UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2024 Daniel Mládek

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Řešení vybraných typů diferenciálních rovnic s podporou Matlabu

Bakalářská práce

2024 Daniel Mládek

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Daniel Mládek**
Osobní číslo:	**I21054**
Studijní program:	**B0714A150008 Automatizace**
Téma práce:	**Řešení vybraných typů diferenciálních rovnic s podporou Matlabu.**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Uvést základní definici pojmu diferenciální rovnice prvního i vyššího řádu pro funkce jedné reálné proměnné. Nastudovat a uvést numerické metody (Runge-Kutta) řešení Cauchyových diferenciálních rovnic. Vypracovat M-skripty pro řešení diferenciálních rovnic prostředky Matlabu. Nastudovat a uvést, jak lze v Matlabu v toolboxu Symbolické matematiky řešit lineární rovnice prvního a vyšších řádů. Vytvořit kolekci ilustrativních příkladů, popř. uvést aplikaci řešení diferenciálních rovnic v oboru.

Rozsah pracovní zprávy: **30 – 60**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Zahrádka, J. *Diskrétní matematika pro SII – diskretizační metody numerické matematiky*. Pardubice, 2014. ISBN 9788073958411.
2. Zahrádka, J. Matematický seminář – MATLAB. Pardubice: Univerzita Pardubice, 2013. ISBN 978-80--7395-691-2.

Vedoucí bakalářské práce: **RNDr. Jaromír Zahrádka, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 12. ledna 2024

Prohlašuji:

Práci s názvem Řešení vybraných typů diferenciálních rovnic s podporou Matlabu jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10.5.2024 Daniel Mládek v. r.

PODĚKOVÁNÍ

Rád bych vyjádřil vděčnost vedoucímu bakalářské práce RNDr. Jaromíru Zahrádkovi, Ph.D. za poskytnutý čas, zodpovězené dotazy a rady ohledně formální úpravy bakalářské práce. Dále jsem vděčný za všechny nabyté vědomosti, které jsem získal během bakalářského studia na této fakultě.

ANOTACE

Tato práce se zaměřuje na základní definici diferenciálních rovnic prvního a vyššího řádu a na studium numerických metod, zejména metody Runge-Kutta. Dále se zabývá zavedením těchto metod do programovacího prostředí Matlab.

KLÍČOVÁ SLOVA

diferenciální rovnice, Matlab, numerické metody, metody Runge-Kutta

TITLE

Solving selected types of differential equations with Matlab support

ANNOTATION

This thesis focuses on the basic definition of differential equations of the first and higher order and on the study of numerical methods, especially the Runge-Kutta method. It also deals with the implementation of these methods in the Matlab programming environment.

KEYWORDS

differential equations, Matlab, numerical methods, Runge-Kutta methods

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK A ZNAČEK

GUI……Graphic user interface (grafické uživatelské rozhraní)

Matlab…….…….Matrix laboratory

ODR……Obyčejná diferenciální rovnice

SW……..Software

TZV……..……Takzvaný/ně

ÚVOD

Diferenciální rovnice patří mezi základní koncepty matematické analýzy a mají široké uplatnění v různých vědeckých oborech, inženýrství a dalších aplikovaných disciplínách. Tyto rovnice popisují vztahy mezi neznámými funkcemi a jejich derivacemi a představují základní nástroj pro modelování dynamických systémů.

Cílem této bakalářské práce je představit základní definice diferenciálních rovnic, se zaměřením na rovnice prvního a vyššího řádu pro funkce jedné reálné proměnné, a prostřednictvím této teoretické základny se seznámit s numerickými metodami pro jejich řešení. Konkrétně se zaměřením na metodu Runge-Kutta a její implementaci v programovacím prostředí Matlab.

První část práce bude věnována krátkému úvodu do programovacího prostředí Matlab, který je v této práci využíván. Druhá část bude zaměřena na teorii o diferenciálních rovnicích, typy diferenciálních rovnic a výpočty. Třetí část bude věnována numerickým metodám, kterými se diferenciální rovnice počítají. V části čtyři budou uvedeny příklady použití diferenciálních rovnic v oboru.

Celkovým cílem práce je nejenom demonstrovat teoretické znalosti v oblasti diferenciálních rovnic, ale především prakticky aplikovat tyto znalosti na konkrétní příklady, které budou zapsány zejména pomocí programovacího vybavení Matlab.

1 PROGRAMOVACÍ PROSTŘEDÍ MATLAB

Zkratkou Matlab je z anglického označení Matrix laboratory, tj. maticová laboratoř. Matlab je výkonný programovací jazyk se zaměřením pro technické účely. V uživatelském prostředí zahrnuje výpočty, programování a vizualizaci. Řešení je vyjádřeno pomocí matematických symbolů. Typické použití je:

- Matematika
- Vývoj algoritmů
- Modelování a simulace
- Vědecká a inženýrská vizualizace
- Analýza dat
- Tvorba grafického uživatelského rozhraní (GUI)

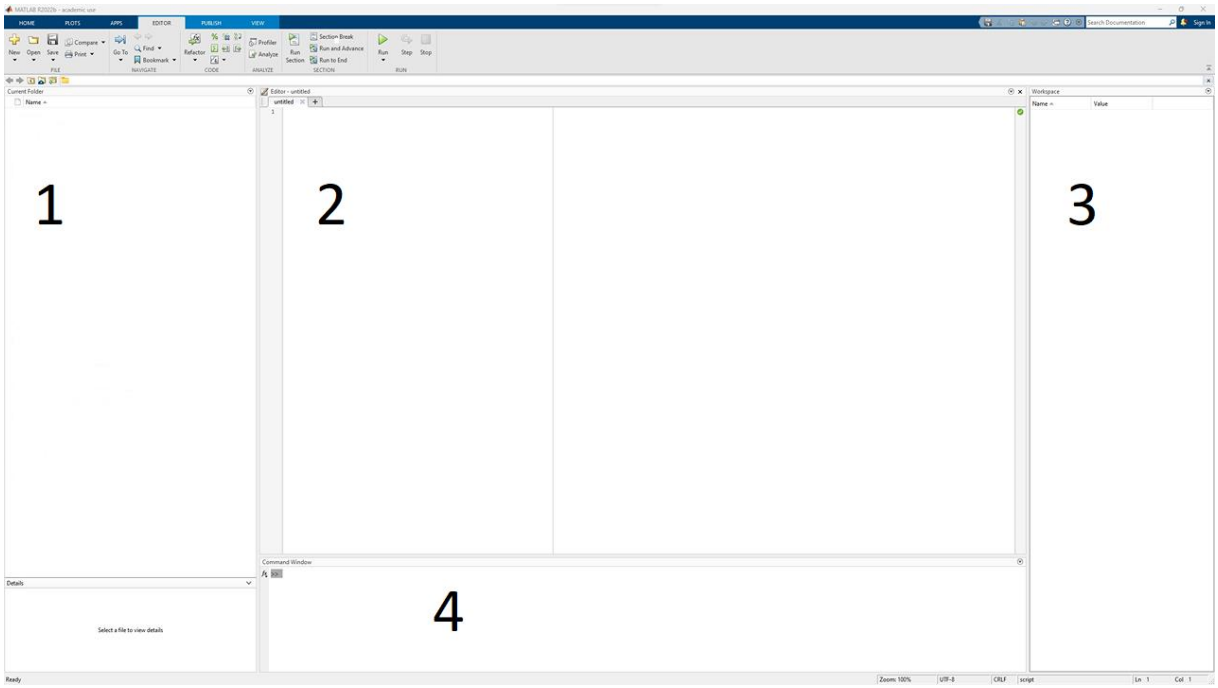
V akademickém prostředí se tento jazyk hojně využívá pro matematické výpočty. V oblasti průmyslu se Matlab využívá pro vysokou produktivitu, analýzu a vývoj. Matlab je interaktivní systém, jehož základním prvkem dat je pole, které nevyžaduje dimenzování.[6]

Matlab obsahuje aplikace pro specifické řešení úloh nazvané toolboxy. Tyto toolboxy jsou velice důležité, neboť rozšiřují programové prostředí o jeho funkčnost.[6]

V této práci bude využit toolbox SymMath.

Pomocí tohoto toolboxu lze poté využít příkazu $dsolve$ pro řešení diferenciálních rovnic. Příkaz $dsolve$ je ve tvaru $dsolve(´D-equation´, ´condition_1´, \dots, ´condition_n)$, kde $D-equation$ je řešená diferenciální rovnice v syntaxi Matlabu. Podmínky $condition_1, \dots, condition_n$ jsou nepovinné počáteční podmínky. Pokud nejsou počáteční podmínky zadány, je výsledkem obecné řešení diferenciální rovnice se zastoupením obecných konstant, které jsou označeny jako $C2, C3, \dots, C_n$.[11]

Obrázek 1- Prostředí Matlabu



Prostředí Matlabu se dělí do čtyř základních segmentů:

1 Aktuální složka – přístup k souborům.

2 Editor – zápis kódu.

3 Workspace – zobrazení dat, která se vytvoří nebo importují ze souborů.

4 Příkazová řádka – zadávání příkazů.

2 SEZNÁMENÍ S DIFERENCIÁLNIMI ROVNICEMI

2.1 Obyčejná diferenciální rovnice

Obyčejná diferenciální rovnice, zkráceně a dále jen ODR, je typ rovnice, kde je neznámou funkce jedné reálné proměnné a zahrnuje minimálně jednu derivaci této neznámé funkce. Neznámá funkce se v diferenciální rovnici nemusí vyskytovat, ale je nezbytné, aby rovnice obsahovala alespoň jednu z jejích derivací, ať už první, druhou, nebo až n-tou. Neznámá funkce se obvykle zapisuje jako y a označení x se používá pro nezávislou proměnnou.[10]

Diferenciální rovnice může obsahovat i počáteční podmínky. Počáteční podmínky stanovují funkční hodnoty neznámé funkce nebo jejich derivací. V případě, že diferenciální rovnice prvního řádu obsahuje jednu počáteční podmínku, pak platí rovnost $y_0 = f(x_0)$. Tato rovnost určuje hodnotu neznámé funkce v bodě x_0. Pro diferenciální rovnici n-tého řádu platí, že může obsahovat až n počátečních podmínek.[10]

Obecně se diferenciální rovnice charakterizují podle několika parametrů, jako je například jejich řád, lineární nebo nelineární charakter a homogenita.

2.2 Lineární ODR

ODR tohoto typu je charakterizována tak, že je lineární ve vztahu k neznámým funkcím a jejích derivacím. To znamená, že neznámé funkce i jejich derivace se v rovnici vyskytují pouze v první mocnině a nejsou mezi sebou násobeny.

2.3 Nelineární ODR

Nelineární ODR se charakterizuje tím, že obsahuje nelineární výrazy vzhledem k neznámým funkcím a jejich derivacím.

2.4 Diferenciální rovnice 1.řádu

Diferenciální rovnice prvního řádu zahrnují pouze první derivaci neznámé funkce. Tyto rovnice mají obecnou formu $F(x, y, y') = 0$, kde $y = y(x)$ představuje hledanou funkci proměnné x a $y' = y'(x)$ je její první derivace. Řešením je funkce $y = y(x)$, která je definována na určitém otevřeném intervalu.[9]

2.5 Rovnice v separovatelné formě

Diferenciální rovnice ve formě umožňující separaci proměnných je typ rovnice, který lze upravit tak, aby na jedné straně byl výraz obsahující neznámou (hledanou) funkci y násobenou diferenciálem dy, zatímco na druhé straně rovnice by byla funkce proměnné x násobená diferenciálem dx. Obecně je toto označováno jako separace proměnných.[3]

ODR 1. řádu ve tvaru $y' = f(x)g(y)$ se říká rovnice se separovatelnými proměnnými. Pokud f a g jsou na otevřených intervalech spojité funkce a zároveň platí, že $g \neq 0$, pak každé řešení lze získat postupem, kde $y' = \frac{dy}{dx}$. Formálně lze tedy diferenciální rovnici v separovatelné formě zapsat:[3]

$$\frac{dy}{dx} = f(x)g(y),$$

kde $\frac{dy}{dx}$ je derivace neznámé funkce y podle x a $f(x)$ a $g(y)$ jsou funkce proměnných x a y. Tento tvar umožňuje oddělení proměnných x a y na jednotlivé strany rovnice a následné řešení pomocí integrace. Obecný postup:

$$\frac{dy}{dx} = f(x)g(y)$$

Převedení x a y na jejich příslušné strany:

$$\frac{dy}{g(y)} = f(x)dx$$

Získaná rovnice se zintegruje a řešením je:

$$\int \frac{dy}{g(y)} = \int f(x)dx$$

Postup řešení

1. Rozdělení proměnných. Přeformulování diferenciální rovnice tak, aby na jedné straně byly pouze proměnné x a na druhé straně pouze y.
2. Integrace. Integrování obou stran rovnice s ohledem na jejich proměnné a následné přičtení integrační konstanty C.
3. Řešení počátečních podmínek. Dosazení počátečních podmínek do obecného řešení a následný výpočet konstanty integrace C.
4. Získání finálního řešení. Dosazení nalezené konstanty zpět do obecného řešení.

Tabulka 1- Diferenciální rovnice v separovatelné formě

$$\frac{dy}{dx} = x^2 \cdot y$$

Řešení

Po oddělení diferenciálu:

$dy = x^2 \cdot y dx$

Po vydělení rovnice y:

$\frac{1}{y} dy = x^2 dx$

Po tomto kroku lze rovnici integrovat:

$\int \frac{1}{y} dy = \int x^2 dx$

Z čehož se získá výsledek:

$\ln(|y|) + C = \frac{x^3}{3} + C$, kde $C \in R$ je integrační konstanta.

Převedením logaritmu do exponenciální formy, kde $\ln(x) = b, x = e^b$:

$|y| = e^{\frac{x^3}{3} + C}$

Jelikož $a^{m+n} = a^m \cdot a^n$, potom tedy:

$|y| = e^{\frac{x^3}{3}} \cdot e^C$

Protože e^C je konstanta, obecným řešením diferenciální rovnice je:

$y = C \cdot e^{\frac{x^3}{3}}$

Při zadané počáteční podmínce, např. $y(0) = 2$ se získá rovnice pro neznámou C:

$2 = C \cdot e^0$

$C = 2$

Po dosazení konstanty C do původního výrazu se získá požadované partikulární řešení:

$y = 2 \cdot e^{\frac{x^3}{3}}$

Řešení pomocí SW Matlab

```
%Definice symbolických proměnných
syms y(x);

%Zadaná diferenciální rovnice
rovnice = x^2 * y == diff(y);

%Převedení rovnice do separovatelné formy
sep_form = dsolve(rovnice);

%Vypsání výsledku
disp(sep_form);
```

Výstupem je funkce:

$y = C_1 \cdot e^{\frac{x^3}{3}}$

2.6 Diferenciální rovnice řešené přímou integrací

Tento typ řešení diferenciálních rovnic je ve tvaru $y = \int f(x)dx$. Tento tvar je ale ovšem možný, pokud diferenciální rovnice prvního řádu neobsahuje proměnnou y a zároveň pokud je možné samotnou diferenciální rovnici upravit na tvar $y' = f(x)$. Obecné řešení pak bude obsahovat integrační konstantu C, kde $C \in$ R.[9]

Pro diferenciální rovnice n-tého řádu lze toto řešení použít taky. Rovnice musí splňovat stejné podmínky, neobsahuje proměnnou y a lze ji upravit na tvar $y^n = f(x)$. Řešení takové rovnice poté závisí na čísle n a podle něho je určen počet integrací. Např.: $y^{n-2} = \int(\int f(x)dx)dx$. Obecné řešení pak obsahuje n integračních konstant $C_1, \dots, C_n$, $C \in$ R.[9]

Postup řešení:

1. Rozdělení proměnných. Pokud je diferenciální rovnice 1. řádu, lze použít metodu separace proměnných.
2. Integrace. Integrování obou stran rovnice s ohledem na proměnné. Tím se vyřeší obecné řešení diferenciální rovnice.
3. Určení konstant. Pokud jsou obsaženy počáteční podmínky.

Tabulka 2- Diferenciální rovnice řešená přímou integrací

$$y' = \frac{x-2}{x-3}$$

Řešení	Řešení pomocí SW Matlab

Tento tvar se převede na tvar integrálu:
$\int \frac{x-2}{x-3}dx$
Následující tvar se dá řešit např. substitucí:
$t = x - 3$
$\int \frac{t+1}{t}dt$
$\int \frac{t}{t} + \frac{1}{t}dt$
$\int 1 + \frac{1}{t}dt$
$\int 1\,dt + \int \frac{1}{t}dt$
	`%Definice symbolických proměnných`
	`syms x;`
	
	`%Zadaná rovnice řešená pomocí metody přímého integrování`
	`vysledek = int((x-2)/(x-3), x);`
	
	`%Vypsání výsledků`
	`disp(vysledek);`
	
	Výstupem je funkce:
	$y = x + \ln(x-3)$

Po integraci:

$t + \ln(|t|) + C$, kde $C \in R$ je integrační konstanta

Po tomto kroku lze provést desubstituci:

$t = x - 3$

Výsledek je:

$x - 3 + \ln(|x - 3|) + C$

$x + \ln(|x - 3|) + C$

2.7 Diferenciální rovnice homogenní

Homogenní diferenciální rovnice je definována tak, že $y' = f(\frac{y}{x})$, kde f je spojitou funkcí na určitém otevřeném intervalu. Homogenní tvar rovnice je v následujícím tvaru, pravá strana rovnice obsahuje proměnné x a y pouze ve spojení $\frac{y}{x}$ či naopak $\frac{x}{y}$.[5]

Zároveň, termínem homogenní jsou označovány i rovnice, které mají pravou stranu rovnu nule. Tyto rovnice mají tvar $y' + P(x)y = Q(x)$, kde P i Q jsou funkce jedné proměnné. Pro homogenní typ rovnice platí, že Q(x) = 0.[5]

Tento typ rovnice lze řešit pomocí substituční metody, kde se zavede nová proměnná, například $u = \frac{y}{x}$, což umožňuje převést rovnici na separabilní formu a následně ji integrovat.

Postup řešení

1. Zavedení substituce $u = \frac{y}{x}$ a odtud $y = u \cdot x$. Potom $y' = u' \cdot x + u$, kde $u' = \frac{du}{dx}$.
2. Užití těchto tvarů a jejich dosazení do původní rovnice. $u' \cdot x + u = f(u)$, potom $u' = \frac{1}{x} \cdot (f(u) - u)$. Pro tento tvar rovnice lze řešení $u(x)$ najít pomocí separací proměnných.
3. Obecné řešení se poté nachází pomocí vztahu $y(x) = x \cdot u(x)$.
4. Určení konstant. Pokud jsou k dispozici.

Tabulka 3- Diferenciální rovnice homogenní

$$y' = \frac{2 \cdot y - x}{x}$$

Řešení

Zavede se substituce podle postupu uvedeného výše. A dosadí se do původní rovnice.

$$u' \cdot x + u = \frac{2 \cdot (u \cdot x) - x}{x}$$

$$u' \cdot x + u = 2 \cdot u - 1$$

$$u' \cdot x = u - 1$$

Tento tvar je řešitelný pomocí separace proměnných. Rozdělí se tedy rovnice na strany obsahující u a x.

$$\frac{du}{u - 1} = \frac{dx}{x}$$

A obě strany rovnice se zintegrují.

$$\int \frac{du}{u - 1} = \int \frac{dx}{x}$$

$\ln|u - 1| = \ln|x| + C$, kde C je integrační konstanta. Po tomto kroku se využije exponenciální funkce k odstranění logaritmů. Obecně platí $e^{\ln(x)} = x$.

$$u - 1 = C \cdot x$$

$$u = C \cdot x + 1$$

Dosadí se $u = \frac{y}{x}$.

$$\frac{y}{x} = C \cdot x + 1$$

Výsledkem je:

$$y = C \cdot x^2 + x$$

Řešení pomocí SW Matlab

```
%Definice symbolických proměnných
syms y(x);

%Zadaná diferenciální rovnice
rovnice = (2*y-x)/x == diff(y);

%Vypočítání rovnice
vysledek = dsolve(rovnice);

%Vypsání výsledku
disp(vysledek);
```

Výstupem je funkce:

$$y = C_1 \cdot x^2 + x$$

2.8 Diferenciální rovnice nehomogenní

Obecným řešením nehomogenních diferenciálních rovnic je tvar $y = y_h + y_p$. Partikulární část řešení se označuje jako y_p a y_h je homogenní část řešení n-tého řádu příslušné k rovnici:

$$a_n y^n(x) + a_{n-1} y^{n-1}(x) + \cdots + a_1 y'(x) + a_0 y(x) = f(x),$$

kde $a_0, a_1, \ldots a_n$ jsou konstanty, přičemž platí, že $a_n \neq 0$ a $f(x)$ je spojitá a nenulová funkce na určitém intervalu.

Obecné řešení rovnice je ve tvaru:

$$a_n y^n(x) + a_{n-1} y^{n-1}(x) + \cdots + a_1 y'(x) + a_0 y(x) = 0[5]$$

Řešení rovnice lze nalézt například pomocí metody neurčitých koeficientů.

Postup řešení

1. Nalezení řešení homogenní rovnice. Vyřešení příslušné homogenní rovnice čili položení levé strany rovnice vůči nule a nalezení obecného řešení. To bude část $y_h(x)$.
2. Nalezení partikulárního řešení nehomogenní rovnice. Druhým krokem je najít partikulární řešení nehomogenní rovnice. To bude část $y_p(x)$.
3. Nalezení celkového řešení. Nalezení celkového řešení nehomogenní rovnice zahrnuje součet obecného řešení homogenní rovnice a partikulárního řešení nehomogenní rovnice. Tedy $y(x) = y_h(x) + y_p(x)$.
4. Určení konstant, pokud jsou k dispozici.

Tabulka 4- Diferenciální rovnice nehomogenní

$$\frac{dy}{dx} + 2 \cdot y = 4 \cdot x$$

Řešení

Vyřešení homogenního tvaru rovnice:

$$\frac{dy}{dx} + 2 \cdot y = 0$$

Pomocí separace proměnných:

$$y_h = \frac{C}{e^{2x}}$$

y_p je partikulární řešení. Protože pravá strana rovnice je polynom, lze předpokládat, že y_p bude také polynom stejného stupně jako pravá strana, tedy $y_p = A \cdot x + B$, kde A i B jsou neznámé konstanty, které se musí určit.

Dosazením y_p do původní nehomogenní rovnice:

$$\frac{d}{dx}(A \cdot x + B) + 2 \cdot (A \cdot x + B) = 4 \cdot x$$

Po derivaci:

$$A + 2 \cdot A \cdot x + 2 \cdot B = 4 \cdot x$$

Porovnáním koeficientů na levé a pravé straně rovnice:

1. Pro x^1 členy: $2 \cdot A = 4$, takže $A = 2$.
2. Pro x^0 členy: $A + 2 \cdot B = 0$, dosazením části $A = 2$ je výsledek $2 + 2 \cdot B = 0$, takže část $B = -1$.

Partikulární řešení $y_p = 2 \cdot x - 1$

Celkové řešení nehomogenní lineární diferenciální rovnice je součet homogenního a partikulárního řešení:

$y = y_h + y_p = \frac{C}{e^{2x}} + 2 \cdot x - 1$, kde $C \in R$ je libovolná konstanta.

Řešení pomocí SW Matlab

```
%Definice symbolických proměnných
syms y(x);

%Zadaná nehomogenní diferenciální
rovnice
odr = diff(y) + 2*y == 4*x;

%Řešení nehomogenní diferenciální
rovnice
vysledek = dsolve(odr);

%Vypsání výsledku
disp(vysledek);
```

Výstupem je funkce:

$$y = 2 \cdot x + C_1 . e^{-2 \cdot x} - 1$$

2.9 Diferenciální rovnice 2. řádu

Diferenciální rovnice vyššího řádu se vyznačují tím, že obsahují více derivací neznámé funkce, nikoli pouze jednu. Diferenciální rovnice vyššího řádu má tvar $F(x, y, y', y'', \dots y^n)$, kde x je nezávislá proměnná, y je hledaná funkce, $y', y'', \dots, y^n$ jsou postupy derivování a F je funkce, která může obsahovat $x, y, y', y'', \dots, y^n$.[9]

Tabulka 5- Diferenciální rovnice druhého řádu

$$y'' - 3 \cdot y' + 2 \cdot y = 0$$

Řešení	Řešení pomocí SW Matlab

Charakteristická rovnice pro lineární diferenciální rovnici 2. řádu má tvar:
$r^2 + 3 \cdot r + 2 = 0$
Kořeny této rovnice lze nalézt například pomocí kvadratické formule či faktorizace:
$(r - 1) \cdot (r - 2) = 0$
Z toho vyplývá, že kořen $r_1 = 1$ a kořen $r_2 = 2$
Obecné řešení této diferenciální rovnice bude kombinace exponenciální funkce $e^{r_1 \cdot x}$ a $e^{r_2 \cdot x}$, kde r_1 a r_2 jsou kořeny charakteristické rovnice:
$y(x) = C_1 \cdot e^x + C_2 \cdot e^{2x}$, kde C_1 a C_2 jsou libovolné konstanty, $C \in R$.
	`%Definice symbolických proměnných`
	`syms y(x);`
	
	`%Zadaná diferenciální rovnice druhého řádu`
	`odr = diff(y, 2) - 3*diff(y) + 2*y == 0;`
	
	`%Řešení diferenciální rovnice druhého řádu`
	`vysledek = dsolve(odr);`
	
	`%Vypsání výsledku`
	`disp(vysledek);`
	
	Výstupem je funkce:
	$y = C_1 . e^x + C_2 \cdot e^{2x}$

3 NUMERICKÉ METODY PRO ŘEŠENÍ ODR

Numerické metody mají společný znak, řešení se nehledá jako spojitá funkce v celém zkoumaném intervalu $\langle a, b \rangle$, ale vypočítávají se hodnoty přibližného řešení v konečném počtu bodů $a = x_0 < x_1 < \cdots < x_n = b$, kde jednotlivé body se nazývají uzlové body a množina hodnot $x_0, x_1, \ldots, x_n$ se nazývá síť. Rozdíl v síti mezi jednotlivými uzly se nazývá krok h. Hodnoty řešení se poté značí jako $y_0, y_1, \ldots, y_n$.[2]

Numerické metody mají princip takový, že hledají přibližné hodnoty řešení v uzlových bodech a jejich cílem je nalezení aproximace $y_0, y_1, \ldots, y_n$ řešení $y(x_0), y(x_1), \ldots, y(x_n)$.[8]

Kategorie numerických metod:

1. Jednokrokové

Pro výpočet nové aproximace y_{i+1} v bodě x_{i+1} využívají pouze předchozích hodnot aproximace y_i.

2. Vícekrokové

Pro výpočet aproximace y_{i+1} v bodě x_{i+1} se používají předchozí aproximace $y_i, \ldots, y_n$.[8]

Následující metody budou provedeny v programovacím prostředí Matlab.

3.1 Eulerova metoda 1. řádu

Eulerova metoda je nejjednodušší jednokrokovou metodou pro numerické řešení úloh.

Pro rovnici $y' = f(x, y), y(x_0) = y_0$ s pravidelnými uzly $(x_0, x_1, \ldots, x_n)$ a krokem h bude ve všech bodech uzlů platit:

$$y'(x_i) = f\big(x_i, y(x_i)\big)$$

Derivace levé strany se poté dá přepsat do tvaru:

$$\frac{y(x_{i+1}) - y(x_i)}{h} \approx f\big(x_i, y(x_i)\big)$$

Nahrazením $y(x_i)$ přibližnou hodnotou y_i, lze dostat vyjádření přibližné hodnoty $y(x_{i+1})$:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Z tohoto vzorce lze poté vypočítat přibližnou hodnotu řešení v dalším uzlu pomocí uzlu předchozího.[2]

Tabulka 6- Eulerova metoda prvního řádu

$$y' = x^2 - y,\ \langle 0\,;0{,}6\rangle,\ y(0) = 3, h_1 = 0{,}1, h_2 = 0{,}2$$

Řešení pomocí SW Matlab

```
%Počáteční podmínky
x0 = 0;
y0 = 3;

%Konečný bod intervalu
x_max = 0.6;

%Velikost kroku
h = 0.1;
h2 = 0.2;

%Počet kroků
n = ceil((x_max - x0) / h);
n2 = ceil((x_max - x0) / h2);

%Inicializace polí pro ukládání hodnot
x_values = zeros(1, n+1);
y_values = zeros(1, n+1);
x2_values = zeros(1, n2+1);
y2_values = zeros(1, n2+1);

%Počáteční hodnoty
x_values(1) = x0;
y_values(1) = y0;
x2_values(1) = x0;
y2_values(1) = y0;

%Eulerova metoda 1. řádu pro h = 0,1
for i = 1:n
x = x_values(i);
y = y_values(i);
f = x^2 - y;
x_values(i+1) = x + h;
y_values(i+1) = y + h * f;
end

%Eulerova metoda 1. řádu pro h = 0,2
for i = 1:n2
x = x2_values(i);
y = y2_values(i);
f = x^2 - y;
x2_values(i+1) = x + h2;
y2_values(i+1) = y + h2 * f;
end

%Vypsání výsledků
disp('Pro h = 0,1:');
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);
disp('Pro h = 0,2:');
disp(['Hodnoty x: ', num2str(x2_values)]);
disp(['Hodnoty y: ', num2str(y2_values)]);
```

Výstup

h = 0,1	
x	**y**
0	3
0,1	2,7
0,2	2,431
0,3	2,1919
0,4	1,9817
0,5	1,7995
0,6	1,6446
h = 0,2	
x	**y**
0	3
0,2	2,4
0,4	1,928
0,6	1,5744

Přesné řešení této rovnice vyřešené pomocí příkazu $dsolve$ je $y(x) = C_1 e^{-x} + x^2 - 2x + 2$. Pro porovnání výsledků slouží následující tabulka. Z tabulky lze vidět, že při zvyšujícím se h a zároveň při zvyšujícím se x klesá přesnost. Odchylky vznikají z aproximace řešení, Eulerova metoda aproximuje skutečné řešení diferenciální rovnice pomocí interpolace mezi jednotlivými body. Čím větší je krok h, tím větší kroky jsou mezi body aproximace a tím je i aproximace hrubší.

Tabulka 7- Porovnání velikosti kroku pro přesnost Eulerovy metody prvního řádu

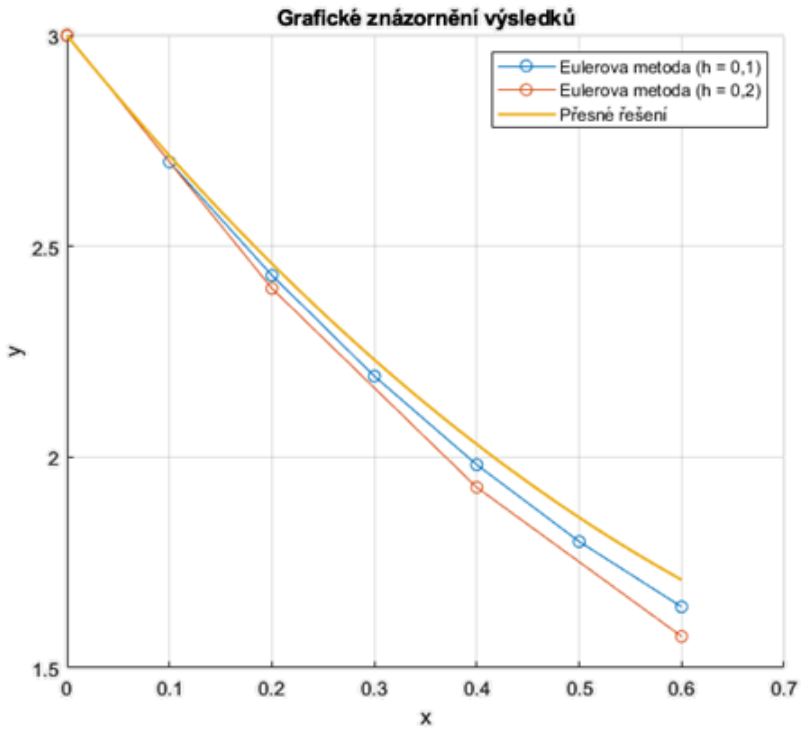
x	y - přesné	h = 0,1		h = 0,2	
		y	odchylka	y	odchylka
0	3,0000	3,0000	0	3,0000	0
0,1000	2,7148	2,7000	- 0,0148	-	-
0,2000	2,4587	2,431	- 0,0277	2,4000	- 0,0587
0,3000	2,2308	2,1919	- 0,0389	-	-
0,4000	2,0303	1,9817	- 0,0486	1,9280	- 0,1023
0,5000	1,8565	1,7995	- 0,0570	-	-
0,6000	1,7088	1,6446	- 0,0642	1,5744	- 0,1344

Grafické porovnání výsledků lze v Matlabu zajistit následujícím kódem, který by se přidal pod kód předešlý pro výpočet Eulerovy metody 1. řádu.

Přídavná část kódu pro grafické znázornění výsledků

```
%Přesné řešení
syms x y(x);
rovnice = diff(y) == x^2 - y;
y_presne = dsolve(rovnice, y(0) == y0);

%Hodnoty pro přesné řešení
x_presne = linspace(x0, x_max, 100);
y_presne_hodnoty = double(subs(y_presne, x_presne));

%Vykreslení grafu
figure;
hold on;
plot(x_values, y_values, '-o', 'DisplayName', 'Eulerova metoda (h = 0,1)');
plot(x2_values, y2_values, '-o', 'DisplayName', 'Eulerova metoda (h = 0,2)');
plot(x_presne,  y_presne_hodnoty,  'LineWidth',  1.5,  'DisplayName',  'Přesné
řešení');
xlabel('x');
ylabel('y');
title('Grafické znázornění výsledků');
legend;
grid on;
hold off;
```

Obrázek 2- Porovnání kroků Eulerovy metody 1. řádu



3.2 Eulerova metoda 2. řádu

Eulerova metoda 2. řádu je vylepšená verze původní Eulerovy metody. Tato metoda se odlišuje od původní verze tím, že nevyužívá jen derivace v bodě y_i, ale bere v úvahu derivace ve dvou různých bodech.

Tento typ metody je zařazen mezi jednokrokové metody Runge-Kutta, které umožňují nalezení přesnější aproximace řešení diferenciální rovnice $y' = f(x, y)$, kde počáteční podmínka $y(x_0) = y_0$. Pro aproximaci y_{i+1} v bodě uzlu x_{i+1} platí vztah:[10]

$$y_{i+1} = y_i + \frac{h}{2} f(x_i + h, y_i + hf(x_i, y_i))$$

Tento vztah lze přepsat do tvaru:

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + h, y_i + k_1),$$

kde k_1 je krok vypočítaný na začátku intervalu a k_2 je krok vypočítaný na konci intervalu.

A potom se aproximace y_{i+1} rovná vztahu:

$$y_{i+1} = y_i + \frac{k_1 + k_2}{2}$$

Pomocí následujícího kódu se definuje požadovaná funkce, počáteční podmínka, interval a velikost kroku. Poté se tyto parametry volají ve funkci pro počítání.

Kód zadaný do příkazové řádky

```
%Definice diferenciální rovnice
f = @(x, y) x^2 - y;
%Počáteční podmínka
x0 = 0;
y0 = 3;
%Konečný bod intervalu
x_max = 0.6;
%Velikost kroku
h = 0.1;
%Volání funkce pro Eulerovu metodu 2. řádu
[x_values, y_values] = eulerova_2rad(f, x0, y0, x_max, h);
```

Vstupní proměnné

f reprezentuje pravou stranu ODR.

x_0 je počáteční hodnota nezávislé proměnné x.

y_0 je počáteční hodnota závislé proměnné y v bodě $x = x_0$.

x_max je horní mezinterva-lu.

h je velikost kroku.

Výstupní proměnné

x_values je pole hodnot nezávislé proměnné od x_0 do x_max s krokem h.

y_values je pole hodnot, do kterého budou postupně ukládány vypočítané hodno-ty závislé proměnné odpoví-dající hodnotám v x_values.

Eulerova metoda 2. řádu

Cyklem for se iteruje přes všechny body x_values.

V každé iteraci se spočítá hodnota k_1 a k_2 podle vzorců Eulerovy metody 2. řádu.

Na základě těchto hodnot se vypočítá další hodnota y pomocí příslušného vzorce Eulerovy metody 2. řádu.

Vypsání výsledků

Hodnoty x a y se vypíší do příkazové řádky pomocí příkazu $disp$.

Vykreslení grafu

Pomocí příkazu $figure$ dojde k vytvoření nového okna pro graf. Příkaz $plot$ vykresluje graf zadaných dat. Příkazy $xlabel$ a $ylabel$ slouží k přidání popisku osy x a y. Příkaz $title$ přidá titulek grafu a příkaz $grid\ on$ povolí mřížku.

Pomocí kódu v tabulce 8 a předešlého kódu pro definici parametrů se vypočte Eulerova metoda 2. řádu včetně grafického vykreslení.

Tabulka 8- Eulerova metoda druhého řádu

$$y' = x^2 - y, \langle 0; 0{,}6 \rangle, y(0) = 3, h = 0{,}1$$

Řešení pomocí SW Matlab

```
function [x_values, y_values] = eulerova_2rad(f,
x0, y0, x_max, h)
x_values = x0:h:x_max;
y_values = zeros(size(x_values));
y_values(1) = y0;

%Eulerova metoda 2. řádu
for i = 1:(length(x_values) - 1)
k1 = h * f(x_values(i), y_values(i));
k2 = h * f(x_values(i) + h, y_values(i) + k1);
y_values(i + 1) = y_values(i) + (k1 + k2) / 2;
end

%Vypsání výsledků
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);

%Vykreslení grafu
figure;
plot(x_values, y_values, 'b-', 'LineWidth', 1.5);
xlabel('x');
ylabel('y');
title("Řešení diferenciální rovnice");
grid on;
end
```

Výstup

h = 0,1	
x	**y**
0	3
0,1	2,7155
0,2	2,46
0,3	2,2326
0,4	2,0325
0,5	1,8591
0,6	1,7118

Za pomocí stejného principu lze obdobně vypočítat metody další. Rozdíl bude v těle funkce, ve které se samotná metoda počítá.

3.3 Modifikovaná Eulerova metoda 2. řádu

Modifikovaná Eulerova metoda 2. řádu, též známá jako Heunova metoda je časově nenáročná a snadná, nicméně stále méně přesná. Příčinou nepřesností je zde aplikace neměnné hodnoty derivace $f(x_i, y_i)$ na celém intervalu při přechodu z kroku x_i do x_{i+1}.Tato hodnota derivace se ale v průběhu kroku mění.[7]

Tato metoda je rovněž zařazena mezi metody druhého řádu, a proto je úroveň přesnosti modifikované Eulerovy metody druhého řádu porovnatelná s tradiční Eulerovou metodou druhého řádu. Tato metoda využívá pro výpočet aproximace y_{i+1} vztahu:[10]

$$y_{i+1} = y_i + hf(x_i + \frac{h}{2}, y_i + \frac{h}{2} f(x_i, y_i))$$

Postupem výpočtu je vypočítání hodnot k_1 a k_2:

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$

A výsledná aproximace se poté vypočítá pomocí vztahu:

$$y_{i+1} = y_i + k_2$$

Tabulka 9- Modifikovaná Eulerova metoda druhého řádu

Rovnice, podmínky i interval je stejný jako u tabulky 8.

Řešení pomocí SW Matlab

```
function [x_values, y_values] =
eulerova_2rad_mod(f, x0, y0, x_max, h)
x_values = x0:h:x_max;
y_values = zeros(size(x_values));
y_values(1) = y0;

%Modifikovaná Eulerova metoda 2.řádu
for i = 1:(length(x_values) - 1)
k1 = h * f(x_values(i),y_values(i));
k2 = h * f(x_values(i) + h/2, y_values(i) + k1/2);
y_values(i + 1) = y_values(i) + k2;
end

%Vypsání výsledků
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);

%Vykreslení grafu
figure;
plot(x_values, y_values, 'b-', 'LineWidth', 1.5);
xlabel('x');
ylabel('y');
title("Řešení diferenciální rovnice");
grid on;
end
```

Výstup

x	y
0	3
0,1	2,7153
0,2	2,4595
0,3	2,2319
0,4	2,0317
0,5	1,8581
0,6	1,7106

Funkce se poté volá zadáním tohoto kódu do příkazové řádky:

```
%Definice diferenciální rovnice
f = @(x, y) x^2 - y;

%Počáteční podmínka
x0 = 0;
y0 = 3;

%Konečný bod intervalu
x_max = 0.6;

%Velikost kroku
h = 0.1;

%Volání funkce pro modifikovanou Eulerovu metodu 2. řádu
[x_values, y_values] = eulerova_2rad_mod(f, x0, y0, x_max, h);
```

3.4 Runge-Kuttova metoda 3. řádu

Další z numerických metod je Runge-Kuttova metoda 3. řádu. Obecně je tato metoda pro řešení diferenciálních rovnic typu $y' = f(x, y)$ s počáteční podmínkou $y(x_0) = y_0$ ještě přesnější, účinnější, ale složitější. Její složitost plyne z použití více kroků výpočtu k, což umožňuje přesnější aproximaci řešení napříč celým intervalem. Metoda Runge-Kutta třetího řádu pro odhad nové hodnoty využívá čtyři kroky. Pro výpočet aproximace se používají následující hodnoty:[10]

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{3},\ y_i + \frac{k_1}{3})$$

$$k_3 = hf(x_i + \frac{2h}{3}, y_i + \frac{2k_2}{3}),$$

kde k_1 je krok vypočítaný na začátku intervalu, k_2 je krok vypočítaný ve třetině kroku a k_3 je krok vypočítaný na konci intervalu.

Aproximace y_{i+1} je pak dána tímto vztahem:

$$y_{i+1} = y_i + \frac{k_1}{4} + \frac{3k_3}{4}$$

Tabulka 10- Runge-Kuttova metoda třetího řádu

Rovnice, podmínky i interval je stejný jako u tabulky 8.

Řešení pomocí SW Matlab

```
function [x_values, y_values] = rungekutt_3rad(f, x0, y0,
x_max, h)
x_values = x0:h:x_max;
y_values = zeros(size(x_values));
y_values(1) = y0;

%Runge-Kuttova metoda 3.řádu
for i = 1:(length(x_values) - 1)
k1 = h * f(x_values(i),y_values(i));
k2 = h * f(x_values(i) + h/3, y_values(i) + k1/3);
k3 = h * f(x_values(i) + (2*h)/3, y_values(i) + (2*k2)/3);
y_values(i + 1) = y_values(i) + k1/4 + 3*k3/4;
end

%Vypsání výsledků
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);

%Vykreslení grafu
figure;
plot(x_values, y_values, 'b-', 'LineWidth', 1.5);
xlabel('x');
ylabel('y');
title("Řešení diferenciální rovnice");
grid on;
end
```

Výstup

x	y
0	3
0,1	2,7148
0,2	2,4587
0,3	2,2308
0,4	2,0303
0,5	1,8565
0,6	1,7088

Funkce se poté volá obdobně jako u modifikované Eulerovy metody 2. řádu, rozdíl bude pouze u příkazu pro volání funkce, kde se změní název funkce pro danou metodu.

```
%Volání funkce pro Runge-Kuttovu metodu 3. řádu.
[x_values, y_values] = rungekutt_3rad(f, x0, y0, x_max, h);
```

3.5 Runge-Kuttova metoda 4. řádu

Runge-Kuttova metoda čtvrtého řádu je poslední jednokrokovou numerickou metodou pro řešení diferenciálních rovnic v této kapitole. Jedná se o vylepšenou verzi Runge-Kuttovy metody třetího řádu, která poskytuje ještě vyšší přesnost. Tato metoda vyžaduje čtyřnásobné vyhodnocení funkce pro každý bod, což zahrnuje výpočet hodnot k_1, k_2, k_3 *a* k_4. V této kapitole je to nejpřesnější metoda, ale zároveň také ta nejsložitější a časově nejnáročnější.[10]

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_i + h, y_i + k_3),$$

kde k_1 je krok vypočítaný na začátku intervalu, k_2 a k_3 jsou kroky vypočítané v polovině kroku a jsou váženy dvakrát, což znamená, že mají dvojnásobný vliv na výslednou aproximaci a k_4 je krok vypočítaný na konci intervalu.

Aproximace y_{i+1} je pak dána tímto vztahem:

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

Tabulka 11- Runge-Kuttova metoda čtvrtého řádu

Rovnice, podmínky i interval je stejný jako u tabulky 8.

Řešení pomocí SW Matlab

```
function [x_values, y_values] = rungekutt_4rad(f, x0, y0,
x_max, h)
x_values = x0:h:x_max;
y_values = zeros(size(x_values));
y_values(1) = y0;

%Runge-Kuttova metoda 4.řádu
for i = 1:(length(x_values) - 1)
k1 = h * f(x_values(i),y_values(i));
k2 = h * f(x_values(i) + h/2, y_values(i) + k1/2);
k3 = h * f(x_values(i) + h/2, y_values(i) + k2/2);
k4 = h * f(x_values(i) + h,   y_values(i) + k3);
y_values(i + 1) = y_values(i) + (k1 + 2*k2 + 2*k3 +k4)/6;
end

%Vypsání výsledků
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);

%Vykreslení grafu
figure;
plot(x_values, y_values, 'b-', 'LineWidth', 1.5);
xlabel('x');
ylabel('y');
title("Řešení diferenciální rovnice");
grid on;
end
```

Výstup

x	y
0	3
0,1	2,7148
0,2	2,4587
0,3	2,2308
0,4	2,0303
0,5	1,8565
0,6	1,7088

Funkce se poté volá obdobně jako u modifikované Eulerovy metody 2. řádu, rozdíl bude pouze u příkazu pro volání funkce, kde se změní název funkce pro danou metodu.

```
%Volání funkce pro Runge-Kuttovu metodu 4. řádu.
[x_values, y_values] = rungekutt_4rad(f, x0, y0, x_max, h);
```

3.6 Porovnání jednokrokových metod

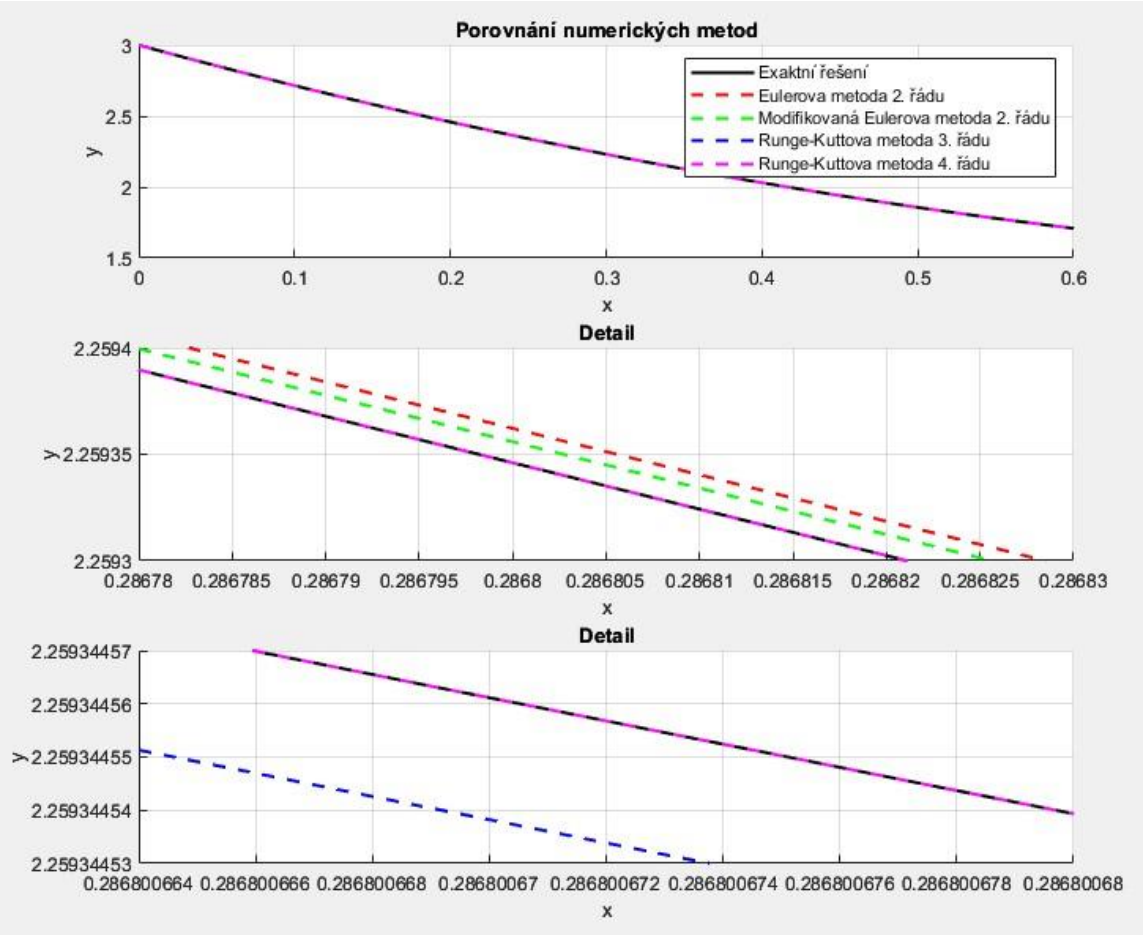
Pomocí následujícího kódu lze porovnat všechny výše zmíněné metody prostřednictvím jednoho grafu, Eulerovu metodu 2. řádu, Modifikovanou Eulerovu metodu 2. řádu, Runge-Kuttovu metodu 3. řádu a Runge-Kuttovu metodu 4. Řádu. Eulerova metoda 1. řádu zde zahrnuta není, je to z toho důvodu, že metoda je proti ostatním velice nepřesná.

```
function porovnani_metod()
%Počáteční podmínky
x0 = 0;
y0 = 3;
%Konečný bod intervalu
x_max = 0.6;
%Velikost kroku
h = 0.01;
%Exaktní řešení
x_presne = x0:0.01:x_max;
y_presne = 2 + exp(-x_presne) + x_presne.^2 - 2*x_presne;
%Eulerova metoda 2. řádu
[euler2_x, euler2_y] = eulerova_2rad(@(x, y) x^2 - y, x0, y0, x_max, h);
%Modifikovaná Eulerova metoda 2. řádu
[mod_euler_x, mod_euler_y] = eulerova_2rad_mod(@(x, y) x^2 - y, x0, y0, x_max,
h);
%Runge-Kuttova metoda 3. řádu
[rk3_x, rk3_y] = rungekutt_3rad(@(x, y) x^2 - y, x0, y0, x_max, h);
%Runge-Kuttova metoda 4. řádu
[rk4_x, rk4_y] = rungekutt_4rad(@(x, y) x^2 - y, x0, y0, x_max, h);
%Vykreslení grafu
figure;
%První podgraf
subplot(3, 1, 1);
hold on;
plot(x_presne, y_presne, 'k-', 'LineWidth', 1.5, 'DisplayName', 'Exaktní
řešení');
plot(euler2_x, euler2_y, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Eulerova metoda
2. řádu');
plot(mod_euler_x, mod_euler_y, 'g--', 'LineWidth', 1.5, 'DisplayName',
'Modifikovaná Eulerova metoda 2. řádu');
plot(rk3_x, rk3_y, 'b--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
3. řádu');
plot(rk4_x, rk4_y, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
4. řádu');
xlabel('x');
ylabel('y');
title('Porovnání numerických metod');
legend();
grid on;
hold off;
%Druhý podgraf
subplot(3, 1, 2);
hold on;
plot(x_presne, y_presne, 'k-', 'LineWidth', 1.5, 'DisplayName', 'Exaktní
řešení');
plot(euler2_x, euler2_y, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Eulerova metoda
2. řádu');
plot(mod_euler_x, mod_euler_y, 'g--', 'LineWidth', 1.5, 'DisplayName',
```

```
'Modifikovaná Eulerova metoda 2. řádu');
plot(rk3_x, rk3_y, 'b--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
3. řádu');
plot(rk4_x, rk4_y, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
4. řádu');
xlabel('x');
ylabel('y');
title('Detail');
grid on;
xlim([0.28678, 0.28683]);
ylim([2.2593, 2.2594]);
hold off;
%Třetí podgraf
subplot(3, 1, 3);
hold on;
plot(x_presne, y_presne, 'k-', 'LineWidth', 1.5, 'DisplayName', 'Exaktní
řešení');
plot(euler2_x, euler2_y, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Eulerova metoda
2. řádu');
plot(mod_euler_x, mod_euler_y, 'g--', 'LineWidth', 1.5, 'DisplayName',
'Modifikovaná Eulerova metoda 2. řádu');
plot(rk3_x, rk3_y, 'b--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
3. řádu');
plot(rk4_x, rk4_y, 'm--', 'LineWidth', 1.5, 'DisplayName', 'Runge-Kuttova metoda
4. řádu');
xlabel('x');
ylabel('y');
title('Detail');
grid on;
xlim([0.286800664, 0.28680068]);
ylim([2.25934453, 2.25934457]);
hold off;
end
```

První podgraf znázorňuje exaktní řešení společně se všemi metodami včetně legendy. Druhý podgraf znázorňuje Eulerovy metody 2. řádu, kde klasická Eulerova metoda 2. řádu vychází jako méně přesná. Poslední podgraf je přiblížen na Runge-Kuttovy metody, kde metoda 4. řádu vychází jako přesnější metoda. Runge-Kuttova metoda 4. řádu je tedy nejpřesnější jednokrokovou metodou pro řešení ODR.

Obrázek 3- Porovnání jednokrokových metod



3.7 Vícekrokové metody

Vícekrokové metody jsou metody, které využívají předchozích hodnot aproximace $y_i, \dots, y_n$ pro výpočet aproximace nové y_{i+1}, proto tyto metody mají složitější začátek, počáteční podmínka totiž udává řešení pouze v jednom bodě. Potřebná řešení v dalších bodech je nutné vypočítat pomocí vhodné jednokrokové metody. Další nevýhodou je obtížná úprava kroku. Při jeho změně je zapotřebí zpětně dopočítat hodnoty řešení v chybějících bodech. [7]

Výhodou vícekrokových metod oproti jednokrokovým je, že pro dosažení vyššího řádu přesnosti u jednokrokových metod je nutné počítat hodnoty pravé strany ve více bodech. Vícekrokové metody tuto vlastnost odstraňují, neboť hodnoty přibližného řešení předchozích kroků jsou pro daný okamžik už započteny.[4]

Tyto metody jsou dané předpisem.

$$\sum_{i=0}^{n} \alpha_i y_{n+1} = h \sum_{i=0}^{n} \beta_i f_{n+1},$$

kde α i β jsou zvolené koeficienty. [4]

$$n = 0,1, \ldots$$

Metoda je explicitní, pokud v řešení nového bodu není třeba znát derivaci v tomto bodě. V této metodě je hodnota nového bodu určena pouze pomocí známých hodnot v předchozích bodech. To znamená, že výpočet nového bodu nepotřebuje znát hodnotu derivace v tomto novém bodě.[7]

Metoda je implicitní, pokud v řešení nového bodu je třeba znát derivaci v tomto bodě. V této metodě je hodnota nového bodu určena pomocí známých hodnot v předchozích bodech a také hodnoty derivace v novém bodě.[7]

3.8 Metoda prediktor-korektor

Metoda prediktor-korektor kombinuje dvě metody, jedna je implicitní a druhá explicitní. Pro dosažení optimální přesnosti je vhodné, aby obě metody byly stejného řádu. Principem této metody je prvotní vypočtení počáteční aproximace pomocí prediktoru, což je metoda explicitní. Výsledná hodnota se následně zpřesní pomocí implicitní metody, tedy korektoru.[2]

Prediktor

$$y_{n+1}^{P} = y_n + hf(x_n, y_n)$$

$$f_{n+1}^{P} = f(x_{n+1}, y_{n+1}^{P})$$

Korektor

$$y_{n+1}^{K} = y_n + \frac{h}{2}(f_{n+1}^{P} + f_n)$$

První odhad řešení y_{n+1}^{P} se vypočítá pomocí prediktoru, v uvedeném vzorci Eulerovou metodou. V tomto bodě se určí hodnota derivace f_{n+1}^{P}, a poté se pomocí korektoru zpřesní odhad řešení. Proces výpočtu derivace a aplikace korektoru se opakuje, dokud se nedosáhne požadované přesnosti u korektoru.[7]

Přidáním tzv. modifikátoru mezi dvojici prediktor-korektor lze dosáhnout větší stability a přesnosti výpočtu.[7]

Tabulka 12- Metoda prediktor-korektor

$$y' = x^2 - y, \langle 0; 0{,}6\rangle, y(0) = 3, h = 0{,}1$$

Řešení pomocí SW Matlab

```
function [x_values, y_values] = prediktor_korektor(f, x0,
y0, x_max, h)
x_values = x0:h:x_max;
y_values = zeros(size(x_values));
y_values(1) = y0;

%Prediktor-korektor
for i = 1:(length(x_values) - 1)

%Prediktor (Eulerova metoda)
prediktor = h * f(x_values(i), y_values(i));
y_prediktor = y_values(i) + prediktor;

%Korektor
korektor = h * f(x_values(i+1), y_prediktor);
y_values(i + 1) = y_values(i) + (prediktor + korektor) / 2;
end

%Vypsání výsledků
disp(['Hodnoty x: ', num2str(x_values)]);
disp(['Hodnoty y: ', num2str(y_values)]);

%Vykreslení grafu
figure;
plot(x_values, y_values, 'b-', 'LineWidth', 1.5);
xlabel('x');
ylabel('y');
title("Řešení diferenciální rovnice");
grid on;
end
```

Výstup

x	y
0	3
0,1	2,7155
0,2	2,46
0,3	2,2326
0,4	2,0325
0,5	1,8591
0,6	1,7118

Funkce se poté volá obdobně jako u modifikované Eulerovy metody 2. řádu, rozdíl bude pouze u příkazu pro volání funkce, kde se změní název funkce pro danou metodu.

```
%Volání funkce pro metodu prediktor-korektor.
[x_values, y_values] = prediktor_korektor(f, x0, y0, x_max, h);
```

4 VYUŽITÍ DIFERENCIÁLNÍCH ROVNIC V OBORU

Diferenciální rovnice jsou užitečným nástrojem v oblasti elektrotechniky či automatizace. Obecně lze říci, že diferenciální rovnice slouží jako nástroj pro analýzu, návrh a simulaci systémů s dynamickým chováním. Jejich užití může být následující:

1. Analýza obvodů

   V elektrotechnice se diferenciální rovnice používají k popisu chování elektrických obvodů. Konkrétním příkladem může být sériový obvod rezistoru a kondenzátoru, kde by se následně užilo Kirchhoffových zákonů a diferenciálních rovnice, které by popisovaly proudy a napětí v obvodu v závislosti na čase.
2. Návrh regulátoru

   V oblasti automatizace se diferenciální rovnice hojně využívají pro navrhování regulátorů, jako jsou PID (P – proporcionální, I – integrační, D – derivační) regulátory. Regulátory se navrhují tak, aby systém dosáhl požadovaného chování, stavu či stability. Diferenciální rovnice mohou být použity k odvození přenosových funkcí (tj. funkce, která popisuje vztah mezi vstupem a výstupem dynamického systému) systému a k návrhu regulátoru, který ovlivňuje chování systému.
3. Modelování systému

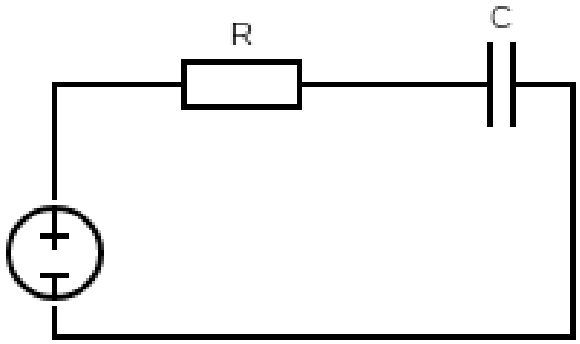
   Dalším využitím diferenciálních rovnic v oblasti automatizace může být modelování dynamických systémů. Konkrétním příkladem může být modelování chování motoru či tepelných systémů.
4. Simulace a predikce

   Posledním zmíněným příkladem použití diferenciálních rovnic v praxi je simulace a predikce chování systémů. Pomocí počítačových programů můžeme řešit diferenciální rovnice k předpovězení, jak se systém bude chovat v různých podmínkách

4.1 Napětí na kondenzátoru

Je obvod, kde je v sérii připojený kondenzátor a rezistor (RC obvod). Tento obvod je připojený k zdroji s konstantním napětím. Pomocí diferenciálních rovnic můžeme zjistit, jak se bude na kondenzátoru měnit napětí v čase.

Obrázek 4- RC obvod



Diferenciální rovnice, která popisuje toto chování, je založena na Kirchhoffově zákoně pro smyčky (Kirchoffův zákon napětí), tento zákon říká, že součet napětí v uzavřené smyčce v obvodu je roven nule.

1. Napětí v obvodu

   $U_{in} - U_r - U_c = 0,$

kde U_{in} je napětí od zdroje, U_r je napětí na rezistoru a U_c je napětí na kondenzátoru. Obdobně pro proud I.

2. Proud

   Jelikož jsou prvky zapojeny v sérii, pak platí:

   $I = I_r = I_c$

Pro rezistor platí Ohmův zákon: $U_r = R \cdot I_r$, kde R je hodnota odporu rezistoru.

Kapacita kondenzátoru je definována jako poměr náboje Q, který je schopen uložit mezi elektrody s napětím U, $C = \frac{Q}{U}$. Proud i, který kondenzátorem teče je definován jako rychlost změny náboje Q s časem t, což znamená derivace náboje podle času, $i(t) = \frac{dQ}{dt}$. Pokud se provede integrace proudu $i(t)$ podle času t od počátečního času t_0 do času t, získá se celkový náboj, který kondenzátorem protekl za časový interval, $Q = \int_{t_0}^{t} i(t)dt$. Dosazením zpět do vzorce pro kapacitu kondenzátoru vzniká vztah $C = \frac{1}{U} \cdot \int_{t_0}^{t} i(t)dt$ odkud lze vyjádřit

vztah pro napětí $U = \frac{1}{C} \cdot \int_{t_0}^{t} i(t)dt$.

Dosazením těchto vztahů do K. zákonů se získá diferenciální rovnice, která popisuje chování obvodu v čase.

$$U_{in} - R \cdot I_r - \frac{1}{C} \cdot \int_0^t I_c \, dt = \ 0$$

Z definice kapacity $C = \frac{Q}{U}$ lze vyjádřit náboj Q jako $Q = C \cdot U$. Pokud se tento vztah derivuje podle času t, potom $\frac{dQ}{dt} = C \cdot \frac{dU}{dt}$. Dosazením do vzorce pro proud $i(t) = C \cdot \frac{dU}{dt}$.

$$U_{in} - R \cdot I_r - \frac{1}{C} \cdot \int_0^t C \cdot \frac{dU_c}{dt} dt = 0$$

Výsledkem je:

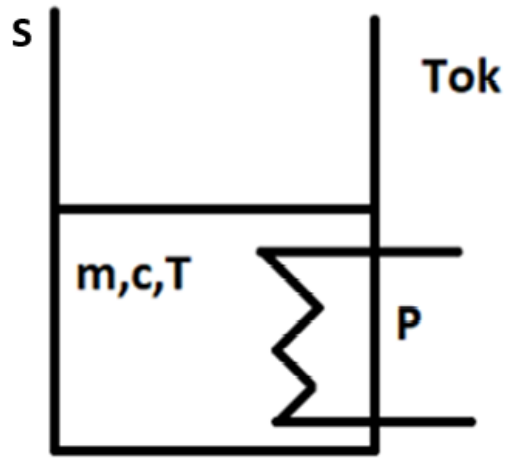
$$U_{in} - R \cdot I_r - \frac{U_c}{C} = 0$$

Řešením této rovnice lze získat konkrétní funkci $U_c(t)$, což umožňuje předpovídat, jak se bude napětí na kondenzátoru měnit v čase a jak bude obvod reagovat na různé podněty.

4.2 Teplota kapaliny v nádobě

Dalším příkladem využití diferenciálních rovnic v oboru může být predikování změny teploty kapaliny v nádobě v závislosti na čase. Pro nádobu, ve které se nachází kapalina, která je ohřívačem s ohřívána platí vztah $\frac{dA}{dt} = Q_{in} - Q_{out}$, kde $\frac{dA}{dt}$ je změna tepelného obsahu $A[J]$ v čase dt, $Q_{in}[W]$ je vstupující tepelný tok a $Q_{out}[W]$ je vystupující tepelný tok (ztráty).[1]

Obrázek 5- Nádoba a její parametry



Měrná tepelná kapacita $c[J \cdot kg^{-1} \cdot K]$ udává množství tepla potřebné k ohřátí jednotkové hmotnosti materiálu o jednu jednotku teploty, $m[kg]$ udává hmotnost kapaliny, $Tok[°C]$ je teplota okolí, $P[W]$ je výkon topné spirály, $S[m^2]$ je plocha stěny nádoby a $T[°C]$ je teplota kapaliny.

Pro vstupní tepelný tok Q_{in} platí, že $Q_{in} = P$, Vzorec pro výstupní tepelný tok určuje tok tepla mezi dvěma body s různými teplotami, $Q_{out} = k \cdot S \cdot (T - T_{ok})$, kde konstanta k je tepelná vodivost materiálu, která závisí na konkrétním materiálu. A pro tepelný obsah A platí, že $A = m \cdot c \cdot T$.

Dosazením do tepelné bilance vzniká vztah $\frac{m \cdot c \cdot dT}{dt} = P - k \cdot S \cdot (T - T_{ok})$, což se následně rovná $\frac{dT}{dt} = \frac{P - k \cdot S \cdot (T - T_{ok})}{m \cdot c}$. Pomocí této diferenciální rovnice lze analyzovat, jak se teplota mění v čase, tj. jak rychle se těleso ohřívá nebo ochlazuje. Samotná diferenciální rovnice poté může být řešena numerickými metodami a tím lze získat hodnoty teploty v průběhu času.

Předešlý příklad s konkrétními parametry v programovacím prostředí Matlab

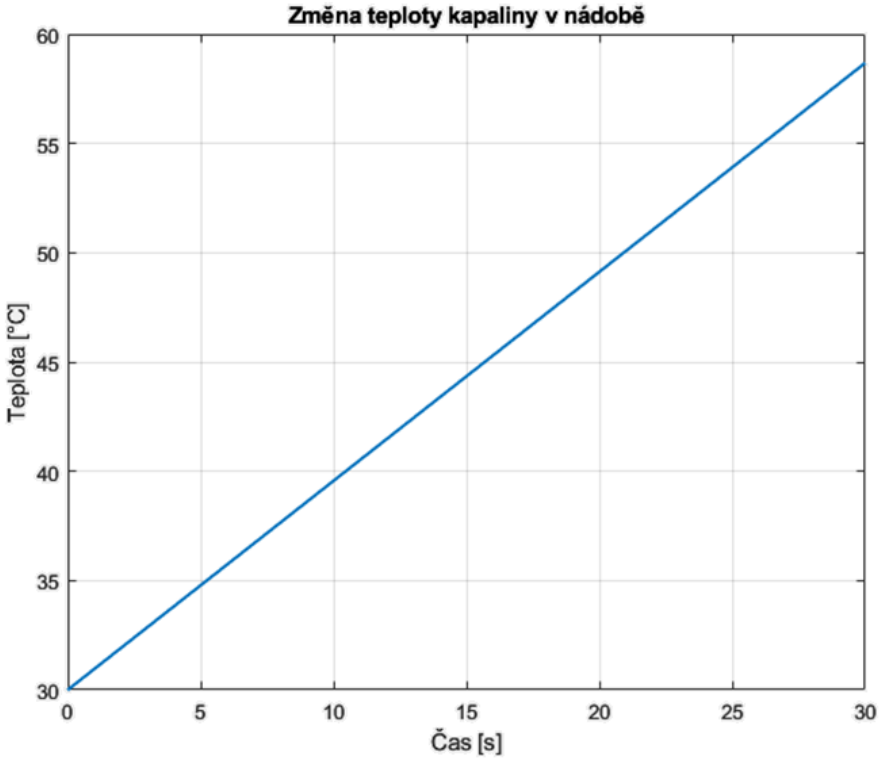
```
function dT_dt = teplota_v_case(t, T, P, k, S, T_ok, m, c)
%Tepelný obsah
A = m * c * T;

%Vstupní tepelný tok
Q_in = P;

%Výstupní tepelný tok
Q_out = k * S * (T - T_ok);

%Diferenciální rovnice pro změnu teploty
dT_dt = (Q_in - Q_out) / (m * c);
end
```

Volací funkce zadaná do příkazové řádky:

```
%Parametry
T0 = 30;
P = 2000;
k = 0.3;
S = 0.25;
T_ok = 20;
m = 0.5;
c = 4180;

%Časový interval
interval = [0,30];

%Volání funkce pro řešení diferenciální rovnice
[t, T] = ode45(@(t, T) teplota_v_case(t, T, P, k, S, T_ok, m, c), interval, T0);

%Vykreslení změny teploty v čase
plot(t, T, 'LineWidth', 1.5);
xlabel('Čas [s]');
ylabel('Teplota [°C]');
title('Změna teploty kapaliny v nádobě');
grid;
```

Parametry v tomto kódu jsou nastaveny tak, aby co nejvíce odpovídaly těm reálným. Tento příklad je pouze jednoduchá ukázka simulace ohřevu kapaliny uvnitř nádoby do 100°C.

Obrázek 6- Dynamika teploty kapaliny v nádobě



Do nádoby bylo nalito 0,5l vody o teplotě 30°C a poté byla tato kapalina zahřívána po dobu 30s ve varné konvici po 30s byla kapalina odlita a byla změřena její teplota. Praktický výsledek je totožný s výsledkem teoretickým, který byl zpracován v Matlabu.

Obrázek 7- Praktické ověření

ZÁVĚR

Práce se zabývala analýzou a řešením diferenciálních rovnic prostředí v Matlab. Cílem bylo seznámit s různými typy diferenciálních rovnic a numerickými metodami pro jejich řešení. V rámci této práce byly nejprve představeny základní pojmy spojené s diferenciálními rovnicemi a programovacím prostředí Matlab a následně byly rozebrány různé typy diferenciálních rovnic. Důležitou součástí práce bylo seznámení s numerickými metodami pro řešení diferenciálních rovnic, včetně Eulerovy metody, Runge-Kuttovy metody a vícekrokové metody prediktor-korektor. Pro každou metodu byly prezentovány principy a kódy pro výpočet, a bylo demonstrováno jejich použití na konkrétním příkladu. Závěrem části tři pro jednokrokové numerické metody bylo grafické ověření, že Runge-Kuttova metoda 4. řádu je nejpřesnější jednokrokovou metodou pro řešení diferenciálních rovnic. Volba mezi jednokrokovou a vícekrokovou metodou závisí na konkrétním problému, který je řešen. Pokud je důležitější přesnost než jednoduchost implementace, pak vícekrokovou metody mohou být výhodnější. Praktické aplikace diferenciálních rovnic byly demonstrovány na příkladech z elektrotechniky a termodynamiky. Tato práce poskytla základní přehled o analýze a řešení diferenciálních rovnic.

POUŽITÁ LITERATURA

[1] CVEJN, Jan. *Automatizace 1*. Online. Pardubice: Univerzita Pardubice, FEI, 2023. Dostupné pouze pro studenty předmětu Automatizace 1 na FEI UPa.

[2] FAJMON, Břetislav a RŮŽIČKOVÁ, Irena. *Matematika 3*. Online. Brno: FEKT VUT, 2005. Dostupné také z: http://matika.umat.feec.vutbr.cz/inovace/materialy/skripta/bma3.pdf.

[3] KUBEN, Jaromír. *Obyčejné diferenciální rovnice*. Online. Brno: Vydavatelství Univerzity Palackého, 1995. Dostupné také z: https://www1.karlin.mff.cuni.cz/~halas/MA/MA3/ODR.pdf.

[4] KUČERA, Radek. *Numerické metody*. Online. Ostrava: VŠB – Technická Univerzita, 2006. ISBN 80-248-1198-7. Dostupné také z: https://homel.vsb.cz/~skn002/dl/NM.pdf.

[5] ŘEZNÍČKOVÁ, Jana. *Diferenciální rovnice*. Online. Zlín: Ústav matematiky FAI UTB, 2015. Dostupné také z: http://msc.utb.cz/wp-content/uploads/2016/11/A3DRI_stud_text.pdf.

[6] THE MATHWORKS. *Matlab*. Online. Natick, MA: The Mathworks, 1996. Dostupné také z: https://www-eio.upc.edu/lceio/manuals/matlab/TECHDOC/PDFDOCS/GETSTART.PDF.

[7] VICHER, Miroslav. *Numerická matematika*. Online. Ústí nad Labem: Univerzita J.E.Purkyně, 2003. Dostupné také z: https://physics.ujep.cz/~mlisal/nm_2-chomutov/vicher_nm1.pdf.

[8] VONDRÁK, Vít a POSPÍŠIL, Lukáš. *Numerické metody I*. Online. Ostrava: Vysoká škola báňská – Technická Univerzita, 2011. Dostupné také z: https://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/numericke_metody.pdf.

[9] ZAHRÁDKA, Jaromír. *Diferenciální rovnice*. Online. Pardubice: Univerzita Pardubice, FEI, 2022. Dostupné pouze pro studenty předmětu Matematika II na FEI UPa.

[10] ZAHRÁDKA, Jaromír. *Diskrétní matematika pro SII: diskretizační metody numerické matematiky*. Pardubice: Univerzita Pardubice, 2014. ISBN 978-80-7395-841-1.

[11] ZAHRÁDKA, Jaromír. *Matematický seminář – MATLAB*. Pardubice: Univerzita Pardubice, 2013. ISBN 978-80-7395-691-2.