

# **Numerické metody řešení obyčejných diferenciálních rovnic s programem Mathematica**

Numerical methods for solving ordinary differential equations  
in Mathematica

Zdenka Pochopová

---

Bakalářská práce  
2011



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Zdenka POCHOPOVÁ**

Osobní číslo: **A08254**

Studijní program: **B 3902 Inženýrská informatika**

Studijní obor: **Informační a řídicí technologie**

Téma práce: **Numerické metody řešení obyčejných diferenciálních rovnic s programem Mathematica**

Zásady pro vypracování:

1. Definujte pojmy obyčejná diferenciální rovnice (ODR), počáteční úloha, numerické řešení ODR.
2. Odvoďte Eulerovu metodu a její modifikace pro přibližné řešení ODR, vysvětlete pojem Taylorova řada a možnosti jejího využití při řešení ODR, popište metodu Runge-Kutta.
3. Popište způsob odhadu chyby numerického řešení.
4. Popište příkaz NDSolve ze software Mathematica, jeho parametry a metody, které využívá při řešení ODR.
5. Naprogramujte v software Mathematica výše zmíněné metody řešení. Na jednoduchých příkladech pak ukažte rozdíly v přesnosti výpočtu, náročnosti výpočtu apod. Ukázkové příklady řešte i pomocí příkazu NDSolve.
6. Řešte konkrétní praktickou úlohu (návrh matematického modelu, řešení v software Mathematica, zhodnocení výsledků).

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. VITÁSEK, E.; PRÁGER, M.; BABUŠKA, I. Numerické řešení diferenciálních rovnic. Praha : Státní nakladatelství technické literatury, 1964. 238 s.
2. VITÁSEK, E. Základy teorie numerických metod pro řešení diferenciálních rovnic. Praha : Academia Praha, 1994. 409 s.
3. BRAUN, M. Differential Equations And Their Applications. New York : Springer, 1993. 718 s.
4. KUČERA, R. Numerické metody. Ostrava, 2008. 143 s. Vytvořeno v rámci projektu Operačního programu Rozvoje lidských zdrojů. VYSOKÁ ŠKOLA BÁŇSKÁ: TECHNICKÁ UNIVERZITA OSTRAVA.
5. KUFNER A. Obvyčejné diferenciální rovnice. Plzeň, 1993. 159 s. Skripta. Západočeská univerzita.

Vedoucí bakalářské práce: **Mgr. Vladimír Polášek, Ph.D.**  
Ústav matematiky

Datum zadání bakalářské práce: **25. února 2011**

Termín odevzdání bakalářské práce: **7. června 2011**

Ve Zlíně dne 25. února 2011

  
prof. Ing. Vladimír Vašek, CSc.  
*děkan*



  
prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## ABSTRAKT

Tato bakalářská práce pojednává o numerických metodách řešení pro obyčejné diferenciální rovnice. V teoretické části definuji pojmy diferenciálních rovnic, jejich způsoby řešení, důležitost ve vědě. Dále představuji software *Mathematica*, příkazy pro numerická řešení, jejich parametry. V praktické části představuji algoritmy pro jednotlivé metody řešení, které ukazuji na ukázkových příkladech. Dále řeším konkrétní úlohu, která vede na obyčejnou diferenciální rovnici.

Klíčová slova: obyčejné diferenciální rovnice, numerické metody, Mathematica

## ABSTRACT

This bachelor thesis deals with numerical methods of solving ordinary differential equations. In the theoretical part I explain what are differential equations, how to solve them and their importance in sciences. Furthermore I introduce the software *Mathematica*. I show commands for numerical solutions and their parameters. In the practical part there are demonstrated creating algorithms for solving particular methods and I show them in sample examples. Also I solve a practical problem, which leads to an ordinary differential equation.

Keywords: ordinary differential equations, numerical methods, Mathematica

Na tomto místě bych chtěla poděkovat vedoucímu bakalářské práce panu Mgr. Vladimíru Poláškoví, Ph.D. za trpělivost, kterou se mnou měl a za cenné rady a názory. V neposlední řadě chci poděkovat svým rodičům, kteří mě ve studiu na Univerzitě Tomáše Bati po celou dobu podporovali nejen finančně, ale především psychicky.

*„Každý úspěch, jehož dosáhneme, nám udělá nepřátele.“*

*Chceme-li být oblíbení, musíme být průměrní.“*

*Oscar Wilde*

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracovala samostatně a použitou literaturu jsem citovala. V případě publikace výsledků budu uvedena jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## OBSAH

<b>ÚVOD.....</b>	<b>9</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>10</b>
<b>1 DIFERENCIÁLNÍ ROVNICE (DR) .....</b>	<b>11</b>
1.1 ROZDĚLENÍ DIFERENCIÁLNÍCH ROVNIC .....	11
1.2 ŘÁD DIFERENCIÁLNÍCH ROVNIC .....	12
1.3 POČÁTEČNÍ ÚLOHA .....	12
1.4 HISTORIE DIFERENCIÁLNÍCH ROVNIC .....	13
1.5 APLIKACE DIFERENCIÁLNÍCH ROVNIC .....	13
<b>2 NUMERICKÉ ŘEŠENÍ ODR.....</b>	<b>14</b>
2.1 NUMERICKÉ DERIVOVÁNÍ .....	15
2.1.1 Odvození Eulerovy metody .....	17
2.1.2 Modifikace Eulerovy metody .....	17
2.2 METODA RUNGE-KUTTA .....	18
2.3 TAYLOROVA ŘADA .....	20
2.4 ODHAD CHYBY NUMERICKÉHO ŘEŠENÍ .....	21
<b>3 SOFTWARE MATHEMATICA .....</b>	<b>23</b>
3.1 VZNIK .....	23
3.2 KLÍČOVÁ MYŠLENKA .....	23
3.3 KLÍČOVÉ PRVKY .....	24
3.3.1 Notebookový dokumentační systém .....	24
3.3.2 Programovací jazyk .....	24
3.3.3 Interaktivní nápověda .....	25
3.3.4 Symbolické a numerické výpočty .....	26
3.3.5 Grafika .....	26
3.3.6 Nástrojové palety .....	26
3.4 VYUŽITÍ SOFTWARE MATHEMATICA .....	27
3.5 SOUČASNOST .....	28
3.6 PŘESNOST VÝSLEDKŮ .....	29
3.7 NUMERICKÁ ŘEŠENÍ OBYČEJNÝCH DIFERENCIÁLNÍCH ROVNIC V SOFTWARE MATHEMATICA .....	29
<b>II PRAKTICKÁ ČÁST .....</b>	<b>31</b>
<b>4 MATEMATICKÝ MODEL PRO ŘEŠENÍ OBYČEJNÝCH DIFERENCIÁLNÍCH ROVNIC .....</b>	<b>32</b>
4.1 ŘEŠENÍ UKÁZKOVÝCH PŘÍKLADŮ .....	32
4.1.1 NDSolve .....	33
4.1.2 Eulerova metoda .....	36
4.1.3 Metoda Runge-Kutta .....	38

4.1.4	Srovnání řešení.....	40
<b>5</b>	<b>APLIKACE ODR.....</b>	<b>43</b>
5.1	SLOVNÍ ZADÁNÍ.....	43
5.2	VÝPOČET.....	43
5.3	PŘESNÉ ŘEŠENÍ POMOCÍ SOFTWARE MATHEMATICA.....	46
5.4	PŘIBLIŽNÉ ŘEŠENÍ POMOCÍ SOFTWARE MATHEMATICA.....	48
5.4.1	Přibližné řešení pomocí NDSolve.....	48
5.4.2	Přibližné řešení Eulerovou metodou.....	50
5.4.3	Přibližné řešení Runge-Kuttovou metodou.....	52
5.5	ZHODNOCENÍ VÝSLEDKŮ.....	54
	<b>ZÁVĚR.....</b>	<b>55</b>
	<b>CONCLUSION.....</b>	<b>56</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>57</b>
	<b>SEZNAM POUŽITÝCH PŘÍKAZŮ SOFTWARE MATHEMATICA.....</b>	<b>58</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>60</b>
	<b>SEZNAM OBRÁZKŮ.....</b>	<b>61</b>
	<b>SEZNAM TABULEK.....</b>	<b>62</b>
	<b>SEZNAM GRAFŮ.....</b>	<b>63</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>64</b>



## ÚVOD

Numerické metody mají a budou mít stále větší význam v matematických aplikacích. Souvisí s tím rozvoj a stále se rozšiřující užívání počítačů. Tímto se neustále zdůrazňuje význam numerických metod a současně se staví do popředí studium některých problémů, jež se dříve plně neprojevily. V dnešní době jsme schopni provádět v krátkém čase statisíce a milióny početních operací. Rychlost stroje a relativně malé finanční náklady na výpočet vedou k tomu, že se řeší stále složitější problémy. Díky výpočetní technice tedy výpočty, které by člověku trvaly několik hodin, ba dokonce i dní, počítač zvládne vyřešit během několika vteřin. V této práci se zaměřuji na numerické metody, pomocí nichž lze řešit obyčejné diferenciální rovnice. Diferenciální rovnice jsou nezbytným pomocným prostředkem každého inženýra i přírodovědce, protože vedou k řešení mnohých přírodních, fyzikálních či technických, ale i společenských problémů.

Teoretická část vysvětluje, co jsou to diferenciální rovnice, kde se s nimi můžeme setkat, jaké místo zastávají ve vědě. Dále je třeba diferenciální rovnice nějakým způsobem řešit. Tato práce se zabývá obyčejnými diferenciálními rovnicemi a jejich numerickým řešením. Tyto řešení zde také popisuji. V teoretické části představuji i program *Mathematica*, který používám v praktické části.

V praktické části vytvářím algoritmy pro jednotlivé metody pro řešení obyčejných diferenciálních rovnic. V programu *Wolfram Mathematica 8* řeším ukázkové příklady ale i konkrétní praktickou úlohu.

## **I. TEORETICKÁ ČÁST**

## 1 DIFERENCIÁLNÍ ROVNICE (DR)

Diferenciální rovnice je matematická rovnice, v níž neznámou je funkce a v níž se vyskytují derivace této funkce.

Pro názorný příklad mějme spojitou funkci  $f(x)$ . Hledáme funkci  $y = y(x)$ , pro kterou na intervalu  $I$  platí:

$$y'(x) = f(x). \quad (1.1)$$

Řešením bude

$$y(x) = \int f(x)dx + C, \quad (1.2)$$

kde  $\int f(x)dx$  je libovolná primitivní funkce k funkci  $f(x)$  na  $I$  a  $C$  je integrační konstanta.

[5]

### 1.1 Rozdělení diferenciálních rovnic

Základní rozdělení diferenciálních rovnic vychází z typu obsažených derivací v rovnici.

- **Obyčejné diferenciální rovnice (ODR)** jsou rovnice, ve kterých neznámou je funkce jedné proměnné a ve kterých se vyskytuje aspoň jedna derivace této funkce.

V obecném tvaru pro neznámou funkci  $y = y(x)$  ji zapisujeme

$$\text{v implicitním tvaru } F(x, y, y', y'', \dots, y^{(n)}) = 0, \quad (1.3)$$

$$\text{v explicitním tvaru } y^{(n)} = f(x, y, y', y'', \dots, y^{(n-1)}). \quad (1.4)$$

$F$  je funkce  $n+2$  proměnných.

- **Parciální diferenciální rovnice (PDR)** jsou rovnice ve kterých neznámou je funkce více proměnných a ve kterých se vyskytují parciální derivace této funkce. [5]

V obecném tvaru pro neznámou funkci  $z(x_1, x_2, \dots, x_n)$   $n$  proměnných ji zapisujeme:

$$F(x_1, x_2, \dots, x_n, z, \frac{\partial z}{\partial x_1}, \dots, \frac{\partial z}{\partial x_n}, \frac{\partial^2 z}{\partial x_1^2}, \frac{\partial^2 z}{\partial x_1 x_2}, \dots, \frac{\partial^2 z}{\partial x_1 x_n}, \frac{\partial^2 z}{\partial x_2^2}, \dots, \frac{\partial^k z}{\partial x_n^k}, \dots) = 0 \quad (1.5)$$

## 1.2 Řád diferenciálních rovnic

Nejvyšší řád derivace hledané funkce, která se v rovnici vyskytuje, označujeme jako řád diferenciální rovnice. Podle řádu se diferenciální rovnice dále dělí na *diferenciální rovnice prvního řádu* a *diferenciální rovnice vyšších řádů*.

**Obecné řešení** je takové řešení diferenciální rovnice, které obsahuje libovolnou integrační konstantu. Počet konstant se rovná řádu DR.

**Partikulární řešení** diferenciální rovnice, je takové řešení, které získáme přiřazením určité číselné hodnoty každé integrační konstantě obecného řešení. V případě jednoduchých diferenciálních rovnic můžeme partikulární řešení vypočítat analyticky. Avšak ve velkém množství případů je analytické řešení příliš obtížné. V tom případě se diferenciální rovnice řeší numericky. [7]

## 1.3 Počáteční úloha

V praktických úlohách je často potřebné nalézt takové řešení DR, které splňuje předem dané podmínky. Vztahují-li se podmínky na jeden bod, nazývají se *počáteční podmínky*, vztahují-li se k více bodům, nazývají se *okrajové podmínky*. [6]

Najít řešení rovnice, která splňuje zadanou počáteční podmínku, se nazývá *počáteční úloha*. Řešením je funkce, která splňuje zadanou podmínku, např.  $y(x_0) = y_0$  a je řešením rovnice na intervalu obsahujícím bod  $x_0$ .

## 1.4 Historie diferenciálních rovnic

Historie diferenciálních rovnic sahá do 2. poloviny 17. století a je spojena se jmény G.W. Leibnize a I. Newtona, kteří již uměli řešit rovnice se separovanými proměnnými. Dalšími významnými matematiky byli bratři Bernoulliiové, L. Euler nebo J. L. Lagrange. [6]

## 1.5 Aplikace diferenciálních rovnic

Diferenciálními rovnicemi lze formulovat spoustu vědních problémů. Objevují se tedy snad ve všech vědních oborech. Veliké zastoupení mají v matematice, fyzice, automatizaci, optimalizaci, chemii, ekologii a dalších. Často se můžeme setkat s diferenciálními rovnicemi, které popisují závislost veličin na čase. Ve fyzice jsou to například pohybové rovnice. Bez diferenciálních rovnic by nebylo možné provádět výpočty související s pružností a pevností materiálu, s řízením jaderných reakcí, s lety do vesmíru apod. [1]

## 2 NUMERICKÉ ŘEŠENÍ ODR

Většinu v praxi se vyskytujících diferenciálních rovnic nelze vyřešit *přímo*. Numerické řešení se používá v případech, kdy by bylo nalezení přesného (analytického) řešení náročné nebo nemožné. Numerickým řešením jsme schopni získat přibližné řešení obyčejných diferenciálních rovnic.

Při hledání řešení diferenciálních rovnic prvního řádu, ve kterých je derivace přímo vyjádřena, hledáme reálnou funkci, která splňuje následující rovnici

$$\frac{dy(x)}{dx} = f(x, y). \quad (2.1)$$

Funkcí, které splňují tuto rovnici, existuje nekonečně mnoho. Jedno konkrétní řešení určíme *počáteční podmínkou*, která je nezbytnou součástí zadání úlohy. Počáteční podmínka má následující tvar

$$y(x_0) = y_0, \quad (2.2)$$

kde  $x_0$  a  $y_0$  jsou zadané hodnoty. [2] [11]

Základním principem numerických metod je diskretizace proměnných, kdy přibližné řešení se nekonstruuje jako funkce spojitá. Postupně se generuje posloupnost uzlů  $x_0, x_1, x_2, \dots$  a pro ty se stanoví čísla  $y_0, y_1, y_2, \dots$ , která aproximují hodnoty přesného řešení  $y(x_0), y(x_1), y(x_2), \dots$

Vyjdeme z bodu  $x_0$ , ve kterém máme zadánu počáteční podmínku. Poté se snažíme najít řešení v dalších bodech  $x_n$ . Numerické metody můžeme rozdělit podle počtu použitých předchozích bodů k výpočtu nové hodnoty.

- Jednokrokové metody používají k výpočtu nové hodnoty pouze jednu předchozí hodnotu.
- Mnohokrokové metody používají k výpočtu nové hodnoty  $k$  předchozích bodů.

## 2.1 Numerické derivování

Numerická derivace je numerická metoda odhadu derivace funkce na základě hodnoty této funkce v konečně mnoha bodech. Numerická derivace se používá především v případech, kdy nejsme schopni derivaci určit analyticky. [2] [8]

Metody pro numerické derivování vycházejí z definice derivace, např.:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \quad (2.3)$$

Derivace znamená směrnici tečny ke grafu funkce v bodě.

K funkci  $f$  sestavíme interpolační polynom  $p_n$  a ten pak derivujeme místo  $p$ . Interpolační polynom v Newtonovém tvaru:

$$p_n(x) = f_0 + f[x_1, x_0](x - x_0) + \dots + f[x_n, \dots, x_0](x - x_0) \dots (x - x_{n-1}). \quad (2.4)$$

- 1) Pro uzly  $x_0 = x$  a  $x_1 = x + h$  sestavíme lineární interpolační polynom a vyjádříme jeho první derivaci:

$$p_1(t) = f(x) + f[x+h, x](t-x) \Rightarrow p_1'(t) = f[x+h, x] \quad (2.5)$$

Z definice poměrných diferencí dostaneme

$$p_1'(x) = \frac{f(x+h) - f(x)}{h} \approx f'(x). \quad (2.6)$$

2) Pro uzly  $x_0 = x-h$ ,  $x_1 = x$  a  $x_2 = x+h$  sestavíme kvadratický interpolační polynom:

$$p_2(t) = f(x-h) + f[x, x-h](t-x+h) + f[x+h, x, x-h](t-x+h)(t-x) \quad (2.7)$$

První a druhá derivace mají tvar:

$$\begin{aligned} p_2'(t) &= f[x, x-h] + f[x+h, x, x-h](2t-2x+h), \\ p_2''(t) &= 2f[x+h, x, x-h] \end{aligned} \quad (2.8)$$

Dosazením  $t = x$  a úpravou dostáváme:

$$p_2'(x) = \frac{f(x) - f(x-h)}{h} + \frac{f(x+h) - 2f(x) + f(x-h)}{2h^2} h = \frac{f(x+h) - f(x-h)}{2h} \approx f'(x) \quad (2.9)$$

$$p_2''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \approx f''(x) \quad (2.10)$$



### 2.1.1 Odvození Eulerovy metody

Nejjednodušší metodou pro numerické řešení diferenciálních rovnic je Eulerova metoda. Pokud v diferenciální rovnici (2.1) nahradíme derivaci přibližným vzorcem, dostaneme

$$\frac{y_{n+1} - y_n}{h} = f(x_n, y_n), \quad (2.11)$$

kde  $h = x_{n+1} - x_n$  a označuje vzdálenost dvou sousedních bodů. Hovoříme o něm jako o tzv. *kroku řešení*. Když vyjádříme  $y_{n+1}$  dostaneme Eulerovu metodu

$$y_{n+1} = y_n + h \cdot f(x_n, y_n). \quad (2.12)$$

Tento vzorec nám umožňuje při znalosti řešení v bodě  $x_n$  vypočítat řešení v bodě  $x_{n+1}$ . Jedná se o *jednokrokovou metodu*. [11]

### 2.1.2 Modifikace Eulerovy metody

Eulerova metoda patří mezi jednoduché metody, avšak je velice nepřesná. Hlavní nepřesnost je způsobena tím, že při kroku z  $x_n$  do  $x_{n+1}$  používáme na celém intervalu konstantní hodnotu derivace  $f(x_n, y_n)$ . Tato hodnota se správně v průběhu kroku mění. Tuto chybu se snaží odstranit modifikace Eulerovy metody. [3] [11]

První modifikace se snaží vystihnout změnu derivace tím, že namísto derivace na začátku intervalu použije derivaci uprostřed intervalu. Doprostřed intervalu se dostaneme obyčejnou Eulerovou metodou s poloviční délkou kroku. Výsledný vzorec je:

$$y_{n+1} = y_n + h \cdot f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot f(x_n, y_n)\right) \quad (2.13)$$

Tento vztah lze zapsat ve tvaru:

$$\begin{aligned}k_1 &= f(x_n, y_n) \\k_2 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_1\right) \\y_{n+1} &= y_n + h \cdot k_2\end{aligned}\tag{2.14}$$

Druhá modifikace používá průměr z derivace na začátku a na konci intervalu:

$$y_{n+1} = y_n + \frac{1}{2}h \cdot [f(x_n, y_n) + f(x_n + h, y_n + h \cdot f(x_n, y_n))]\tag{2.15}$$

Tento vztah lze zapsat ve tvaru:

$$\begin{aligned}k_1 &= f(x_n, y_n) \\k_2 &= f(x_n + h, y_n + h \cdot k_1) \\y_{n+1} &= y_n + h \cdot \frac{k_1 + k_2}{2}\end{aligned}\tag{2.16}$$

## 2.2 Metoda Runge-Kutta

Dalšími modifikacemi lze Eulerovu metodu dále vylepšovat. Získáme tzv. Runge-Kuttovy metody (RK). Při těchto metodách získáme několik odhadů derivace  $k_i$  v  $s$  různých bodech. Pro celý krok potom použijeme vážený průměr těchto derivací s váhami  $w_i$

$$y_{n+1} = y_n + h \cdot (w_1 k_1 + \dots + w_s k_s).\tag{2.17}$$

Odhady derivací se vypočítají následujícími vztahy:

$$\begin{aligned}
 k_1 &= f(x, y) \\
 k_2 &= f(x + \alpha_2 h, y + h\beta_{21}k_1) \\
 k_3 &= f(x + \alpha_3 h, y + h\beta_{31}k_1 + h\beta_{32}k_2) \\
 k_i &= f(x + \alpha_i h, y + h\sum_{j=1}^{i-1} \beta_{ij}k_j)
 \end{aligned} \tag{2.18}$$

Pokud použijeme jen jeden bod (tj.  $s = 1$ ), odvodíme metodu prvního řádu – Eulerovu metodu. Pokud použijeme dva body, můžeme dostat modifikované Eulerovy metody.

První modifikaci odpovídá volba:

$$w_1 = 0 \quad w_2 = 1 \quad \alpha_2 = \beta_{21} = \frac{1}{2} \tag{2.19}$$

Druhé modifikace odpovídá volba:

$$w_1 = \frac{1}{2} \quad w_2 = \frac{1}{2} \quad \alpha_2 = \beta_{21} = 1 \tag{2.20}$$

S rostoucím počtem bodů dostáváme metody vyšších řádů. Například při čtyřech odhadech derivace můžeme odvodit následující metodu čtvrtého řádu (RK4):

$$\begin{aligned}
 k_1 &= hf(x_n, y_n) \\
 k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{h}{2}k_1) \\
 k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{h}{2}k_2) \\
 k_4 &= hf(x_n + h, y_n + hk_3) \\
 y_{n+1} &= y_n + \frac{1}{6}h \cdot (k_1 + 2k_2 + 2k_3 + k_4)
 \end{aligned} \tag{2.21}$$

U dosud uvedených RK metod se řád metody vždy rovnal použitému počtu odhadů derivace. Tak to ovšem vždy není, například při pěti odhadech derivace se dá sestavit pouze metoda čtvrtého řádu. I z toho důvodu patří uvedená metoda RK4 k velice oblíbeným. [11]

Mezi výhody RK metod patří především přesnější výsledek. Ačkoliv v jednom kroku (při skoku o jedno  $h$ ) musíme funkci  $f$  vyčíslit vícekrát než u Eulerovy metody, dostaneme metodu vyššího řádu, který nám zvýšené nároky na výpočet funkce  $f$  kompenzuje.

## 2.3 Taylorova řada

Taylorova řada je nekonečná geometrická řada. Za určitých předpokladů o funkci  $f(x)$  v okolí bodu  $a$  lze tuto funkci rozvinout jako mocninnou řadu. Toto vyjádření funkce prostřednictvím Taylorovy řady se označuje jako Taylorův rozvoj.

Obecně lze Taylorovu řadu zapsat:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \quad (2.22)$$

Pro přibližné vyjádření hodnot funkce není nutné vyjadřovat všechny členy Taylorovy řady, můžeme zanedbat členy s vyššími derivacemi. Získáme tím tzv. **Taylorův polynom**.

Taylorův polynom aproximuje hodnoty funkce, která má v daném bodě derivaci, pomocí polynomu, jehož koeficienty závisí na derivacích funkce v tomto bodě. Taylorův polynom je speciálním příkladem Taylorovy řady, kdy od určitého  $n$  jsou všechny vyšší derivace nulové. [7]

Využití Taylorovy řady při řešení obyčejných diferenciálních rovnic vidíme např. při metodě RK. Koeficienty jsou vypočteny tak, aby metoda řádu  $p$  odpovídala Taylorovu polynomu funkce  $y(t)$  stejného řádu.

## 2.4 Odhad chyby numerického řešení

Při výpočtu přibližných hodnot derivací mohou výsledek podstatně ovlivnit zaokrouhlovací chyby. Jmenovatele vzorců totiž obsahují parametr  $h$ , který musí být malý, abychom dostali dostatečně přesnou aproximaci derivace. Zároveň ovšem malá hodnota jmenovatele zlomku zvětšuje zaokrouhlovací chyby v čitateli. Největší dosažitelná přesnost proto musí být jistým kompromisem mezi chybou aproximace a chybou zaokrouhlení.

Součtem odhadu chyby aproximace  $e(h)$  a chyby zaokrouhlení  $z(h)$  získáme horní odhad celkové chyby  $E_C(h)$ :

$$E_C(h) = e(h) + z(h) \quad (2.23)$$

Dále budeme předpokládat, že při výpočtu  $f(x)$  dostaneme vlivem zaokrouhlení porušenou hodnotu  $f^*(x)$ , přičemž velikost poruchy je nejvýše  $\kappa$ :

$$|f^*(x) - f(x)| \leq \kappa \quad (2.24)$$

Pro vzorec (2.6) dostáváme následující odhad:

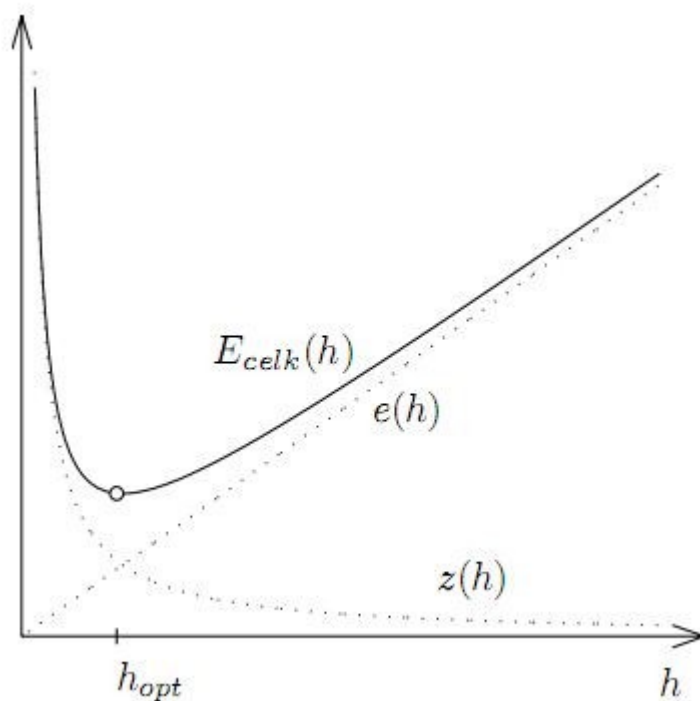
$$\begin{aligned} \left| \frac{f(x+h) - f(x)}{h} - \frac{f^*(x+h) - f^*(x)}{h} \right| &\leq \frac{1}{h} (|f^*(x) - f(x)| + |f^*(x+h) - f(x+h)|) \leq \\ &\leq \frac{2\kappa}{h} = z(h) \end{aligned} \quad (2.25)$$

Odhad celkové chyby má tvar  $E_C(h) = Ch + \frac{2\pi}{h}$ . Funkce  $E_C(h)$  má jediné minimum v bodě

$$h_{opt} = \sqrt{\frac{2\pi}{c}}. \quad (2.26)$$

Odpovídající hodnota  $E_C(h_{opt})$  představuje nejmenší horní odhad celkové chyby. Při výpočtu derivace podle rovnice (2.6) tedy dojde pro  $h$  menší než  $h_{opt}$  ke ztrátě přesnosti.

[4]



Obrázek 1 - Odhad celkové chyby  $E_C(h)$

### 3 SOFTWARE MATHEMATICA

Software *Mathematica* je jedním z prostředí, jež integruje nástroje pro numerickou a symbolickou matematiku, grafický a dokumentační systém a zajišťuje pokročilé propojení s dalšími aplikacemi. [9]



Obrázek 2 - Logo software *Mathematica 8*

#### 3.1 Vznik

V roce 1987 byla založena Stephenem Wolframem společnost Wolfram Research, Inc., ve které byl a je tento produkt vytvářen. První vydání software *Mathematica* bylo v roce 1988. Znamenalo zásadní význam pro způsob využívání počítačů v řadě technických i jiných odvětví. Uvádí se, že právě vydáním software *Mathematica* začal nový věk tzv. technical computing. Od 60. let minulého století existovali samostatné sady. Avšak byly vhodné vždy jen pro určité specifické úlohy. Jednalo se například o sady pro úlohy numerické, algebraické, grafické či jiné. Software *Mathematica* představuje zvrát v tom, že integruje všechny potřebné sady do jednoho produktu. [9]

#### 3.2 Klíčová myšlenka

Důležité pro software *Mathematica* bylo objevení nového symbolického jazyka. Symbolické programování umožňuje reprezentovat data, funkce, grafy, programy a dokonce i celý dokument jednotným způsobem jako symbolický výraz. Ukázkovým

příkladem může být  $f[x]$ . Tento výraz může být matematickou funkcí, grafikou, zvukem, programem i celým *Mathematica* dokumentem. Může fungovat jako vstup i výstup jiné funkce a umožňovat tak stručné a jednoduché kódování. [9]

Další technickou inovací v softwaru *Mathematica* je využití nezávislého interakčního dokumentu, známého jako notebook. Notebookové rozhraní kombinuje textový procesor se zřetelně definovanou představou „buněk“, které jsou seřazeny svisle v rozbalovacím okně jako odstavce textu. Buňky jsou důležité, oddělují vizuálně a funkčně text na vstupy, výstupy, grafiku, nadpisy atd. Jsou dostatečně přizpůsobitelné, aby podporovaly jakýkoliv typ a velikost výrazu, umožňují jednoduchou úpravu a vložení, a jsou snadno rozšiřitelné pro velké výpočty a dokumenty.

### 3.3 Klíčové prvky

#### 3.3.1 Notebookový dokumentační systém

*Mathematica* notebook zajišťuje kompletní technický dokumentační systém, zahrnující sazbu matematických výrazů, formátovaného textu, zvuky, grafiky, animací a hypertextových odkazů. Notebook může být převeden do jiného formátu, např. HTML, TeX, je možno jej odeslat emailem nebo umístit na webovou stránku, FTP server a to bez poškození kvality. [9]

#### 3.3.2 Programovací jazyk

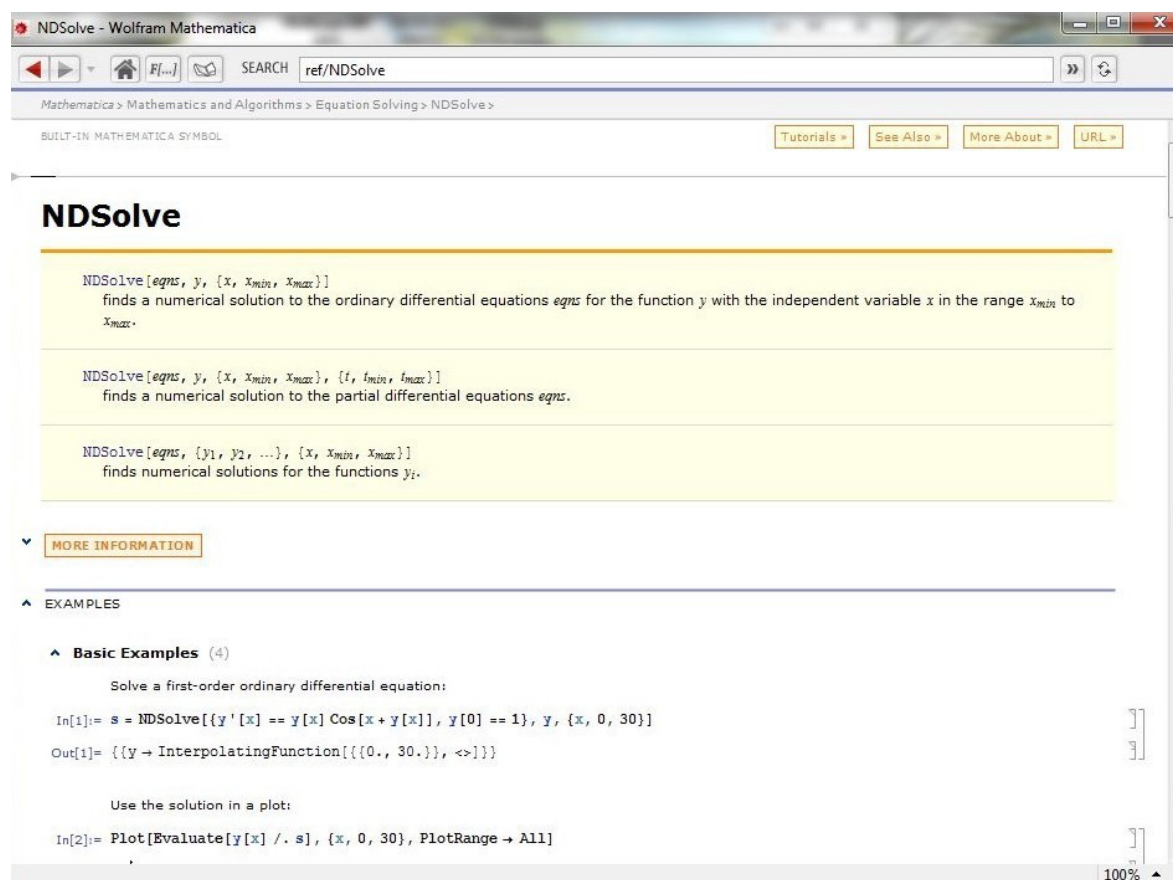
*Mathematica* poskytuje silné programové vývojové prostředí. Programový kód *Mathematica* dokáže odrážet specifičnost problému, což jej dělá kratším a snadněji čitelným. Tato ojedinělá pružnost činní přechod z jiného programovacího jazyka jednodušším a efektivnějším. Ani pro dříve neprogramující není obtížné začít tvořit nové programy. Při použití *Mathematica* programovacího jazyka není potřeba předeklarovat typ proměnné, dimenzi matice, překládat program ani přímé řízení paměti. Běžné procedury jako seřazení, vyhledání, manipulace se soubory a daty jsou vestavěné. Programy napsané



v *Mathematica* jsou díky těmto vlastnostem co do velikosti pouze 5 až 10 procentní oproti programům napsaným v tradičních jazycích nebo numerických systémech. [9]

### 3.3.3 Interaktivní nápověda

Nápověda v software *Mathematica* zahrnuje kompletní dokumentaci pro všechny funkce v software *Mathematica* s úplným textem ***The Mathematica Book*** jako plně indexovaný notebook s pokročilými vyhledávacími schopnostmi včetně množství hypertextových odkazů. Nápověda také obsahuje interaktivní příklady, které demonstrují použití funkcí, jejich hlavní schopnosti a nejlepší způsob, jak je využít. Na rozdíl od jiných softwarů *Mathematica* umožňuje upravovat a vyhodnocovat příklady uvedené v nápovědě. Uživatelské změny nejsou trvalé. V případě vymazání části textu či příkladu je pouze potřeba znovu zavést tuto stranu a tím se obnoví originální informace. [9]



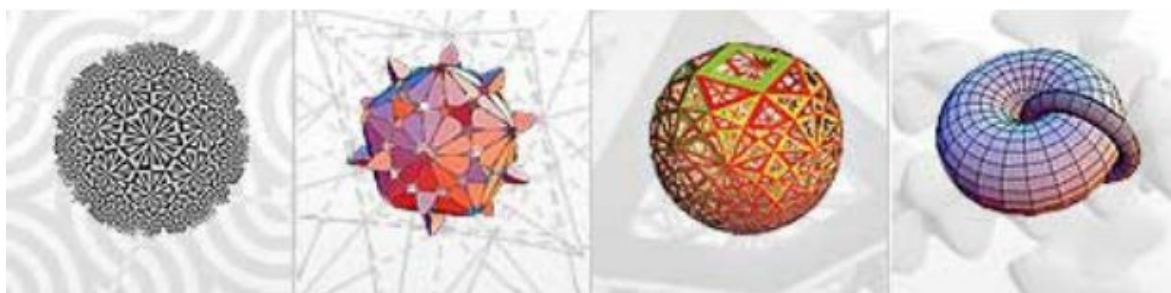
Obrázek 3 - Ukázka interaktivní nápovědy

### 3.3.4 Symbolické a numerické výpočty

Každá funkce, která je v prostředí *Mathematica* zavedena, může pracovat jak s numerickými tak se symbolickými vstupy. Dokáže vyhodnocovat funkce na jakoukoliv přesnost.

### 3.3.5 Grafika

*Mathematica* obsahuje velké množství vestavěných grafických vzorů pro vizualizaci výsledků, zahrnuje 2D, 3D, vrstevnicové grafy, grafy hustoty a navíc také specializované ekonomické a statistické grafy.

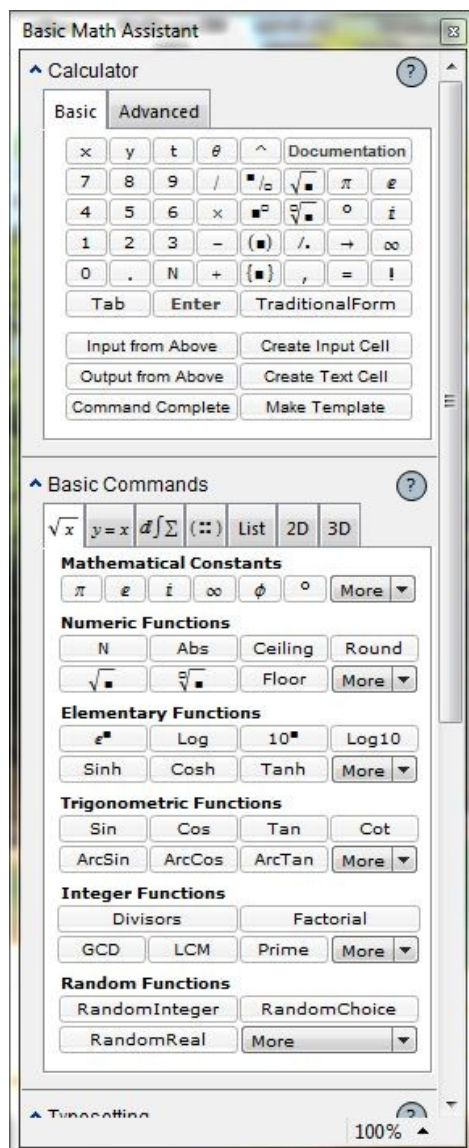


Obrázek 4 - Ukázka možností *Mathematica*

*Mathematica* také umožňuje provádět animace a vytvářet zvukové záznamy velmi jednoduchým způsobem. *Mathematica* má tzv. grafický jazyk, kterým lze dle požadovaných specifikací přizpůsobit vestavěné grafické vzory či tvořit vlastní vzory. [9]

### 3.3.6 Nástrojové palety

Software *Mathematica* obsahuje kolekci nástrojových palet připravených k okamžitému použití. Palety jsou vytvořené pro základní typy matematických operací, typů písmen, symbolů z funkcí vestavěných. Je možnost vytvořit i vlastní palety např. nejpoužívanějších symbolů.



Obrázek 5 - Nástrojová paleta

### 3.4 Využití software Mathematica

Software *Mathematica* nabízí široké uplatnění pro plnění úkolů, jakými jsou:

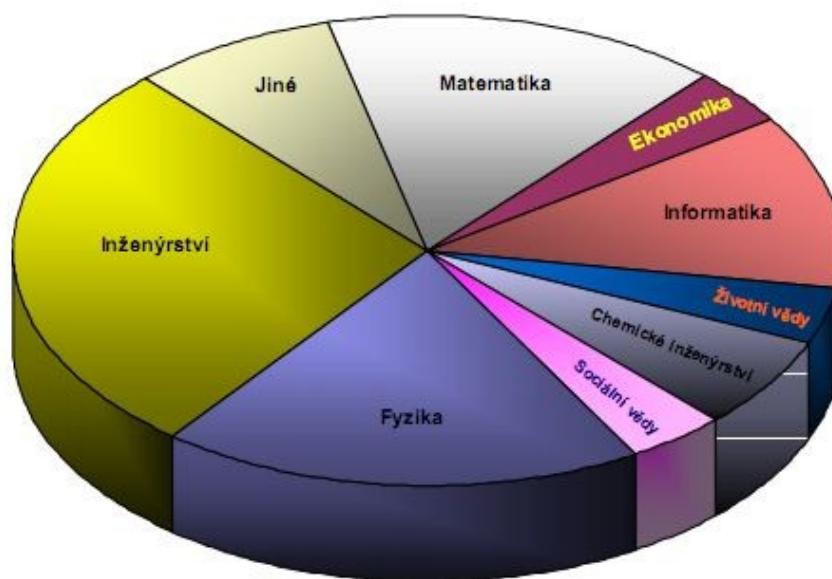
- Zpracování komplexních symbolických výrazů, které obsahují velké množství členů
- Zavádění, analýza a vizualizace dat
- Řešení rovnic, diferenciálních rovnic a minimalizace problémů numericky nebo symbolicky

- Výzkum chování numerických modelů a simulací, od jednoduchých regulačních systémů až po srážku galaxií, finančních nebo biologických systémů, chemických reakcí či vlivů na prostředí
- Výroba profesionálně kvalitních, interaktivních technických dokumentů pro elektronickou distribuci či tisk
- Ilustrace matematických nebo vědeckých konceptů pro studenty
- Sazba technických informací
- Provádění odborných prezentací a seminářů
- Usnadnění rychlého rozvoje engineeringových společností a finančních institucí [9]

### 3.5 Současnost

V současnosti je počet uživatelů software *Mathematica* přes jeden milion. Z počátku se software *Mathematica* nejvíce využíval ve fyzice, engineeringu a matematice. Během let se začal používat v překvapivě širokém okruhu oborů. V dnešní době se uplatňuje ve vědách přírodních, biologických, sociálních i jiných. Software *Mathematica* hraje důležitou roli v řadě objevů a je základem pro tisíce vědeckých prací. Se software *Mathematica* pracují především odborníci z nejrůznějších oblastí, avšak používá se také pro studijní účely. S dostupností studentské verze se stává důležitým nástrojem pro technicky i netechnicky vzdělávající se studenty po celém světě.

Software *Mathematica* je stále vyvíjen a zdokonalován ve společnosti Wolfram Research, Inc. nejlepšími světovými odborníky pod vedením Stephena Wolframa. V dnešních dnech existuje téměř sto specializovaných komerčních sad pro tento software, několik periodik a více než dvě stě knih věnovaných tomuto systému. [9]

Obrázek 6 - Rozsah uživatelů *Mathematica*

### 3.6 Přesnost výsledků

Výsledky výpočtů v *Mathematice* jsou vždy přesnými, exaktními výsledky, i kdyby měl mít výsledek např. 31 řádů. Výsledky mohou mít i numericky aproximovaný tvar jako při použití kalkulátoru. To lze provést prostřednictvím funkce `//N`, respektive `N[]`.

*Mathematica* může rovněž provádět výpočty s uživatelsky definovanou přesností. K tomu lze opět využít funkce `N[]`, jejíž druhý argument udává přesnost. [9]

### 3.7 Numerická řešení obyčejných diferenciálních rovnic v softwaru

#### **Mathematica**

Funkce `NDSolve` umožňuje nalézt numerické řešení diferenciálních rovnic. V systému obyčejných diferenciálních rovnic může být libovolný počet neznámých funkcí  $y_i$ , ale všechny tyto funkce musí záviset na jediné nezávislé proměnné  $x$ , která je stejná pro každou funkci.

*NDSolve* [{*equation*, *initials condition*}, *function*, {*variable*, *interval*}] vrátí aproximaci řešení funkce rovnice *equation* s počáteční podmínkou *initials condition* na intervalu  $(a,b)$ .

*NDSolve* [*eqns*, *y*, {*x*, *x min*, *x max*}] řeší obyčejné diferenciální rovnice *eqns* pro funkci *y* s nezávisle proměnnou *x* v rozsahu  $x_{\min}$  až  $x_{\max}$ .

*NDSolve* představuje řešení pro funkce  $y_i$  jako *InterpolatingFunction* – přibližná funkce, jejíž hodnoty se zjišťují interpolací. Tyto objekty poskytují přiblížení k  $y_i$  v rozsahu hodnot  $x_{\min}$  až  $x_{\max}$  pro nezávislé proměnné *x*.

*NDSolve* hledá řešení iteračně. Začíná na určité hodnotě *x*, dále pokračuje v posloupnosti kroků, kde se snaží zahrnout celou řadu od  $x_{\min}$  po  $x_{\max}$ .

*MaxSteps* určuje maximální počet kroků, které *NDSolve* bude mít ve snaze najít řešení. Pro obyčejné diferenciální rovnice je výchozí nastavení *MaxSteps*  $\rightarrow 10000$ . Při *MaxSteps*  $\rightarrow \text{Infinity}$  neexistuje žádný horní limit na počet použitých kroků. Pokud chceme zadat velikost prvního kroku, poslouží příkaz *StartingStepSize*. Tím se vyhneme problému, že by první krok byl větší, což by se odrazilo v našem výsledku.

Diferenciální rovnice musí obsahovat počáteční nebo okrajové podmínky pro určení celkového řešení  $y_i$ . Počáteční a okrajové podmínky se obvykle uvádějí ve tvaru  $\mathbf{y}[\mathbf{x}_0] = \mathbf{c}_0$ ,  $\mathbf{y}'[\mathbf{x}_0] = \mathbf{D}\mathbf{C}_0$ , atd. Vestavěná funkce *D*, najde derivaci výrazu. Tyto podmínky určují hodnoty pro  $y_i[x]$  a pro případné derivace  $y'_i[x]$  v konkrétním bodě *x*. Počáteční nebo okrajové podmínky musí být zadány dostatečně k určení celkového řešení. [10] [12]

## **II. PRAKTICKÁ ČÁST**

## 4 MATEMATICKÝ MODEL PRO ŘEŠENÍ OBYČEJNÝCH DIFERENCIÁLNÍCH ROVNIC

Součástí této bakalářské práce bylo naprogramovat výše zmíněné metody řešení obyčejných diferenciálních rovnic. Pro realizaci byl použit software *Mathematica*.

### 4.1 Řešení ukázkových příkladů

Na dvou ukázkových příkladech ukážu, jak získat přibližné řešení počáteční úlohy ve tvaru  $f(x, y)$ ,  $y(x_0) = y_0$  použitím rovnoměrně rozložených bodů v intervalu  $\langle x_0, x_N \rangle$ .

$$\begin{aligned}
 x_N &= x_0 + nh \text{ pro } n = 0, \dots, N && \text{rovnoměrně rozložené body intervalu} \\
 h &= \frac{x_N - x_0}{N} && \text{délka kroku} \\
 y_n &&& \text{aproximovaná hodnota } y(x_n) \\
 f_n &= f(x_n, y_n)
 \end{aligned}
 \tag{4.1}$$

- **1. ukázkový příklad:**  $y' = \cos x$ ,  $x(0) = 0$  (4.2)

Jedná se o jednoduchý příklad, nenáročný na čas výpočtu.

$$\text{Přesné řešení: } y = \sin x \tag{4.3}$$

- **2. ukázkový příklad:**  $y' = x^2 + \frac{y}{x}$ ,  $x(1) = 2$  (4.4)

Jedná se o složitější příklad, náročnější na čas výpočtu.

$$\text{Přesné řešení: } y = \frac{x^3}{3} + y \cdot \ln|x| \tag{4.5}$$



#### 4.1.1 NDSolve

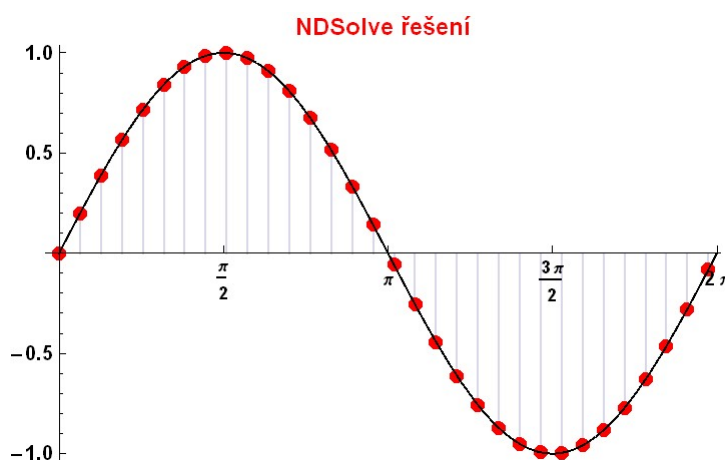
Není vždy možné přesně spočítat ODR. V tom případě je nevyhovující příkaz *DSolve*, který na výstup vypíše přesně to, co je vloženo jako vstup. ODR je možné vyřešit přibližně. K takovému řešení ODR se používá příkaz *NDSolve*. Příklad má jediné řešení, jde o počáteční problém. *NDSolve* má syntaxi:

***NDSolve [{equation, initials condition}, function, {variable, interval}]***

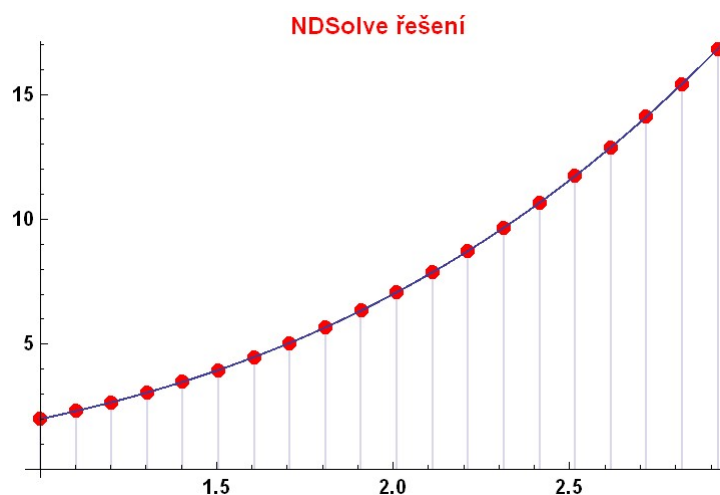
*NDSolve* vrátí aproximaci řešení rovnice *equation* s počáteční podmínkou *initials condition* na intervalu  $(a,b)$ . Výstup z funkce *NDSolve* je funkce definovaná za podmínek interpolační funkce. Proto musí být zkonvertována do tabulky hodnot, aby bylo možné vykreslit získané přibližné řešení. K tomu použijeme následující metodu:

***f[x\_] = y[x] /. solve1[[1]]***

Nyní můžeme získat funkční hodnotu v jakémkoliv bodě zadaného intervalu. Tabulku rovnoměrně rozložených hodnot v intervalu získáme za použití kroku  $h=Pi/9,8$  v prvním příkladě,  $h=0,105$  v příkladě druhém. Pro zobrazení řešení, použijeme funkce *ListPlot*.



Graf 1 - Řešení 1. ukázkového příkladu pomocí *NDSolve*



Graf 2 - Řešení 2. ukázkového příkladu pomocí *NDSolve*

V *Grafu 1* a *Grafu 2* můžeme vidět porovnání přesného řešení, vykreslenou křivkou, s přibližným řešením, zobrazeným diskretními body. Z grafů lze usuzovat, že řešení je téměř přesné.

Pokud vypíšete přesné hodnoty řešení do *Tabulky 1* a *Tabulky 2* a porovnáte je s hodnotami, které vychází z algoritmu, je patrné, že malá nepřesnost zde je.

Řešení zobrazuji ve 20 krocích. U příkazu *NDSolve* nastavuji počet kroků posledním z parametrů příkazu *Table*, který značí velikost kroku.

***TableForm[Table[{x, f[x]}, {x, 0, 2Pi, Pi/9.8}]]***

Hodnoty  $x$  a  $y$  nám představují vypočítané hodnoty. Hodnoty  $x^*$  a  $y^*$  nám představují přesné hodnoty naší funkce.

	x	y		x*	y*
1	0.	$-2.85212 \times 10^{-21}$	1	0.	0.
2	0.320571	0.315108	2	0.320571	0.315108
3	0.641141	0.598111	3	0.641141	0.598111
4	0.961712	0.820172	4	0.961712	0.820172
5	1.28228	0.958668	5	1.28228	0.958668
6	1.60285	0.999486	6	1.60285	0.999486
7	1.92342	0.938468	7	1.92342	0.938468
8	2.24399	0.781831	8	2.24399	0.781831
9	2.56457	0.545534	9	2.56457	0.545535
10	2.88514	0.253655	10	2.88514	0.253655
11	3.20571	-0.0640701	11	3.20571	-0.0640702
12	3.52628	-0.375267	12	3.52628	-0.375267
13	3.84685	-0.648227	13	3.84685	-0.648228
14	4.16742	-0.855142	14	4.16742	-0.855143
15	4.48799	-0.974927	15	4.48799	-0.974928
16	4.80856	-0.995379	16	4.80856	-0.995379
17	5.12913	-0.91441	17	5.12913	-0.914413
18	5.4497	-0.740276	18	5.4497	-0.740278
19	5.77027	-0.490717	19	5.77027	-0.490718
20	6.09084	-0.191158	20	6.09084	-0.191159

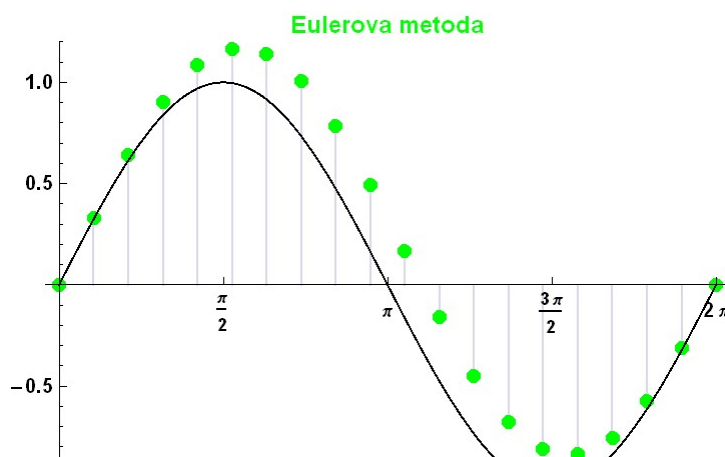
Tabulka 1 - Porovnání výsledků 1. př. pomocí *NDSolve*

	x	y		x*	y*
1	1.	2.	1	1.	2.
2	1.105	2.33212	2	1.105	2.33212
3	1.21	2.70078	3	1.21	2.70078
4	1.315	3.10947	4	1.315	3.10947
5	1.42	3.56164	5	1.42	3.56164
6	1.525	4.06079	6	1.525	4.06079
7	1.63	4.61037	7	1.63	4.61037
8	1.735	5.21387	8	1.735	5.21387
9	1.84	5.87475	9	1.84	5.87475
10	1.945	6.59649	10	1.945	6.59649
11	2.05	7.38256	11	2.05	7.38256
12	2.155	8.23644	12	2.155	8.23644
13	2.26	9.16159	13	2.26	9.16159
14	2.365	10.1615	14	2.365	10.1615
15	2.47	11.2396	15	2.47	11.2396
16	2.575	12.3994	16	2.575	12.3994
17	2.68	13.6444	17	2.68	13.6444
18	2.785	14.978	18	2.785	14.978
19	2.89	16.4038	19	2.89	16.4038
20	2.995	17.9251	20	2.995	17.9251

Tabulka 2 - Porovnání výsledků 2. př. pomocí *NDSolve*

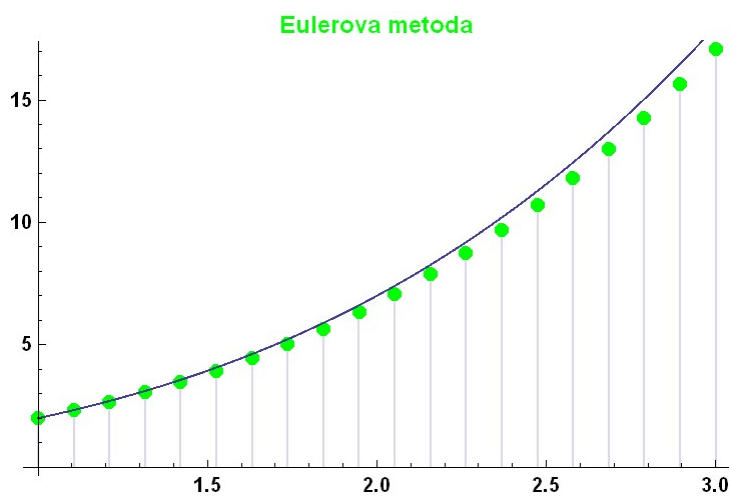
#### 4.1.2 Eulerova metoda

Eulerova metoda se řadí mezi méně přesné metody z jednokrokových metod. To si ukážeme i na ukázkových příkladech. Vycházíme ze vztahu (2.12) a postupujeme iteračně, dokud nesplníme počet kroků. Graf zobrazíme pomocí funkce *ListPlot*.



Graf 3 - Řešení 1. ukázkového příkladu

Eulerovou metodou



Graf 4 - Řešení 2. ukázkového příkladu

Eulerovou metodou

Porovnáním přesného a vypočteného řešení v *Grafu 3* a *Grafu 4* je zřetelné, že Eulerova metoda skutečně patří mezi méně přesné metody. O tom se přesvědčíme i v *Tabulce 3* a *Tabulce 4*.

	x	y		x*	y*
1	0	0	1	0.	0.
2	0.330694	0.330694	2	0.330694	0.324699
3	0.661388	0.64347	3	0.661388	0.614213
4	0.992082	0.904434	4	0.992082	0.837166
5	1.32278	1.08531	5	1.32278	0.9694
6	1.65347	1.16649	6	1.65347	0.996584
7	1.98416	1.13918	7	1.98416	0.915773
8	2.31486	1.00634	8	2.31486	0.735724
9	2.64555	0.782367	9	2.64555	0.475947
10	2.97625	0.491531	10	2.97625	0.164595
11	3.30694	0.165347	11	3.30694	-0.164595
12	3.63763	-0.160837	12	3.63763	-0.475947
13	3.96833	-0.451673	13	3.96833	-0.735724
14	4.29902	-0.675646	14	4.29902	-0.915773
15	4.62972	-0.808485	15	4.62972	-0.996584
16	4.96041	-0.835793	16	4.96041	-0.9694
17	5.2911	-0.754613	17	5.2911	-0.837166
18	5.6218	-0.57374	18	5.6218	-0.614213
19	5.95249	-0.312776	19	5.95249	-0.324699
20	6.28319	$-4.996 \times 10^{-16}$	20	6.28319	$-2.44929 \times 10^{-16}$

Tabulka 3 - Porovnání výsledků 1. př. Eulerovou metodou

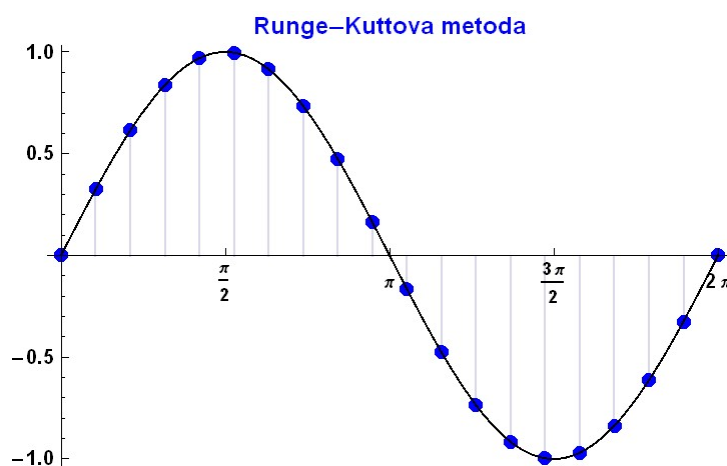
	x	y		x*	y*
1	1	2	1	1.	2.
2	1.10526	2.31579	2	1.105	2.33212
3	1.21053	2.66493	3	1.21	2.70078
4	1.31579	3.05091	4	1.315	3.10947
5	1.42105	3.47723	5	1.42	3.56164
6	1.52632	3.94737	6	1.525	4.06079
7	1.63158	4.46483	7	1.63	4.61037
8	1.73684	5.0331	8	1.735	5.21387
9	1.84211	5.65567	9	1.84	5.87475
10	1.94737	6.33605	10	1.945	6.59649
11	2.05263	7.07772	11	2.05	7.38256
12	2.15789	7.88419	12	2.155	8.23644
13	2.26316	8.75894	13	2.26	9.16159
14	2.36842	9.70548	14	2.365	10.1615
15	2.47368	10.7273	15	2.47	11.2396
16	2.57895	11.8279	16	2.575	12.3994
17	2.68421	13.0108	17	2.68	13.6444
18	2.78947	14.2794	18	2.785	14.978
19	2.89474	15.6373	19	2.89	16.4038
20	3.	17.088	20	2.995	17.9251

Tabulka 4 - Porovnání výsledků 2. př. Eulerovou metodou

Hodnoty  $x$  a  $y$  nám představují vypočítané hodnoty. Hodnoty  $x^*$  a  $y^*$  nám představují přesné hodnoty naší funkce. Vidíme, že ve stejných hodnotách osy  $x$ , vychází hodnoty  $y$  s větším rozptylem než u předchozí metody. Z hodnot je patrné, že Eulerova metoda není metodou příliš přesnou.

#### 4.1.3 Metoda Runge-Kutta

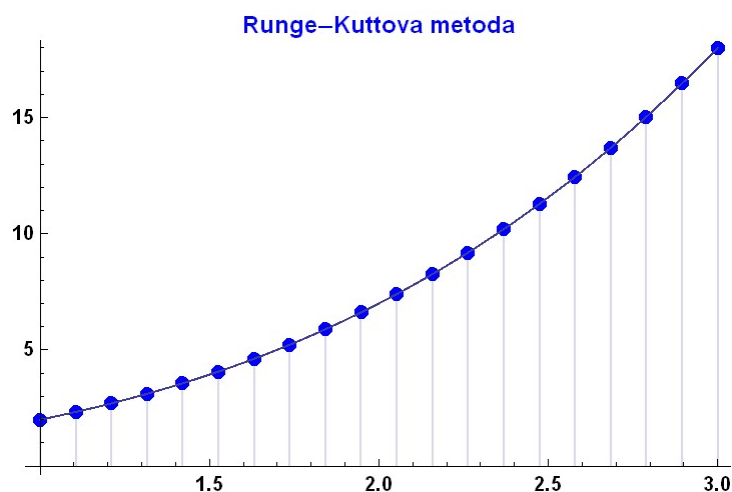
Pro výpočet numerické přibližné hodnoty řešeného problému s počáteční hodnotou  $y' = f(x, y)$  s počáteční podmínkou  $y(x_0) = y_0 = \alpha$  na intervalu  $[x_0, b] = [a, b]$  vycházíme z rovnic (2.21). Graf vykreslíme pomocí funkce *ListPlot*.



Graf 5 - Řešení 1. ukázkového příkladu

RK metodou





Graf 6 - Řešení 2. ukázkového příkladu

RK metodou

Z grafů je patrné, že Runge-Kuttova metoda patří mezi přesnější numerické metody řešení. Diskrétní body, které představují naše výpočty v *Mathematice*, leží v souřadnicích, kterými prochází křivka vykreslující přesné řešení. Přesvědčíme se v *Tabulce 5* a *Tabulce 6*.

	x	y		x*	y*
1	0	0	1	0.	0.
2	0.330694	0.324701	2	0.330694	0.324699
3	0.661388	0.614215	3	0.661388	0.614213
4	0.992082	0.83717	4	0.992082	0.837166
5	1.32278	0.969404	5	1.32278	0.9694
6	1.65347	0.996589	6	1.65347	0.996584
7	1.98416	0.915777	7	1.98416	0.915773
8	2.31486	0.735727	8	2.31486	0.735724
9	2.64555	0.475949	9	2.64555	0.475947
10	2.97625	0.164595	10	2.97625	0.164595
11	3.30694	-0.164595	11	3.30694	-0.164595
12	3.63763	-0.475949	12	3.63763	-0.475947
13	3.96833	-0.735727	13	3.96833	-0.735724
14	4.29902	-0.915777	14	4.29902	-0.915773
15	4.62972	-0.996589	15	4.62972	-0.996584
16	4.96041	-0.969404	16	4.96041	-0.9694
17	5.2911	-0.83717	17	5.2911	-0.837166
18	5.6218	-0.614215	18	5.6218	-0.614213
19	5.95249	-0.324701	19	5.95249	-0.324699
20	6.28319	$-1.11022 \times 10^{-15}$	20	6.28319	$-2.44929 \times 10^{-16}$

Tabulka 5 - Porovnání výsledků 1. př. RK metodou

	x	y		x*	y*
1	1	2	1	1.	2.
2	1.10526	2.33299	2	1.105	2.33212
3	1.21053	2.70273	3	1.21	2.70078
4	1.31579	3.1127	4	1.315	3.10947
5	1.42105	3.56641	5	1.42	3.56164
6	1.52632	4.06735	6	1.525	4.06079
7	1.63158	4.61904	7	1.63	4.61037
8	1.73684	5.22496	8	1.735	5.21387
9	1.84211	5.88861	9	1.84	5.87475
10	1.94737	6.6135	10	1.945	6.59649
11	2.05263	7.40312	11	2.05	7.38256
12	2.15789	8.26097	12	2.155	8.23644
13	2.26316	9.19055	13	2.26	9.16159
14	2.36842	10.1954	14	2.365	10.1615
15	2.47368	11.2789	15	2.47	11.2396
16	2.57895	12.4447	16	2.575	12.3994
17	2.68421	13.6962	17	2.68	13.6444
18	2.78947	15.0369	18	2.785	14.978
19	2.89474	16.4703	19	2.89	16.4038
20	3.	18.	20	2.995	17.9251

Tabulka 6 - Porovnání výsledků 2. př. RK metodou

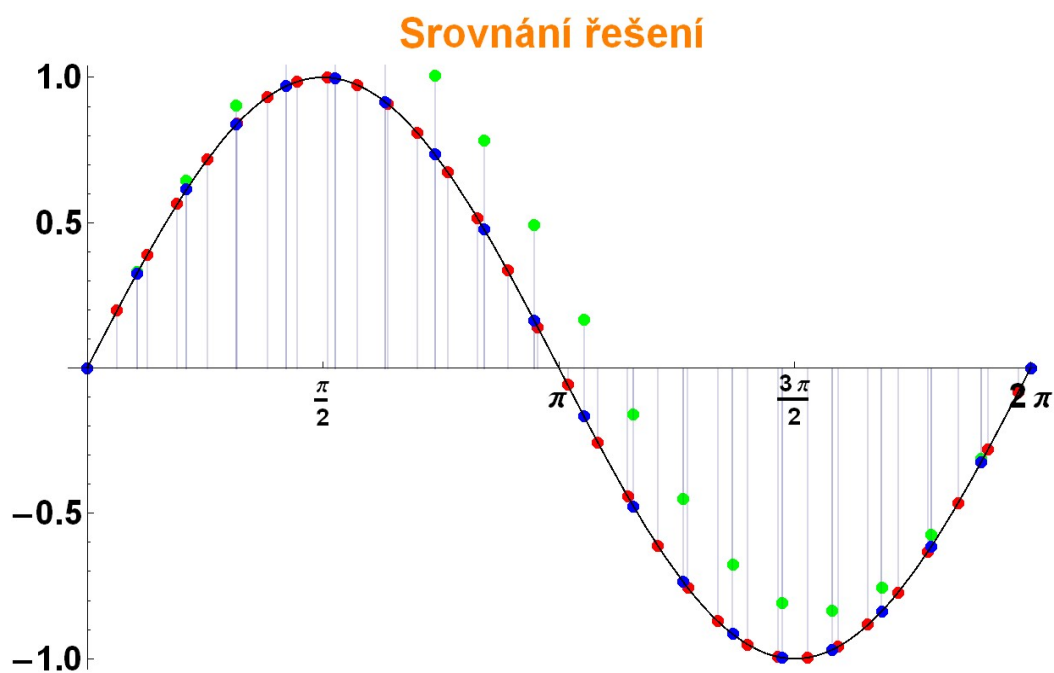
Hodnoty  $x$  a  $y$  nám představují vypočítané hodnoty. Hodnoty  $x^*$  a  $y^*$  nám představují přesné hodnoty naší funkce. Řešení zobrazují ve 20 stejných krocích.

Vidíme, že ve stejných hodnotách osy  $x$ , vychází hodnoty  $y$  v numerickém a přesném řešení téměř stejné. Rozdíl mezi přesným a numerickým řešením je v metodě Runge-Kutta malý.

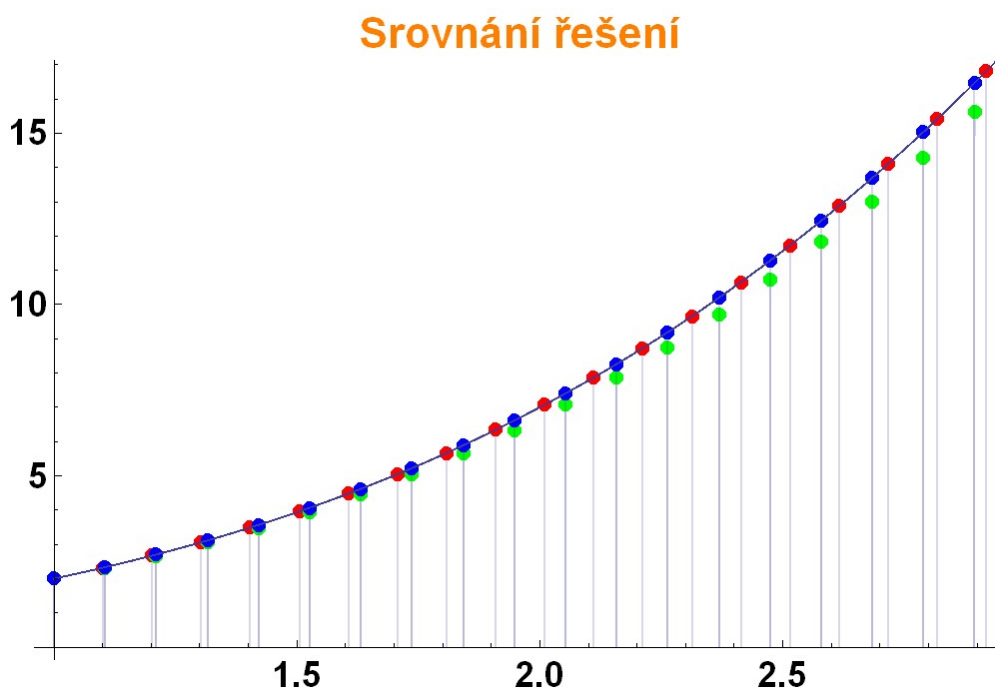
#### 4.1.4 Srovnání řešení

Pro srovnání výsledků vynesu všechny 3 výše zmíněné metody do jednoho společného grafu.





Graf 7 - Srovnání výsledků metod na 1. příkladu



Graf 8 - Srovnání výsledků metod na 2. příkladu

V *Grafu 7* a *Grafu 8* můžeme porovnat všechny výše zmíněné metody řešení ODR pro první a druhý příklad. Spojitá křivka představuje přesné řešení. Diskrétní body, rozlišené barevně, znázorňují numerické metody vypočítané v programu *Mathematica*. Červená – řešení *NDSolve*, zelená – Eulerova metoda, modrá – Runge-Kuttova metoda.

Přesvědčila jsem se, že řešení metodou Runge-Kutta je z výše zmíněných metod nejpřesnější. Ovšem přesnost z grafu lze odvodit jen částečně. Proto hodnoty vypisuji do tabulek. Srovnání provedu na prvním příkladě.

Metoda, při které použiji příkaz *NDSolve* s délkou kroku  $\pi/9,8$  je s přesností na 6 desetinných míst. Pokud se krok zvětší na  $\pi/4,8$  je přesnost 5 desetinných míst. Zvětšením kroku dosáhnou menšího počtu kroků. Eulerova metoda je nejméně přesná, pouze s přesností na 1 desetinné místo v případě 20 kroků. Pokud zvětšíme krok, přesnost se pohybuje jen v celém čísle, maximálně 1 desetině. Metoda Runge-Kutta je s přesností na 3 desetinná místa v případě 20 i 10 kroků.

	x	y		x*	y*
1	0.	$-2.85212 \times 10^{-21}$	1	0.	0.
2	0.654498	0.608761	2	0.654498	0.608761
3	1.309	0.965926	3	1.309	0.965926
4	1.9635	0.923879	4	1.9635	0.92388
5	2.61799	0.5	5	2.61799	0.5
6	3.27249	-0.130526	6	3.27249	-0.130526
7	3.92699	-0.707107	7	3.92699	-0.707107
8	4.58149	-0.991444	8	4.58149	-0.991445
9	5.23599	-0.866024	9	5.23599	-0.866025
10	5.89049	-0.382683	10	5.89049	-0.382683

Tabulka 7 – Řešení pro zvětšený krok *NDSolve* 1. příkladu

## 5 APLIKACE ODR

Konkrétní příklad, který uvádím, spadá do vědní oblasti demografie. Demografie je věda, která se zabývá reprodukcí (obnovou) lidské populace. Zkoumá především počet narozených dětí.

### 5.1 Slovní zadání

Počet narozených dětí ve velkoměstě je v období  $t \in \langle 0,6 \rangle$  ( $t$  je udáno v rocích) je modelován funkcí

$$N'(t) = -\frac{1}{4}t^2 + t + 4, \quad (5.1)$$

(kde  $N'(t)$  je v tisících dětí). Kolik dětí se v průběhu šesti let narodí celkem, jaký je průměrný počet narozených dětí v jednom roce a ve kterém roce byl počet narozených dětí největší?

### 5.2 Výpočet

Celkový počet narozených dětí získáme integrací funkce  $N'(t)$  na intervalu  $t \in \langle 0,6 \rangle$ .

$$P = \int_0^6 -\frac{1}{4}t^2 + t + 4 dt = \left[ -\frac{1}{4} \frac{t^3}{3} + \frac{1}{2}t^2 + 4t \right]_0^6 = 24 \quad (5.2)$$

Dosazením mezí do integrované funkce získáme celkový počet narozených dětí v období 6 let, tedy 24 tisíc.

Průměrný počet narozených dětí v 1 roce získáme podle věty o střední hodnotě, tedy podílem integrované funkce velikostí intervalu.

$$\bar{P} = \frac{\int_0^6 -\frac{1}{4}t^2 + t + 4dt}{6-0} = \frac{24}{6} = 4 \quad (5.3)$$

Z výpočtu vidíme, že průměrně se během 1 roku narodí 4 tisíce dětí.

Abychom zjistili, v kterém roce byl počet narozených dětí největší, dosadíme do integrované funkce postupně  $t \in \langle 0,6 \rangle$ . Tedy za  $t$  dosadíme hodnoty 1, 2, 3, 4, 5, 6.

$$R(t) = \int_0^t \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.4)$$

$$R_1 = \int_0^1 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.5)$$

$$R_1 = -\frac{1}{12} + \frac{1}{2} + 4 = \frac{53}{12} = 4,416$$

Za první rok se narodilo 4,416 tisíce dětí.

$$R_2 = \int_0^2 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.6)$$

$$R_2 = -\frac{8}{12} + 2 + 8 = \frac{28}{3} = 9,33$$

$$R_2 - R_1 = 4,914$$

Za druhý rok se narodilo 4,914 tisíce dětí.

$$R_3 = \int_0^3 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.7)$$

$$R_3 = -\frac{27}{12} + \frac{9}{2} + 12 = \frac{171}{12} = 14,25$$

$$R_3 - R_2 = 4,92$$

Za třetí rok se narodilo 4,92 tisíce dětí.

$$R_4 = \int_0^4 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.8)$$

$$R_4 = -\frac{64}{12} + 8 + 16 = \frac{224}{12} = 18,66$$

$$R_4 - R_3 = 4,41$$

Za čtvrtý rok se narodilo 4,41 tisíce dětí.

$$R_5 = \int_0^5 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.9)$$

$$R_5 = -\frac{125}{12} + \frac{25}{2} + 20 = \frac{265}{12} = 22,083$$

$$R_5 - R_4 = 3,423$$

Za pátý rok se narodilo 3,423 tisíce dětí.

$$R_6 = \int_0^6 \left( -\frac{1}{4}x^2 + x + 4 \right) dx \quad (5.10)$$

$$R_6 = -\frac{216}{12} + 18 + 24 = \frac{288}{12} = 24$$

$$R_6 - R_5 = 1,917$$

Za šestý rok se narodilo 1,917 tisíce dětí.

Dle výpočtů byl největší počet narozených dětí ve třetím roce s počtem 4,92 tisíce narozených dětí. Naopak nejméně dětí se narodilo v roce šestém s počtem 1,917 tisíce narozených dětí. Tento výsledek se může lišit od přesného řešení v *Mathematica*, protože hodnoty zaokrouhluji.

### 5.3 Přesné řešení pomocí software Mathematica

Celkový počet narozených dětí v průběhu šesti let v programu *Mathematica* vyřešíme integrací naší zadané funkce. Průměrný počet narozených dětí v jednom roce vyřešíme integrací funkce dělenou délkou doby, tj. 6.



Graf 9 - Celkový počet dětí

Dále bylo nutné vytvořit posloupnost hodnot  $a$  integrované funkce, abychom získali počty narozených dětí v jednotlivých letech. Počet narozených dětí v jednotlivých letech jsem zjistila pomocí příkazu *Differences*. Maximální hodnotu z této nové posloupnosti vybereme lehce, pomocí příkazu *Max*.

***Roz = Differences [a];***

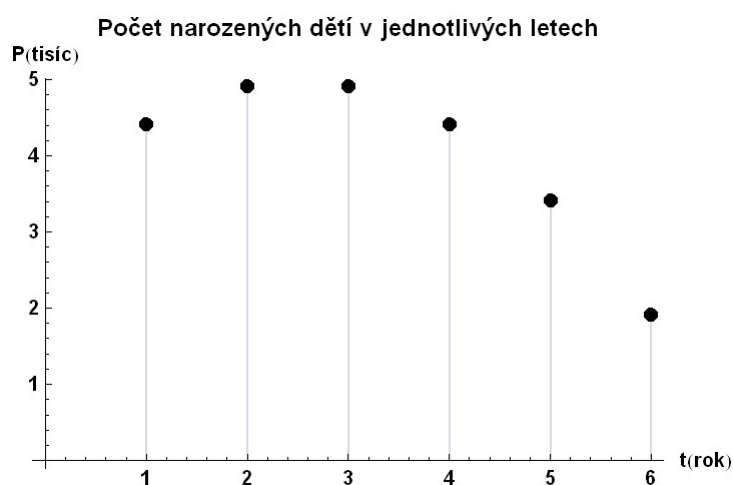
***nej = Max [roz];***

---

Celkový počet všech narozených dětí v období 6 let je: 24 tisíc  
 Průměrný počet narozených dětí v 1 roce je: 4 tisíce  
 Nejvyšší počet narozených dětí v jednom roce je: 4.91667

Obrázek 7 – Řešení v Mathematica

Výsledky jsem vynesla do tabulky a do grafu.



Graf 10 - Počet dětí v každém roce

Podle grafu můžeme vyčíst, že nejvíc dětí se narodilo v druhém a třetím roce. Vypíši tyto hodnoty do tabulky.

```
Out[473]/TableForm=
1.rok | 4.41667
2.rok | 4.91667
3.rok | 4.91667
4.rok | 4.41667
5.rok | 3.41667
6.rok | 1.91667
```

Tabulka 8 – Jednotlivé roky

Z tabulky můžeme vyčíst počty narozených dětí v jednotlivých letech. Nejvyšší počet narozených dětí je ve druhém a třetím roce s počtem 4,91667 tisíce dětí.

## 5.4 Přibližné řešení pomocí software Mathematica

Jelikož samotné výpočty jednotlivých metod se liší, pro každou z metod jsem musela tvořit vlastní algoritmy pro výpis a dopočítání výsledků.

### 5.4.1 Přibližné řešení pomocí NDSolve

Výpočet pomocí příkazu *NDSolve* nám vytvoří příkazem *Table* posloupnost hodnot v intervalu od 0 do 6. Získáme posloupnost, která stále narůstá, protože nám značí celkový počet narozených dětí v tomto intervalu. Posloupnost vytvářím, abych mohla pracovat s jednotlivými hodnotami v pozicích. Pokud chci zjistit celkový počet narozených dětí během šesti let, stačí vypsat poslední pozici.



Graf 11 - Celkový počet dětí pomocí *NDSolve*

Pro průměrnou hodnotu narozených dětí v jednom roce provedu podíl celkového počtu dětí a počtu let. Maximální hodnotu získáme příkazem *Max*, z jednotlivých hodnot pozic 1 - 6. Do výpisu se promítne, že se jedná o posloupnost, proto zde zůstanou složené závorky.



---

Celkový počet všech narozených dětí v období 6 let je: {24.} tisíc  
 Průměrný počet narozených dětí v 1 roce je: {4.} tisíce  
 Nejvyšší počet narozených dětí v jednom roce je: 4.91667 tisíc dětí v roce: {{2}, {3}}.

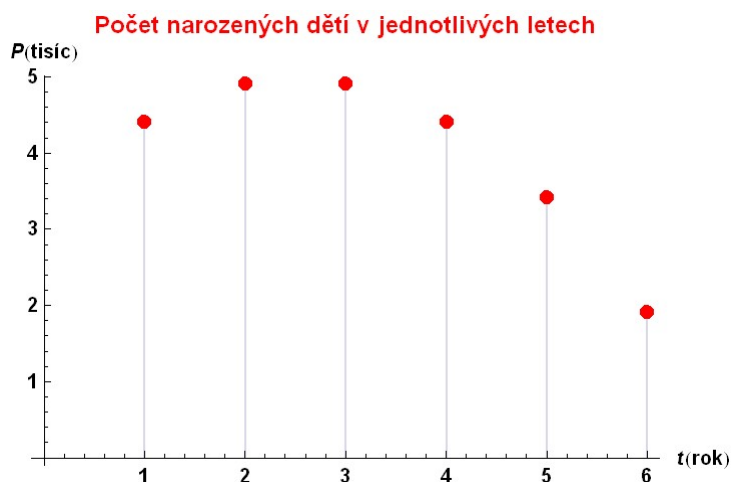
---

Obrázek 8 - Řešení pomocí *NDSolve*

Abych získala počty narozených dětí v jednotlivých letech, použila jsem příkaz *Differences*. Ten jsem použila na vypočítanou posloupnost hodnot.

$$Roz = Differences[Table[f[x], \{x, 0, 6, 1\}]];$$

Výsledky jsem vynesla do tabulky a do grafu.

Graf 12 - Počet dětí v každém roce pomocí *NDSolve*

Z grafu můžeme vyčíst, že nejvíc dětí se narodilo v druhém roce a třetím roce. Vypíši tyto hodnoty do tabulky.

```
Out[1043]/TableForm=
1.rok|4.41667
2.rok|4.91667
3.rok|4.91667
4.rok|4.41667
5.rok|3.41667
6.rok|1.91667
```

Tabulka 9 - Jednotlivé roky

Výpisem hodnot jsem se přesvědčila, že řešení pomocí *NDSolve* je shodné s přesným řešením. Dle výpisu hodnot v tabulce vidíme, že nejvyšší počet narozených dětí byl v druhém a třetím roce. Což se shoduje s výsledky přesného řešení.

#### 5.4.2 Přibližné řešení Eulerovou metodou

Výpočtem funkce Eulerovou metodou získáme posloupnost hodnot, která nám představuje nárůst počtu dětí v závislosti na počtu roků. Poslední krok znamená celkový počet narozených dětí v období šesti let.



Graf 13 - Celkový počet dětí Eulerovou metodou

Pro průměrnou hodnotu narozených dětí v jednom roce provedu podíl celkového počtu dětí a počtu let. Maximální hodnotu získáme příkazem *Max*, z jednotlivých hodnot pozic 1 - 7

(začínáme nultou pozicí), ale protože je *eulerlist* dvouhodnotový, při výpisu specifikuji příkazem *Position* výpis druhého prvku ve všech krocích.

***Print [Position [roz][[All,2]], nej];***

---

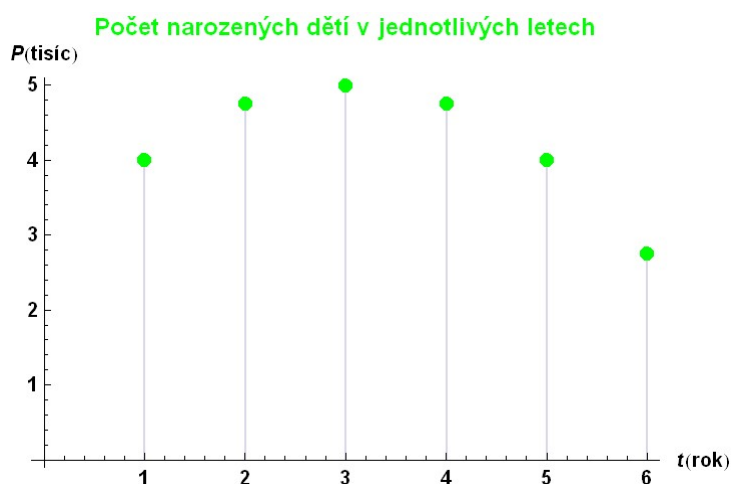
```
Celkový počet všech narozených dětí v období 6 let je: 25.25 tisíc
Průměrný počet narozených dětí v 1 roce je: 4.20833 tisíce
Nejvyšší počet narozených dětí v jednom roce je: 5. tisíc dětí v roce: {{3}}.
```

Obrázek 9 - Řešení Eulerovou metodou

Abych získala počty narozených dětí v jednotlivých letech, použila jsem příkaz *Differences*. Ten jsem použila na vypočítanou posloupnost hodnot. Výsledky jsem vynesla do tabulky a do grafu.

***Roz = Differences [Take [eulerlist, {1,7}]];***

***Nej = Max [ Differences [Take [eulerlist, {1,7}]]];***



Graf 14 - Počet dětí v každém roce Eulerovou metodou

Z grafu můžeme vyčíst, že nejvyšší počet dětí se narodil v třetím roce. Vypíši tyto hodnoty do tabulky.

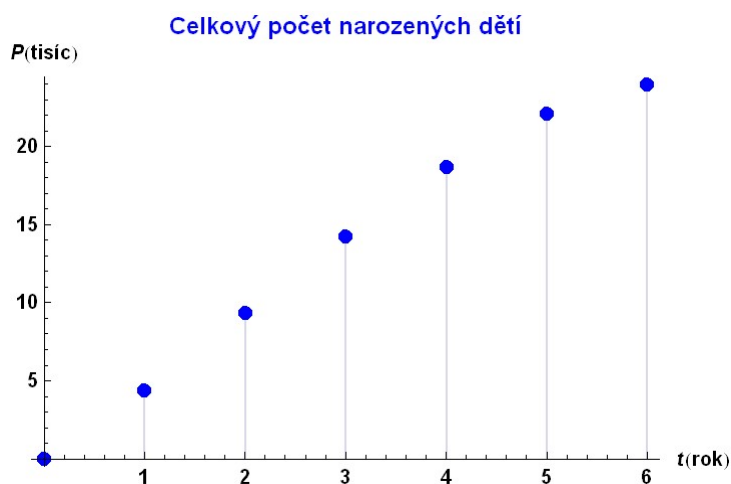
```
Out[1006]/TableForm=  
1.rok | 4.  
2.rok | 4.75  
3.rok | 5.  
4.rok | 4.75  
5.rok | 4.  
6.rok | 2.75
```

Tabulka 10 - Jednotlivé roky

Výpisem hodnot je skutečně ve třetím roce narozeno nejvíc dětí. Také jsem se přesvědčila, že Eulerova metoda není příliš přesnou metodou. V porovnání s přesným řešením je patrná chyba numerického řešení.

### 5.4.3 Přibližné řešení Runge-Kuttovou metodou

Výpočtem funkce Runge-Kuttovou metodou získáme posloupnost hodnot, která nám představuje nárůst počtu dětí v závislosti na počtu roků. Poslední krok znamená celkový počet narozených dětí v období šesti let.



Graf 15 - Celkový počet dětí RK metodou

Pro průměrnou hodnotu narozených dětí v jednom roce provedu rozdíl celkového počtu dětí a počtu let. Maximální hodnotu získáme příkazem *Max*, z jednotlivých hodnot pozic 1 - 7 (začínáme nultou pozicí), ale protože je *runelist* dvouhodnotový, při výpisu specifikuji příkazem *Position* výpis druhého prvku ve všech krocích.

---

```
Celkový počet všech narozených dětí v období 6 let je: 24. tisíc
Průměrný počet narozených dětí v 1 roce je:  $\frac{24.}{6}$  tisíce
Nejvyšší počet narozených dětí v jednom roce je: 4.91667 tisíc dětí v roce: {{2}, {3}}.
```

---

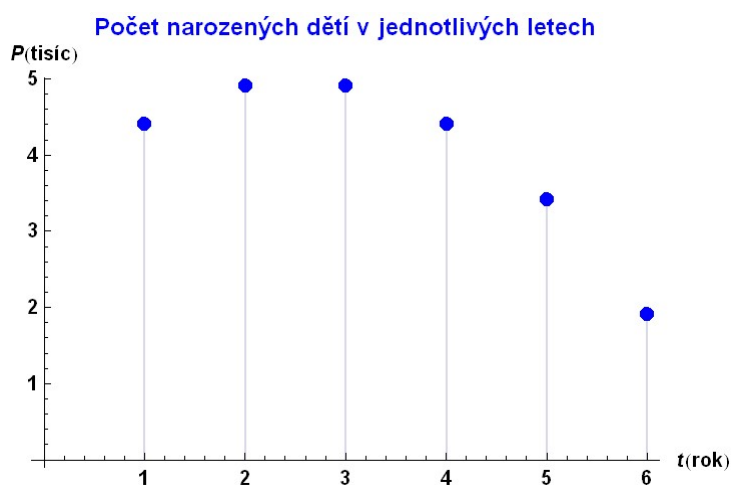
Obrázek 10 - Řešení RK metodou

Abych získala počty narozených dětí v jednotlivých letech, použila jsem příkaz *Differences*. Ten jsem použila na vypočítanou posloupnost hodnot.

***Roz = Differences [Take [runelist, {1,7}]];***

***Nej = Max [Differences [Take [runelist, {1,7}]]];***

Výsledky jsem vynesla do tabulky a do grafu.



Graf 16 - Počet dětí v každém roce RK metodou

Z grafu je patrné, že nejvyšší počet narozených dětí je ve druhém a třetím roce. Jednotlivé hodnoty vypíši do tabulky.

```
Out[1028]/TableForm=
1.rok|4.41667
2.rok|4.91667
3.rok|4.91667
4.rok|4.41667
5.rok|3.41667
6.rok|1.91667
```

Tabulka 11 - Jednotlivé roky

Výpisem hodnot je vidět, že nejvyšší počet narozených dětí je ve druhém a třetím roce. Porovnáním s přesným řešením jsem se přesvědčila, že metoda Runge-Kutta patří mezi přesnější numerické metody řešení.

## 5.5 Zhodnocení výsledků

V programu *Mathematica* jsem vypočítala přesné řešení praktické úlohy. Porovnáním výsledků s třemi metodami řešení jsem dospěla k závěru, že Eulerova metoda patří sice mezi jednodušší metody pro výpočet numerických řešení ODR, avšak výsledky nejsou zcela vyhovující. Oproti Eulerově metodě je metoda Runge-Kutta mnohem přesnější metodou. To má samozřejmě vliv na čas výpočtu.

Výsledky získané pomocí *NDSolve* a Runge-Kuttovou metodou se porovnáním s přesným řešením v tomto příkladě neliší. Volila jsem šest kroků, abych docílila toho, že získám řešení přímo v letech 1 – 6.

## ZÁVĚR

Cílem této bakalářské práce bylo vytvořit matematický model a numericky řešit obyčejné diferenciální rovnice v programu *Mathematica*. Diferenciálními rovnicemi lze formulovat velké množství vědních problémů. Objevují se snad ve všech vědních oborech. Veliké zastoupení mají v matematice, fyzice, automatizaci, optimalizaci, chemii, ekologii ale např. i v demografii, z které uvádím praktický příklad.

K vytvoření algoritmů jsem vycházela z matematických definicí pro jednotlivé metody řešení. K porovnání výsledků tří výše zmíněných metod nejlépe poslouží grafy, na kterých je vidět rozdíl mezi přesným a vypočítaným řešením. Přesvědčila jsem se, že Eulerova metoda je nejméně přesnou metodou pro numerická řešení, ovšem není tak náročná na čas výpočtu. Mnohem přesnějšími metodami jsou metoda Runge-Kutta a řešení pomocí příkazu *NDSolve*. Avšak přesnost z grafu lze odvodit jen částečně, je to dost zkreslené. Numerické chyby, které zde jsou, jsou malé, v řádech tisícín, desítitisícín. Ovšem tyto přesnější metody jsou náročnější na čas výpočtu, proto je lepší volit menší počet kroků. S menším počtem kroků, jsou metody Eulerova i výsledky pomocí *NDSolve* méně přesnými. Runge-Kuttova metoda je s přesností na tři desetinná místa neměnná i v případě zmenšení počtu kroků. V praktickém příkladu jsem volila šest kroků, abych docílila správných výsledků v jednotlivých letech. Metody Runge-Kutta a řešení pomocí *NDSolve* zde vycházejí dostatečně přesně.

Program *Mathematica* je výbornou pomůckou nejen pro studenty vysokých škol, ale určitě ho využijí i lidé z různých oblastí vědy, techniky či ekonomie. Software je jednoduchý a přijatelný pro každého. Navíc obsahuje nápovědu, která zahrnuje *The Mathematica book* s velkým množstvím příkladů, na kterých je ukázána syntaxe jednotlivých funkcí.

## CONCLUSION

The aim of this bachelor thesis was to develop a mathematical model and numerically solve ordinary differential equations in the software *Mathematica*. In differential equations it can be formulated a large number of scientific problems. ODE appears in almost all scientific disciplines. They have a great representation in mathematics, physics, automation, optimization, chemistry, ecology and also in demographics, from which I present a practical example.

I created algorithms based on mathematical definition for the various methods of solution. To compare the results of the three above mentioned methods I use the graph in which you can see the differences between the exact solution and the numerical solution. I am convinced that the Euler method is less accurate method for numerical solution, but it's not so time-consuming calculation. Runge-Kutta method and solution using the *NDSolve* are much more accurate methods. Their numerical mistakes are acceptable, in hundredths or even in thousandths. However these methods are more demanding to the time consuming calculation. It is better to choose fewer steps. The Runge-Kutta methods and result using *NDSolve* are less accurate using fewer steps. In a practical example I chose six steps to achieve correct results in each of individual years. The Runge-Kutta method, the solution using *NDSolve* and the exact solution have the same result.

The software *Mathematica* is an excellent program not only for university students, but it's also suitable for people from different fields of science, engineering or economics. The software is simple and acceptable for anyone. *Mathematica* includes *the Mathematica book*, which contains the program's manual and a many examples with syntax instructions.



## SEZNAM POUŽITÉ LITERATURY

- [1] KUFNER Alois. *Obyčejné diferenciální rovnice*. Plzeň, 1993. 159 s. Západočeská univerzita. Skripta.
- [2] VITÁSEK, Emil, PRÁGER, Milan, BABUŠKA, Ivo. *Numerické řešení diferenciálních rovnic*. Praha : Státní nakladatelství technické literatury, 1964. 238 s.
- [3] VITÁSEK, Emil. *Základy teorie numerických metod pro řešení diferenciálních rovnic*. Praha : Academia Praha, 1994. 409 s.
- [4] KUČERA, Radek. *Numerické metody*. Ostrava, 2008. 143 s. Vytvořeno v rámci projektu Operačního programu Rozvoje lidských zdrojů. Vysoká škola Báňská : Technická univerzita Ostrava.
- [5] ŘEZNÍČKOVÁ Jana. *Diferenciální rovnice* [online]. 2009 [cit.2011-03-12]. Dostupný z WWW: <<http://www.vyuka.fai.utb.cz/>>.
- [6] MOUČKA, Jiří, RÁDL, Petr. *Matematika pro studenty ekonomie*. Praha : Grada, 2010. 272 s.
- [7] Wikipedie : otevřená encyklopedie [online]. 2002 [cit. 2011-03-12]. Dostupný z WWW: <<http://www.wikipedia.cz/>>.
- [8] BRAUN, Martin. *Differential Equations And Their Applications*. New York : Springer, 1993. 718 s.
- [9] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica* [online]. 2006 [cit.2011-03-26]. Dostupný z WWW: <<http://www.vyuka.fai.utb.cz/>>.
- [10] Wolfram Research, Inc. *Numerical Solution of Differential Equations* [online]. 2011 [cit. 2011-04-03]. Wolfram Mathematica 8 Documentation Center. Dostupný z WWW: <<http://reference.wolfram.com/mathematica/ref/NDSolve.html>>
- [11] VICHER, Miroslav. *Numerická matematika*. Ústí nad Labem, 2003. 85s. Skripta.
- [12] The Mathematica Book, manuál pro software Mathematica

**SEZNAM POUŽITÝCH PŘÍKAZŮ SOFTWARE MATHEMATICA**

NDSolve	Nalezne numerická řešení obyčejných diferenciálních rovnic.
TableForm	Zobrazí posloupnost hodnot v tabulkové formě.
Table	Zobrazí posloupnost hodnot.
Map	Vytvoří vektor.
For	Vyhodnocuje výrazy uvnitř těla cyklu, dokud je podmínka pravdivá. V každém cyklu zvyšuje hodnotu proměnné. Počáteční hodnota proměnné je dána.
Append	Spojuje objekty do jednoho řetězce.
ListPlot	Vykreslení bodového grafu, kde souřadnice bodů jsou dány.
Show	Funkce pro zobrazení grafů.
N	Převod na numerické hodnoty.
Clear	Maže hodnoty a definice.
Module	Umožňuje nastavit lokální proměnné pro výrazy.
Return	Vrací hodnotu funkce.
Transpose	Transponuje matici.
Manipulate	Generuje výsledky s přidáním ovládacích prvků, umožňující změny hodnot.
Column	Výpis výsledků pod sebe.
Cos[x]	Trigonometrická funkce Cosinus.
Sin[x]	Trigonometrická funkce Sinus.
ReplaceAll	Dočasná změna hodnot probíhajícího výpočtu.
Max	Vybere maximální hodnotu ze seznamu hodnot.
Differences	Rozdíl dvou sousedních hodnot v seznamu.
Take	Vybírá element ze seznamu v určitém rozsahu.
Print	Vypisuje výstupy na obrazovku.

Position      Uvádí pozici hledané hodnoty v řetězci.

Přesný popis jednotlivých příkazů spolu s jejich parametry nalezneme v nápovědě software *Mathematica*.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ODR    **O**byčejná **D**iferenciální **R**ovnice

PDR    **P**arciální **D**iferenciální **R**ovnice

DR     **D**iferenciální **R**ovnice

RK     Metoda **R**unge-**K**utta

RK4    Metoda **R**unge-**K**utta **4.** Řádu

ODE    **O**rdinary **D**ifferential **E**quations

např.   **N**apříklad

apod.   **A** podobně

tj.      **T**o je

**SEZNAM OBRÁZKŮ**

Obrázek 1 - Odhad celkové chyby $E_C(h)$ .....	22
Obrázek 2 - Logo software Mathematica 8.....	23
Obrázek 3 - Ukázka interaktivní nápovědy.....	25
Obrázek 4 - Ukázka možností Mathematica.....	26
Obrázek 5 - Nástrojová paleta.....	27
Obrázek 6 - Rozsah uživatelů Mathematica .....	29
Obrázek 7 – Řešení v Mathematica .....	47
Obrázek 8 - Řešení pomocí <i>NDSolve</i> .....	49
Obrázek 9 - Řešení Eulerovou metodou .....	51
Obrázek 10 - Řešení RK metodou .....	53

**SEZNAM TABULEK**

Tabulka 1 - Porovnání výsledků 1. př. pomocí <i>NDSolve</i> .....	35
Tabulka 2 - Porovnání výsledků 2. př. pomocí <i>NDSolve</i> .....	35
Tabulka 3 - Porovnání výsledků 1. př. Eulerovou metodou .....	37
Tabulka 4 - Porovnání výsledků 2. př. Eulerovou metodou .....	37
Tabulka 5 - Porovnání výsledků 1. př. RK metodou .....	39
Tabulka 6 - Porovnání výsledků 2. př. RK metodou .....	40
Tabulka 7 – Řešení pro zvětšený krok <i>NDSolve</i> 1. příkladu .....	42
Tabulka 8 – Jednotlivé roky.....	47
Tabulka 9 - Jednotlivé roky .....	50
Tabulka 10 - Jednotlivé roky .....	52
Tabulka 11 - Jednotlivé roky .....	54

**SEZNAM GRAFŮ**

Graf 1 - Řešení 1. ukázkového příkladu pomocí <i>NDSolve</i> .....	33
Graf 2 - Řešení 2. ukázkového příkladu pomocí <i>NDSolve</i> .....	34
Graf 3 - Řešení 1. ukázkového příkladu .....	36
Graf 4 - Řešení 2. ukázkového příkladu .....	36
Graf 5 - Řešení 1. ukázkového příkladu .....	38
Graf 6 - Řešení 2. ukázkového příkladu .....	39
Graf 7 - Srovnání výsledků metod na 1. příkladu .....	41
Graf 8 - Srovnání výsledků metod na 2. příkladu .....	41
Graf 9 - Celkový počet dětí .....	46
Graf 10 - Počet dětí v každém roce .....	47
Graf 11 - Celkový počet dětí pomocí <i>NDSolve</i> .....	48
Graf 12 - Počet dětí v každém roce pomocí <i>NDSolve</i> .....	49
Graf 13 - Celkový počet dětí Eulerovou metodou .....	50
Graf 14 - Počet dětí v každém roce Eulerovou metodou .....	51
Graf 15 - Celkový počet dětí RK metodou .....	52
Graf 16 - Počet dětí v každém roce RK metodou .....	53

## SEZNAM PŘÍLOH

- P I Zdrojový kód naprogramovaných numerických metod pro řešení 1. uk. příkladu
- P II Zdrojový kód naprogramovaných numerických metod pro řešení 2. uk. příkladu
- P III Zdrojový kód naprogramovaných numerických metod pro řešení praktického příkladu

Zdrojové kódy k příkladům jsou vloženy na CD v příloze.