparciu o idee programowania równoległego. Głównym zamysłem jest tworzenie oprogramowania w taki sposób, aby poszczególne obliczenia mogły być realizowane przez oddzielne procesory równocześnie. W znacznym stopniu usprawnia to prace programu poprzez przyspieszenie jego działania. W tym celu w oprogramowaniu została zaimplementowana biblioteka 'pthread' [21]. Standard ten służy to tworzenia, zarządzania i usuwania wątków. Posiada on zabezpieczenia przed równoczesnym użyciem tego samego zasobu przez różne wątki tzw. mutexy. Do opracowania tego rozdziału posłużyła książka Z. Weissa "Programowanie współbieżne i rozproszone w przykładach i zadaniach"[16].

Podstawowym pojęciem z jakim powinniśmy się zapoznać jest proces. Jak wiemy z wielu dziedzin życia proces jest to wykonywanie czynności w ustalonej kolejności. W tym przypadku kolejność tą ustala nam algorytm. Niezbędnymi narzędziami do wykonywania procesu są pamięć operacyjna odpowiadająca za zapamiętanie kodu oraz procesor, który odpowiada za dokonywanie kolejnych zmian. Proces może być zarówno skończony jak i nieskończony. Każdy z nas spotkał się kiedyś z nieskończonym procesem podczas nauki programowania, tworząc nieskończoną pętlę. Jak wiadomo nie jest to efekt zamierzony, a raczej wynikający z błędnej deklaracji warunków. Są jednak sytuacje, w których chcemy uzyskać taki właśnie nieskończony proces. Idealnym przykładem jest system operacyjny, od którego wymagamy wręcz, aby nigdy nie kończył swojej pracy. Przechodząc do meritum, z procesami współbieżnymi mamy do czynienia w przypadku, gdy jakiś proces rozpoczął swoją prace, zanim zakończył ją uprzednio rozpoczęty. Zauważmy, że nieskończony proces będzie współbieżny z każdym procesem, który został uruchomiony po nim. Na naszym przykładzie, system operacyjny jest współbieżny z każdym procesem uruchamianym przez użytkownika.

Oprócz pojęcia procesu, wprowadzamy pojęcie wątku (ang. thread). Różnica pomiędzy wątkiem a procesem polega na sposobie wykorzystania zasobów. W ramach jednego procesu może być wykonywanych wiele wątków i dzielą one pamięć. Biorąc pod uwagę takie rozumowanie, proces nazwiemy zadaniem, w którym wykonywany jest jeden wątek. Użycie wątków pozwala nam na realizacje równocześnie wielu żądań skierowanych do jednego taska. Wątki dzielimy na synchroniczne, takie które same przekazują sobie sterowanie oraz asynchroniczne. Kiedy spotykamy się z wątkami asynchronicznymi, potrzebujemy mechanizmu umożliwiającego nam synchronizację działania przy dostępie do współdzielonych danych.

6.2 Blokada i zagłodzenie

Procesy wykonywane w programowaniu współbieżnym wymagają wzajemnej synchronizacji lub muszą komunikować się ze sobą. Przez tą własność, w niektórych przypadkach procesy będą wstrzymywane przez oczekiwanie na informacje od innego procesu. O poprawnie działającym programie współbieżnym mówimy kiedy w każdym z procesów nastąpiło oczekiwane zdarzenie. Kiedy program działa niepoprawnie, może wystąpić zjawisko blokady lub zagłodzenia.

- BLOKADA [16] Zjawisko blokady występuje w przypadku kiedy zbiór procesów utrzymuje się w stanie zatrzymania, z powodu oczekiwania na zdarzenie, które powinno być wykonane przez jeden z procesów należących do tego zbioru. Jeśli w programie współbieżnym może wystąpić blokada, nie znaczy to, że wystąpi ona za każdym razem przy działaniu programu. Czasami unikanie blokady jest bardzo kosztowne, dlatego decydujemy się wtedy na pogodzenie z jej istnieniem i uruchamianie mechanizmów, które o niej informują oraz usuwają ją.
- ZAGŁODZENIE [16] Zjawisko zagłodzenia inaczej zwane zjawiskiem wykluczenia jest specyficznym przypadkiem nieskończonego wstrzymania jakiegoś procesu. Zachodzi ono w sytuacji kiedy wiele procesów czeka na sygnał lub komunikat synchronizacyjny. Zagłodzony proces to taki, który pomimo tego, że zdarzenie zostało wykonane już jakąś liczbę razy, to on nadal czeka. Dzieje się tak, ponieważ za każdym razem został wybrany jakiś inny proces. Problem zagłodzenia występuje, gdy do ustalenia kolejności wykonywanych procesów użyjemy kolejki priorytetowej. W przypadku kiedy o kolejności decyduje kolejka prosta, czyli procesy są wznawiane w takiej samej kolejności w jakiej zostały zatrzymane, nie musimy się martwić o zagłodzenie któregoś z procesów. Zjawisko jest bardzo ciężkie do wykrycia, jednakże przeważnie godzimy się z jego istnieniem, ponieważ bierzemy pod uwagę, że dane zdarzenie za które odpowiada proces jest naprawdę mało prawdopodobne.

6.3 Klasyczne problemy współbieżności

6.3.1 Problem producenta i konsumenta

Problem producenta i konsumenta [16] polega na synchronizacji dwóch procesów. Pierwszy proces - producent produkuje porcję informacji i przekazuję ją do drugiego procesu - konsumenta, który pobiera informacje i konsumuje ją. W przypadku, kiedy w danym momencie producent nie może przekazać informacji do konsumenta, musi czekać. Podobnie konsument czeka, kiedy nie może w danym momencie pobrać informacji. Założenie jest takie, że porcje powinny być konsumowane w takiej samej kolejności w jakiej zostały wyprodukowane.

Można rozważać wiele wariantów tego problemu, najprostszym z nich zakładam, że producent przekazuje informacje bezpośrednio do konsumenta. Jednak najbardziej popularnym jest wariant, w którym pomiędzy procesami istnieje n-elementowy bufor (taki, że $n>0$). W tym przypadku, producent wstawia porcje informacji do niepełnego bufora, a konsument pobiera informacje z niepustego bufora. Trzeci rozważany przypadek, to taki w którym bufor jest nieskończony, musimy wtedy zapewnić, aby operacje konsumenta i producenta tak się synchronizowały, żeby wzajemnie wykluczać pobieranie i wstawianie do tego samego elementu bufora w tym samym czasie.

Efektywność przyjętego rozwiązania jest ściśle powiązana z rozmiarem buforu i zależna od czasu produkcji i konsumpcji danych. Kiedy średni czas produkcji jest dłuższy od średniego czasu konsumpcji, bufor może okazać się nieprzydany, ponieważ proces konsumpcji będzie pracował na bieżąco. W odwrotnym przypadku, gdy produkcja zachodzi szybciej od konsumpcji, bufor w pewnym momencie okazuje się być za mały. Kiedy średnie czasy produkcji i konsumpcji są równe, wielkość bufora dobierana jest w zależności od wariancji średnich czasów tych procesów.

Architektura kodu omawianego w części praktycznej mojej pracy została oparta na problemie producenta i konsumenta.

6.3.2 Problem czytelników i pisarzy/producent konsument

Problem ten jest abstrakcją problemu polegającego na synchronizacji procesów, które korzystają ze wspólnej bazy danych [16]. Zatem czytelnicy oraz pisarze przynależą do dwóch grup cyklicznych, które konkurują ze sobą o dostęp do czytelni. Każdy z czytelników może korzystać z czytelni, kiedy znajduje się w niej ktoś poza nim - inny proces czytelnik. U pisarzy sprawa nie jest, aż tak prosta, ponieważ aby korzystać z czytelni potrzebują całkowitej samotności. Proces czytelnik odpowiada za odczyt danych, a proces pisarz za ich dodawanie. Ponadto zakładany, że procesy są skończone. Reasumując, dane z bazy danych mogą być sczytywane przez wiele procesów jednocześnie, ale modyfikować je może jeden proces. Przedstawia się trzy odrębne rozwiązania tego problemu, dwa z nich zakładają możliwość zagłodzenia pisarzy lub czytelników.

- **Rozwiązanie z możliwością zagłodzenia pisarzy** Jak zakłada problem, czytelnicy mogą wchodzić do czytelni kiedy znajdują się w niej inni przedstawiciele tej grupy społecznej, jednakże nie mogą do niej wejść kiedy znajduje się w niej pisarz. Kiedy czytelnia zajęta jest przez pisarza, czytelnicy czekają, aż stanie się pusta. Podobnie pisarz, kiedy czytelnia jest zajęta przez czytelników, musi zaczekać na wejście. Rozwiązanie zakłada, że czytelnicy mogą na stałe zawładnąć czytelnią, przez co pisarze zostają zagłodzeni, ponieważ nigdy nie doczekali się pustej czytelni.

- **Rozwiązanie z możliwością zagłodzenia czytelników** Uznajemy, że jeśli pisarze chcą coś napisać, należy dać im taką możliwość. Nie ma sensu, aby czytelnicy czytali przestarzałe informację. Pisarze ustawiają się w kolejce do wejścia, czekając aż czytelnia będzie pusta. W momencie kiedy taki pisarz czeka, żaden czytelnik nie może wejść już do czytelni, ponieważ chcemy, aby informacje były jak najszybciej zaktualizowane. W momencie gdy czytelnia jest już pusta, zajmują ją intensywnie pracujący pisarze, w takiej sytuacji może dojść do zagłodzenia czytelników, ponieważ wystarczy, że jeden pisarz będzie czekał w kolejce i już wszyscy czytelnicy mają zablokowany dostęp do czytelni.

- **Rozwiązanie poprawne** Rozwiązanie zakłada, że nie będziemy zagładzać żadnej z grup procesów. Zarówno czytelnicy jak i pisarze wpuszczani są do czytelni w kolejności zgłoszeń. Z uwagi na to, że czytelnicy mogą przebywać w czytelni jednocześnie, dlatego razem z czytelnikiem, na którego przyszła kolej wpuszczani są wszyscy inni czekający czytelnicy. Takie rozwiązanie zapewnia nam optymalne działanie, w którym procesy nie zostaną zagłodzone.

Ten wzorzec został użyty w całkowaniu równań różniczkowych na płaszczyźnie zespolonej w pracy [1], na której się opieram. Problem ten jest użyteczny zawsze, gdy zadanie możemy podzielić na małe porcje, które mogą być przetwarzane równolegle, a w naszym przypadku całkowanie równania wzdłuż zadanej drogi stanowi taką niezależną porcję.

6.3.3 Problem 5 filozofów

Problem pięciu filozofów jest sformułowany w następujący sposób [16]: Przy okrągłym stole siedzi pięciu filozofów. Każdy z nich rozmyśla, jednak może robić to tylko jeśli jest najedzony. Na środku stołu leży duży talerz ze spaghetti, a przed każdym z filozofów pusty talerz i widelec po lewej i prawej stronie. Kiedy któryś z filozofów zgłodnieje sięga po oba widelce i je do syta. Takie zachowanie sprawia, że sąsiadujący myśliciel nie ma dostępu do dwóch widelców, co jest warunkiem koniecznym

do rozpoczęcia spożywania posiłku. Problem ten nie ma odzwierciedlenia w rzeczywistości jednak jest on świetnym przykładem do wyjaśniania zjawiska blokady oraz zagłodzenia.

- **Rozwiązanie z możliwością blokady** Każdy z filozofów postępuje według tych samych zasad. Jeśli widelec po jego lewej stronie jest wolny, to podnosi go i czeka, aż filozof po prawej stronie skończy posiłek, żeby podnieść widelec po prawej stronie i zacząć jeść. Po skończonym posiłku odkłada oba widelce. W tym przypadku, łatwo o blokowanie wszystkich procesów, ponieważ jeśli każdy z filozofów podniesie widelec po swojej lewej stronie i będzie czekać aż drugi widelec będzie wolny, to wszyscy filozofowie pozostaną w stanie oczekiwania.

- **Rozwiązanie z możliwością zagłodzenia** Opisany we wcześniejszym przykładzie sposób postępowania filozofów można ulepszyć o zasadę, że filozof podnosi widelce tylko jeśli oba są wolne. W tym przypadku może się okazać, że sąsiedzi któregoś z mężczyzn będą zainteresowani głównie jedzeniem i nigdy nie nastąpi moment, w którym żaden z nich nie je. W tym przypadku filozof siedzący pomiędzy nimi zostanie zagłodzony.

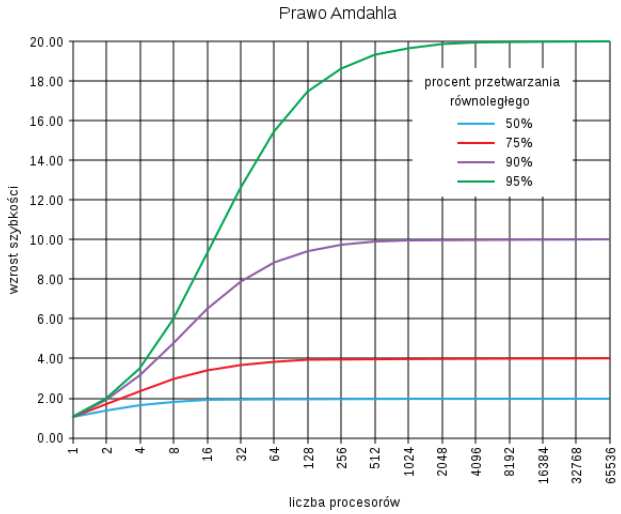
- **Rozwiązanie poprawne** Okazuje się, że filozofowie nie potrafią sami rozwiązać problemu odżywiania się, jeśli będą obstawać przy założeniu, że każdy z nich ma takie same prawo do posiłku. Potrzebny jest im lokaj, który będzie nadzorował ucztę. W momencie kiedy czterech filozofów konkuruje o widelce, problem blokowania oraz zagłodzenia znika. Dlatego zadaniem lokaja jest powstrzymywanie jednego z filozofów do momentu, aż pozostała czwórka skończy posiłek.

6.4 Prawo Amdahla

Prawo Amdahla [18] stosowane jest do oszacowania maksymalnego spodziewanego zwiększenia wydajności całkowitej programu, w przypadku jeśli tylko jakaś część tego programu została ulepszona. Prawo to reprezentowane jest przez następujący wzór:

$$S(n) = \frac{T(n)}{T(1)} = \frac{1}{B + \frac{1}{n}(1 - B)}, \tag{6.4.0.1}$$

gdzie n jest liczbą wątków do wykonania, $T(n)$ czasem wykonania n wątków, a B jest proporcją programu, który może zostać poddany zrównolegleniu. Prawo to zostało przedstawione na wykresie 6.1. Widzimy, że zwiększenie liczby procesorów w pewnym momencie nie ma już wpływu na zwiększenie szybkości działania programu. Dzieje się tak, ponieważ wzrost szybkości działania programów równoległych jest ograniczony przez sekwencyjny podział programu.



Rysunek 6.1: Wykres zależności wzrostu szybkości działania programu od przyrostu liczby procesorów reprezentujący prawo Amdahla [20].

Część II

Część praktyczna

Rozdział 7

Oprogramowanie

Jak wspominałam w części teoretycznej rozwiązywanie równań różniczkowych zwyczajnych za pomocą funkcji elementarnych nie jest w praktyce możliwe. Dlatego ważnym zadaniem jest wybór odpowiednich metod, aby otrzymać optymalne wyniki. Część praktyczną mojej pracy stanowi rozwój oprogramowania do szukania osobliwości. Oprogramowanie postanowiłam rozszerzyć o metody numeryczne opisane w części teoretycznej mojej pracy.

7.1 Opis działania oprogramowania

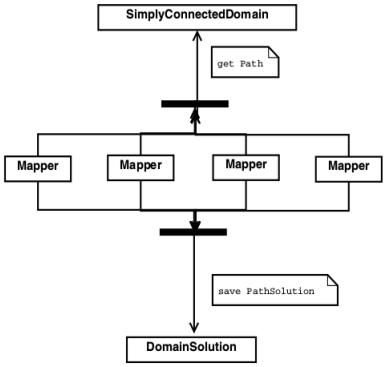
Szczegółowy opis oprogramowania, znajduje się w publikacji [1], a kod źródłowy dostępny jest pod adresem [24]. Sposób działania oprogramowania został przedstawiony na schemacie 7.1. Widzimy, że program został zrównoleglony poprzez podział obliczeń na wątki. Na początku wyznaczamy ścieżki, po których będziemy poruszać się w procesie całkowania. W oprogramowaniu zostały zaimplementowane dwie wersję, w pierwszej poruszamy się po prostej ścieżce, a w drugiej przyjmuje ona postać spirali. Są one tworzone w obiekcie SimplyConnectedDomain, po czym zostają przekazane do procesu całkowania, który wykonywany jest równolegle przez funkcje Mapper. W tym miejscu używamy opisywanych w pracy metod numerycznych. Zastosowana jest bariera synchronizacyjna (poziome linie), za sprawą której wymuszamy pełne pobranie ścieżki,48 po której całkujemy przez konkretny wątek. Wyniki całkowania zapisywane są w obiekcie DomainSolution, do którego dostęp również jest kontrolowany przez barierę synchronizacyjną. Bariera zapewnia synchronizację zapisu rozwiązania wzdłuż ścieżki w całości w kontenerze DomainSolution.

7.2 Metody numeryczne Rungego-Kutty

Pierwotna wersja oprogramowania posiadała zaimplementowaną metodę numeryczną Rungego-Kutty IV rzędu. Zaimplementowanie metod niższego oraz wyższego rzędu miało dać nam informację czy początkowo wybrana metoda była rozwiązaniem optymalnym. Działanie programu zostało przetestowane na przykładzie równania różniczkowego zwyczajnego będącego uogólnieniem równania Lane-Emdena występującego w astrofizyce jako najprostszy model gwiazdy [22, 23]

$$\frac{d^2u(x)}{dx^2} + \frac{\alpha}{x}\frac{du(x)}{dx} + x^n u(x)^p = 0. \tag{7.2.0.1}$$

Z literatury [22, 23] wiemy, że dla $n = 1$, rozwiązanie analityczne (wokół $x = 0$) tego równania rozwiązanie posiada trzy ruchome osobliwości rozłożone na płaszczyźnie zespolonej w pobliżu $x = 0$.



Rysunek 7.1: Schemat przedstawia równoległość metody.[1]

Wszystkie opracowane metody działają w podobny sposób. W funkcji main zostają utworzone wątki odpowiadające za całkowanie po ścieżkach.

```
thread_args[i].mapper = new Mapper( new Equation(),
new Initializer(), new Domain ( xLeftUp, xRightDown ),
new fixedRK4Stepper( new Equation(), 1000.0, 0.000001, 0.0001 ) );
```

Funkcja fixedRK4Stepper, odpowiada za całkowanie numeryczne i zwraca nam wartość true jeśli krok jest dobry lub false w przeciwnym razie. Przyjmuje zmienne odpowiadające za deklaracje równania, wartość maksymalną, w której całkowanie zostanie zatrzymane, czyli znaleziona zostanie oczekiwana osobliwość, oraz wartości hmin i hmax. Dba o to, aby rozmiar kroku był odpowiedni, to znaczy nie większy od ustalonego hmax i nie mniejszy od hmin.

7.2.1 Metody Rungego-Kutty II rzędu

Metoda Heun'a

W metodzie Heun'a obliczenia numeryczne znajdują się w funkcji RK2, która na wejście przyjmuje wartości: cmplx x, cvector y, cmplx h. Wartość cmplx x określa punkt początkowy, cmplx y początkową wartość rozwiązania, a cmplx h jest rozmiarem kroku. Funkcja zwraca nam wektor ytmp, będący rozwiązaniami całkowania. Wykonane obliczenia są zgodne z opisanymi w części teoretycznej.

```
cvector rk2h::RK2( cmplx x, cvector y, cmplx h){

    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );
    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
```

```
        xtmp = x + h;
        ytmp[i] = y[i] + h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * ( k1[i]/2.0 + k2[i]/2.0);
    }

    return( ytmp );
};
```

Metoda punktu środkowego

Metoda działa analogicznie do metody Rungego-Kuty 2 rzędu Heun'a.

```
cvector rk2ps::RK2p( cmplx x, cvector y, cmplx h){
    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + h * 0.5;
        ytmp[i] = y[i] + 0.5 * h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * k2[i];
    }

    return( ytmp );

};
```

Metoda Ralstona

Metoda ta również działa na tej samej zasadzie co wcześniej opisane metody.

```
cvector rk2r::RK2ra( cmplx x, cvector y, cmplx h){

    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + h * 3.0/4.0;
        ytmp[i] = y[i] + 3.0/4.0 * h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * (k1[i]/3.0 + 2.0/3.0 * k2[i]);
    }

    return( ytmp );

};
```

7.2.2 Metoda Rungego-Kutty III rzędu

Metoda trzeciego rzędu działa podobnie do metod rzędu drugiego, w tym przypadku zwiększa się liczba obliczeń, ponieważ jak wynika ze wzoru występuje tu dodatkowo poza zmiennymi k1 i k2, zmienna k3.

```
cvector fixedRK3Stepper::RK3( cmplx x, cvector y, cmplx h){
    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );
    cvector k3( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );
```

```
    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + 0.5 * h;
        ytmp[i] = y[i] + 0.5 * h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + h;
        ytmp[i] = y[i] -k1[i]*h + 2.0 * h * k2[i];
    }

    k3 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * ( k1[i] + 4.0 * k2[i] + k3[i]) / 6.0;
    }

    return( ytmp );
};
```

7.2.3 Metoda Rungego-Kutty IV rzędu

W tej metodzie również obliczenia są analogiczne do metod poprzednich.

```
cvector fixedRK4Stepper::RK4( cmplx x, cvector y, cmplx h){

    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );
    cvector k3( eq->getRank() + 1 );
    cvector k4( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + 0.5 * h;
        ytmp[i] = y[i] + 0.5 * h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
```

```
        xtmp = x + 0.5 * h;
        ytmp[i] = y[i] + 0.5 * h * k2[i];
    }

    k3 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + h;
        ytmp[i] = y[i] + h * k3[i];
    }

    k4 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * ( k1[i] + 2.0 * k2[i] + 2.0 * k3[i] + k4[i] ) / 6.0;
    }

    return( ytmp );
};
```

7.2.4 Metoda Rungego-Kutty V rzędu

W metodzie tej wykorzystane jest najwięcej obliczeń. Przewidujemy, że znacznie wydłuży nam to czas działania, ale da najbardziej dokładne przybliżenie.

```
cvector fixedRK5Stepper::RK5( cmplx x, cvector y, cmplx h){
    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );
    cvector k3( eq->getRank() + 1 );
    cvector k4( eq->getRank() + 1 );
    cvector k5( eq->getRank() + 1 );
    cvector k6( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    k1 = eq->derivs( x, y );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + 0.5 * h;
        ytmp[i] = y[i] + 0.5 * h * k1[i];
    }

    k2 = eq->derivs( xtmp, ytmp );
```

```
    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + 0.5 * h;
        ytmp[i] = y[i] + h * k1[i]/4.0 + h * k2[i]/4.0;
    }

    k3 = eq->derivs( xtmp, ytmp );

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        xtmp = x + h;
        ytmp[i] = y[i] - h * k2[i] + h * 2.0 * k3[i];
    }

    k4 = eq->derivs( xtmp, ytmp );

    for(int i=1; i < eq->getRank() + 1; i++){
        xtmp = x + 2.0/3.0 * h;
        ytmp[i] = y[i] + 7.0/27.0 * h * k1[i]
        + 10.0/27.0 *  h * k2[i] + 1.0/27.0 * h * k4[i];
    }

    k5 = eq->derivs( xtmp, ytmp);

    for(int i=1; i < eq->getRank() + 1; i++){
        xtmp = x + 1.0/5.0 * h;
        ytmp[i] = y[i] + 28.0/625.0 * h * k1[i] - 1.0/5.0 * h * k2[i]
        + 546.0/625.0 * h * k3[i] + 54.0/625.0 * h * k4[i]
        - 378.0/625.0 * h * k5[i];
    }

    k6 = eq->derivs(xtmp, ytmp);

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        ytmp[i] = y[i] + h * ( 1.0/24.0 * k1[i]
        + 5.0/48.0 * k4[i] + 27.0/56.0 * k5[i]
        + 125.0/336.0 * k6[i] );
    }

    return( ytmp );
};
```

7.2.5 Metoda podwojonego kroku

Krok całkowania w metodzie jest liczony na bieżąco, przed obliczeniem kolejnych wartości ytmp. Jeśli delta jest różna od deltamax, to krok całkowania wynosi h. W przypadku kiedy delta jest równa deltamax, czyli wartość jest bliska zeru, to krok z którym się poruszamy jest podwajany.

```
cvector doublestep::RKds( cmplx x, cvector y, cmplx h) {

    assert( y.size() == eq->getRank() + 1);

    cvector k1( eq->getRank() + 1 );
    cvector k2( eq->getRank() + 1 );
    cvector k3( eq->getRank() + 1 );
    cvector k4( eq->getRank() + 1 );

    cmplx xtmp;
    cvector ytmp( eq->getRank() + 1 );

    cmplx delta = (0.0,0.0);
    cmplx deltamax=(0.0,0.0);

    for( int i = 1; i < eq->getRank() + 1; i++ ){
        if(delta != deltamax){
            k1 = eq->derivs( x, y );
            xtmp = x + 0.5 * h;
            ytmp[i] = y[i] + 0.5 * h * k1[i];

            k2 = eq->derivs( xtmp, ytmp );
            xtmp = x + 0.5 * h;
            ytmp[i] = y[i] + 0.5 * h * k2[i];

            k3 = eq->derivs( xtmp, ytmp );
            xtmp = x + h;
            ytmp[i] = y[i] + h * k3[i];

            k4 = eq->derivs( xtmp, ytmp );

            ytmp[i] = y[i] + h * ( k1[i] + 2.0 * k2[i] + 2.0 * k3[i] + k4[i] ) / 6.0;

        }else{
            h=h*2.0;
            k1 = eq->derivs( x, y );

            xtmp = x + 0.5 * h;
            ytmp[i] = y[i] + 0.5 * h * k1[i];

            k2 = eq->derivs( xtmp, ytmp );

            xtmp = x + 0.5 *h;
            ytmp[i] = y[i] + 0.5 *h * k2[i];
```

```
            k3 = eq->derivs( xtmp, ytmp );

            xtmp = x + h;
            ytmp[i] = y[i] + h * k3[i];

            k4 = eq->derivs( xtmp, ytmp );

            ytmp[i] = y[i] + h * ( k1[i] + 2.0 * k2[i] + 2.0 * k3[i] + k4[i] ) / 6.0;
        }

        delta =ytmp[i]-ytmp[i-1];
    }

    return( ytmp );
};
```

7.3 Wyniki

7.3.1 Czas działania

Porównując zaimplementowane metody chcielibyśmy dowiedzieć się, która z nich jest optymalna. Pierwszym kryterium wyboru jest porównanie czasu działania poszczególnych metod. W kodzie wykorzystałam funkcję clock() z biblioteki time.h, odpowiadającą za pobieranie czasu systemowego. Zostało to zrealizowane w następujący sposób:

```
#include <ctime>

...
    //pobieramy czas początkowy
    clock_t begin = clock();

        for(int i=0; i<NUM_THREADS; ++i ){
            thread_args[i].mapper = new Mapper( new Equation(),
            new Initializer(), new Domain ( xLeftUp, xRightDown ),
            new fixedRK4Stepper( new Equation(), 1000.0, 0.000001, 0.0001 ) );

            thread_args[i].paths = & paths;
            thread_args[i].domainSolution = & domainSolution;
            thread_args[i].mutexRead  = mutexReadTmp;
            thread_args[i].mutexWrite = mutexWriteTmp;
            thread_args[i].threadNr = i;

            cout << "MAIN :: creating thread " << i << endl;

            retCode = pthread_create(&threads[i], NULL, TaskCode, (void *) &thread_args[i]);
            assert(0 == retCode);
        }

        for(int i=0; i<NUM_THREADS; ++i){
            retCode = pthread_join(threads[i], NULL);
            delete thread_args[i].mapper;
            cout << "MAIN :: join of thread " << i << endl;
            assert(0 == retCode);
        }
    //pobieramy czas końcowy oraz wyliczamy czas działania metody
    clock_t end = clock();
    double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;

    cout << "Czas działania metody:" << elapsed_secs;

...
```

Wyniki pomiaru czasów działania poszczególnych metod prezentuje tabela (7.2). We wszystkich metodach zastosowaliśmy takie same dane początkowe (hmin=0.000001, hmax=0.0001), aby otrzymane wyniki dały nam obiektywne porównanie czasów działania metod:

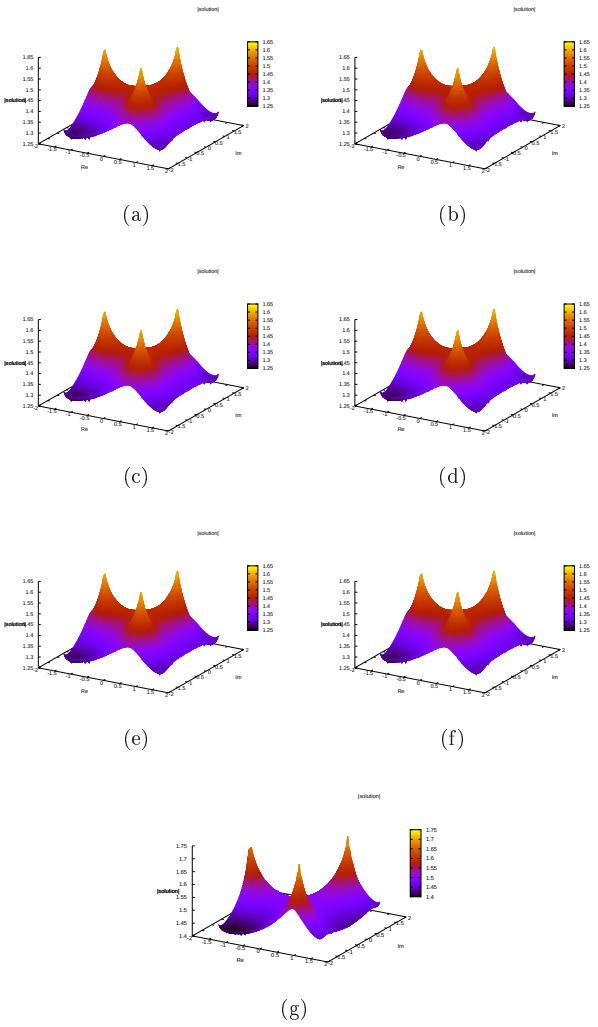
Nazwa metody	Heun'a	Punktu środkowego	Ralstona	RK3	RK4	RK5	Podwojonego kroku
Czas działania [s]	178.581	178.015	183.177	250.068	306.894	480.593	469.452

Rysunek 7.2: Tabela zawiera zestawienie czasów działania poszczególnych metod.

Tak jak się spodziewaliśmy, metody niższych rzędów ze względu na mniejszą liczbę obliczeń działają szybciej.

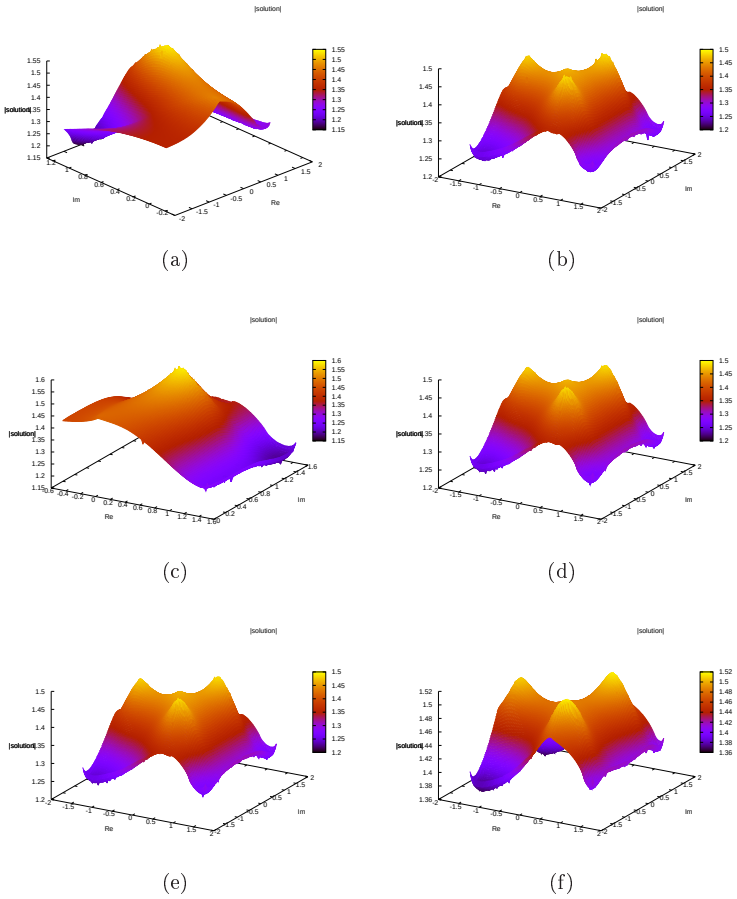
7.3.2 Wykresy

Przy odpowiednio małym kroku całkowania każda z metod zadziałała poprawnie, pokazując, że równanie posiada trzy osobliwości i wykresy są na pozór identyczne. Jednak w przypadku kiedy zastosujemy duży rozmiar kroku, oszacowania numeryczne dla metod niższych rzędów okazują się być niedokładne, a w krytycznych przypadkach nie odszukamy osobliwości. Poniżej zamieszczam zestawienie wykresów.



(a) (b) (c) (d) (e) (f) (g)

Rysunek 7.3: Zestawienie wykresów dla poszczególnych metod przy doborze wartości początkowych hmin=0.000001, hmax=0.0001. Na rysunku (a) Metoda Rungego-Kutty 2 rzędu Heun'a, (b) Metoda Rungego-Kutty 2 rzędu punktu środkowego, (c) Metoda Rungego-Kutty 2 rzędu Ralstona, (d) Metoda Rungego-Kutty 3 rzędu, (e) Metoda Rungego-Kutty 4 rzędu, (f) Metoda Rungego-Kutty 5 rzędu, (g) Metoda podwojonego kroku.



(a) (b)

(c) (d)

(e) (f)

Rysunek 7.4: Zestawienie wykresów dla poszczególnych metod dla wartości początkowych hmin=0.001 hmax=0.01. Na rysunku (a) Metoda Rungego-Kutty 2 rzędu Heun'a, (b) Metoda Rungego-Kutty 2 rzędu punktu środkowego, (c) Metoda Rungego-Kutty 2 rzędu Ralstona, (d) Metoda Rungego-Kutty 3 rzędu, (e) Metoda Rungego-Kutty 5 rzędu, (f) Metoda podwojonego kroku.

Rozdział 8

Podsumowanie

W pracy została opisana teoria osobliwości nieliniowych równań różniczkowych zwyczajnych, a także metody numeryczne całkowania takich równań na płaszczyźnie zespolonej. Wybrane metody zostały zaimplementowane w kodzie opisanym w części praktycznej.

Okazuje się, że przy dobrym doborze rozmiaru kroku, czyli przy wyborze małego kroku, rezultaty są bardzo podobne. Wszystkie metody pozwoliły nam na odszukanie osobliwości, a zatem wszystkie działają poprawnie. Możemy wyciągnąć wniosek, że w takim przypadku nie mamy potrzeby używania metod wyższych rzędów niż drugiego, aczkolwiek nie jest to najbardziej wydajne rozwiązanie. Okazuje się, że metody niskich rzędów przy użyciu większego rozmiaru kroku zawodzą i dają nam wyniki, które bardzo odbiegają od rzeczywistości.

Bibliografia

[1] R.A. Kycia, *Scheme for numerical investigation of movable singularities of the complex valued solutions of ordinary differential equations*, V.P. Gerdt et al. (Eds.): CASC Workshop 2014, LNCS 8660, pp. 288-303 (2014)

[2] F. Leja, *Funkcje zespolone*, PWN, wydanie 6, 2006

[3] A. Palczewski, *Równania różniczkowe zwyczajne. Teoria i metody numeryczne z wykorzystaniem programu rachunków symbolicznych*, WNT, Warszawa 2004, wyd II

[4] K. Golec-Biernat, *Metody matematyczne dla fizyków II. Równania różniczkowe,* Instytut Fizyki Jądrowej PAN w Krakowie, Instytut Fizyki Uniwersytetu Rzeszowskiego, Kraków/Rzeszów, 2006

[5] http://duch.mimuw.edu.pl/ rybka/dydaktyka/ref_14_11.pdf
(data dostępu: 16.09.2018r.)

[6] http://www.math.ualberta.ca/ xinweiyu/334.1.11f/334.1.11.lec21.pdf
(data dostępu: 16.09.2018r.)

[7] E. Hille, *Ordinary Differential Equations in the Complex Domain*, Dover, 1997

[8] J. B. Conway *Functions of One Complex Variable I.* Springer, 1986

[9] J. B. Conway *Functions of One Complex Variable II.* Springer, 1995

[10] P. Henrici, *Applied and Computational Complex Analysis 1*, John Wiley & Sons, 1974

[11] L. V. Ahlfors, *Complex Analysis*, McGraw-Hill, New York, 1979

[12] M. J. Ablowitz, P. A. Clarkson, *Solitons, nonlinear evolution equations and inverse scattering*, London Mathematical Society Lecture Note Series, 149, Cambridge University Press, 1991

[13] http://mini.pw.edu.pl/ kotus/www/pdf/W_12.pdf
(data dostępu: 16.09.2018r.)

[14] D. Kincaid, W. Cheney, *Numerical analysis mathematics of scientific computing*, California, 1991

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C. The Art of Scientific Computing*, 2nd Edition, 1992

[16] Z. Weiss, T. Gruźlewski, *Programowanie współbieżne i rozproszone w przykładach i zadaniach*, WNT, Warszawa, 1993.

[17] G. M. Amdahl, *Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities* AFIPS Conference Proceedings (30): 483–485, 1967

[18] D. P. Rodgers *Improvements in multiprocessor system design* ACM SIGARCH Computer Architecture News. New York, NY, USA: ACM. 13 (3): 225–231, June 1985

[19] http://www.ikb.poznan.pl/almamater/wyklady/metody_komputerowe_03-04/03.pdf (data dostępu: 16.09.2018r.)

[20] https://pl.wikipedia.org/wiki/Prawo_Amdahla (data dostępu: 17.09.2018r.)

[21] `http://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html` (data dostępu: 17.09.2018r.)

[22] R.A. Kycia, G. Filipuk, *On the generalized Emden–Fowler and isothermal spheres equations*, Applied Mathematics and Computation, 265, 1003–1010 (2015)

[23] R.A. Kycia, G. Filipuk, *On the Singularities of the Emden–Fowler Type Equations*, Current Trends in Analysis and Its Applications Trends in Mathematics, 93–99, Springer (2015)

[24] Oprogramowanie do całkowania rozwiązań na płaszczyźnie zespolonej: `https://www.mimuw.edu.pl/~rkycia/software.html`, `http://fizyk.ifpk.pk.edu.pl/~rkycia/research.html` (data dostępu: 17.09.2018r.)