UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

Eulerova metoda

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **Mgr. Jana Burkotová, Ph.D.**
Vypracoval(a): **Helena Paulasová**
Studijní program: B1101 Matematika
Studijní obor Matematika a její aplikace
Forma studia: prezenční
Rok odevzdání: 2017

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Helena Paulasová

Název práce: Eulerova metoda

Typ práce: Bakalářská práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: Mgr. Jana Burkotová, Ph.D.

Rok obhajoby práce: 2018

Abstrakt: Tato práce se zabývá Eulerovou metodou pro numerické řešení počáteční úlohy obyčejných diferenciálních rovnic prvního řádu. Eulerova metoda je představena v následujících verzích a úpravách: explicitní, implicitní, adaptivní velikost kroku, Euler-Cromerova metoda a Heunova metoda. V textu je provedena základní analýza chyby a konvergence u explicitní Eulerovy metody. Na závěr jsou metody použity k přibližnému řešení diferenciálních rovnic s impulzy.

Klíčová slova: numerické metody, Eulerova metoda, obyčejné diferenciální rovnice, MATLAB, počáteční úloha

Počet stran: 75

Počet příloh: 1

Jazyk: Český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Helena Paulasová

Title: Euler's Method

Type of thesis: Bachelor's

Department: Department of Mathematical Analysis and Applications of Mathematics

Supervisor: Mgr. Jana Burkotová, Ph.D.

The year of presentation: 2018

Abstract: This thesis deals with Euler's method for solving initial value problems of ordinary differential equations numerically. Euler's method is introduced in the following versions and modifications: explicit, implicit, adaptive stepsize, Euler-Cromer's method and Heun's method. Convergence and error analysis of explicit Euler's method is discussed. The paper closes with examples using the method to solve differential equations with impulses.

Key words: numerical methods, Euler's method, ordinary differential equations, MATLAB, initial value problem

Number of pages: 75

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně pod vedením paní Mgr. Jany Burkotové, Ph.D. a všechny použité zdroje jsem uvedla v seznamu literatury.

V Olomouci dne ....

podpis

Obsah

Poděkování

Ráda bych poděkovala své vedoucí bakalářské práce Mgr. Janě Burkotové, Ph.D. za odborné vedení, cenné rady, trpělivost a ochotu, kterou mi věnovala.

Úvod

Eulerova metoda je numerická metoda pro řešení počáteční úlohy obyčejných diferenciálních rovnic. Řešit diferenciální rovnici numericky znamená najít přibližné hodnoty přesného řešení. Důvodem pro hledání přibližných řešení je skutečnost, že najít přesné řešení pomocí metod matematické analýzy je často nemožné. Přibližné řešení hledáme pouze pokud víme o existenci řešení přesného.

Numerické metody pro řešení počátečních úloh diferenciálních rovnic prvního řádu se dělí do dvou skupin: metody jednokrokové a mnohokrokové. Jednokroková metoda k nalezení další přibližné hodnoty využívá pouze informace získané z jedné předchozí přibližné hodnoty. Mnohokroková metoda využije informace získané z několika předchozích kroků. Eulerova metoda je nejjednodušší metodou jednokrokovou a spadá do třídy metod kterým se říká Runge-Kutta. Přednosti Eulerovy metody spočívají v jednoduchém zápisu, na kterém se dají ukázat koncepty, dále používané u složitějších metod, aniž by se čtenář-začátečník ztratil v technických detailech. Dalo by se namítnout, že Eulerova metoda je zastaralá či naivní. Důmyslnější metody totiž rychleji konvergují a používají se tak jako standartní nástroj pro řešení počátečních úloh v softwarech jako je Matlab. Nicméně chceme-li vyřešit složitější problém obsahující počáteční úlohu, tak si raději zvolíme metodu, jejíž zápis není příliš komplikovaný. Lze se tak lépe soustředit na detaily problému, které se numerické metody přímo netýkají. Příkladem budiž modelování trajektorie pohybující se částice uvnitř prostoru ohraničeného nějakou bariérou, který řeším v kapitole Numerické experimenty. Výhoda Eulerovy metody by se tedy dala shrnout frází v jednoduchosti je síla.

1 Základní pojmy

V této části definujeme některé dále používané pojmy z teorie diferenciálních rovnic, aby bylo jasné, co se těmito pojmy rozumí.

Definice 1.1. Systémem n diferenciálních rovnic 1. řádu rozumíme systém,

$$\begin{cases} y_1' = f_1(t, y_1, ..., y_n) \\ y_2' = f_2(t, y_1, ..., y_n) \\ \vdots \\ y_n' = f_n(t, y_1, ..., y_n), \end{cases} \tag{1}$$

kde $f_i : G \subset \mathbb{R}^{n+1} \to \mathbb{R}$, $y_i : \mathbb{R} \to \mathbb{R}$, $t_0 \leq t \leq b$ a G je oblast (otevřená, souvislá množina).
Pro zkrácení se používá vektorový zápis:

$$\mathbf{y} = \mathbf{f}(t, \mathbf{y}),$$

kde $\mathbf{f} = (f_1, ..., f_n)$ a $\mathbf{y} = (y_1, ..., y_n)$.

Definice 1.2. Analytickým (přesným) řešením systému (1), na intervalu $[t_0, b]$ je funkce $\mathbf{y} = (y_1(t), ..., y_n(t))$ pokud splňuje

1. $y_i \in \mathcal{C}^1([t_0, b])$, tj. její složky jsou na tomto intervalu definované, spojité a mají spojité derivace prvního řádu
2. $(t, \mathbf{y}(t)) \in G \subset \mathbb{R}^{n+1}$ $\forall t \in [t_0, b]$
3. $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$ $\forall t \in [t_0, b]$

Definice 1.3. Počáteční úlohou rozumíme systém (1) spolu s počáteční podmínkou

$$\mathbf{y}(t_0) = \mathbf{Y_0} = (y_1(t_0), ..., y_n(t_0)) \tag{1.2}$$

Definice 1.4. Řešením počáteční úlohy je řešení systému (1), které splňuje počáteční podmínku (1.2).

Počáteční úlohu budeme dále zapisovat následujícím způsobem:

$$\begin{cases} \mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \ t \in [t_0, b] \\ \mathbf{y}(t_0) = \mathbf{Y_0} \end{cases} \tag{1.3}$$

Věta 1.1. ([1], str. 19, Věta 1.1) *Věta o existenci a jednoznačnosti*
Nechť $\mathbf{Y_0}$ je libovolný bod v $\mathbb{R}^n$ a platí:

1. $\mathbf{f}$ je definovaná a spojitá jako funkce *n+1* proměnných v
   $G = \{(t, \mathbf{y}) \in \mathbb{R} \times \mathbb{R}^n; t_0 \leq t \leq b, \|\mathbf{y}\| < \infty\}$

2. $\mathbf{f}$ splňuje v G Lipschitzovu podmínku vzhledem k $\mathbf{y}$, tj. existuje $L > 0$ a platí:
   $$\|\mathbf{f}(t, \mathbf{y_1}) - \mathbf{f}(t, \mathbf{y_2})\| \leq L\|\mathbf{y_1} - \mathbf{y_2}\|$$
   pro každé $t \in [t_0, b]$ a libovolná $\mathbf{y_1}$ a $\mathbf{y_2}$.

Pak existuje jediné řešení počáteční úlohy (1.3).

Druhý předpoklad o pravé straně diferenciální rovnice je globální a značně silný, a tak se často stane, že nebude splněn. Většinou však předem máme nějakou představu o poloze přesného řešení. Díky tomu lze upravit definici pravé strany v těch částech množiny G, kde řešení zaručeně neleží. Tím dosáhneme splnění druhého předpokladu.

Definice 1.5. Diferenciální rovnicí n-tého řádu v normálním tvaru rozumíme rovnici

$$y^{(n)} = f(t, y(t), y'(t), ..., y^{(n-1)}(t)), \tag{1.4}$$

kde $f : \mathbb{R}^{n+1} \to \mathbb{R}$.
Tuto rovnici lze převést na systém n rovnic prvního řádu. Zavedeme substituci $x_1 = y,\ x_2 = y',\ x_3 = y'', ...,\ x_n = y^{(n-1)}$. Pak dostáváme soustavu n diferenciáních rovnic prvního řádu

$$\begin{cases} x_1' = x_2 \\ x_2' = x_3 \\ \vdots \\ x_n' = f(t, x_1, ..., x_n). \end{cases} \tag{1.5}$$

Každou diferenciální rovnici n-tého řádu lze převést na systém n rovnic prvního řádu. Stačí tedy dále uvažovat pouze soustavu rovnic prvního řádu.

Protože se budeme zabývat rychlostí konvergence Eulerovy metody, dále definujeme symbol velké $\mathcal{O}$. Písmeno $\mathcal{O}$ je z anglického slova *order* znamenající řád.

Definice 1.6. Nechť $\phi(h)$ je funkce definovaná v intervalu $(0, h_0]$ a $p \geq 1$. Řekneme, že funkce $\phi(h)$ je řádu $\mathcal{O}(h^p)$ a píšeme $\phi(h) = \mathcal{O}(h^p)$, jestliže existuje $C > 0$ takové, že pro všechna $0 < h \leq h_0$ platí $|\phi(h)| \leq Ch^p$.

Na závěr této části ještě uveďme jedno pomocné lemma, které později využijeme při odvozování odhadu tzv. celkové diskretizační chyby.

Lemma 1.1. *(Bernoulliho nerovnost)*([2], str. 270, Lemma 5.7)
$\forall m \in \mathbb{N}_0$ a $x \geq -1,\ x \in \mathbb{R}$ platí

$$0 \leq (1 + x)^m \leq e^{mx}$$

2 Explicitní Eulerova metoda

V této kapitole odvodíme algoritmus známý pod názvem explicitní Eulerova metoda. Odvození provedeme pro případ, kdy diferenciální rovnice v počáteční úloze (1.3) je skalární. Nedílnou součástí každého přibližného řešení je také analýza chyby. Budeme se věnovat jejich popisu a vlivu na konvergenci. Na závěr této kapitoly ilustrujeme použití Eulerovy metody na příkladech.

Přibližným řešením dále budeme označovat konečnou posloupnost hodnot $y_0, y_1, ..., y_N$, kde y_i je přibližná hodnota přesného řešení v bodě t_i, tj.

$$y_i \approx y(t_i).$$

Prozatím si zjednodušíme situaci a omezíme se na případ, kdy je interval $[t_0, b]$ rozdělen na síť stejně vzdálených bodů, pro které platí

$$t_i = t_0 + ih, \quad i = 0, ..., N.$$

To je zajištěno zvolením kladného přirozeného čísla $N > 0$. Pro velikost integračního kroku h, tak platí $h = (b - t_0)/N$. Rekurzivní předpis pro posloupnost bodů sítě pak je

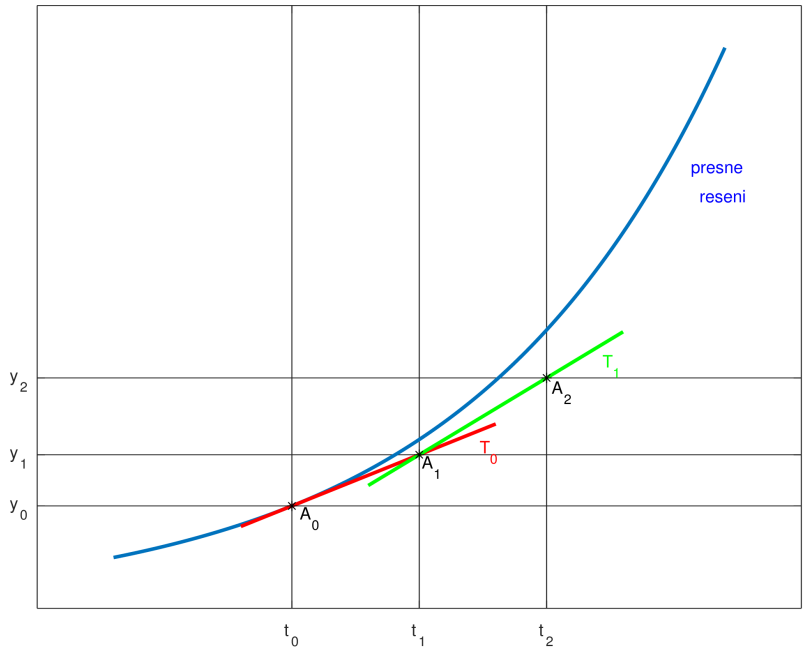
$$t_{i+1} = t_i + h, \quad i = 0, ..., N - 1. \tag{2.6}$$

Odvození

Předpis explicitní Eulerovy metody lze například odvodit pomocí geometrického významu počáteční úlohy. Jednoduše řečeno: začněte v bodě počáteční podmínky a malými kroky sledujte směrové pole dané diferenciální rovnice. Přesněji

řečeno: na začátku známe souřadnice bodu $A_0 = (t_0, y_0)$ a směrnici tečny ke grafu přesného řešení v tomto bodě, tedy $f(t_0, y_0)$. Po tečně se posuneme do bodu $A_1 = (t_1, y_1)$, jehož druhá souřadnice je první člen ve výše zmíněné posloupnosti. Označme úsečku spojující body A_0 a A_1 jako T_0.

Obrázek 2.1: grafické odvození explicitní Eulerovy metody



Pro její směrnici platí

$$\frac{y_1 - y_0}{t_1 - t_0} = f(t_0, y_0),$$

úpravou dostáváme

$$y_1 = y_0 + h \cdot f(t_0, y_0),$$

kde y_0 je hodnota z počáteční podmínky. Hodnota y_1 je $T_0(t_1)$. Ve druhém kroku bychom se chtěli posunout do bodu $A_2 = (t_2, y_2)$. Úsečku spojující body A_1 a A_2

označme T_1. Pro její směrnici platí

$$\frac{y_2 - y_1}{t_2 - t_1} = f(t_1, y_1).$$

Hodnota $f(t_1, y_1)$ je už pouze přibližnou hodnotou směrnice tečny ke grafu přesného řešení. Hodnota y_2 je opět $T_1(t_2)$. Tento postup je shrnut na Obrázku 2.1. Takto pokračujeme pro další A_i. Předpis Eulerovy metody je tedy rekurzivní vztah pro $(i+1)$-ní člen posloupnosti přibližného řešení, tj.

$$\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hf(t_i, y_i), \quad \forall i = 0, ..., N-1, \ t_i \in [t_0, b] \end{cases} \tag{2.7}$$

První člen, y_0, je daný z počáteční úlohy.

K rekurzivnímu předpisu lze dojít i jinými postupy. Uveďme alespoň jeden z nich.
O přesném řešení počáteční úlohy (1.3) předpokládáme, že má spojité derivace do řádu dva. Taylorův polynom takové funkce v okolí bodu $t = t_i$ je

$$y(t_i + h) = y(t_i) + y'(t_i)h + \frac{y''(t_i)}{2!}h^2 + \mathcal{O}(h^3), \tag{2.8}$$

Dosazením $y(t_i+h) = y(t_{i+1})$ a $y'(t_i) = f(t_i, y(t_i))$ a zanedbáním posledních dvou sčítanců, tak dostaneme:

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

Právě kvůli výše zmíněnému zanedbání rovnost neplatí přesně, již tedy nelze psát $y(t_{i+1})$, ale je nutné použít označení pro přibližnou hodnotu y_{i+1}.

Chyba a konvergence

Každá přibližná hodnota se od té přesné liší o nějakou chybu. V tomto případě chyby dělíme na diskretizační a zaokrouhlovací. Chybu diskretizační dále rozlišujeme celkovou a lokální.

Definice 2.7. Celkovou diskretizační chybou v bodě t_i rozumíme

$$e_i = y_i - y(t_i).$$

Při vyšetřování konvergence uvažujeme celkovou diskretizační chybu v pevném t_i jako funkci integračního kroku, přičemž při $h \to 0$ je $i \to \infty$.

Definice 2.8. Řekneme, že jednokroková metoda je konvergentní, jestliže pro každou počáteční úlohu (1.3), kde f splňuje Lipschitzovu podmínku a pro každé $t \in [t_0, b]$ platí

$$\lim_{\substack{h \to 0^+ \\ t_i = t}} y_i = y(t_i).$$

Věta 2.2. ([1], str.22, Věta 2.1) Nechť jsou splněny předpoklady 1. a 2. z Věty 1.1, nechť y_i je přibližné řešení úlohy (1.3) získané Eulerovou metodou (2.7) s integračním krokem h a nechť y je přesné řešení. Pak platí

$$|y_i - y(t_i)| \leq \omega(h) E_L(t_i - t_0), \qquad i = 0, ..., N,$$

kde N je největší přirozené číslo, pro něž je $t_N \in [t_0, b]$, funke E_L je definovaná jako

$$E_L(t) = \frac{e^{Lt} - 1}{L}, \ pro\ L > 0$$
$$= t, \ pro\ L = 0.$$

Funkce ω je modul spojitosti první derivace přesného řešení, tj.

$$\omega(h) = \sup_{\substack{|t - t^*| \leq h \\ t, t^* \in [t_0, b]}} |y'(t) - y'(t^*)|.$$

Důkaz Věty 2.2 lze najít v [1], strana 22 Věta 2.1. Věta 2.2 udává (formální) odhad celkové diskretizační chyby. Díky tomu, že E_L je omezená funkce, a modul spojitosti je funkce zprava spojitá konvergující pro $h \to 0$ k nule, z této věty konvergence vyplývá. Odhad chyby je však pouze formální, protože často nevíme

přesný tvar modulu spojitosti. Na přesné řešení lze klást přísnější předpoklady než jsou ve Větě 2.2. Odvození odhadu celkové chyby je tak jednodušší a navíc tento odhad nebude pouze formální. Bude z něj plynout nejen konvergence, ale také její rychlost. Na následujících řádcích odvození takového odhadu celkové diskretizační chyby provedeme.

Předpokládáme, že pravá strana diferenciální rovnice splňuje předpoklady Věty 1.1, a že přesné řešení má spojité derivace do řádu dva. Označme

$$M = \max_{t\in[t_0,b]} |y''(t)| = \max_{t\in[t_0,b]} \left|\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt}\right|. \tag{2.9}$$

Také platí

$$y(t_i) = y(t_{i-1}) + hf(t_{i-1}, y(t_{i-1})) + \frac{h^2}{2!}y''(\eta_{i-1}) \qquad \eta_{i-1} \in [t_{i-1}, t_i] \tag{2.10}$$

Chceme ukázat, že

$$\lim_{\substack{h\to\ 0^+ \\ t_i\ =\ t}} |e_i| = 0.$$

Všimneme si, že $f(t_{i-1}, y_{i-1}) = f(t_{i-1}, y(t_{i-1}) + e_{i-1})$. Pak lze rekurzivní vztah (2.7) psát ve tvaru $y_i = y_{i-1} + hf(t_{i-1}, y(t_{i-1}) + e_{i-1})$. Odečteme od něj (2.10):

$$y_i - y(t_i) = y_{i-1} - y(t_{i-1}) + h[f(t_{i-1}, y(t_{i-1}) + e_{i-1}) - f(t_{i-1}, y(t_{i-1}))] - \frac{h^2}{2!}y''(\eta_{i-1})$$

Použijeme Lipschitzovu podmínku, předpoklady o y'' a přidáme absolutní hodnoty:

$$|e_i| \le |e_{i-1}| + Lh|e_{i-1}| + \frac{Mh^2}{2} = |e_{i-1}|(1 + Lh) + \frac{Mh^2}{2}$$

Zároveň však $|e_{i-1}| \leq |e_{i-2}|(1+Lh) + \frac{Mh^2}{2}$ a tedy

$$\begin{aligned}
|e_i| &\leq \big[|e_{i-2}|(1+Lh) + \frac{Mh^2}{2}\big](1+Lh) + \frac{Mh^2}{2} \\
&= |e_{i-2}|(1+Lh^2) + \frac{Mh^2}{2}[(1+Lh^2)^0 + (1+Lh^2)^1] \\
&\vdots \\
&\leq |e_0|(1+Lh)^i + \frac{Mh^2}{2}[(1+Lh)^0 + ... + (1+Lh)^{i-1}]
\end{aligned}$$

Jenomže $|e_0| = |y_0 - y(t_0)| = 0$ a

$$\sum_{j=0}^{i-1}(1+Lh)^j = \frac{1-(1+Lh)^i}{-Lh} = \frac{1}{Lh}[(1+Lh)^i - 1],$$

tedy je třeba ukázat, že

$$|e_i| \leq \frac{Mh}{2L}[(1+Lh)^i - 1] \qquad i = 0, 1, ..., N.$$

Předchozí nerovnost dokážeme matematickou indukcí.
$i = 0$:

$$|e_0| = 0 \leq \frac{Mh}{2L}[(1+Lh)^0 - 1] = 0.$$

Předpokládáme, že nerovnost platí pro i a ukážeme, že platí také pro $i+1$. Výše jsme ukázali, že $|e_{i+1}| \leq |e_i|(1+Lh) + \frac{Mh^2}{2}$. Použijeme na $|e_i|$ indukční předpoklad a máme:

$$|e_{i+1}| \leq (1+Lh)\Big(\frac{Mh}{2L}[(1+Lh)^i - 1]\Big) + \frac{Mh^2}{2}$$

po roznásobení

$$|e_{i+1}| \leq \frac{Mh}{2L}[(1+Lh)^{i+1} - 1].$$

Tím jsme ukázali, že nerovnost platí také pro $(i+1)$-ní člen. Podle lemmatu 1.1 platí $0 \leq (1+Lh)^i \leq e^{Lih}$. Jelikož $t_i = t_0 + ih$, tak

$$|e_i| \leq \frac{Mh}{2L}(e^{L(t_i - t_0)} - 1). \tag{2.11}$$

Výraz na pravé straně nerovnosti je hledaný horní odhad celkové chyby. Ten při $h \to 0,\ t_i = t$ konverguje k nule, neboť $\frac{M}{2L}(e^{L(t_i - t_0)} - 1)$ je pro všechna $t_i \in [t_0, b]$ omezená. Tedy platí $|e_i| \to 0$.
Horní odhad (2.11) je příliš hrubý. Jeho hlavní význam spočívá v tom, že je lineární funkcí h, říkáme tedy že Eulerova metoda má lineární rychlost konvergence. Nebo také, že (teoretický) řád konvergence je jedna. Teoretický řád konvergence lze pozorovat při řešení konkrétního příkladu spočtením numerického řádu konvergence.

Označme p hledaný numerický řád konvergence. V případě Eulerovy metody očekáváme, že p s $h \to 0$ bude konvergovat k jedničce. Předpokládáme, že celková chyba v i-tém kroku je funkcí h_i a p v následujícím smyslu: $e_i = Ch_i^p,\ C > 0$. Pro poměr dvou po sobě jdoucích výpočtů tak dostáváme

$$\left(\frac{e_i}{e_{i+1}}\right) = \left(\frac{h_i}{h_{i+1}}\right)^p$$

$$ln\left(\frac{e_i}{e_{i+1}}\right) = p \cdot ln\left(\frac{h_i}{h_{i+1}}\right)$$

$$p = \frac{ln\left(\frac{e_i}{e_{i+1}}\right)}{ln\left(\frac{h_i}{h_{i+1}}\right)} \tag{2.12}$$

Poznámka: Výpočet numerického řádu konvergence podle vzorce (2.12) lze použít jen pro Eulerovu metodu s pevným integračním krokem.

Příklad 2.1. Uvažujme úlohu

$$\begin{cases} y' = y \cdot \cos t,\ t \in [0, 1] \\ y(0) = 1. \end{cases}$$

K nalezení Lipschitzovy konstanty použijeme Větu o střední hodnotě v proměnné y, když t je pevné. $\forall t \in [0,1],\ \exists\ \xi\ \in (y_1, y_2)$:

$$\frac{|f(t,y_2) - f(t,y_1)|}{|y_2 - y_1|} = \left|\frac{\partial f}{\partial y}(t,\xi)\right|$$

$$|f(t,y_2) - f(t,y_1)| = \cos t |y_2 - y_1| \leq 1 \cdot |y_2 - y_1|$$

Lipschitzova konstanta je $L = 1$. Přesným řešením je $y(t) = e^{sint}$. Hodnota přesného řešení v koncovém bodě intervalu je

$$y(1) = 2.3198.$$

Díky znalosti přesného řešení spočteme konstantu M.

$$M = \max_{t\in[0,1]} |e^{\sin t}(\cos^2(t) - \sin(t))|$$

Neboť $y''(t) = y'(t) \cdot cos(t) - y(t) \cdot sin(t) = y(t)(cos^2(t) - sin(t))$. Označme funkci uvnitř absolutní hodnty $w(t)$. Platí $w'(t) = e^{\sin t}\cos t(\cos^2 t - 3\sin t - 1)$, nulové body má $w'(t)$ v $t = 0$ a $t_k = \pi/2 + k\pi$, k je celé číslo. Platí $[0,1] \subset (0, \pi/2)$. Funkce $w'(t)$ je na $(0, \pi/2)$ záporná, tedy $w(t)$ je zde klesající. Pro krajní body intervalu platí $w(0) = 1$ a $w(1) = -1.2748$. Tedy lze tvrdit, že $M = 1.2748$.

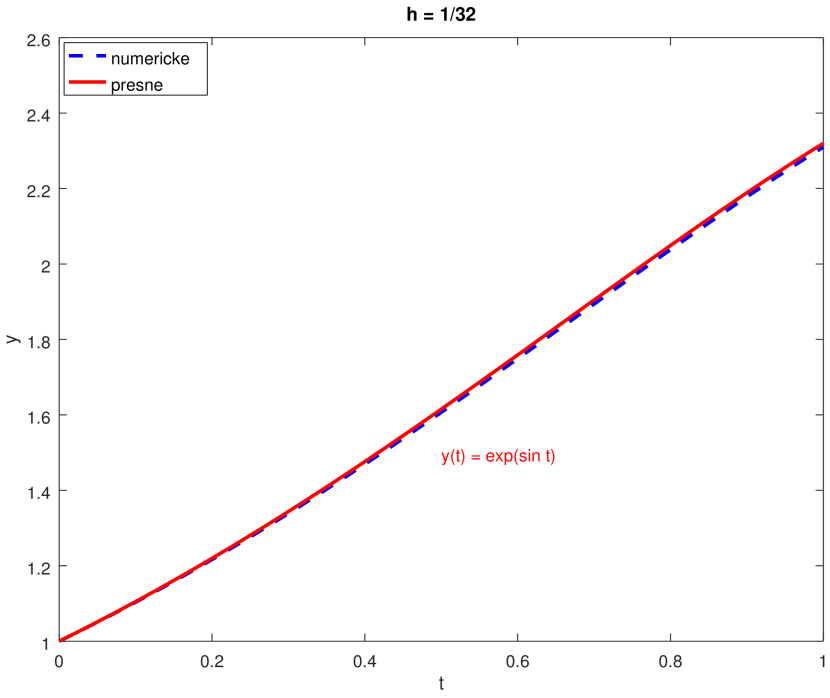
Budeme zkoumat přibližnou hodnotu v koncovém bodě intervalu, tj. pro $t = 1$,

Tabulka 2.1: Chyba v t = 1, numerický řád konvergence

h_i	1/2	1/4	1/8	1/16	1/32	1/ 64	1/128	1/256
y_i	2.1582	2.2398	2.2803	2.3002	2.3100	2.3149	2.3173	2.3186
e_i	0.1616	0.0800	0.0395	0.0196	0.0098	0.0049	0.0024	0.0012
$\hat{e}_i$	0.5476	0.2738	0.1369	0.0685	0.0342	0.0171	0.0086	0.0043
p	0	1.0150	1.0169	1.0118	1.0069	1.0037	1.0019	1.0010

s postupně se zmenšujícím krokem. V Tabulce 2.1 jsou shrnuté výsledky. Numerický řád konvergence p je vždy spočten podle vzorce (2.12) a je vidět, že podle očekávání se posloupnost hodnot p blíží k 1. K výpočtu $\hat{e}_i$ jsme použili vzorec pro odhad (2.11).

Obrázek 2.2: Graf přibližného a přesného řešení příkladu 2.1



Příklad 2.2. Uvažujme úlohu

$$\begin{cases} y' = \frac{y}{t} - \left(\frac{y}{t}\right)^2, \ t \in [1,2] \\ y(1) = 1. \end{cases}$$

Přesným řešením této počáteční úlohy je $y(t) = t/(1 + ln(t))$. Pro $t \in [1,2]$ hodnoty této funkce neopustí interval $[1,2]$. Pokud bychom chtěli nalézt Lipschitzovu konstantu analogicky jako v předchozím příkladě máme
$\forall t \in [1,2] \ \exists \xi \ \in (y_1, y_2) \subset [1,2]$:

$$\left|\frac{\partial f}{\partial y}(t,\xi)\right| = \left|\frac{1}{t} - \frac{2\xi}{t^2}\right| = \left|\frac{t - 2\xi}{t^2}\right|$$

Pro $t \in [1,2]$ a $\xi \in [1,2]$ bude jmenovatel výrazu v absolutní hodnotě vždy kladný a čitatel bude vždy záporný. Největší záporná hodnota nastane pro $\xi = 2$ a $t = 1$. Tedy $L = 3$ a platí pouze ve čtverci $D = \{(t,y) \in [1,2] \times [1,2]\}$. Na

základě znalosti přesného řešení také spočteme konstantu M:

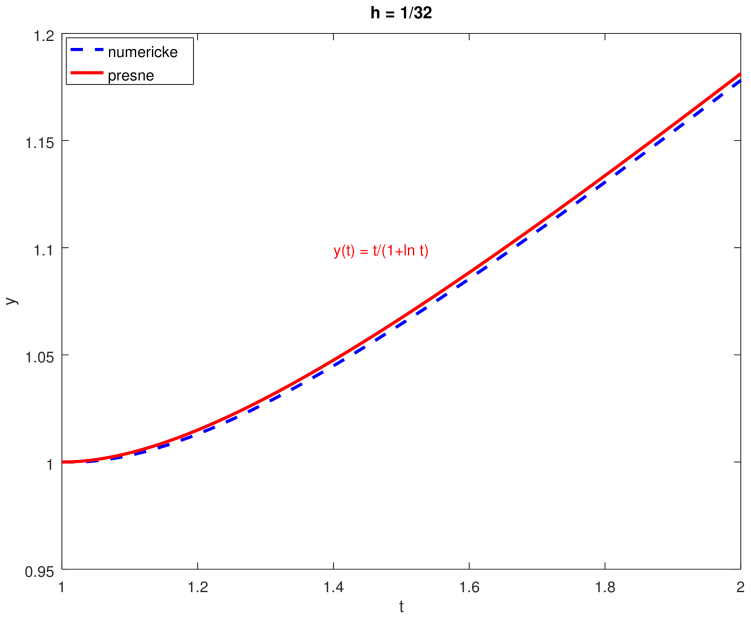
$$M = \max_{t\in[1,2]} \left| \frac{1 - \ln t}{t(1 + \ln t)^3} \right|.$$

Označme opět funkci v absolutní hodnotě jako $w(t)$. Pak

$$w'(t) = \frac{-5 + 2\ln t + \ln^2 t}{t^2(1 + \ln t)^4}$$

Položíme-li $w'(t) = 0$, tak najdeme nulové body v $t = e^{-\sqrt{6}-1}$ a $t = e^{\sqrt{6}-1}$. Platí $[1, 2] \subset (e^{-\sqrt{6}-1}, e^{\sqrt{6}-1})$. Funkce $w'(t)$ je na tomto intervalu záporná, tedy $w(t)$ je na intervalu $[1, 2]$ klesající, $w(1) = 1$ a $w(2) \approx 0.0316$, tedy $M = 1$.
Opět budeme zkoumat přibližnou hodnotu v koncovém bodě intervalu, tj. pro $t = 1$, s postupně se zmenšujícím krokem. Tabulka 2.2 je analogická Tabulce 2.1 z předchozího příkladu.

Obrázek 2.3: Graf přibližného a přesného řešení příkladu 2.2



Tabulka 2.2: Chyba v t = 2, numerický řád konvergence

h_i	1/2	1/4	1/8	1/16	1/32	1/ 64	1/128	1/256
y_i	1.1111	1.1518	1.1678	1.1748	1.1781	1.1797	1.1805	1.1808
e_i	0.0701	0.0295	0.0135	0.0064	0.0032	0.0016	0.0008	0.00004
$\hat{e}_i$	1.5905	0.7952	0.3976	0.1988	0.0994	0.0497	0.0249	0.0124
p	0	1.2510	1.1298	1.0639	1.0316	1.0157	1.0078	1.0039

Celková diskretizační chyba je výsledkem nahromadění tzv. lokálních diskretizačních chyb. Lokální diskretizační chyby se dopouštíme provedením jednoho kroku metody za předpokladu, že počítáme s přesnými údaji.

Definice 2.9. Lokální diskretizační chybu zavádíme vztahem

$$\tau_i = y(t_{i+1}) - (y(t_i) + hf(t_i, y(t_i))),$$

kde $y(t)$ je přesné řešení.

Pokud máme případ jedné rovnice v počáteční úloze (1.3), lze vyjádřit lokální diskretizační chybu v závislosti na h následujícím způsobem.
Podobně jako výše použijeme Taylorův polynom:

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2!}y''(\eta_i) \quad \eta_i \in (t_i, t_{i+1}).$$

Dosadíme do definice $|\tau_i|$

$$|\tau_i| = \frac{h^2}{2!}|y''(\eta_i)|.$$

Tato rovnost znamená, že lokální diskretizační chyba je úměrná druhé mocnině integračního kroku. To platí i při použití explicitní Eulerovy metody na systém, ale odvození je složitější.

Vliv zaokrouhlování

Nyní se budeme stručně věnovat vlivu zaokrouhlování na výpočet přibližného řešení. V důsledku toho, že aritmetické operace lze provádět jen s konečným

počtem čísel o konečném počtu cifer, tak místo y_i vlastně spočteme $\widetilde{y}_i$, které nesplňuje rekurzivní vztah (2.7) přesně. Blíže skutečnosti je zadání počáteční úlohy v podobně (2.13). Přibližné řešení je tedy zatíženo nejen chybou diskretizační, ale navíc také celkovou zaokrouhlovací chybou.

$$\begin{cases} \widetilde{y}_0 = y_0 \\ \widetilde{y}_i = \widetilde{y}_{i-1} + hf(t_{i-1}, \widetilde{y}_{i-1}) + \delta_{i-1} \end{cases} \tag{2.13}$$

Symbol δ_i označuje lokální zaokrouhlovací chybu. Lokální zaokrouhlovací chyba je ovlivněna především přesností s jakou spočítáme součet, součin a funkční hodnotu ve druhé rovnici. O lokální zaokrouhlovací chybě budeme dále předpokládat, že je v každém kroku omezená, tj. $\exists \delta > 0 : |\delta_i| \leq \delta$.

Celkovou chybu definujeme následovně: $r_i = \widetilde{y}_i - y(t_i)$. Analogicky jako u celkové diskretizační chyby budeme chtít najít horní odhad celkové chyby. Jak je vidět z rovnic (2.13), předpokládáme, že počáteční podmínka je zadána přesně, takže platí:

$$r_0 = \widetilde{y}_0 - y_0 = y_0 - y(t_0) = 0.$$

Odečteme druhou z rovnic (2.13) od (2.10)

$$\widetilde{y}_i - y(t_i) = \widetilde{y}_{i-1} - y(t_{i-1}) + h[f(t_{i-1}, \widetilde{y}_{i-1}) - f(t_{i-1}, y_{i-1})] + \delta_{i-1} - \frac{h^2}{2!} y''(\eta_{i-1}),$$

kde $\eta_{i-1} \in [t_{i-1}, t_i]$. Dále budeme předpokládat, že řešení splňuje předpoklady Věty 1.1. Analogicky jako u hledání odhadu celkové diskretizační chyby navíc předpokládáme splnění podmínky nutné k existenci konstanty M ze vztahu (2.9). Přidáním absolutní hodnoty a využitím předpokladu omezenosti δ_i přechozí rovnici převedeme na nerovnost

$$|r_i| \leq |r_{i-1}|(1 + Lh) + \delta + \frac{Mh^2}{2}.$$

Protože $|r_0| = |y_0 - y(t_0)|$ je podle předpokladu nula, použijeme stejný argument jako dříve v odvození odhadu celkové diskretizační chyby a dojdeme k závěru

$$|r_i| \leq \frac{1}{L}\left(\frac{\delta}{h} + \frac{Mh}{2}\right)(e^{L(t_i - t_0)}) \quad \forall i = 1, ..., N.$$

Nyní však dostáváme:

$$\lim_{\substack{h\to\ 0^+ \\ t_i\ =\ t}} \left(\frac{\delta}{h} + \frac{Mh}{2}\right) = \infty.$$

To znamená, že při zmenšování integračního kroku lze dospět k určité spodní hranici, kterou již většinou nemá smysl překračovat. Zaokrouhlovací chyba by převládla, a zvětšovala chybu celkovou. Tato spodní hranice závisí na dané numerické metodě a velikosti δ_i, říkáme jí mezní přesnost. Definujme $E(h) = \frac{Mh}{2} + \frac{\delta}{h}$. Potom položme $E'(h) = 0$, abychom našli stacionární bod $h = \sqrt{\frac{2\delta}{M}}$. Protože $E(h)$ je pro $h > 0$ konvexní funkcí, tak stacionární bod je bodem minima, tj. mezní přesnosti.

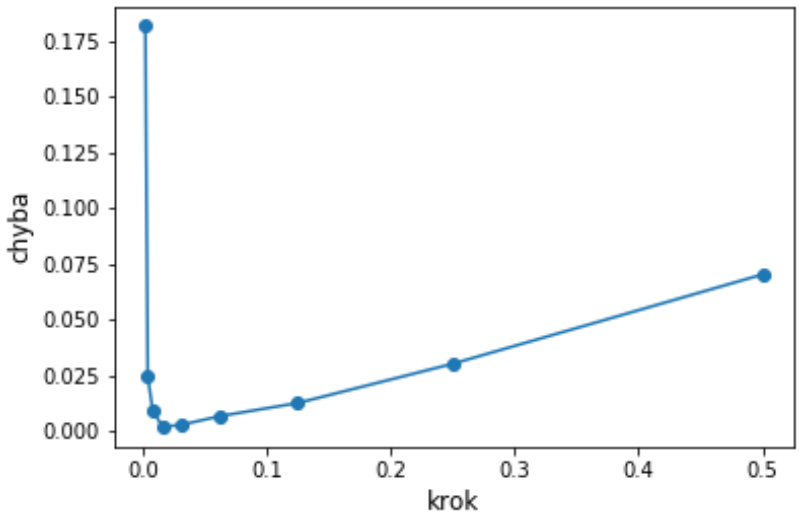
Příklad 2.3. Mějme počáteční úlohu z Příkladu 2.2. Spočteme hodnotu přibližného řešení v $t = 2$ a užijeme přitom integrační kroky $h_s = 1/2^s,\ s = 1, ..., 9$. Pro každý h_s spočteme celkovou chybu, tj. $r_s = |y_N - y(2)|$. Navíc budeme počítat s přesností *binary16*, kde každé číslo je v počítači reprezentováno pomocí 16 bitů. To je oproti běžné reprezentaci pomocí 64 bitů nepřesné, takže zaokrouhlovací chyba se projeví již na relativně velkých krocích. V Tabulce 2.3 jsou uvedené celkové chyby pro jednotlivé velikosti kroků. Pokud chceme vypočíst mezní přesnost, je třeba zjistit δ, tj. horní odhad lokální zaokrouhlovací chyby δ_i. Jako δ se volí tzv. strojové epsilon. To je rozdíl mezi jedničkou a dalším nejbližším číslem, které se dá v dané přesnosti reprezentovat na počítači. V přesnosti *binary*16 je $\delta = 9.7656 \cdot 10^{-4}$. Pak mezní přesnost vychází $h_M = \sqrt{\frac{2\delta}{M}} = 0.0442$. Při použití $h_6 = 0.0156 < h_M$ ještě zaokrouhlovací chyba nepřevládne, ale při použití $h_7 = 0.0078$ již ano. Mezní přesnost h_M tedy lze chápat jako odhad nejmenšího použitelného kroku.

Na Obrázku 2.4 je graf r_s jako funkce h_s. Většina hodnot je však nakupená v oblasti blízko nuly, protože zde dochází k nějvětším změnám. Graf se dá rozdělit na dvě části. V první části, než převáží zaokrouhlovací chyba, je celková chyba hlavně tvaru Ah, kde $A > 0$ je nějaká konstanta. Podobně ve druhé části, po

Tabulka 2.3: Celková chyba v závislosti na velikosti kroku

s	1	2	3	4	5	6	7	8	9
$1/2^s$	0.5	0.25	0.125	0.0625	0.0312	0.0156	0.0078	0.0039	0.002
e	0.0703	0.0303	0.0127	0.0068	0.0029	0.002	0.0098	0.0244	0.1816

Obrázek 2.4: Celková chyba v závisloti na h



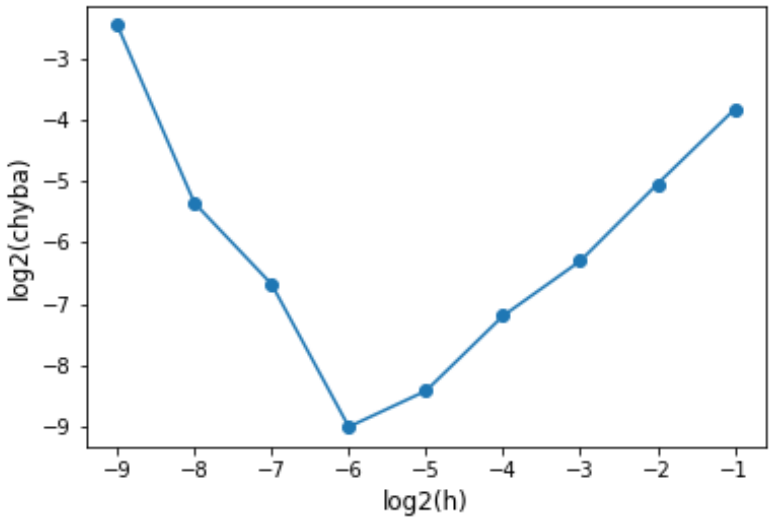
převážení zaokrouhlovací chyby, je celková chyba tvaru B/h, $B > 0$. To jsou obecně mocninné funkce $f(z) = z^p$, kde $|p| = 1$. Vezmeme-li jejich logaritmus, tak dostaneme rovnici dvou přímek. Základ logaritmu zvolíme dva, neboť

$$\log_2 h_s = \log_2 1/2^s$$
$$\log_2 h_s = -s.$$

Na Obrázku 2.5 v první části odpovídá směrnice grafu (tj. ≈ 1) řádu Eulerovy metody, který plyne z faktu, že celková diskretizační chyba je $\mathcal{O}(h)$. Podobně ve druhé části, kde převažuje chyba zaokrouhlovací, směrnice grafu (tj. ≈ -1) odpovídá řádu zaokrouhlovací chyby, která je $\mathcal{O}(\frac{1}{h})$.

Jak již bylo zmíněno, běžně se počítá s přesností 64 bitů, kde strojové epsilon

Obrázek 2.5: loglog graf, abs. hodnota směrnice je přibližně jedna



je dostatečně malá hodnota. Tedy se nestává, že bychom museli zvolit tak malé h, aby byla překročena mezní přesnost.

3 Implicitní Eulerova metoda

Rekurzivní předpis pro $(i+1)$-ní člen posloupnosti přibližných hodnot je v explicitní metodě vyjádřen pomocí již spočtených hodnot. V metodě implicitní je y_{i+1} vyjádřeno v obecně nelineární implicitní rovnici, kterou je třeba v každém kroce vyřešit pomocí jiné numerické metody (například Newtonova metoda a její modifikace). V této kapitole odvodíme předpis implicitní Eulerovy metody, které se někdy také říká zpětná Eulerova metoda, stručně popíšeme tvar diskretizačních chyb a uvedeme motivační příklad. Implicitní Eulerova metoda je někdy označována jako zpětná Eulerova metoda.

Uvažujme počáteční úlohu (1.3), případ skalární rovnice. Interval $[t_0, b]$ rozdělíme na síť stejně vzdálených bodů. Známe souřadnice bodu $A_0 = (t_0, y_0)$ z počáteční podmínky. Tečnu však tentokrát pouze symbolicky sestrojíme v bodě $A_1 = (t_1, y_1)$. Proč symbolicky? Neboť neznáme hodnotu y_1, tak ve skutečnosti tečnu v tomto bodě sestrojit nelze. Její směrnice, hodnoda $f(t_1, y_1)$, je neznámé číslo. Představme si však, že se po této symbolické tečně chceme posunout *zpět* do bodu $A_0 = (t_0, y_0)$. Odsud plyne důvod pro druhý název *zpětná* Eulerova metoda. Porovnáním dvou výrazů pro směrnici

$$\frac{y_1 - y_0}{h} = f(t_1, y_1)$$

a úpravou dostáváme

$$y_1 = y_0 + hf(t_1, y_1).$$

Nyní je třeba najít hodnotu y_1. Uvažujeme funkci $g(y_1) = y_1 - y_0 - hf(t_1, y_1)$ a hledáme řešení rovnice $g(y_1) = 0$ pomocí numerické metody pro řešení ne-

lineárních rovnic.
Rekurzivní předpis pro $(i+1)$-ní člen posloupnosti přibližných řešení je tvaru

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) \qquad i = 0, ..., N-1, \tag{3.14}$$

tedy předpis implicitní Eulerovy metody tvoří vztahy $t_{i+1} = t_i + h$ a (3.14). V každém kroku je třeba vyřešit rovnici $g(y_{i+1}) = 0$, což je výpočetně náročné. To je nevýhoda implicitní metody. O její výhodě se dozvíme v motivačním příkladě, který je uveden dále v této kapitole.

Rekurzivní předpis lze odvodit i pomocí Taylorova polynomu. Ve výrazu (2.8) provedeme substituci $t_i + h = t_{i+1} - h$, $t_i = t_{i+1}$.

$$y(t_{i+1} - h) = y(t_{i+1}) - y'(t_{i+1})h - \frac{y''(t_i)}{2!}h^2 + \mathcal{O}(h^3)$$

Dosadíme za $y(t_{i+1} - h) = y(t_i)$ a $y'(t_{i+1}) = f(t_{i+1}, y(t_{i+1}))$. Zanedbáním posledních dvou sčítanců a převedením součinu na druhou stranu dostáváme

$$y_i + hf(t_{i+1}, y_{i+1}) = y_{i+1}, \tag{3.15}$$

tedy stejný vzorec jako (3.14).

Nyní odvodíme závislost lokální diskretizační chyby na integračním kroku. Podle definice

$$|\tau_i| = |y(t_{i+1}) - y(t_i) - hf(t_{i+1}, y(t_{i+1}))|$$

K osamostatnění výrazu s h opět použijeme Taylorův polynom:

$$y(t_i) = y(t_{i+1}) - hf(t_{i+1}, y(t_{i+1})) - \frac{y''(\eta_i)}{2!}h^2$$

Zpětným dosazením dostáváme

$$|\tau_i| = \frac{h^2}{2!}|y''(\eta_i)|.$$

To je stejný výsledek jako u metody explicitní. Odvození závislosti celkové diskretizační chyby na velikosti h provádět nebudeme. Avšak z následující úvahy plyne, že také celková chyba je opět lineárně závislá na h.
Pokud celkem provedeme N kroků, kdy se v každém dopustíme lokální chyby řádu $\mathcal{O}(h^2)$, máme celkem chybu řádu $\mathcal{O}(h)$, neboť

$$NKh^2 = \frac{b - t_0}{h} Kh^2.$$

Zatím jsme viděli, že implicitní metoda má stejný řád konvergence jako metoda explicitní. Navíc má nevýhodu - je třeba najít kořen rovnice $g(y_{i+1}) = 0$ pro každé $0 \leq i \leq N - 1$. Mohlo by se tedy zdát, že je zbytečné ji zavádět. Existují však úlohy, které je lepší řešit implicitní metodou, protože se v těchto úlohách projeví velký nedostatek metody explicitní.

Příklad 3.4.

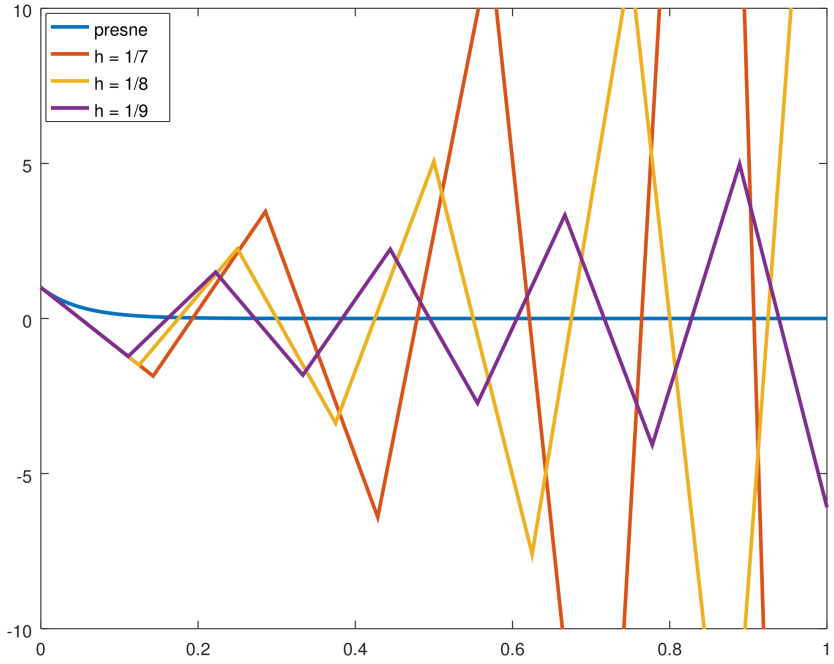
$$\begin{cases} y' = -20y \\ y(0) = 1, \ t \in [0, 1] \end{cases}$$

Přesným řešením je $y(t) = e^{-20t}$. Budeme tento příklad řešit explicitní Eulerovou metodou s velikostmi kroků postupně 1/7, 1/8 a 1/9. Na Obrázku 3.6 jsou vykresleny grafy přibližných řešení spolu s grafem přesného řešení.

Na první pohled je vidět, že je něco špatně. Explicitní metoda by si samozřejmě vedla lépe s menšími velikostmi kroků. Pojďme se však podívat, jak si s těmito relativně velkými h vede implicitní Eulerova metoda.

Rozdíly v chování těchto dvou metod jsou projevem vlastnosti, které se říká *stabilita*. Na základě srovnání grafů přibližných řešení na Obrázcích 3.6, kde je příklad řešen explicitní metodou, a 3.7, kde je řešen implicitní metodou, bychom mohli intuitivně usoudit, že implicitní Eulerova metoda je nějakým způsobem stabilnější než ta explicitní. Přesné definici pojmu stabilita se věnují následující

Obrázek 3.6: Porovnání přibližných řešení spočtené explicitní metodou, Příklad 3.4



odstavce.

Stabilita

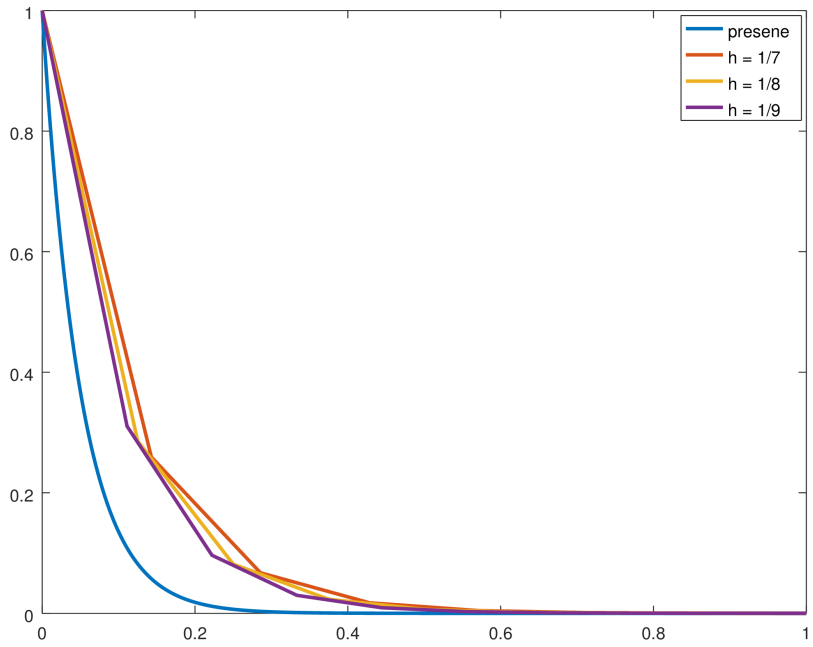
Existuje více druhů stability; dále se podíváme na absolutní stabilitu. Analýza této vlastnosti u numerické metody, je založená na analýze následující testovací diferenciální rovnice spolu s počáteční podmínkou:

$$y' = \lambda y, \;\; y(0) = y_0 \tag{3.16}$$

Tato počáteční úloha má přesné řešení $y(t) = y_0 e^{\lambda t}$, kde $\lambda \in \mathbb{C}$. Číslo λ lze psát v algebraickém tvaru $\lambda = a + ib$. Přesné řešení přepíšeme do tvaru

$$y(t) = y_0 e^{(a+ib)t} = y_0 e^{at}(\cos bt + i \sin bt).$$

Obrázek 3.7: Porovnání přibližných řešení spočtené implicitní metodou, Příklad 3.4



Pro $t \to \infty$ je sčítanec v závorce vždy omezený, díky vlastnostem funkcí sinus a cosinus. Výraz $y_0 e^{at}$ bude růst nad všechny meze pro $a > 0$, naopak bude konvergovat k nule pro $a = Re(\lambda) < 0$. Je žádoucí, aby přibližné řešení mělo stejnou vlastnost, což z grafů přibližných řešení na Obrázku 3.6 nepozorujeme.

Uvažujme diferenční rovnici

$$y_{i+1} = \xi(\lambda h) y_i \tag{3.17}$$

Jak explicitní, tak implicitní Eulerova metoda má tento tvar. Aplikujeme-li explicitní metodu na rovnici (3.16), dostáváme

$$y_{i+1} = y_i + h\lambda y_i.$$

Potom $\xi(\lambda h) = (1 + h\lambda)$. Podobně pro implicitní metodu dostáváme

$$y_{i+1} = y_i + h\lambda y_{i+1},$$

a $\xi(\lambda h) = \frac{1}{1-h\lambda}$.

Dále aplikujeme rovnici (3.17) rekurzivně. Pro explicitní metodu získáme:

$$y_{i+1} = y_i(1 + h\lambda) = y_{i-1}(1 + h\lambda)^2 = ... = y_0(1 + h\lambda)^i = y_0\xi^i(h\lambda),$$

podobně pro implicitní Eulerovu metodu:

$$y_{i+1} = \frac{y_i}{1 - h\lambda} = \frac{y_{i-1}}{(1 - h\lambda)^2} = ... = \frac{y_0}{(1 - h\lambda)^i} = y_0\xi^i(h\lambda).$$

Definice 3.10. Jednokroková metoda je absolutně stabilní, jestliže pro $\lambda h \in \mathbb{C}$ platí $|\xi(\lambda h)| < 1$. Oblastí absolutní stability rozumíme množinu $A = \{\lambda h \in \mathbb{C};\ |\xi(\lambda h)| < 1\}$.

Uvažujeme-li pevné reálné $\lambda < 0$, tak podle předchozí definice je oblastí absolutní stability explicitní Eulerovy metody interval $\lambda h \in (-2, 0)$. Pro krok h odsud plyne $-1 < 1 + h\lambda < 1 \iff -2 < h\lambda < 0 \iff$

$$h < \frac{2}{|\lambda|}. \tag{3.18}$$

Pro implicitní metodu dostáváme:

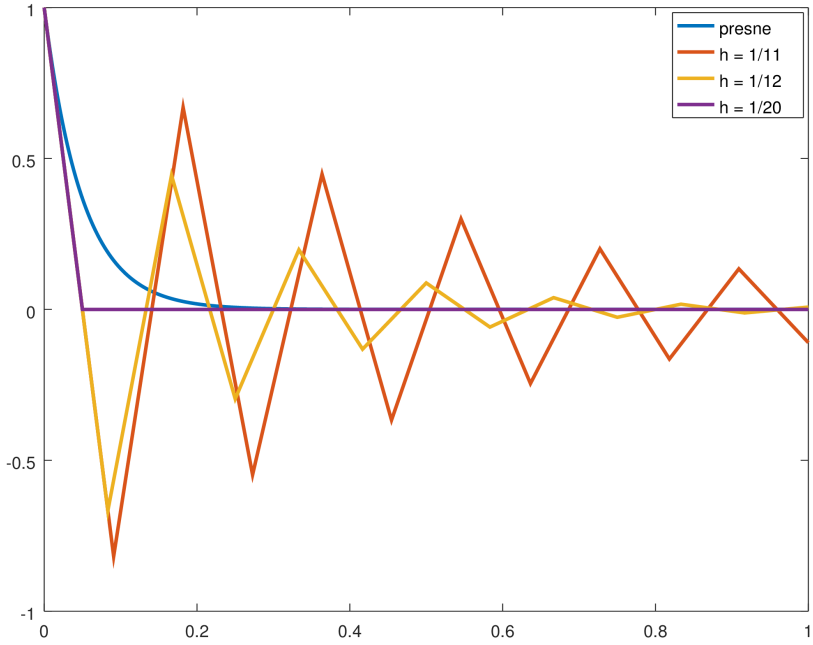
$$\begin{aligned} -1 < \frac{1}{1 - h\lambda} < 1 \\ -(1 - h\lambda) < 1 < 1 - h\lambda \\ 1 - h\lambda > -1 > -1 + h\lambda \\ 2 - h\lambda > 0 > h\lambda \end{aligned}$$

Protože $\lambda < 0$ a $h > 0$, tak vždy platí $0 > h\lambda$, ze stejného důvodu také platí $2 - h\lambda > 0$. Implicitní Eulerova metoda je absolutně stabilní při pevném $\lambda < 0$ pro všechna $h > 0$.

Pro takto zvolená λ a h má přibližné řešení testovací úlohy (3.16) stejnou vlastnost jako řešení přesné, tj. s $t \to \infty$ je $y_i \to 0$.

Vrátíme-li se nyní k příkladu 3.1, tak $|\lambda| = 20$ a tudíž explicitní Eulerova metoda bude pro tento příklad stabilní, zvolíme-li $h < \frac{1}{10}$. Velikosti kroků 1/7, 1/8 a 1/9 tuto podmínku nesplňují. Zkusíme použít kroky 1/11, 1/12 a 1/20, které splňují podmínku stability. Výsledky jsou uvedeny na Obrázku 3.8. Na první pohled je situace jiná než u Obrázku 3.6, kde se s $t_i \to 1$ bylo $y_i \to \infty$, nyní pozorujeme $y_i \to 0$, což je v souladu s definicí absolutní stability.

Obrázek 3.8: Explicitní metoda - stabilní



Tabulka 3.4: Srovnání implicitní a explicitní metody

i	h	e_i explicitní	e_i implicitní
1	1/30	0.1801	0.0964
2	1/40	0.1179	0.0766
3	1/50	0.0893	0.0632
4	1/60	0.0716	0.0540

Pojďme se podívat na srovnání implicitní a explicitní metody s velikostmi

kroků, pro které je explicitní metoda stabilní. Výsledky jsou uvedeny v Tabulce 3.4. Je zde uvedená maximální celková chyba e_i na intervalu $[0, 1]$ při použití příslušného h_i. Z Tabulky pozorujeme, že pro každou velikost kroku si implicitní metoda vedla lépe.

V této kapitole jsme ilustrovali koncept implicitní numerické metody na případě implicitní Eulerovy metody. Výhoda implicitní Eulerovy metody je velká oblast absolutní stability, je tedy možné užívat velkých integračních kroků. Přibližné výsledky přitom zůstávají rozumné. Používání relativně velkých integračních kroků u explicitní metody naproti tomu vede k velmi špatným přibližným hodnotám, protože má menší oblast absolutní stability.

4 Adaptivní krok

V této části opustíme myšlenku dělení intervalu $[t_0, b]$ na stejně vzdálené body. Toho dosáhneme tak, že v každé iteraci změníme velikost kroku. Důvodem pro změny velikosti kroku je splnění dvou požadavků. Zaprvé požadujeme, aby řešení bylo spočteno s předem zvolenou přesností. Zároveň chceme volit délku kroku největší možnou, aby objem výpočtů byl co nejmenší. Postup je zhruba následující. Při výpočtu vždy spočteme dvě různé přibližné hodnoty a porovnáme odhad chyby s tolerancí. Je-li odhad příliš velký, zmenšíme krok a zopakujeme výpočet. Novou přibližnou hodnotu přijmeme pouze pokud je odhad chyby menší než povolená tolerance.

Odhad globální chyby (2.11) uvedený dříve není příliš vhodný, protože je dost hrubý. Proto odvodíme jiný odhad globální chyby založený na asymptotickém vzorci pro chybu.

Věta 4.3. ([1], str. 27, Věta 2.4) *Asymptotický vzorec pro chybu*
Nechť jsou splněny předpoklady (a) a (b) Věty 1.1 a nechť navíc má pravá strana f dané diferenciální rovnice spojité první a druhé parciální derivace podle obou proměnných. Pak celková diskretizační chyba e_i přibližného řešení spočteného Eulerovou metodou se dá psát ve tvaru

$$e_i = e(t_i)h + \mathcal{O}(h^2),$$

kde funkce e je řešením diferenciální rovnice

$$e' = f_y(t, y(t))e - \frac{1}{2}y''(t)$$

s počáteční podmínkou $e(t_0) = 0$.

Na základě předchozího tvrzení odvodíme odhad celkové chyby pomocí metody polovičního kroku.
Zvolme pevný bod $t \in [t_0, b]$. Značením $y(h;t)$ pro tuto chvíli budeme rozumět přibližné řešení v tomto bodě spočtené s krokem h. Jsou-li splněny předpoklady předchozí věty, pak platí

$$y(h;t) = y(t) + e(t)h + \mathcal{O}(h^2).$$

Použijeme-li krok poloviční, dostáváme

$$y(h/2;t) = y(t) + \frac{1}{2}e(t)h + \mathcal{O}(h^2).$$

Odečtením předchozích rovnic:

$$y(h;t) - y(h/2;t) = \frac{1}{2}e(t)h + \mathcal{O}(h^2),$$

a tedy pro asymptotický odhad celkové diskretizační chyby platí

$$e(t)h = 2[y(h;t) - y(h/2;t)] + \mathcal{O}(h^2). \tag{4.19}$$

Dále ukážeme, jak odhad (4.19) použijeme při výpočtu.
Je dána počáteční úloha $y' = f(t,y), y(t_0) = y_0$, startovní velikost kroku h_0 a tolerance ϵ. V i-tém kroce algoritmu z trojice t_i, y_i, h_i spočítáme dvě různé hodnoty y_{i+1} spočtené s různým h, pojmenujme je A_1 a A_2.
A_1: spočteno pomocí jedné iterace explicitní Eulerovy metody s krokem h_i,

$$A_1 = y_i + h_i f(t_i, y_i).$$

A_2: spočteno pomocí dvou iterací, každá provedena s krokem $h_i/2$,

$$A_2 = y_i + \frac{h_i}{2}f(t_i, y_i) + \frac{h_i}{2}f\Big(t_i + \frac{h_i}{2}, y_i + \frac{h_i}{2}f(t_i, y_i)\Big).$$

Provedeme tedy dva poloviční kroky takto: $[t_i, y_i] \to [t_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}] \to [t_{i+1}, A_2]$, kde

$$t_{i+\frac{1}{2}} = t_i + \frac{h_i}{2}$$

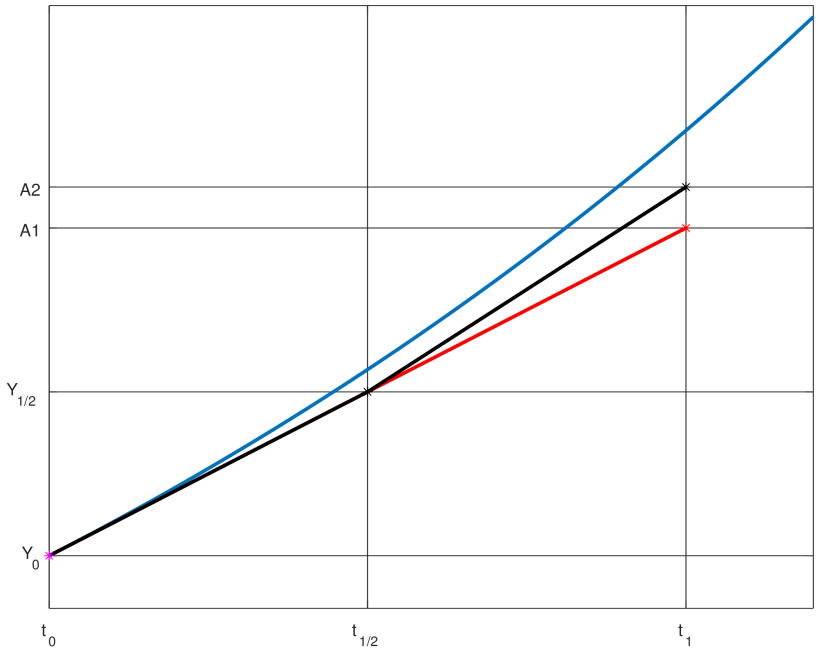
a

$$y_{i+\frac{1}{2}} = y_i + \frac{h_i}{2} f(t_i, y_i)$$

Na obrázku (4.9) jsou znázorněny hodnoty A_1 a A_2, modrá křivka je graf přesného řešení, červená úsečka znázorňuje výpočet hodnoty A_1 s krokem h a černá úsečka znázorňuje výpočet hodnoty A_2 s krokem $h/2$.

Shodně s dřívějším značením je $y(h;t) = A_1$ a $y(h/2;t) = A_2$. Zanedbáním

Obrázek 4.9: Srovnání jednoho celého a dvou poloviních kroků



posledního sčítance ve vztahu (4.19) tak dostaneme následující pravidlo: je-li

$$\hat{e}_i = 2|A1 - A2| > \epsilon, \tag{4.20}$$

zmenšíme krok h_i na

$$h_i' = 0.9 \cdot h_i \cdot \min\left(\max\left(\frac{\epsilon}{\hat{e}_i}, 0.3\right), 2\right) \tag{4.21}$$

a zopakujeme výpočet hodnot A_1, A_2 a $\hat{e}_i$ s tímto novým krokem, dokud $\hat{e}_i \leq \epsilon$. V tomto případě položíme $t_{i+1} = t_i + h_i$, $y_{i+1} = A2$ a h_{i+1} zvolíme podle (4.21).

Poznámky:

Odhad (4.19) je odhadem celkové chyby. Je zkonstruovaný tak, že nejprve všechno spočteme s krokem h, tím získáme jednu posloupnost přibližných řešení. Pak s krokem $h/2$, tím získáme jinou posloupnost. Rozdíl těchto dvou posloupností je odhad celkové chyby v příslušném t_i. Při použití metody adaptivního kroku však takto nepostupujeme, místo toho v každém bodě vždy spočteme dvě přibližné hodnoty jednou pomocí h a jednou pomocí $h/2$. V další iteraci už počítáme většinou s jiným h. Z tohoto důvodu má používaný odhad (4.21) lokální charakter a někdy se v litearatuře uvádí jako odhad lokální diskretizační chyby, ačkoli vychází z odhadu celkové chyby.

Vzorců jako je (4.21) existuje více. Mají společné to, že pokud je momentální odhad lokální diskretizační chyby menší než tolerance ϵ, tak může být krok zvětšen, protože $\frac{\epsilon}{\hat{e}_i} > 1$. Zároveň ho ale nechceme zvětšit příliš, tedy volíme $\min(\frac{\epsilon}{\hat{e}_i}, q_{max})$, kde q_{max} je maximální zvětšení kroku v případě dodržení tolerance pro chybu. Ve vzorci (4.21) je $q_{max} = 2$. Pokud tolerance splněna nebyla, je třeba krok zmenšit, ale opět ho nechceme zmenšit příliš. Vybíráme proto $\max(\frac{\epsilon}{\hat{e}_i}, q_{min})$, kde q_{min} je minimální zmenšení kroku při nedodržení zadané tolerance. Ve výrazu (4.21) je $q_{min} = 0.3$. Pokud je $\frac{\epsilon}{\hat{e}_i} \approx 1$, tak je lepší krok trochu zmenšit, proto přenásobení číslem 0.9. Místo zlomku $\frac{\epsilon}{\hat{e}_i}$ se také často používá jeho druhá odmocnina. Důvodem je lokální charakter odhadu $\hat{e}_i$ vysvětlen v předchozím odstavci a fakt, že lokální diskretizační chyba u Eulerovy metody je $\mathcal{O}(h^2)$, takže chceme-li z ní vyjádřit h, je třeba vzít její odmocninu.

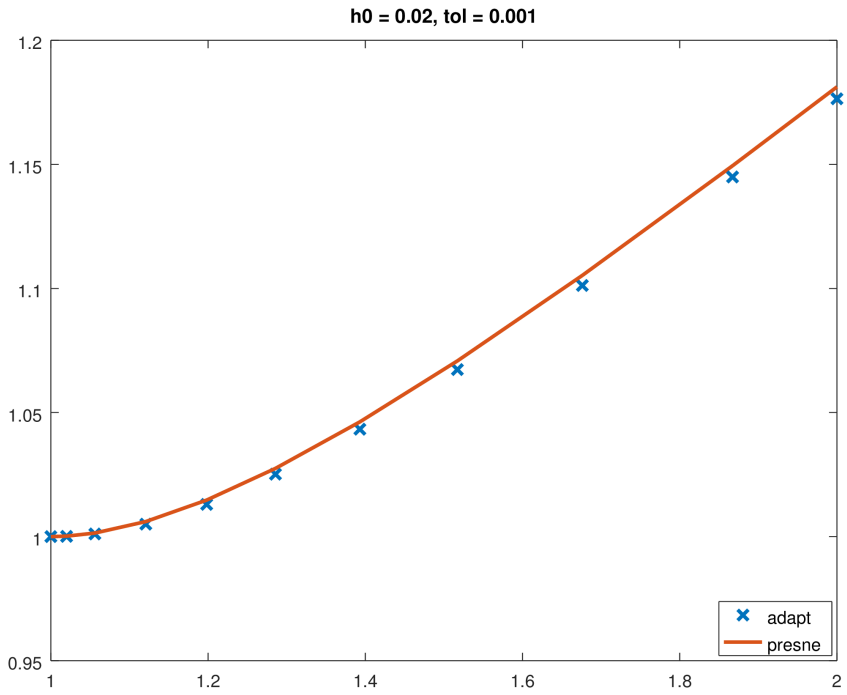
Příklad 4.5. Mějme opět počáteční úlohu z Příkladu 2.2. Ukážeme srovnání

Tabulka 4.5: Srovnání odhadů chyb

h_i	1/2	1/4	1/8	1/16	1/32	1/ 64	1/128	1/256
y_i	1.1111	1.1518	1.1678	1.1748	1.1781	1.1797	1.1808	1.1808
e_i	0.0701	0.0295	0.0135	0.0064	0.0032	0.0016	0.0008	0.0004
$\hat{e}_i$(4.19)	0.0813	0.0320	0.0140	0.0066	0.0032	0.0016	0.0008	0.0004
$\hat{e}_i$(2.11)	1.5905	0.7952	0.3976	0.1988	0.0.994	0.0497	0.0249	0.0124

odhadů celkové diskretizační chyby pomocí vzorců (2.11) a (4.19). Připomeňme, že přesná hodnota v $t = 2$ je 1.1812. V Tabulce 4.5 jsou výsledky. Je vidět, že při každé velikosti h je odhad chyby (4.19) v koncovém bodě intervalu menší než odhad (2.11).

Obrázek 4.10: Srovnání grafů přesného a přibližného řešení spočteného pomocí metody adaptivního kroku



Příklad 4.6. V tomto příkladě ukážeme, jak adaptivní Eulerova metoda volí

malý krok v oblastech, kde je řešení proměnlivé, a velký krok jinde.

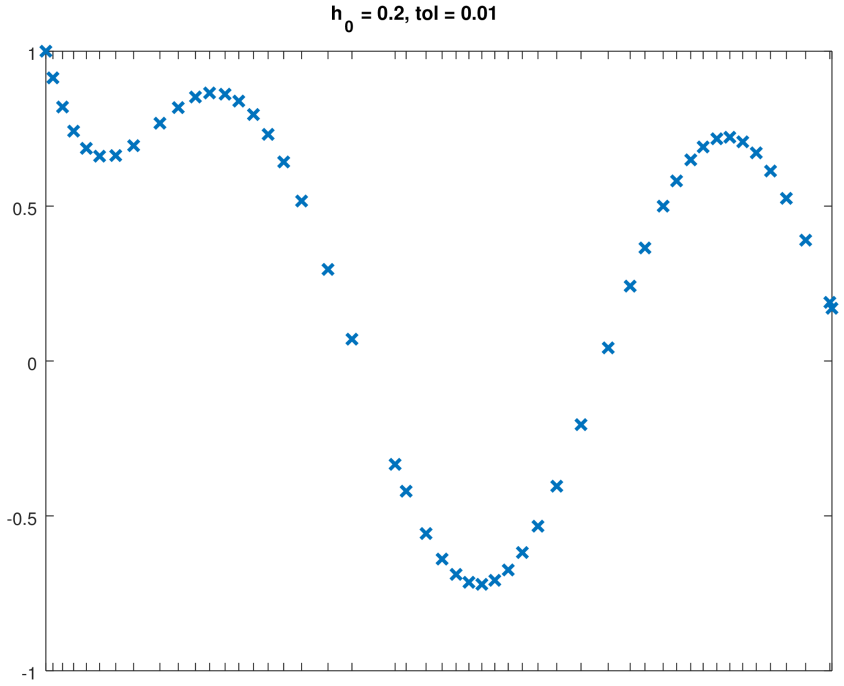
$$\begin{cases} y' = \sin t - y \\ y(0) = 1, \ t \in [0, 10] \end{cases}$$

Nejprve ověříme existenci a jednoznačnost řešení $\forall t \in [0, 10], \ \forall y_1, y_2 \ \exists \xi \in (y_1, y_2)$:

$$\frac{\partial f}{\partial y} = -1,$$

tedy $L = 1$ a existuje jediné řešení počáteční úlohy. Na Obrázku 4.11 je vidět

Obrázek 4.11: Efekt proměnlivosti kroku, příklad 4.6



proměnlivost délky kroku, protože přibližné hodnoty nejsou od sebe stejně vzdáleny. Příklad byl spočten s počátečním krokem $h_0 = 0.2$ a tolerancí $\epsilon = 0.01$.

Poznámka: Na závěr této části uveďme, že v literatuře, zabývající se adaptivním krokem, se jako další člen posloupnosti $\{y_i\}$ bere hodnota, která je jistým způsobem upravená na základě znalosti odhadu chyby. Této úpravě se říká lokální neboli Richardsonova extrapolace. Takto zvolené y_{i+1} však není vždy přesnější.

5 Systém rovnic

Uvažujme nyní případ, kdy počáteční úloha (1.3) je systém n rovnic prvního řádu (1). Definice diskretizačních chyb a konvergence zůstávají stejné jako ve skalárním případě.

Přibližným řešením budeme označovat konečnou posloupnost vektorů

$$\mathbf{Y}_i = \begin{bmatrix} y_i^1 \\ \vdots \\ y_i^n \end{bmatrix},$$

kde y_i^j je přibližná hodnota j-té složky přesného řešení v bodě t_i, tj. $y_i^j \approx y^j(t_i)$. Dále zaveďme označení

$$\mathbf{F}_i = \begin{bmatrix} f^1(t_i, \mathbf{Y}_i) \\ \vdots \\ f^n(t_i, \mathbf{Y}_i) \end{bmatrix}$$

Pro každou z n složek je rekurzivní předpis Eulerovy metody

$$\mathbf{Y}_{i+1} = \mathbf{Y}_i + h\mathbf{F}_i \qquad \forall i = 0, ..., N-1,$$

po složkách pak

$$y_{i+1}^j = y_i^j + h f^j(t_i, y_i^1, ..., y_i^n) \qquad 1 \leq j \leq n,\ 0 \leq i \leq N-1.$$

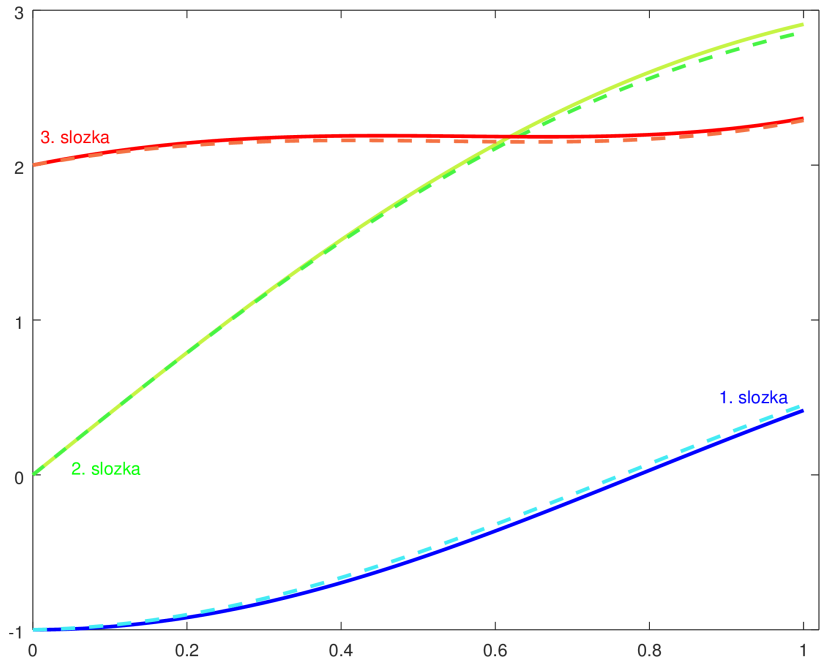
Příklad 5.7. Mějme počáteční úlohu

$$\begin{cases} y_1'(t) = \quad 2y_2(t) \quad - 4t \\ y_2'(t) = -y_1(t) \quad + y_3(t) - e^t + 2 \\ y_3'(t) = y_1(t) - 2y_2(t) + y_3(t) + 4t \end{cases}$$

$\mathbf{y}(0) = (-1, 0, 2)^T$ $t \in [0, 1]$.

Přesným řešením je $\mathbf{y}(t) = (-cos(2t), sin(2t) + 2t, cos(2t) + e^t)$. Použijeme ten-

Obrázek 5.12: Porovnání s přesným řešením, h = 1/64. Přesné řešení je plnou čarou.



tokrát implicitní metodu pro výpočet přibližné hodnoty v koncovém bodě s $h_i = 1/2^i, i = 1, .., 8$. Také opět ověříme numerický řád konvergence podle vzorce (2.12). Výpočty porovnáme při použití Euklidovy a maximové normy (tj. $\|x\|_\infty = max(|x_1|, ..., |x_n|)$).

Tabulka 5.6: Srovnání norem

h_i	1/2	1/4	1/8	1/16	1/32	1/ 64	1/128	1/256
$\|e_i\|_2$	1.2861	0.7278	0.4236	0.2323	0.1220	0.0626	0.0317	0.0159
p_2	0	0.8214	0.7808	0.8667	0.9290	0.9637	0.9817	0.9908
$\|e_i\|_\infty$	0.9217	0.5883	0.3504	0.1902	0.0988	0.0503	0.0254	0.0127
p_∞	0	0.6476	0.7477	0.8814	0.9452	0.9742	0.9876	0.9939

Označením p_2 rozumíme numerický řád konvergence vypočtený s hodnotami $\|e_i\|_2$, analogicky pro p_∞. Pro úplnost jsou v Tabulce 5.7 uvedeny přibližné hodnoty v $t = 1$ pro vybrané kroky. Přesná hodnota je $\mathbf{y}(1) = (0.4161, 2.9093, 2.3021)^T$.

Tabulka 5.7: Přibližné hodnoty v t = 1

i	h_i	$\mathbf{Y_i}^T$		
1	1/2	1.3378	2.0878	2.6622
2	1/4	0.84431	2.32096	2.31618
3	1/8	0.65044	2.55893	2.25985
4	1/16	0.54406	2.71910	2.26434
5	1/32	0.48369	2.81052	2.27832
6	1/64	0.45093	2.85902	2.28889
7	1/128	0.43381	2.88394	2.29517
8	1/256	0.42505	2.89656	2.29856

Obrázku 5.12 odpovídá řádek 6 Tabulky 5.7. Přerušovanou čarou je znázorněn graf přibližného řešení a plnou čarou graf přesného řešení. Lze vidět, že přibližné řešení s přesným souhlasí velmi dobře ve třetí složce, ale v první a ve druhé se ke konci intervalu poděkud liší. To potvrdí výpočet celkové chyby v $t = 1$: $|\mathbf{e}| = (0.034787, 0.050280, 0.013241)^T$, její třetí složka je skutečně nejmenší. Příklad byl spočten pomocí implicitní Eulerovy metody, která pro řešení implicitní rovnice používá upravenou Newtonovu metodu spolu s numerickým výpočtem Jacobiho matice. Viz Kapitola 8 Kódy.

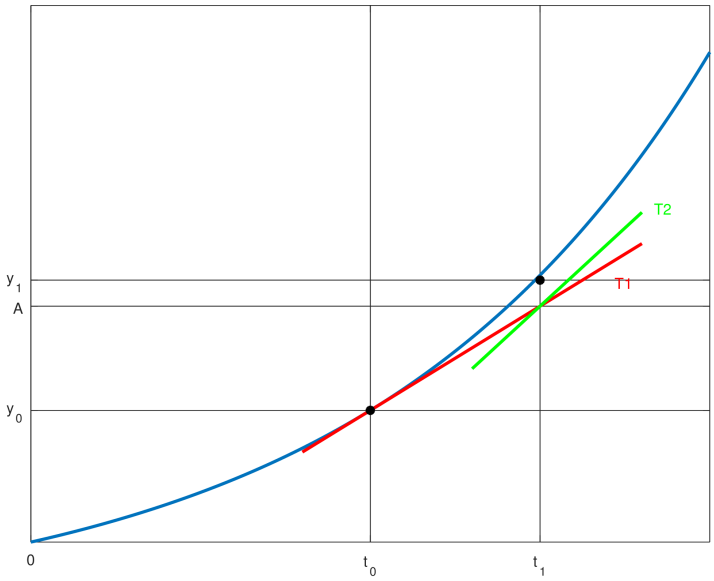
6 Modifikace Eulerovy metody

V této části uvedeme dvě úpravy Eulerovy metody. Konkrétně Heunovu metodu a Euler-Cromerovu metodu. Možných úprav však existuje více.

Heunova metoda

Heunova metoda vychází z metody Eulerovy, ale bere v potaz dodatečnou informaci, která přibližnou hodnotu y_i značně zlepší. Dále popíšeme odvození jejího předpisu, které je znázorněno na Obrázku 6.13.

Obrázek 6.13: Odvození Heunovy metody



Sledujme tečnu ke grafu přesného řešení z bodu $[t_0, y_0]$. Označme ji jako *T1*. Sledujeme tuto lineární funkci do $t_1 = t_0 + h$, kde zaznamenáme hodnotu A. Kdybychom použili Eulerovu metodu, tak bychom za y_1 přijali A. Nicméně nyní místo toho vyčíslíme $f(t_1, A)$, kde $f(t, y)$ je pravá strana diferenciální rovnice z počáteční úlohy (1.3). Tato hodnota je směrnicí přímky, kterou označíme *T2*. Jelikož navíc víme, že na ní leží bod (t_1, A), tak lze najít její směrnicový tvar. Přibližná hodnota y_1 získaná pomocí Heunovy metody je pak tvaru $y_1 = y_0 + h \cdot \bar{S}$. To odpovídá schematu metody Eulerovy. Změna je v hodnotě $\bar{S}$, což není tentokrát směrnice $f(t_0, y_0)$, ale průměr směrnice $f(t_0, y_0)$ přímky *T1* a $f(t_1, A) = f(t_1, y_0 + h \cdot f(t_0, y_0))$ přímky *T2*. Takto spočtená přibližná hodnota je blíže přesnému řešení než hodnota spočtená explicitní Eulerovou metodou. Řád konvergence Heunovy metody je dva. To znamená, že celková diskretizační chyba je $\mathcal{O}(h^2)$. Připomeňme, že explicitní Eulerova metoda je řádu jedna. Analogicky jako v předchozích případech je rekurzivním předpisem zobecnění na index $i = 1, .., N - 1$, tedy

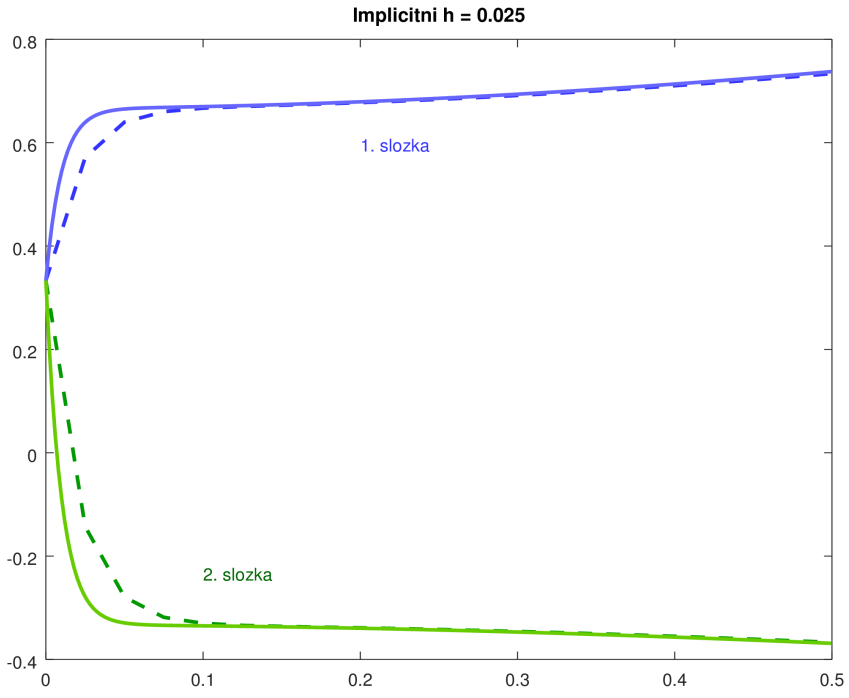
$$y_{i+1} = y_i + h \cdot \frac{f(t_i, y_i) + f(t_{i+1}, y_i + h \cdot f(t_i, y_i))}{2}.$$

Příklad 6.8. Uvažujme následující počáteční úlohu, systém dvou rovnic.

$$\begin{cases} u_1' = 32u_1 + 66u_2 + \frac{2}{3}t + \frac{2}{3}, \ u_1(0) = \frac{1}{3} \\ u_2' = -66u_1 - 133u_2 - \frac{1}{3}t - \frac{1}{3}, \ u_2(0) = \frac{1}{3} \end{cases}$$

$t \in [0, 0.5]$. Přesným řešením je $u_1(t) = \frac{2}{3}t - \frac{2}{3}e^{-t} - \frac{1}{3}e^{-100t}$, $u_2(t) = -\frac{1}{3}t - \frac{1}{3}e^{-t} + \frac{2}{3}e^{-100t}$. Příklad spočítáme pomocí Heunovy a implicitní Eulerovy metody. Nejdříve s krokem $h_1 = 0.025$ a pak s $h_2 = 0.00625$. Na Obrázku 6.14 je srovnání grafů přibližného a přesného řešení spočteného implicitní Eulerovou metodou. Tento systém diferenciálních rovnic je podobný Příkladu 3.4 v tom smyslu, že explicitní metody budou dobře fungovat jen s malými kroky. Na Obrázku 6.15 je srovnání prvních složek přesného a přibližného řešení spočteného Heunovou metodou, které ovšem diverguje. Nyní použijeme menší velikost kroku $h = 0.00625$. Srovnání přibližných výsledků je vidět na Obrázku 6.16. Spočteme-li maximální

Obrázek 6.14: Implicitní metoda, přesné řešení plnou čarou

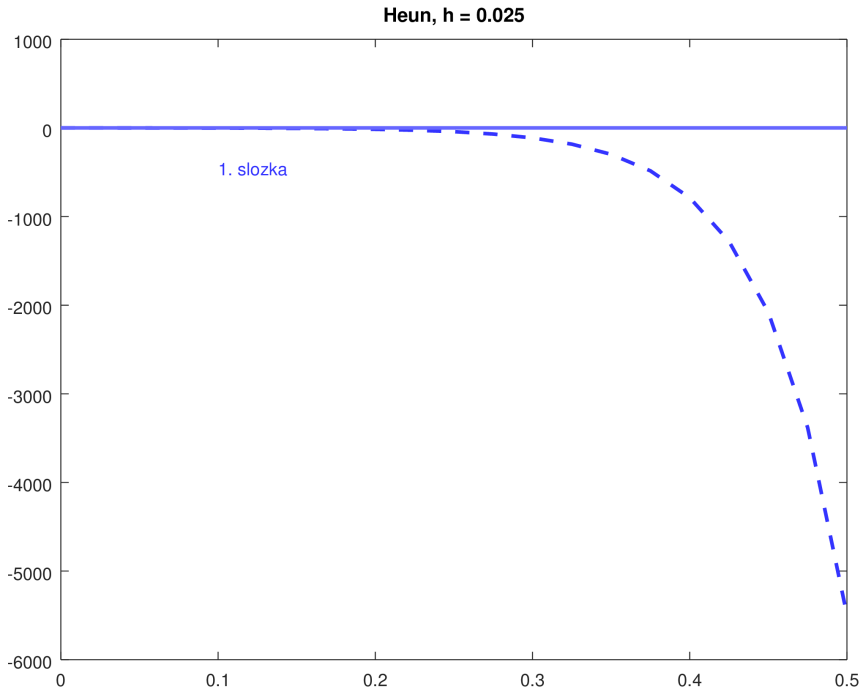


celkovou chybu (tj. největší rozdíl mezi přibližnou a přesnou hodnotou) na intervalu $t \in [0, 0.5]$, tak pro implicitní Eulerovu metodu dostáváme 0.092232 a pro Heunovu metodu 0.038751.

Euler-Cromerova metoda

V této části uvedeme motivační příklad pro zavedení Euler-Cromerovy metody, které se někdy říká Eulerova metoda semi-implicitní. Má lineární rychlost konvergence a také lokální chyba je řádu $\mathcal{O}(h^2)$.
Motivačním příkladem budiž matematické kyvadlo. To je model kyvadla skutečného. Uvažujeme hmotný bod, který je zavěšen na pevném drátku zanedbatelné hmostnosti. Bod při pohybu opisuje oblouk, jedná se tedy pouze o pohyb ve dvou dimenzích. Zanedbáváme tření a žádná energie není v průběhu pohybu dodávána.

Obrázek 6.15: Heunova metoda, pouze 1. složka



Hmotný bod má hmotnost m a drátek má délku l. Úhel θ značí vychýlení ze svislé polohy. Předpokládáme, že jediné síly působící na kuličku jsou gravitace a napětí v drátku. Výsledná síla je

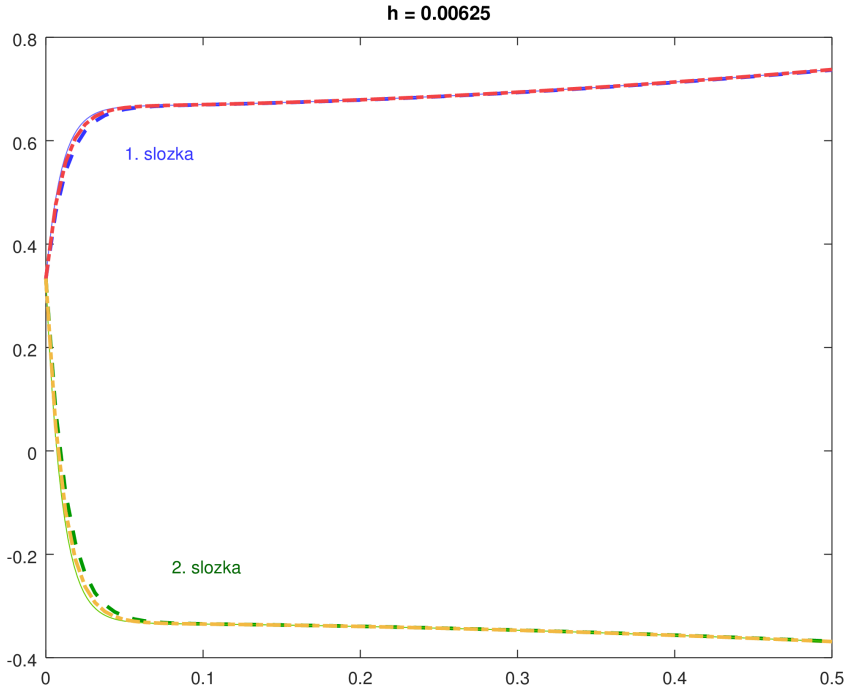
$$F_\theta = -mg \sin\theta$$

Znaménko minus značí, že síla má vždy opačný směr než je vychýlení ze svislé polohy. Podle Newtonova druhého pohybového zákona však zároveň $F_\theta = ma = ms''(t)$, kde $s(t) = l\theta(t)$ je vychýlení v čase. Porovnáním dostáváme

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l} \sin\theta.$$

Pokud uvažujeme jen dostatečně malé výchylky, tak $\sin\theta \approx \theta$ a tudíž dostáváme $\theta''(t) = -\frac{g}{l}\theta$. To je lineární rovnice, její přesné řešení v obecném tvaru je $\theta(t) = c_1 \cos\sqrt{\frac{g}{l}}t + c_2 \sin\sqrt{\frac{g}{l}}t$. Protože budeme příklad řešit Eulerovou metodou, je třeba si rovnici převést na systém dvou rovnic prvního řádu. Zaveďme značení $\theta(t) = x_1$

Obrázek 6.16: Srovnání Heunovy a implicitní Eulerovy metody s přesným řešením



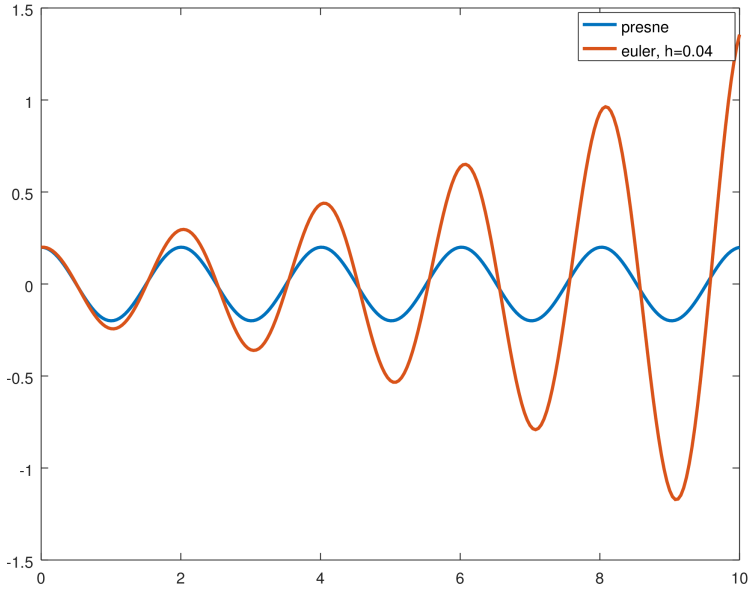
úhel vychýlení vůči svislé poloze a $\theta'(t) = x_2$ rychlost.

$$\begin{cases} x_1' = x_2 \\ x_2' = -\frac{g}{l} x_1 \end{cases}$$

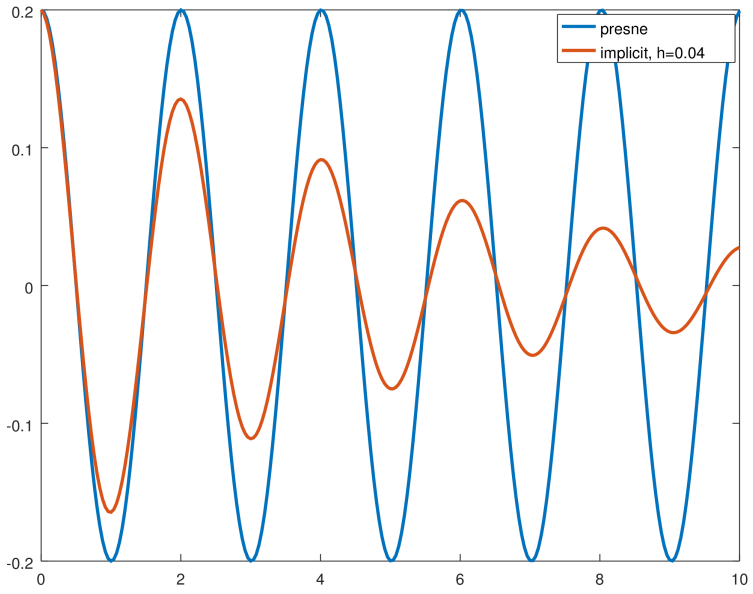
Budeme řešit počáteční úlohu s podmínkou $x_1(0) = 0.2$ (počáteční výchylka), $x_2(0) = 0$ (počáteční rychlost). Přesným řešením je $\theta(t) = 0.2 \cos \sqrt{\frac{g}{l}} t$. Délku drátku uvažujeme $l = 1$. Nejprve zkusíme najít přibližné řešení Eulerovou metodou s krokem $h = 0.04$.

Jak je vidět na Obrázku 6.17, kde jsme příklad řešili explicitní Eulerovou metodou, tak amplitudy přibližného řešení s časem rostou, zatímco u řešení přesného zůstávají konstantní. Pokud zmenšíme krok, tak na stejném intervalu sice dosáhneme lepších hodnot, ale na větším časovém intervalu je vidět, že jsme pouze zpomalili rychlost narůstání chyb. Pokud použijeme implicitní metodu, je chyba stejná, i když v opačném směru: amplitudy se tentokrát zmenšují, viz

Obrázek 6.17: Explicitní metoda



Obrázek 6.18: Implicitní metoda



Obrázek 6.18. Příklad matematického kyvadla je zjednodušením fyzické reality. Zjednodušení například v tom, že neuvažujeme tření a odpor vzduchu, tedy z modelu se žádná energie neztrácí. Zároveň žádnou energii nedodáváme, takže proto se amplitudy přesného řešení nemění. V přibližných řešeních se však energie zvětšuje nebo ztrácí. Podle [4] není Eulerova metoda vhodná pro problémy zahrnující kmitavý pohyb. Metody Runge-Kutta, které mají vyšší řád konvergence, například také Heunova metoda, si s těmito problémy vedou lépe. Smyslem tohoto příkladu však je ukázat drobnou úpravu Eulerovy metody, která sice nemá lepší řád konvergence, ale je více vhodná pro problémy tohoto typu. Této úpravě se říká Euler-Cromerova metoda, nebo také semi-implicitní metoda.

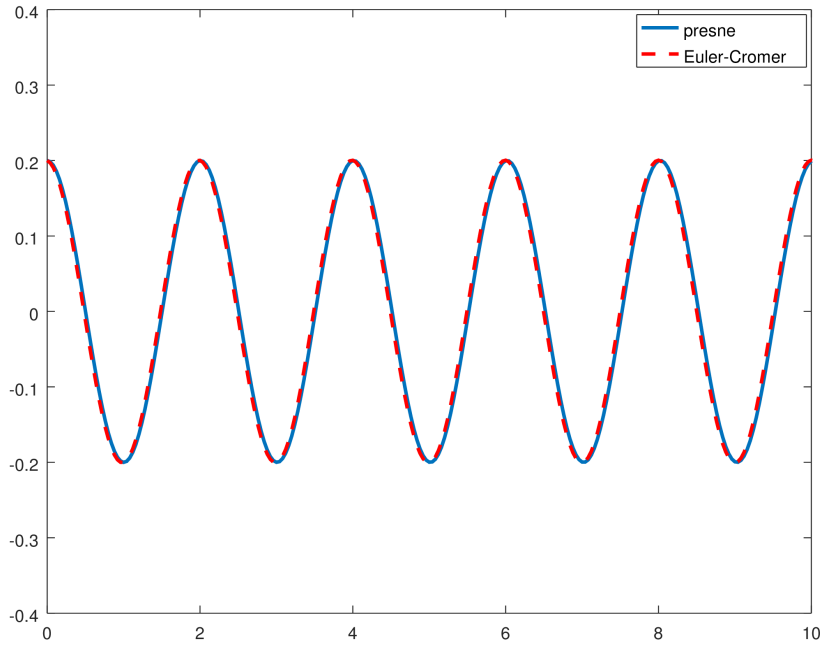
Označme $Y(i,1)$ výchylku v čase t_i a $Y(i,2)$ rychlost. Myšlenka této úpravy je následující: použijeme $Y(i,1)$ a $Y(i,2)$ k nalezení $Y(i+1,2)$. Pak použijeme $Y(i,1)$ a $Y(i+1,2)$ k nalezení $Y(i+1,1)$. Zapsáno v kódu:

```
while t(i) < t_f
        Y(i+1,2) = Y(i,2) - (g/l) * Y(i,1) * h;
        Y(i+1,1) = Y(i,1) + Y(i+1,2) * h;
        t(i+1) = t(i) + h;
        i = i + 1;
end
```

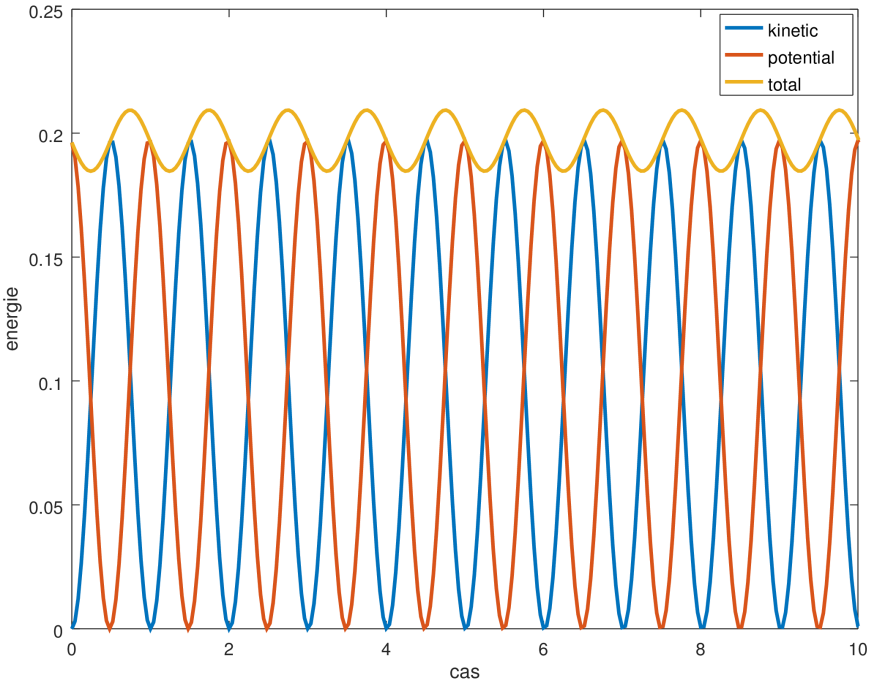
Euler-Cromerova metoda patří mezi tzv. *symplektické* numerické metody pro řešení diferenciálních rovnic. Také metody Runge-Kutta vyšších řádů se dají upravit do symplektické podoby. Jsou to metody používané zejména pro řešení diferenciálních rovnic, které se objevují ve fyzice. Zhruba řečeno: symplektické řešiče zajišťují, že průměrná změna energie na relativně velkém časovém intervalu je malá. Tento jev je vidět na Obrázku 6.20, kde je graf energie v závislosti na čase při řešení příkladu semi-implicitní metodou s $h = 0.04$.

Počáteční celková energie je $E(0) = 0.19620$ a celková energie na konci časového intervalu je $E(10) = 0.19495$. Čísla by se lišila méně, pokud bychom zvolili menší krok. Důležité však je, že jednoduchou úpravou jsme dosáhli mnohem lepších výsledků než u běžné explicitní či implicitní Eulerovy metody.

Obrázek 6.19: Srovnání grafů přesného a přibližného řešení, Euler-Cromerova metoda



Obrázek 6.20: Zachování energie pro semi-implicitní metodu



7 Numerické experimenty

V této kapitole budeme hledat numerické řešení tzv. rovnic s impulzy. Nejdříve řešíme počáteční úlohu, navíc však máme zadanou jistou funkci $G(t, \mathbf{y}(t))$, které budeme říkat funkce bariéry. Pokud v určitém bodě $(t_k, y(t_k))$ nastane $G(t_k, y(t_k)) = 0$, tak došlo k dotyku s bariérou. Přesná řešení rovnic s impulzy nemají při dotyku se zadanou bariérou spojité derivace. Body dotyku závisí na řešení a nejsou předem známy.

Příklady v této kapitole budeme obecně řešit následujícím postupem:
1. Je třeba najít interval $[t_i, t_{i+1}]$, ve kterém dojde k dotyku, tj. tento interval obsahuje bod (t_k, y_k), kde $G(t_k, y_k) = 0$. To znamená, že ověření, zdali se v takovém intervalu nacházíme, lze provést porovnáním znamének hodnot $G(t_i, y_i)$ a $G(t_{i+1}, y_{i+1})$.
2. Bod (t_k, y_k) zpřesníme metodou bisekce. Iterační proces zpřesňování zastavíme, když $|G(t_k, y_k)| < tol$.
3. Řešíme novou počáteční úlohu na intervalu $[t_k, t_f]$.

Příklad 7.9. Uvažujme padající těleso, které se odráží od bariéry s koeficientem restituce, *cor*. Pokud je $cor \in (0, 1)$, tak dochází ke ztrátě energie při odrazu. Počáteční výšku zvolíme 1 m a počáteční rychlost nulovou. Také budeme brát v potaz odpor vzduchu. Řešíme tak počáteční úlohu

$$\begin{cases} y''(t) = -g - by'(t), \ t \in [0, t_f] \\ y(0) = 1, y'(0) = 0 \end{cases}$$

Kde $y(t)$ je výška nad zemí v čase t a b a g jsou fyzikální konstanty, g gravitační zrychlení a b odpor vzduchu. Tuto rovnici druhého řádu opět převedeme na systém dvou rovnic prvního řádu. Zaveďme značení $x_1 = y(t)$ a $x_2 = y'(t)$.

$$\begin{cases} x_1' = x_2, & x_1(0) = 1 \\ x_2' = -g - bx_2, & x_2(0) = 0 \end{cases}$$

V tomto příkladě je bariérou povrch země. Řešení má dvě složky: výšku nad zemí a rychlost. Zajímá nás, kdy dojde k dotyku se zemí, tedy kdy je první složka nulová. Proto zvolíme bariéru tvaru $G(t, \mathbf{Y}) = -Y_1(t)$.
Protože se v počítači pohybujeme v diskrétních krocích, tak se nám nepodaří přesně spočítat okamžik dotyku tělesa s bariérou. Hodnotu $G(t_i, y_i) = 0$ tak budeme brát s jistou tolerancí *tol*. Zajímá nás tak podmínka $|G(t_i, Y_i)| < tol$. Jak bylo řečeno v úvodu této kapitoly, v každé iteraci je třeba zkontrolovat, zdali numerické řešení již bariéru nepřeskočilo. K tomu dojde je-li splněna podmínka (zapsáno v Matlabu):

```
if G(t(i),Y(i,:))*G(t(i+1),Y(i+1,:)) < 0.
```

V takovém případě si zapamatujeme počáteční bod t_i intervalu $[t_i, t_{i+1}]$ a přibližnou hodnotu Y_i v tomto bodě. Budeme počítat nové t_i a Y_i vždy s poloviční velikostí předchozího kroku a to dokud platí

```
abs(G(t(i),Y(i,:)) > tol).
```

Jakmile tato podmínka neplatí, našli jsme dostatečně přesný bod dotyku s bariérou. Nyní řešíme novou počáteční úlohu. Nová počáteční podmínka je dána odrazem tělesa od bariéry. Vždy je třeba rychlost tělesa rozložit na dvě složky. Jedna složka je vzhledem k bariéře kolmá a druhá rovnoběžná. Při odraze dochází ke změně znaménka v kolmé složce, zatímco rovnoběžná složka zůstává beze změny. V tomto případě uvažujeme pouze vertikální pohyb, tedy vektor rychlosti má jen jednu složku. Není třeba zkoumat, která je kolmá a která rovnoběžná.

Nyní popíšeme detailně popíšeme program výpočtu v Matlabu. (*Poznámka:* Komentáře v programu uvádím v angličtině, neboť není možné psát diaktriku,

navíc příkazy jako while a if jsou také anglicky.)

```
function [t Y] = bar_puleni(F,G,Y0,tf,h,tol,cor)
%input: F -- function handle, right hand side of diff. eqn
%       G -- function handle, barrier
%       Y0 -- initial condition at t = 0
%       tf -- final time
%       h -- stepsize
%       tol -- error tolerance

i = 1;
t(i) = 0;
Y(i,:) = Y0;
```

Dokud nejsme na konci intervalu, počítáme nové přibližné hodnoty pomocí explicitní Eulerovy metody s adaptivním krokem.

```
while t(i) < tf

        f1 = F(t(i),Y(i,:));
        f2 = F(t(i) + (h/2),Y(i,:) + (h/2).*f1');

        A1 = Y(i,:) + h.*f1';
        A2 = Y(i,:) + (h/2).*f1' + (h/2).*f2';

        r = norm(A1-A2,inf);

        if r > tol
                h = 0.9*h*max(tol/r,0.3);
        else
                t(i+1,:) = t(i) + h;
                Y(i+1,:) = A2;
                h = 0.9*h*min(tol/r,2);
```

V každých po sobě jdoucích dvou iteracích kontrolujeme, zda-li nejsme v intervalu obsahující bod dotyku, tedy poslední **else** obsahuje poněkud více řádků kódu. Jeho příslušné **end** také zvýrazním modře.

```
                if G(t(i),Y(i,:))*G(t(i+1),Y(i+1,:)) < 0
                        t_s = t(i);
                        Y_s = Y(i,:);
```

```
            hod = G(t_s,Y_s);
            h_new = h;
```

Pokud jsme v intervalu obsahující bod dotyku, zapamatujeme si krajní hodnoty (viz popis v předchozích odstavcích).

```
        while abs(G(t(i),Y(i,:))) > tol
            h_new = h_new/2;
            t_1 = t(i) + h_new;
            Y_1 = Y(i,:) + h_new.*F(t(i),Y(i,:))';
```

Dokud nemáme dostatečně přesný bod dotyku, počítáme nové hodnoty s polovičním krokem.

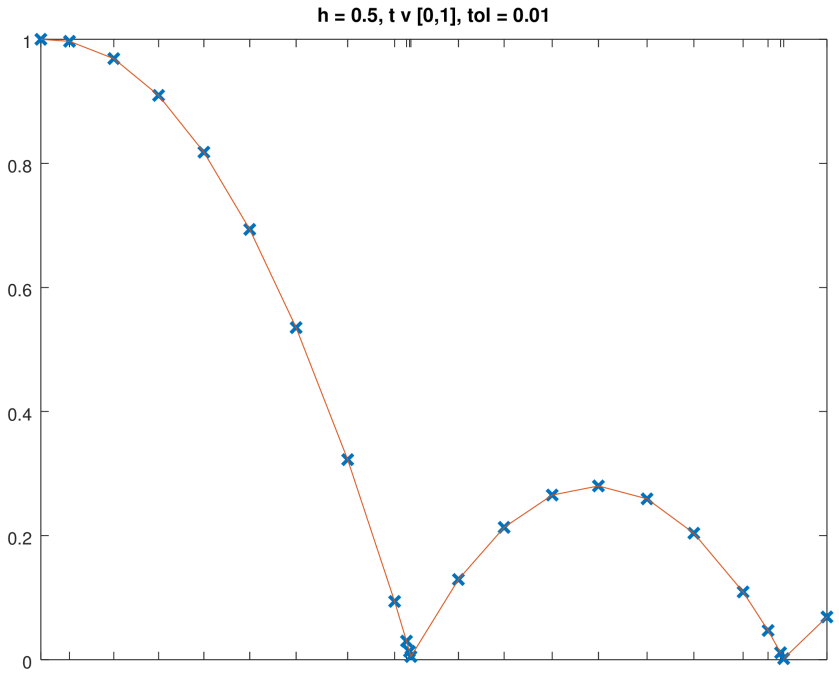
```
            if hod*G(t_1,Y_1) > 0
                t(i+1) = t_1;
                Y(i+1,:) = Y_1;
                i = i +1;
            end
        end
```

Nová hodnota $G(t_1, Y_1)$ může mít stejné znaménko jako hodnota funkce bariéry v krajním bodě intervalu. V takovém případě ji přijmeme do posloupnosti přibližných řešení. V opačném případě nic neděláme, tj. hodnoty t_1, Y_1 si nezapamatujeme neboť podmínka neprojde. Vrátíme se do cyklu while, opět zmenšíme krok na poloviční velikost a spočítáme novou hodnotu. Tímto postupem se přibližné hodnoty nedostanou pod bariéru.

```
            Y(i,2) = -cor*Y(i,2);
        else
            i = i + 1;
    end
end
```

Není-li splněna červená podmínka, tj. přibližná hodnota není blízko bariéry, pouze posuneme index a dále počítáme přibližné hodnoty pomocí adaptivního kroku. Na Obrázku 7.21 je vidět závislost výšky tělesa na čase v intervalu $[0, 1]$, použili jsme počáteční krok $h_0 = 0.5$ a toleranci $tol = 0.01$. Zvětšíme-li graf v oblasti okolo bodů dotyku, je vidět větší hustota sítě. Pak se těleso odrazí a dále pokračujeme s velkým krokem, viz Obrázek 7.22.

Obrázek 7.21: Závislost výšky na čase



Příklad 7.10. Následující úloha je převzata z článku [3] (str. 679, Příklad 2.1), kde je řešena jako okrajová úloha. Autor se nezabývá numerickým řešením daných úloh, ale zkoumá vlastnosti analytických řešení. Uvažovaná okrajová úloha ve [3] je tvaru:

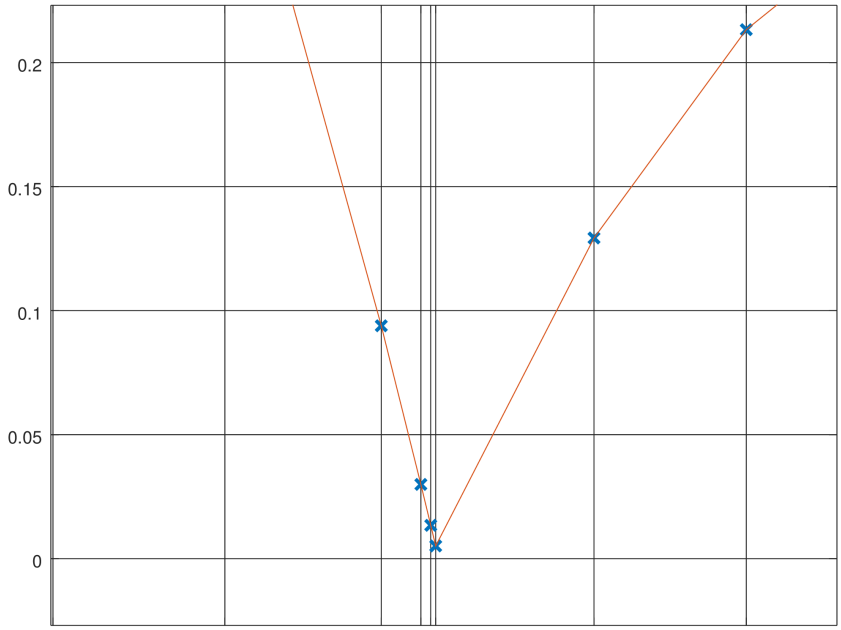
$$\begin{cases} \ddot{x}(t) = 2 \quad t \in [0,1], |x(t)| < \frac{1}{8} \\ \dot{x}(s+) = -\dot{x}(s), \quad |x(s)| = \frac{1}{8} \\ x(0) = x(1) = 0, \end{cases}$$

kde s je bod, kde je splněna podmínka bariéry. Okrajovou úlohu druhého řádu přepíšeme na systém dvou rovnic prvního řádu

$$\begin{cases} x_1'(t) = x_2 \\ x_2'(t) = 2, \end{cases}$$

$t \in [0,1]$, $|x(t)| \leq 1/8$. Tedy chceme, aby se částice pohybovala v pásu $[-1/8, 1/8]$.

Obrázek 7.22: Zvětšení - hustší síť okolo bodu dotyku



Funkce bariéry tak má tvar

$$G(t, \mathbf{x}) = (x_1(t) - 1/8)(x_1(t) + 1/8).$$

Příklad budeme nejdříve řešit s počáteční podmínkou $\mathbf{x}_0^1 = (0, -0.8568)$, a pak s podmínkou $\mathbf{x}_0^2 = (0, -1.76579)$. Hodnoty počátečních podmínek jsou převzaty z článku a jsou voleny tak, aby řešení dané počáteční úlohy splňovalo okrajovou podmínku.

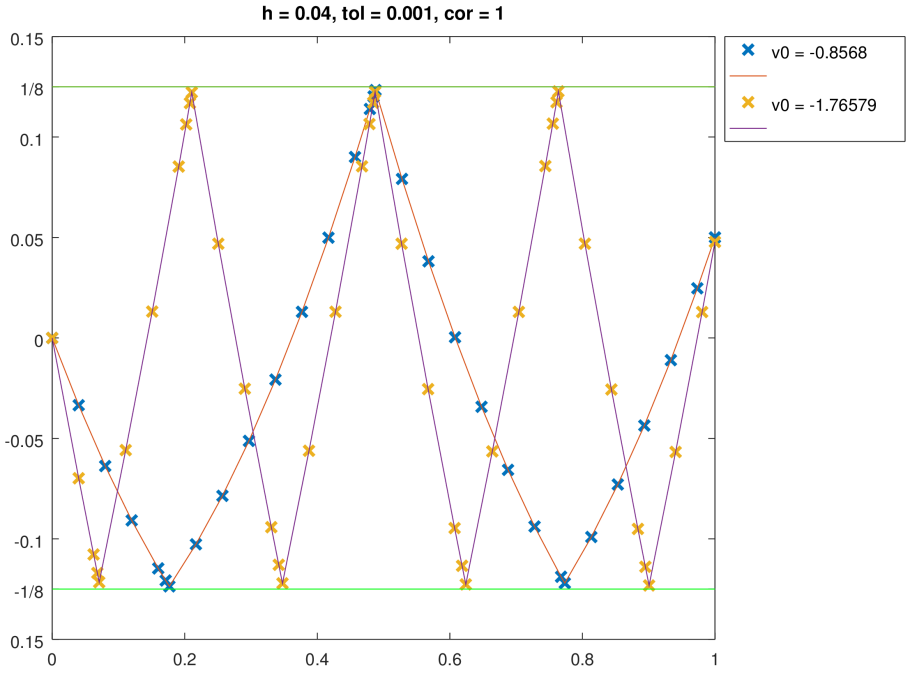
Pro počáteční podmínku $\mathbf{x}_0^1$ nastal první dotyk s bariérou v čase $t = 0.176875$. V článku [3] je uveden bod dotyku přesného řešení s bariérou jako $t = 0.186475$. Další přibližný čas dotyku jsme našli v $t = 0.487812$, přesný čas by měl být přesně $1/2$. Poslední přibližný čas dotyku je 0.773437, přičemž přesný čas je přibližně 0.81353.

Při řešení příkladu s počáteční podmínkou $\mathbf{x}_0^2$ získáme sedm bodů dotyku. Na Obrázku 7.23 jsou znázorněny přibližné hodnoty v čase pro obě počáteční podmínky.

Ve všech případech je použit krok $h = 0.04$, tolerance $tol = 0.001$ a $cor = 1$, uvažujeme tedy totálně elastický odraz.

V obou případech počátečních podmínek jsme dostali přibližná řešení splňující

Obrázek 7.23: Závislost výšky tělesa na čase



okrajovou úlohu s jistou tolerancí.

Příklad 7.11. Uvažujme opět příklad z [3] (str. 680, Příklad 2.2). Okrajová úloha je tvaru

$$\begin{cases} \ddot{x}(t) = 6t, \quad t \in [0,1], \ |x(t)| < \frac{3}{8} \\ \dot{x}(s+) = -\dot{x}(s), \quad |x(s)| = \frac{3}{8} \\ x(0) = x(1) = 0 \end{cases}$$

Tentokrát požadujeme $|\mathbf{x}(t)| < 3/8$, tedy funkce bariéry je

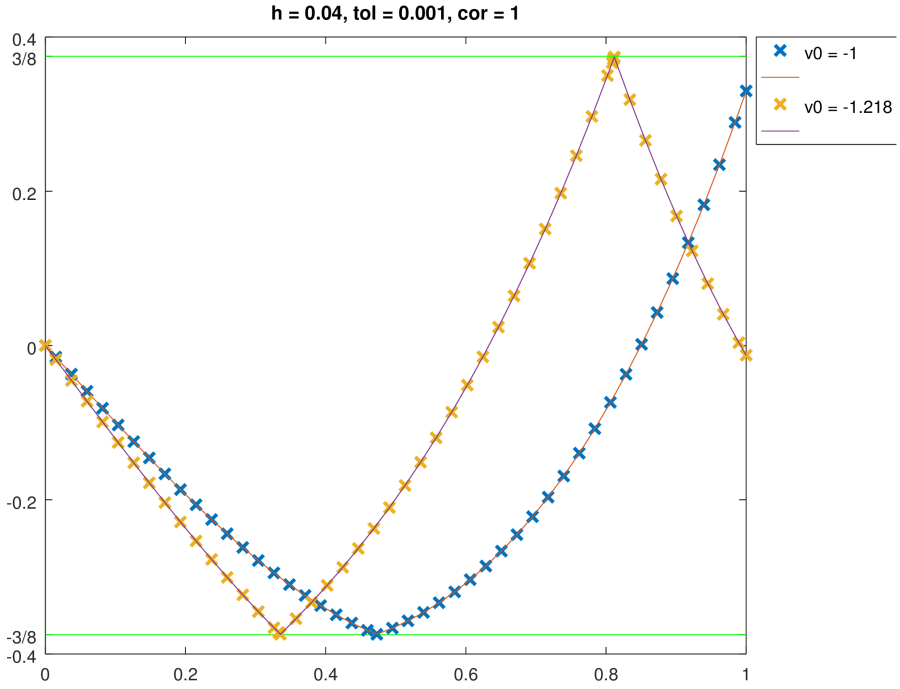
$$G(t, \mathbf{x}) = (x_1(t) - 3/8)(x_1(t) + 3/8).$$

Okrajovou úlohu opět přepíšeme na soustavu dvou rovnic prvního řádu

$$\begin{cases} x_1'(t) = x_2 \\ x_2'(t) = 6t. \end{cases}$$

První počáteční podmínka $\mathbf{x}_0^1 = (0, -1)$ je zvolena bez ohledu na splnění okrajové podmínky. Bod dotyku přesného řešení impulzní úlohy s bariérou nastane podle [3] v čase $t = 0.5$. Přibližný bod dotyku vyšel $t = 0.472944$. Použijeme-li počáteční podmínku $\mathbf{x}_0^2 = (0, -1.218)$, tak dostaneme přibližné řešení impulzní rovnice splňující okrajové podmínky s jistou tolerancí. Přibližná hodnota v koncovém bodě intervalu je $y_N^1 = -0.012594$, horní index 1 značí výšku nad nulou, druhá složka (rychlost) není součástí okrajové podmínky. Přibližné hodnoty obou počátečních úloh jsou znázorněny na Obrázku 7.24. Zvolení druhé počáteční rych-

Obrázek 7.24: Závislost výšky tělesa na čase



losti, pro kterou přibližné řešení splňuje okrajovou podmínku, má své odůvodnění plynoucí z analýzy přesného řešení.

8 Příloha: Kódy

Tato část obsahuje programy napsané v Matlabu, které jsem použila k řešení příkladů v této práci. Výstupem každé funkce je sloupcový vektor **t**, což je síť bodů $t_i,\ i = 0, ..., N-1$ z intervalu $[t_0, b]$ (někde značeno jako $[t_0, t_f]$. A matice či vektor přibližných hodnot $Y(N, n)$, který má N řádků a n sloupců. Matice (či vektor) **Y** obsahuje přibližné hodnoty spočtené právě v časech **t**, počet sloupců odpovídá počtu složek řešení. Počet řádků N odpovídá počtu přibližných hodnot spočtených danou metodou.

Explicitní Eulerova metoda

```
function [t Y] = euler(F,Y0,T,h)
% explicit euler method
%input:
%   F = @(t,X)  function handle
%   T = [t0 tf] t0 = starting time, tf = final time
%   Y0  row vector or a point, initial condition
%   h stepsize; optional argument, default h = 0.02

%check input

if ~exist('h','var') || isempty(h)
    h = 0.02;
end

if (T(1) >= T(2))
    fprintf('Chyba: neplatny interval.\n');
    return
end

if h >= norm(T(2) - T(1),2)
```

```
    fprintf('Chyba: neplatne h.\n');
    return
end

%divide time interval
t = [T(1):h:T(2)]';
n = length(t);
if t(n) < T(2)
  t(n+1) = T(2);
elseif t(n) > T(2)
  t(n) = T(2);
end

%initialize for approximate values
N = length(t);
vars = length(Y0);
Y = zeros(N, vars);
Y(1,:) = Y0;

for i = 1:(N-1)
  %forward euler iteration
    Y(i+1,:) = Y(i,:) + h.*F(t(i),Y(i,:))';%F(t,Y) column
end

end
```

Implicitní Eulerova metoda

Implicitní Eulerovu metodu jsem napsala dvěma způsoby. První verze využívá funkce *fsolve*, která je součástí Matlabu. Tato funkce najde numerické řešení soustavy n nelineárních rovnic o n neznámých tvaru $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. Její základní vstupní argumenty jsou function handle funkce $\mathbf{g}$ z levé strany rovnice a vektor (či jediná hodnota ve skalárním případě) počátečního odhadu řešení. Funkci *fsolve* volám navíc v příkaze *evalc*, který zachytí veškerý výstup určený pro terminál v proměnné *extra*, kterou již dále nevypisuji, takže se tento výstup v terminále neobjeví. Tento výstup je například zpráva, že řešení rovnice bylo nalezeno s danou přesností. Jelikož je funkce v průběhu iteračního procesu Eulerovy metody volána několikrát, tak by tyto zprávy zbytečně zabíraly místo v terminále

Matlabu. Druhá verze využívá k nalezení řešení implicitní rovnice upravenou verzi Newtonovy metody. Připomeňme, že v implicitní Eulerově metodě je následující přibližná hodnota $Y(i+1,:)$ spočtená jako řešení rovnice

$$Y(i+1,:) = Y(i,:) + h.*F(t(i), Y(i+1,:))'$$

Abychom mohli použít funkci *fsolve*, nebo Newtonovu metodu, je třeba tuto rovnici převést na tvar $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. Definuji tedy nové function handle

```
g = @(x) (x - h.*F(t(i+1),x) - Y(i,:)'),
```

které pak předám jako argument funkci *fsolve* spolu s počátečním odhadem kořene. Jako odhad zvolím aktuální přibližnou hodnotu $Y(i,:)$.

```
function [t Y] = implicit_euler(F,Y0,T,h)
%backward euler method for systems/or scalar equation
%using built-in function fsolve
% input:
%   F = @(t,X) function handle
%   Y0  row vector or a point, initial condition
%   T = [t0 tf] t0 = starting time, tf = final time
%   h stepsize; optional argument, default h = 0.02
%

%check input
if ~exist('h','var') || isempty(h)
    h = 0.02;
end

if (T(1) >= T(2))
    fprintf('Chyba: neplatny interval.\n');
    return
end

if h >= norm(T(2) - T(1),2)
    fprintf('Chyba: neplatne h.\n');
    return
end

%divide time interval
t = [T(1):h:T(2)]';
n = length(t);
```

```
if t(n) < T(2)
   t(n+1) = T(2);
elseif t(n) > T(2)
   t(n) = T(2);
end

%initialize for approximate values
N = length(t);
vars = length(Y0);
Y = zeros(N, vars);
Y(1,:) = Y0;

m = 0;


for i=1:(N-1)
     %first guess
     x0 = Y(i,:)';
     g = @(x) (x - h.*F(t(i+1),x) - Y(i,:)');
     [extra x1] = evalc('fsolve(g,x0);');
     m = m + 1;
     Y(i+1,:) = x1';
end

fprintf('pocet reseni rovnice pro y i+1 %f\n',m)

end
```

Ve druhé verzi budeme kořen funkce definované jako

```
g = @(x) (x - h.*F(t(i+1),x) - Y(i,:)')
```

hledat pomocí Newtonova iteračního procesu

$$x_{k+1} = x_k - J_g^{-1}(x_k) * g(x_k).$$

J_g^{-1} je inverze Jacobiho matice, i-tý řádek obsahuje derivace funkce $\mathbf{g}$ podle i-té proměnné. Přitom nederivujeme podle t. Jelikož počítat inverzi matice je náročná operace, upravíme iterační proces na soustavu lineárních rovnic (neboť $J_g(x_k)$ je číselná matice a $\delta_k = x_{k+1} - x_k$) tvaru:

$$J_g(x_k)\delta_k = -g(x_k).$$

Řešení soustav lineárních rovnic je daleko jednodušší než řešení soustav rovnic nelineárních. Také Newtonova metoda potřebuje počáteční odhad kořene, který zvolím jako aktuální přibližnou hodnotu Y_i. Jacobiho matice funkce $\mathbf{g}$ se počítá z Jacobiho matice pravé strany diferenciální rovnice z počáteční úlohy, neboť platí

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} - h\mathbf{F}(\mathbf{x}) - \mathbf{x}_0$$

$$J_g = I - hJ_F$$

```
function [t Y] = implicit_euler_newton(F,Y0,T,h,dF)
%backward euler method for systems/or scalar equation
%using Newton's method
% input:
%   F = @(t,X) function handle
%   Y0  row vector or a point, initial condition
%   T = [t0 tf] t0 = starting time, tf = final time
%   h stepsize; optional argument default h = 0.02
%   dF optional argument, funtion handle, Jacobi of F


%check input
if ~exist('h','var') || isempty(h)
    h = 0.02;
end

if (T(1) >= T(2))
    fprintf('Chyba: neplatny interval.\n');
    return
end

if h >= norm(T(2) - T(1),2)
    fprintf('Chyba: neplatne h.\n');
    return
end

%divide time interval
t = [T(1):h:T(2)]';
n = length(t);
if t(n) < T(2)
  t(n+1) = T(2);
elseif t(n) > T(2)
```

```
    t(n) = T(2);
    end

%initialize for approximate values
N = length(t);
vars = length(Y0);
Y = zeros(N, vars);
Y(1,:) = Y0;



for i=1:(N-1)
    g = @(t,x) (x - h.*F(t,x)- Y(i,:)');

    %newton method set up
    tol = 1e-3;
    max_iters = 50;
    c = true;
    k = 1;

    %initial guess
    x0 = Y(i,:)';
    t0 = t(i);

    %check if dF provided
    if ~exist('dF','var')
                %if not, compute numerical jacobian
        J = jacobian(F,t0,x0,h);
    else
        J = eye(length(x0)) - h.*dF(t0,x0);
    end
    %no inverse version
    delta = J \ (-g(t0,x0));
    x1 = x0 + delta;

    %Newton
    while norm(x1-x0,inf) > tol && c
        x0 = x1;
        %check if dF provided
        if ~exist('dF','var')
                        %if not compute numerical Jacobian
                J = jacobian(F,t0,x0,h);
      else
```

```
                J = eye(length(x0)) - h.*dF(t0,x0);
        end

        delta = J \ (-g(t0,x0));
        x1 = x0 + delta;

        k = k + 1;
        if k == max_iters
          c = false;
          fprintf('newton reached max_iters\n')
        end
      end
Y(i+1,:) = x1';
end
end
```

Pokud programu nedodáme derivace pravé strany diferenciální rovnice z počáteční úlohy, tak spočítá Jacobiho matici pro Newtonovu metodu numericky pomocí následující funkce.

```
function [J] = jacobian(F,t,Y,h)

% numerical jacobian of
%g(y(i+1)) = y(i+1) - y(i) - (hF(t(i+1),y(i+1)))
%input:
%        F = @(x) function handle
%        t = a single value
%        Y = column vector
%        h = stepsize

  Y = Y';
  eps = 1e-5;
  X = [t Y];
  X_d = X;
  fx = feval(F,t,Y);
  n = length(X);
  for j=1:n
    X_d(j) = X_d(j) + eps;
    J(:,j) = (feval(F,X_d(1),X_d(2:n))-fx)/eps;
    X_d(j) = X(j);
  end
    J = J(:,2:n);
    n = size(J,1);
```

```
    I = eye(n);
    J = I - h.*J;
```

end

Derivace je aproximovaná diferencí vpřed. Máme-li

$$\mathbf{F}_J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

Pak

$$\frac{\partial F_i}{\partial x_i} \approx \frac{F_i(x_1, ..., x_i + eps, ..., x_n) - F_i(x_1, ..., x_i, ..., x_n)}{eps}.$$

Adaptivní krok

V kapitole Adaptivní krok byl uveden vzorec pro výpočet nové velikosti kroku v případě, že odhad chyby je vetší než povolená tolerance. Jedná se o vzorec (4.21), který je v následujícím kódu upraven. Komentář k této úpravě je vždy uveden pod příslušným řádkem.

```
function [t_n Y_n] = adapt(F,Y0,T,h0,tol)

        % F = @(t,X)
        % T = [t_0 tf]   t0 = starting time, tf = final time
        % Y0    row vector or a point, initial condition
        % h0    initial stepsize; default = 0.02
        % tol   error tolerance; default = 10^-3


%kontrola vstupu
if ~exist('h0','var') || isempty(h0)
    h0 = 0.02;
end

if ~exist('tol','var') || isempty(tol)
    tol = 0.001;
end
```

```
t0 = T(1);
b = T(2);

i = 1;
m = 0;

t = t0;
Y = Y0';%Y = column
h = h0;

t_n(i,1) = t;
Y_n(i,:) = Y;

count = 0;


    while(t < b)
        f1 = F(t,Y);%column
        A1 = Y + h*f1;

        f2 = F(t + (h/2),Y + (h/2)*f1);%column
        A2 = Y + (h/2)*f1 + (h/2)*f2;


        m = m+2;

        r = norm(A1-A2,inf);

        if(r > tol)
            h = 0.9*h*max(tol/r,0.3);
```

Pro jistotu znovu uveďme vzorec (4.21):

$$h_i' = 0.9 \cdot h_i \cdot \min\left(\max\left(\frac{tol}{r}, 0.3\right), 2\right).$$

Pokud platí, že $r > tol$, tak $1 > \frac{tol}{r}$. Tedy $z = \max(\frac{tol}{3}, 0.3) < 1$ a $\min(z, 2)$ bude vždy z. Proto stačí vzít pouze maximum. Podobně ve větvi *else* této podmínky, kde $r < tol$ bude $1 < \frac{tol}{r}$ a uvažované maximum bude vždy tento zlomek, pak

stačí brát pouze $\min(\frac{tol}{r}, 2)$.

```
            count = count + 1;
        else
            i = i+1;
            t = t+h;

            Y = A2;
            t_n(i,1) = t;
            Y_n(i,:) = Y';
            %after accepting Y_n, can increase stepsize
            h = 0.9*h*min(tol/r,2);

        end

    end

%while can overstep tf
N = length(t_n);
if t_n(N) ~= T(2)
 t_n(N) = T(2);
 end
h = t_n(N) - t_n(N-1);

cor = Y_n(N-1,:) + h.*F(t_n(N-1),Y_n(N-1,:))';
Y_n(N,:) = cor;


    fprintf('pocet vyhodnoceni F %f\n',m)
    fprintf('pocet zamitnuti kroku %d\n',count)
end
```

Heunova metoda

```
function [t Y] = heun(F,Y0,T,h)
%Heun method for systems/or scalar equation
% input:
%   F = @(t,X) function handle
%   Y0  row vector or a point, initial condition
%   T = [t0 tf]  t0 = starting time, tf = final time
%   h stepsize; optional argument; default h = 0.02
```

```
%check input
if ~exist('h','var') || isempty(h)
    h = 0.02;
end

if (T(1) >= T(2))
    fprintf('Chyba: neplatny interval.\n');
    return
end

if h >= norm(T(2) - T(1),2)
    fprintf('Chyba: neplatne h.\n');
    return
end

%divide time interval
t = [T(1):h:T(2)]';
n = length(t);
if t(n) < T(2)
  t(n+1) = T(2);
elseif t(n) > T(2)
  t(n) = T(2);
  end

%initialize for approximate values
N = length(t);
vars = length(Y0);
Y = zeros(N, vars);


Y(1,:) = Y0;
m = 0;


for i = 1:(N-1)
    f1 = feval(F,t(i),Y(i,:));
    f2 = feval(F,t(i+1),Y(i,:) + h.*f1');
    m = m+ 2;
    Y(i+1,:) = Y(i,:) + h.*((f1' + f2')/2);
end
```

```
[t Y];
fprintf('pocet vyhodnoceni F: %f\n',m)


end
```

Závěr

Tématem této práce byla Eulerova metoda. Je to nejjednodušší numerická metoda pro řešení počáteční úlohy systému n diferenciálních rovnic prvního řádu. Tuto práci jsem si vybrala, neboť mi Eulerova metoda přišla jako dobrý úvod do numerických metod pro řešení diferenciálních rovnic. Tímto tématem jsem se v žádném kurzu bakalářského studia nezabývala, takže jsem se naučila něco nového. Explicitní Eulerova metoda je základem numerických řešičů počátečních úloh, navíc její zápis není příliš komplikovaný. Tato meotda se dá jednoduše upravit do jiných podob vhodných pro různé typy problémů.

První taková úprava uvedená v této práci je implicitní verze. Výhoda implicitní Eulerovy metody je stabilita, nevýhodou je výpočetní náročnost. Další úpravou byla myšlenka měnit délku kroku v průběhu výpočtu. Adaptivita kroku se běžně používá ve funkcích, které jsou součástí Matlabu a jiných softwarů. V kapitole Modifikace Eulerovy metody jsem uvedla další dvě úpravy. Jedna z nich, Euler-Cromerova metoda, má stejný řád konvergence jako Eulerova metoda. Euler-Cromerova metoda je zástupcem numerických metod pro řešení diferenciálních rovnic objevujících se v některých oblastech fyziky. Těmto metodám se říká symplektické. Tyto metody v jistém smyslu berou ohled na fyzikální zákony, jako je např. zákon zachování energie. Druhá z úprav představených v této kapitole, Heunova metoda, má řád konvergence o stupeň vyšší. Heunova a Eulerova metoda patří do skupiny metod souhrnně označovaných jako Runge-Kutta. Všechny tyto metody jsou jednokrokové, k určení nové přibližné hodnoty využívají pouze data spočtená z aktuální přibližné hodnoty.

Kromě využití jednoduchého zápisu Eulerovy metody z teoretických důvodů, tj.

odvození různých dalších numerických metod, jsem také využila jednoduchosti zápisu pro výpočet příkladů uvedených v kapitole Numerické experimenty. V této kapitole jsem hledala přibližná řešení počátečních úloh, které se měnily v průběhu výpočtu. Najít přibližný bod nové počáteční úlohy s danou přesností lze různými způsoby. Já jsem použila metodu bisekce, což je sice procedura dost pomalá, ale zato si myslím, že kód je přehledný.

Literatura

[1] Vitásek, E.: Základy teorie numerických metod pro řešení diferenciálních rovnic, Academia, Praha 1994

[2] Burden, Richard L., Faires, Douglas J.: Numerical Analysis, Brooks Cole, Boston, MA, 2011

[3] Gabor, G.: On the Dirichlet problem in billiard spaces, J. Math. Anal. Appl. 440, 677-691 (2016).

[4] Giordano, Nicholas J.: Computational Physics, Prentice-Hall, 1997

[5] I. Horová, J. Zelinka: Numerické metody, Brno, Masarykova univerzita, 2004

[6] Čermák, L.: Numerické metody pro řešení diferenciálních rovnic , Brno : Litera Brno, 2013