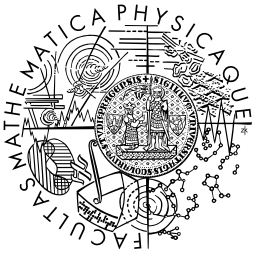


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Petr Záboj

Paralelní časová integrace pro řešení obyčejných diferenciálních rovnic

Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Václav Kučera, Ph.D.

Studijní program: Matematika

Studijní obor: Obecná matematika

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chtěl bych poděkovat vedoucímu práce panu doc. RNDr. Václavu Kučerovi, Ph.D. za jeho vstřícnost a cenné rady. Také bych chtěl poděkovat mé rodině, která mi poskytovala spoustu podpory při studiu na této škole. Nakonec bych chtěl poděkovat za podporu a porozumění mých přátel, které jsem si našel na MFF.

Název práce: Paralelní časová integrace pro řešení obyčejných diferenciálních rovnic

Autor: Petr Záboj

Katedra: Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Václav Kučera, Ph.D., Katedra numerické matematiky

Abstrakt: Tato práce se zabývá problémem paralelizace metod pro numerické řešení obyčejncýh diferenciálních rovnic. Hlavní obsah práce tvoří algortimus Parareal, který je v dnešní době jedním z nejvíce studovaných a využívaných algoritmů využívající paralelní výpočty pro řešení diferenciálních rovnic. Nadále se zabýváme odvozením vzorce pro metodu Parareal pomocí jednokrokových metod a metody vícenásobné střelby. Nakonec jsou provedeny numerické experimenty, na nichž jsou předvedeny vlastnosti tohoto algoritmu.

Klíčová slova: Parareal, Paralelizace, Diferenciální rovnice, Diskretizace

Title: Parallel time integration for ordinary differential equations

Author: Petr Záboj

Department: Department of Numerical Mathematics

Supervisor: doc. RNDr. Václav Kučera, Ph.D., Department of Numerical Mathematics

Abstract: This thesis is about the problem of parallel-in-time integration methods. The main bulk of this thesis consists of the Parareal algorithm, which is one of the most widely used and studied parallel-in-time integration methods. We focus on the derivation of the Parareal algorithm using single-step integration methods and the multiple shooting method. Finally, the properties of this algorithm are demonstrated with numerical experiments.

Keywords: Parareal, Parallelisation, Differential equation, Discretization

Obsah

Úvod

V dnešní době jsme schopni vyrábět obrovské superpočítače, které dokážou provádět paralelně spoustu výpočtů najednou. Pokud ale chceme modelovat fyzikální problémy, potřebujeme přitom často řešit různé diferenciální rovnice. Analytické řešení ve většině případů nejsme schopni najít, proto potřebujeme použít numerické řešení. Numerická integrace je ale bohužel ve většině případech z povahy problému možná provádět pouze sekvenčně, díky čemuž nejsme schopni tyto algoritmy urychlit pomocí výkonější výpočetní techniky. Cílem této práce bude tedy předvést metody, které umožňují zrychlit výpočet numerického řešení pro časově závislé úlohy.

V průběhu historie po vynálezu výpočetní techniky se vědci pokoušeli vyvinout algoritmy pro numerickou integraci diferenciálních rovnic, které umožňují paralelní výpočty. Existuje více přístupů, kterými se dá paralelně integrovat diferenciální rovnice, viz (Gander (2013)), my se zde podíváme na přístup, který je založen na vícenásobné střelbě. Ta spočívá v tom, že si časový interval rozdělíme na více částí a na každém budeme řešit počáteční úlohu, přičemž budeme používat jako počáteční podmínku aproximaci z předchozího intervalu. Tento postup byl poprvné představen v článku (Bellen a Zennaro (1989)), a na jehož principu jsou založeny různé algoritmy, jedním z nichž je algoritmus Parareal, na který se v této práci budeme zaměřovat. Ten byl poprvé představen v článku (Lions a kol. (2001)).

V této práci se ze začátku zaměříme na jednokrokové metody, které se obvykle využívají při sekvenční časové integraci diferenciálních rovnic. V 2. kapitole představíme metodu vícenásobné střelby jako vhodný rámec pro paralelizaci řešení diferenciálních rovnic, a následně si pomocí ní odvodíme vzorec pro algortimus Parareal. Nakonec provedeme numerické experimenty na lineárních a nelineárních diferenciálních rovnicích a ukážeme si funkčnost algoritmu Parareal.

1. Sekvenční jednokrokové metody

1.1 Popis problému

V celé naší práci budeme řešit počáteční úlohu ve tvaru:

$$\begin{aligned} &u'(t) = f(t,u(t)) \text{ na intervalu } (0,T], \\ &u(0) = u^0, \end{aligned} \tag{1.1}$$

$u : \mathbb{R} \longrightarrow \mathbb{R}$ a $f : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$, kde známe $f(t,u)$ a počáteční podmínku u^0. U funkce $f(t,u)$ budeme předpokládat, že je spojitá v t a lipschitzovská v u, aby bylo řešení této úlohy jednoznačné. Naším úkolem bude toto řešení $u(t)$ aproximovat.

V celé naší práci ho budeme aproximovat následujícím způsobem: Na intervalu $(0,T]$ si vybereme posloupnost rovnoměrně rozložených bodů $\{t_n\}_{n=0}^N$, $t_0 = 0$, $t_N = T$, $t_0 < t_1 < ... < t_N$, které budeme nazývat *uzly*. Uzly jsou od sousedních uzlů vzdáleny o konstatní vzdálenost h, kterou budeme nazývat *krok metody*. Dají se uvažovat i metody, které nepoužívají rovnoměrné dělení, ty se hodí v případech, kdy potřebujeme lokálně zlepšit přesnost metody. My ale budeme pro jednoduchost uvažovat rovnoměrné dělení. Řešení rovnice budeme aproximovat pouze v uzlech, tedy budeme hledat posloupnost$\{u_n\}_{n=0}^N$ která nám bude nějakým způsobem aproximovat hodnoty $\{u(t_n)\}_{n=0}^N$.

1.2 Jednokrokové metody

Předtím, než se zaměříme na řešení diferenciálních rovnic pomocí paralelních metod, tak si definujeme sekvenční jednokrokové metody pro řešení těchto úloh, neboť je taky budeme následně používat v metodě Parareal. V následujících kapitolách vycházíme ze skript (Kučera a Feistauer (2014)).

1.2.1 Obecná jednokroková metoda

Mějme tedy nějakou posloupnost uzlů $\{t_n\}_{n=0}^N$ s krokem h a budeme chtít aproximovat hodnoty $\{u(t_n)\}_{n=0}^n$. Obecné jednokrokové metody vycházejí z následujícího vzorce:

$$u_{n+1} = u_n + h\phi(t_n,u_n,h),$$

kde za u_0 zvolíme počáteční podmínku u^0. Funkci $\phi(t,u,h)$ budeme nazývat *přírůstkovou funkcí*. Tato metoda je přirozený způsob, jak se dá aproximovat řešení diferenciální rovnice, neboť pokud si představíme diferenciální rovnici jako vývoj systému v čase, tak jednokrokové metody fungují tak, že se podívají, kde se momentálně v čase nacházíme a následně odhadneme, kde bychom se mohli nacházet dále.

Jak ale můžeme vidět z tohoto vzorce, tak jednokrokové metody jsou obecně sekvenční, neboť abychom mohli vypočítat u_{n+1}, tak musíme znát u_n. To dává

smysl i z heuristického hlediska, neboť potřebujeme znát místo, ze kterého vycházíme, abychom věděli, kam se dostaneme. My ale budeme chtít výpočet aproximace řešení urychlit pomocí paralelních výpočtů na několika procesorech najednou. Jak už povaha této úlohy napovídá, tak sekvenčnímu počítání se kompletně vyhnout nepůjde, proto se nám bude hodit si o sekvenčnímch metodách říct více.

1.2.2 Explicitní Eulerova metoda

Rozvineme-li si $u(t_{n+1})$ do Taylorova polynomu kolem bodu t_n, dostaneme následující vzorec:

$$\begin{aligned} u(t_{n+1}) &= u(t_n) + hu'(t_n) + O(h^2), \\ &\approx u(t_n) + hu'(t_n), \\ &= u(t_n) + hf(t_n,u(t_n)). \end{aligned}$$

Tento přístup nabádá k tomu si vzít za přírůstkovou funkci $\phi(t,u,h) = f(t,u)$, což vede na následující metodu:

Metoda 1 (explicitní Eulerova metoda)**.** *Explicitní Eulerova metoda má následující tvar:*

$$u_{n+1} = u_n + hf(t_n,u_n).$$

Tato metoda má za výhodu to, že je velice levná, neboť vyžaduje pouze jedno vyhodnocení funkce f. To je ale bohužel za cenu toho, že je tato metoda velice nepřesná, konkrétně je pouze 1. řádu. To, co znamená řád metody a jak se pomocí něho dá měřit přesnost jednokrokových metod, se budeme zabývat v následující kapitole.

1.2.3 Analýza chyby metody

Kvůli tomu, že přesné řešení pouze aproximujeme, tak vznikají chyby. Obecně je těžké o těchto chybách říci něco konkrétního, ale my se přesto pokusíme tyto chyby analyzovat, abychom nad nimi mohli mít určitou kontrolu. Pro analýzu chyby si zavedeme následující pojmy:

Definice 1. *Lokální diksretizační chyba je dána vzorcem:*

$$\tau(t,h) := \frac{u(t+h) - u(t)}{h} - \phi(t,u(t),h).$$

Lokální diskretizační chyba nám udává, jak moc se liší přesný přírůstek řešení od přírůstku daný přírůstkovou funkcí, neboli jak moc velkou uděláme chybu při jednom kroku naší metody. Tuto veličinu můžeme vypočítat za předpokladu hladkosti funkce u pomocí Taylorova rozvoje, a výpočet vede k odhadu ve tvaru $|\tau(t,h)| \leq Ch^p$, kde $p \in \mathbb{N}$ a konstanta C nezávisí na h. Pro nás bude důležité číslo p v exponentu kroku metody. Lokální diskretizační chyba nám toho sama o sobě ale o chování metody moc neřekne, proto si zadefinujeme následující pojem:

Definice 2. *Globální chyba metody je dána vzorcem:*

$$e_n := u(t_n) - u_n.$$

Tato chyba nám určuje, jak moc se liší aproximace od přesného řešení, tedy nám ukazuje, jak moc se lokální chyba akumuluje přes všechny kroky. Výpočet této chyby obecně provést nelze, jelikož přesné řešení neznáme, k jejímu odhadu se ale za silnějších předpokladů dá použít následující věta:

Věta 1. *Mějme jednokrokovou metodu, jejíž přírůstková funkce ϕ je lipschitzovksá v proměnné u s konstantou lipschtzovskosti L a nechť $|\tau(t,h)| \leq Ch^p$. Pak pro globální chybu metody platí následující odhad:*

$$\max_{n=1,\dots,N} |e_n| \leq Ch^p \frac{e^{LT}-1}{L}.$$

Pokud jsou předpoklady této věty splněny, tak vidíme, že metoda konverguje ke správnému řešení pro $h \to 0+$, jelikož konstanty nezávisí na h. Vidíme také, že konstanta p je pro odhad globální chyby metody v tomto případě stejná, jako pro lokální diskretizační chybu. Tuto konstantu budeme používat jako míru toho, jak moc je metoda přesná, což vede na následující definici:

Definice 3. *Řekneme, že metoda je řádu $p \in \mathbb{N}$, pokud existuje konstanta $C > 0$ nezávislá na h, pro kterou platí:*

$$\max_{n=1,\dots,N} |u_n - u(t_n)| \leq Ch^p.$$

Pokud budeme mít metodu s vyšším řádem, tak bude v praktickém využití efektivnější, protože pokud budeme zmenšovat krok h, tak mají metody s vyšším řádem zaručenou rychlejší konvergenci ke správnému řešení. Jak už jsme si říkali, tak explicitní Eulerova metoda je 1. řádu, nemůžeme tedy očekávat velikou přesnost.

1.2.4 Runge-Kuttova metoda 4. řádu

Obecné Runge-Kuttovy metody jsou jednokrokové metody, které využívají vyhodnocování funkce ve více různých bodech k získání vyššího řádu metody. Konkrétně tedy budeme hledat přírůstkovou funkci ϕ ve tvaru:

$$\phi(t,u,h) = \sum_{i=1}^{s} \omega_i k_i,$$

kde ω_i jsou konstanty a k_i jsou následujícího tvaru:

$$\begin{aligned}
k_1 &= f(t,u),\\
k_2 &= f(t+\alpha_2 h, u+\beta_{12}k_1),\\
k_3 &= f(t+\alpha_3 h, u+\beta_{13}k_1+\beta_{23}k_2),\\
&\vdots\\
k_s &= f(t+\alpha_s h, u+\sum_{j=1}^{s-1}\beta_{js}k_s).
\end{aligned}$$

Číslo s nám určuje, jak moc je tato metoda náročná na výpočet, neboť čím vyšší s, tím vícekrát musíme vyhodnotit funkci f. Můžeme si všimnout, že problém se sekvenčním vyhodnocováním funkce zde stále přetrvává, nejen že musíme znát předchozí bod u_n, abychom získali bod u_{n+1}, ale zároveň i při získávání bodu u_{n+1} musíme vyhodnocovat funkci sekvenčně, neboť abychom spočítali k_{i+1}, tak předtím musíme spočítat k_i.

My si zde uvedeme jako speciální příklad následující metodu:

Metoda 2 (Runge-Kuttova metoda 4. řádu)**.** *Runge-Kuttova metoda 4. řádu má následující tvar:*

$$\begin{aligned}
k_1 &= f(t_n,u_n),\\
k_2 &= f(t_n + \tfrac{h}{2}, u_n + \tfrac{k_1}{2}),\\
k_3 &= f(t_n + \tfrac{h}{2}, u_n + \tfrac{k_2}{2}),\\
k_4 &= f(t_n + h, u_n + k_3),\\
u_{n+1} &= u_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + 2k_4).
\end{aligned}$$

Tato metoda je 4. řádu a číslo $s = 4$. To může působit tak, že $s = p$, kde p je řád metody, jenže to obecně neplatí. Jakmile máme vyšší řád než 4, pak potřebujeme více vyhodnocení funkce než výsledný řád metody, například pro Runge-Kuttovu metodu 8. řádu potřebujeme funkci vyhodnotit jedenáctkrát, viz (Butcher (1996)). Proto je Runge-Kuttova metoda 4. řádu jedna z nejpoužívanějších metod, neboť je to metoda nejvyššího řádu, pro kterou stále platí $s = p$.

2. Paralelní metody časové integrace

Přístup, který jsme používali doteď v jednokrokových metodách nám paralelně počítat neumožňuje, budeme tedy potřebovat úplně nový nápad, který nám to umožní.

2.1 Metoda vícenásobné střelby

Metoda vícenásobné střelby nám bude sloužit jako základ, na kterém uvidíme, jak by paralelní algoritmy mohli fungovat. My ji následně použijeme k odvození algoritmu Parareal.

2.1.1 Odvození metody vícenásobné střelby

V následující kapitole se podíváme na úplně jiný přístup k získávání aproximací $\{u_n\}_{n=0}^N$ řešení problému (1.1). Vycházím z článku (Gander (2013)), který se zabývá historií paralelní časové integrace. Postup velice podobný metodě vícenásobné střelby byl poprvé představen v článku (Bellen a Zennaro (1989)) jako pokus o vytvoření iteračních metod pro řešení evolučního problému a následně byl použit v kontextu aproximace řešení diferenciálních rovnic v článku (Chartier a Phillipe (1993)).

Máme tedy náš problém (1.1) a máme posloupnost uzlů $\{t_n\}_{n=0}^N$. To si můžeme představit i tak, že jsme rozdělili interval $(0,T]$ na podintervaly $(t_n,t_{n+1}]$. To nám umožňuje přepsat problém (1.1) v následujícím tvaru:

$$\begin{aligned} &\tilde{u}_n'(t) = f(t,\tilde{u}_n(t)) \text{ na intervalu } (t_n,t_{n+1}], \\ &\tilde{u}_n(t_n) = u_n, \\ &n = 0,\ldots,N-1. \end{aligned} \tag{2.1}$$

Abychom ale tyto podúlohy mohli řešit, potřebujeme znát počáteční podmínky $\{u_n\}_{n=0}^{N-1}$. Tyto hodnoty nazýváme *parametry střelby* a my je budeme chápat jako aproximaci řešení. Dále si odvodíme, jak tyto parametry můžeme získat.

Víme, že řešení problému (1.1) musí být spojité, tedy řešení problému (2.1) na intervalu $(t_{n-1},t_n]$ se musí spojitě napojit na řešení na intervalu $(t_n,t_{n+1}]$. Počáteční podmínka se tím pádem musí rovnat koncovému bodu řešení na předchozím intervalu. Pokud nám $\tilde{u}_n(t,u)$ řeší problém (2.1) s počáteční podmínkou u, pak to pro parametry střelby znamená, že musí splňovat následující soustavu rovnic:

$$\begin{aligned} u_0 &= u^0, \\ u_1 &= \tilde{u}_0(t_1,u_0), \\ u_2 &= \tilde{u}_1(t_2,u_1), \\ &\vdots \\ u_N &= \tilde{u}_{N-1}(t_N,u_{N-1}). \end{aligned}$$

Abychom tedy získali naše hodnoty $\{u_n\}_{n=0}^{N-1}$, musíme vyřešit tuto soustavu nelineárních rovnic. Označíme-li si $\mathbf{u} = (u_0,u_1,\ldots,u_N)^T$, pak tuto soustavu rovnic

zapíšeme jako:

$$\mathbf{F}(\mathbf{u}) := \begin{pmatrix} u_0 - u^0 \\ u_1 - \tilde{u}_0(t_1,u_0) \\ u_2 - \tilde{u}_1(t_2,u_1) \\ \vdots \\ u_N - \tilde{u}_{N-1}(t_N,u_{N-1}) \end{pmatrix} = 0.$$

Tímto jsme převedli problém aproximace řešení diferenciální rovnice na problém řešení soustavy nelineárních rovnic určených funkcí $\mathbf{F}$. Pro vyřešení tohoto problému využijeme Newtonovu metodu, kde získáváme aproximace řešení této rovnice pomocí vzorce:

$$\mathbf{u}^{k+1} = \mathbf{u}^k - [\mathbf{F}'(\mathbf{u}^k)]^{-1}\mathbf{F}(\mathbf{u}^k),$$

kde $\mathbf{F}'(\mathbf{u})$ je Jacobiho matice funkce $\mathbf{F}$ v bodě $\mathbf{u}$. V této rovnosti převedeme vektory $\mathbf{u}^{k+1}$ a $\mathbf{u}^k$ na jednu stranu, přenásobíme maticí $\mathbf{F}'(\mathbf{u}^k)$ a dostaneme:

$$\mathbf{F}'(\mathbf{u}^k)(\mathbf{u}^k - \mathbf{u}^{k+1}) = \mathbf{F}(\mathbf{u}^k).$$

Matice $\mathbf{F}'(\mathbf{u})$ má díky tvaru funkce $\mathbf{F}$ jednoduchý bidiagonální tvar. Rovnice pak vypadá následovně:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -\frac{\partial\tilde{u}_0}{\partial u}(t_1,u_0^k) & 1 & 0 & \dots & 0 \\ 0 & -\frac{\partial\tilde{u}_1}{\partial u}(t_2,u_1^k) & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} u_0^k\text{-}u_0^{k+1} \\ u_1^k\text{-}u_1^{k+1} \\ u_2^k\text{-}u_2^{k+1} \\ \vdots \\ u_N^k\text{-}u_N^{k+1} \end{pmatrix} = \begin{pmatrix} u_0^k\text{-}u^0 \\ u_1^k\text{-}\tilde{u}_0(t_1,u_0^k) \\ u_2^k\text{-}\tilde{u}_1(t_2,u_1^k) \\ \vdots \\ u_N^k\text{-}\tilde{u}_{N\text{-}1}(t_N,u_{N\text{-}1}^k) \end{pmatrix}.$$

Pokud si vyjádříme n-tý řádek této rovnice, dostaneme:

$$u_n^k - u_n^{k+1} - \frac{\partial\tilde{u}_{n-1}}{\partial u}(t_n,u_{n-1}^k)(u_{n-1}^k - u_{n-1}^{k+1}) = u_n^k - \tilde{u}_{n-1}(t_n,u_{n-1}^k).$$

Po odečtení prvků u_n^k a vyjádření prvku u_n^{k+1} dostaneme následující vzorec:

$$u_n^{k+1} = \tilde{u}_{n-1}(t_n,u_{n-1}^k) + \frac{\partial\tilde{u}_{n-1}}{\partial u}(t_n,u_{n-1}^k)(u_{n-1}^{k+1} - u_{n-1}^k).$$

To nám tedy umožňuje definovat následující metodu:

Metoda 3 (Metoda vícenásobné střelby)**.** *Metoda vícenásobné střelby má následující tvar:*

$$\begin{aligned} u_0^{k+1} &= u^0, \\ u_n^{k+1} &= \tilde{u}_{n-1}(t_n,u_{n-1}^k) + \frac{\partial\tilde{u}_{n-1}}{\partial u}(t_n,u_{n-1}^k)(u_{n-1}^{k+1} - u_{n-1}^k),\ n = 1,\dots,N. \end{aligned} \tag{2.2}$$

Pomocí tohoto schématu tedy můžeme postupnými iteracemi získávat přesnější aproximaci řešení problému (1.1), ale jeho výpočet není praktický. Hodnotu funkce $\tilde{u}_n(t_{n+1},u_n^k)$ sice můžeme aproximovat pomocí jednokrokových metod, ale ve vzorečku se nám objevuje derivace řešení podle počáteční podmínky. Její aproximaci můžeme získat například pomocí tzv. citlivostních rovnic, viz. (Dickinson

a Gelinas (1976)), což ale není praktické, neboť musíme řešit souběžně s naší diferenciální rovnicí ještě další diferenciální rovnice. My se ale nebudeme praktičností této metody zabývat, neboť ji nebudeme k praktickému výpočtu používat.

Pro nás je důležité to, že díky této metodě jsme schopni využívat paralelní výpočet na několika procesorech najednou, neboť při každé iteraci aproximujeme hodnotu funkce $\tilde{u}_n(t_{n+1},u_n^k)$, a to můžeme provádět paralelně, neboť hodnoty $\{u_n^k\}_{n=0}^{N-1}$ už známe z předchozí iterace. Sice je tato metoda stále sekvenční, neboť pro výpočet u_{n+1}^{k+1} potřebujeme znát u_n^{k+1}, ale část výpočtu se nám podařilo paralelizovat. Dále uvidíme, že to stejné budeme využívat i u metody Parareal: sice bude stále sekvenční, ale díky paralelnímu výpočtu několika částí jsme schopni se dostat k přesnějšímu řešení velice rychle.

2.1.2 Vlastnosti metody vícenásobné střelby

V následující kapitole si ukážeme vlastnosti metody vícenásobné střelby. Díky tomu, že metoda vychází z Newtonovy metody platí následující věta:

Věta 2 (Lokálně kvadratická konvergence, Gander (2018))**.** *Pokud je funkce $f(t,u)$ dvakrát spojitě diferencovatelná podle u, pak metoda vícenásobné střelby konverguje lokálně kvadraticky, tj.*

$$\max_{n=0,\dots,N} |u_n^{k+1} - u(t_n)| \leq C \max_{n=0,\dots,N} |u_n^k - u(t_n)|^2 + O(\max_{n=0,\dots,N} |u_n^k - u(t_n)|^3).$$

To, že se metoda chová takto, je očekávané, avšak následující vlastnost už je unikátní pro tuto metodu.

Věta 3 (Kovergence v konečném počtu kroků)**.** *Pokud $u_0^0 = u^0$, pak má metoda vícenásobné střelby následující vlastnost:*

$$u_n^k = u(t_n) \text{ pro } k \geq n.$$

Důkaz. Důkaz provedeme indukcí podle n.

Pro $n = 0$ platí $u_0^0 = u^0 = u(t_0)$, kde první rovnost platí díky předpokladu a druhá rovnost plyne z počáteční podmínky. Následně ze vzorce pro metodu vícenásobné střelby (2.2) platí $u_0^k = u_0^0 = u(t_0)$ pro $k = 1,2,\dots$, tedy tvrzení platí.

Nyní nechť tvrzení platí pro $n-1$, a my budeme chtít ukázat, že pro $k \geq n$ platí $u_n^k = u(t_n)$. Nechť tedy $k \geq n$. Z toho, že $k \geq n$, plyne, že $k-1 \geq n-1$ a $k \geq n-1$, tedy můžeme použít indukční předpoklad a dostaneme $u_{n-1}^{k-1} = u_{n-1}^k = u(t_{n-1})$. Pro u_n^k tedy ze vzorce pro metodu vícenásobné střelby (2.2) máme:

$$\begin{aligned}
u_n^k &= \tilde{u}_{n-1}(t_n,u_{n-1}^{k-1}) + \frac{\partial \tilde{u}_{n-1}}{\partial u}(t_n,u_{n-1}^{k-1})(u_{n-1}^k - u_{n-1}^{k-1}),\\
&= \tilde{u}_{n-1}(t_n,u_{n-1}^{k-1}) + \frac{\partial \tilde{u}_{n-1}}{\partial u}(t_n,u_{n-1}^{k-1})(u(t_{n-1}) - u(t_{n-1})),\\
&= \tilde{u}_{n-1}(t_n,u_{n-1}^{k-1}),\\
&= \tilde{u}_{n-1}(t_n,u(t_{n-1})),\\
&= u(t_n).
\end{aligned}$$

Poslední rovnost zde platí díky tomu, že pokud $\tilde{u}_{n-1}$ má správnou počáteční podmínku, pak se její funkční hodnoty rovnají u. Tímto je tvrzení dokázáno. □

Tato vlastnost je velice praktická, neboť pokud předpokládáme, že počítáme všechno přesně, tak nám tato vlastnost zaručuje to, že se v konečném čase hodnoty budou rovnat správnému řešení.

Celkově tedy máme jednokrokové sekvenční metody, jejichž výpočet je praktický a jednoduchý, ale které neumožňují paralelní počítání, a nyní máme metodu, která umožňuje část výpočtu počítat paralelně, ale její výpočet je až moc složitý na to, aby byl praktický. V další kapitole tyto dva přístupy zkombinujeme a získáme praktický algoritmus, který dokáže počítat paralelně.

2.2 Metoda Parareal

Metoda Parareal spočívá v tom, že si zavedeme hrubé dělení intervalu $(0,T]$ na podintervaly $(t_{n-1},t_n]$ a dále si tyto intervaly rozdělíme pomocí jemného dělení. Výpočet provádíme tak, že na hrubém dělení získáme rychlou ale špatnou aproximaci pomocí nepřesné metody a pak budeme na jemném dělení aproximaci zlepšovat pomocí přesnější metody.

2.2.1 Propagační operátory

Metodu Parareal si zadefinujeme pomocí následujících propagačních operátorů:

- $G(t_2,t_1,u_1)$ je hrubá aproximace hodnoty $u(t_2)$ řešení problému (1.1) s počáteční podmínkou $u(t_1) = u_1$.
- $F(t_2,t_1,u_1)$ je jemná aproximace hodnoty $u(t_2)$ řešení problému (1.1) s počáteční podmínkou $u(t_1) = u_1$.

Jako G si můžeme představit například explicitní Eulerovu metodu, která nám dá levnou ale nepřesnou aproximaci, a jako F si můžeme představit například Runge-Kuttovu metodu 4. řádu, kterou ještě zpřesníme zjemněním dělení (t_1,t_2). Pro nás je důležité to, že vyhodnocení G je rychlé, ale dává nepřesnou aproximaci, a že výpočet F dává dobrou aproximaci, ale je výpočetně náročný. Algoritmus Parareal funguje tak, že počítáme sekvenčně pouze pomocí G, což je levné, a hodnoty F budeme počítat paralelně, díky čemuž jsme schopni urychlit výpočet.

2.2.2 Odvození metody Parareal

Abychom si odvodili vzoreček pro metodu Parareal pomocí propagačních operátorů, podívejme se zpět na metodu vícenásobné střelby, tj. na vzoreček (2.2):

$$u_{n+1}^{k+1} = \tilde{u}_n(t_{n+1},u_n^k) + \frac{\partial \tilde{u}_n}{\partial u}(t_{n+1},u_n^k)(u_n^{k+1} - u_n^k).$$

Člen $\tilde{u}_n(t_{n+1},u_n^k)$ si aproximujeme pomocí hodnoty $F(t_{n+1},t_n,u_n^k)$. Nyní předpokládejme, že řešení jsou dostatečně hladká, a my si rozvineme $\tilde{u}_n(t_{n+1},u_n^{k+1})$ do Taylorova rozvoje podle bodu u_n^k, díky čemuž dostaneme:

$$\tilde{u}_n(t_{n+1},u_n^{k+1}) = \tilde{u}_n(t_{n+1},u_n^k) + \frac{\partial \tilde{u}_n}{\partial u}(t_{n+1},u_n^k)(u_n^{k+1} - u_n^k) + O((u_n^{k+1} - u_n^k)^2),$$

z čehož nám plyne:

$$\frac{\partial \tilde{u}_n}{\partial u}(t_{n+1},u_n^k)(u_n^{k+1}-u_n^k) \approx \tilde{u}_n(t_{n+1},u_n^{k+1}) - \tilde{u}_n(t_{n+1},u_n^k).$$

My si tyto hodnoty aproximujeme pomocí propagátoru G, což nás vede k následujícímu odhadu:

$$\frac{\partial \tilde{u}_n}{\partial u}(t_{n+1},u_n^k)(u_n^{k+1}-u_n^k) \approx G(t_{n+1},t_n,u_n^{k+1}) - G(t_{n+1},t_n,u_n^k).$$

Toto nás tedy vede na vzoreček metody Parareal, který vypadá následovně:

$$u_{n+1}^{k+1} = F(t_{n+1},t_n,u_n^k) + G(t_{n+1},t_n,u_n^{k+1}) - G(t_{n+1},t_n,u_n^k).$$

Metoda Parareal pak tedy funguje takto:

Metoda 4 (Parareal)**.** *Interval* $(0,T]$ *si rozdělíme na intervaly* $(t_{n-1},t_n]$, $n = 1,\ldots,N$. *Prvně spočítáme sekvenčně hrubou aproximaci řešení pomocí propagačního operátoru* G, *tedy:*

$$\begin{aligned} u_0^0 &= u^0, \\ u_n^0 &= G(t_n,t_{n-1},u_{n-1}^0). \end{aligned} \tag{2.3}$$

Následně pak pro $k = 0,1,\ldots$ *počítáme korekční iterace:*

$$\begin{aligned} u_0^{k+1} &= u^0, \\ u_n^{k+1} &= F(t_n,t_{n-1},u_{n-1}^k) + G(t_n,t_{n-1},u_{n-1}^{k+1}) - G(t_n,t_{n-1},u_{n-1}^k). \end{aligned} \tag{2.4}$$

Můžeme vidět, že každá korekční iterace je sekvenční, neboť pro výpočet u_n^{k+1} musíme znát u_{n-1}^{k+1}, ale počítáme sekvenčně pouze pomocí G, což není drahé. Hodnoty které vkladáme do propagátoru F už známe z minulé korekční iterace, tedy nejdražší výpočet jsme schopni provádět paralelně. Tímto jsme tedy konečně dostali algoritmus který je prakticky použitelný a který je schopný část výpočtů provádět paralelně. Ještě ale nevíme, jaké má tento algoritmus vlastnosti a jestli je dostatečně efektivní, na to se podíváme v následující kapitole.

2.2.3 Vlastnosti metody Parareal

Ukážeme si, že metoda Parareal má stejně jako metoda vícenásobné střelby vlastnost konvergence v konečném počtku kroků, v tomto případě ale konvergujeme k jemné aproximaci řešení dané propagačním operátorem F.

Věta 4 (Konvergence v konečném počtu kroků)**.** *Metoda Parareal (2.3), (2.4) má následující vlastnost:*

$$u_n^k = F(t_n,0,u^0) \text{ pro } k \geq n,\ n = 1,\ldots,N.$$

Důkaz. Důkaz budeme provádět analogicky jako pro metodu vícenásobné střelby, tedy matematickou indukcí podle n.

Pro $n = 1$ víme z (2.3), že $u_0^k = u^0$, $k = 0,1,\ldots$, tedy počáteční počáteční podmínka se nemění. Nechť $k \geq 1$. Pak pro u_1^k ze vzorce (2.4):

$$\begin{aligned} u_1^k &= F(t_1,0,u_0^{k-1}) + G(t_1,0,u_0^k) - G(t_1,0,u_0^{k-1}), \\ &= F(t_1,0,u^0) + G(t_1,0,u^0) - G(t_1,0,u^0), \\ &= F(t_1,0,u^0). \end{aligned}$$

Tedy tvrzení platí.

Předpokládejme tedy, že tvrzení platí pro $n-1$ a nechť $k \geq n$. Z toho plyne, že $k-1 \geq n-1$ a $k \geq n-1$, tedy díky indukčnímu předpokladu dostaneme $u_{n-1}^{k-1} = u_{n-1}^k = f(t_{n-1},0,u^0)$. Pro u_n^k ze vzorce (2.4) máme:

$$\begin{aligned} u_n^k =& F(t_n,t_{n-1},u_{n-1}^{k-1}) + G(t_n,t_{n-1},u_{n-1}^k) - G(t_n,t_{n-1},u_{n-1}^{k-1}), \\ =& F(t_n,t_{n-1},u_{n-1}^{k-1}) + G(t_n,t_{n-1},F(t_{n-1},0,u^0)) - G(t_n,t_{n-1},F(t_{n-1},0,u^0)), \\ =& F(t_n,t_{n-1},u_{n-1}^{k-1}), \\ =& F(t_n,t_{n-1},F(t_{n-1},0,u^0)), \\ =& F(t_n,0,u^0). \end{aligned}$$

Tímto je důkaz dokončen. □

Tato vlastnost zaručuje to, že v přesné aritmetice se v N krocích vždy dostaneme k jemné aproximaci původní rovnice. Tím ale bohužel nedosáhneme žádného zrychlení, neboť v každé iteraci počítáme jedenkrát paralelně pomocí F, tedy jsme počítali pomocí F N-krát, což bychom ale počítali i tehdy, kdybychom F vyhodnocovali sekvenčně. V tomto případě tedy nedochází k žádnému zrychlení, věta nám pouze zaručuje to, že v nejhorším případě se vždy k jemné aproximaci dostaneme. V praktické části ale uvidíme, že algoritmus Parareal obecně konverguje dostatečně blízko k jemné aproximaci výrazně dříve.

Další věta už nám toho ale o síle této metody řekne více.

Věta 5 (Odhad chyby metody Parareal, Gander a Hairer (2007))**.** *Nechť F dává přesnou hodnotu řešení problému (1.1) a nechť G je jednokroková metoda, pro jejíž lokální diskretizační chybu platí $|\tau(t,h)| \leq C_3 h^p$, a která je lispchitzovská, tj.:*

$$||G(t+h,t,u) - G(t+h,t,v)|| \leq (1 + C_2 h)||u - v||.$$

Pokud navíc platí pro h dostatečně malé:

$$F(t_{n+1},t_n,x) - G(t_{n+1},t_n,x) = c_p(x)h^p + c_{p+1}(x)h^{p+1} + \ldots, \tag{2.5}$$

kde c_j, $j = p, p+1, \ldots$ jsou spojitě diferencovatelné, pak v k-té korekční iteraci metody Parareal (2.3), (2.4) platí následující odhad:

$$||u_n^k - u(t_n)|| \leq \frac{C_3(C_1 t_n)^{k+1}}{C_1(k+1)!} e^{C_2(t_n - t_{k+1})} h^{pk},$$

kde konstanta C_1 závisí na funkcích c_j.

Důsledkem této věty je to, že pokud předpokládáme, že nám F dává přesné řešení problému (1.1), pak každou korekční iterací zvyšujeme řád metody Parareal o řád operátoru G. Věta má sice ještě předpoklad (2.5), naštěstí lze ukázat (Gander

a Hairer (2007)), že pokud je f dostatečně hladká a G je dáno například Runge-Kuttovou metodou, pak je tento předpoklad splněn. Předpoklad toho, že F nám řeší rovnici přesně je sice v praxi nemožný splnit, v numerických experimentech ale uvidíme, že pokud bude F dostatečně přesný, pak bude tato vlastnost platit. Metoda Parareal je díky této vlastnosti velice praktická, neboť nám umožňuje globálně zvyšovat řád metody pomocí paralelních výpočtů.

3. Numerické experimenty

V následující kapitole si na příkladech otestujeme metodu Parareal. Nebudeme bohužel využívat paralelní programování a hodnoty propagačního operátoru F budeme počítat sekvenčně, ale to našim experimentům vadit nebude. Numerické experimenty jsou prováděny v prostředí Matlab.

3.1 Detaily implementace

V algoritmu Parareal potřebujeme metody pro řešení problému (1.1). My použijeme explicitní Eulerovu metodu jako propagační operátor G na hrubém dělení a Runge-Kuttovu metodu 4. řádu jako propagační operátor F na jemném dělení. Jejich implementace vypadá následovně:

```
function u = ExpEuler(u1,t1,H,n)
%funkce ktera pomoci explicitni Eulerovy metody provede
%n kroku velikosti H vychazejici z bodu (t1,u1)
%vystup je posledni hodnota teto metody
u = u1;
for i = 1:n
    u = u + H*f(t1+(H*(i-1)),u);
end

function u = RungeKutta(u1,t1,H,n)
%funkce ktera pomoci Runge-Kuttovy metody 4. radu provede
%n kroku o velikosti H vychazejici z bodu (t1,u1)
%vystup je posledni hodnota teto metody
u = u1;
for i = 1:n
    k1 = H*f(t1+H*(i-1),u);
    k2 = H*f(t1+H*(i-1)+H/2,u+k1/2);
    k3 = H*f(t1+H*(i-1)+H/2,u+k2/2);
    k4 = H*f(t1+H*(i-1)+H,u+k3);
    u = u+(k1+2*k2+2*k3+k4)/6;
end
```

Algoritmus Parareal má zaručenou konvergenci v N krocích, ale jak už bylo zmíněno dříve, tak od metody vyžadujeme rychlejší konvergenci, jinak by bylo zbytečné tuto metodu používat. Proto bychom chtěli najít zastavovací kritérium tohoto algoritmu. Jako toto zastavovací kritérium budeme používat kritérium $\max_{n=0,\dots,N} |u_n^{k+1} - u_n^k| \leq tol$, kde požadovanou toleranci si sami určíme (například 10^{-6}). Kód algoritmu Parareal pak může vypadat následovně:

```
function U = Parareal(u0,N,h,j,tol)
%u0...pocatecni podminka, tj u(0)=u0
%N...pocet uzlu, ve kterych budeme reseni aproximovat
%h...hruby krok metody
%j...zjemneni hrubeho kroku, aneb kolikrat bude
```

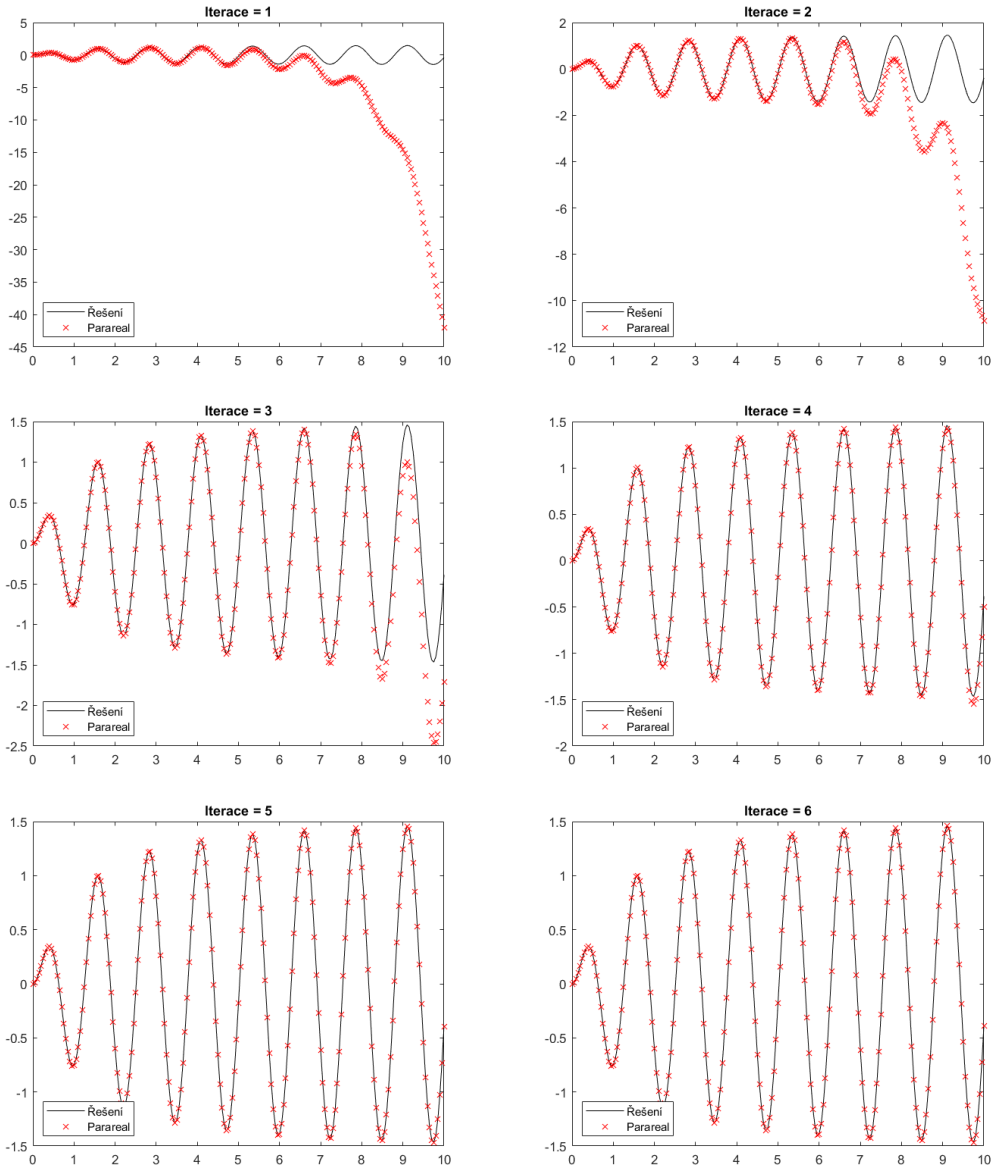
```
%jemny krok mensi nez hruby
%tol...zastavovaci kriterium
%vystup je posledni aproximace, u ktere se algoritmus zastavi
%pomoci zastavovaciho kriteria
hj = h/j;
uzly = zeros(N+1,1);
for i = 1:N+1
    uzly(i,1) = (i-1)*h;
end
%prvni hruba aproximace reseni
GO = zeros(N+1,1);
GO(1) = u0;
for i = 1:N
    GO(i+1) = ExpEuler(GO(i),(i-1)*h,h,1);
end
GN = zeros(N+1,1);
GN(1) = u0;
U = GO;
UO = zeros(N+1,1);
F = zeros(N+1,1);
%korekcni iterace
it = 0;
while norm(UO - U,Inf) > tol
    UO = U;
    it = it+1;
    %paralelni vypocet pomoci jemneho propagatoru
    for i = it:N
        F(i+1) = RungeKutta(U(i),(i-1)*h,hj,j);
    end
    %sekvencni korekcni iterace
    for i = it:N
        GN(i+1) = ExpEuler(U(i),(i-1)*h,h,1);
        U(i+1) = F(i+1) + GN(i+1) - GO(i+1);
    end
    GO = GN;
end
```

3.2 Rychlost konvergence

Rychlost kovergence algoritmu Parareal si ukážeme na následujícím problému:

$$\begin{aligned} u'(t) &= \left(\frac{1}{1+t^2} - arctan(t)\right) sin(5t) + 5arctan(t)cos(5t) + u, \\ u(0) &= 0. \end{aligned} \tag{3.1}$$

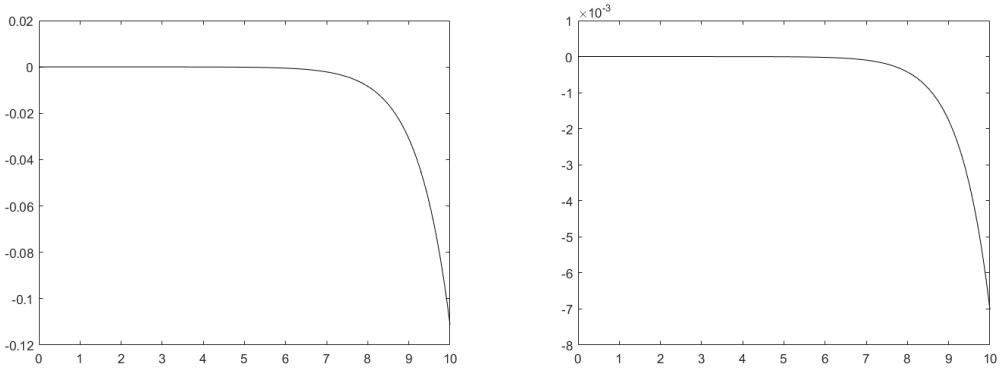
Tento problém řeší funkce $u(t) = arctan(t)sin(5t)$. Na intervalu (0,10] si určíme 200 rovnoměrně rozdělených uzlů a budeme na nich aproximovat řešení pomocí algoritmu Parareal. Toto dělení si zjemníme stokrát při používání jemného řešiče.



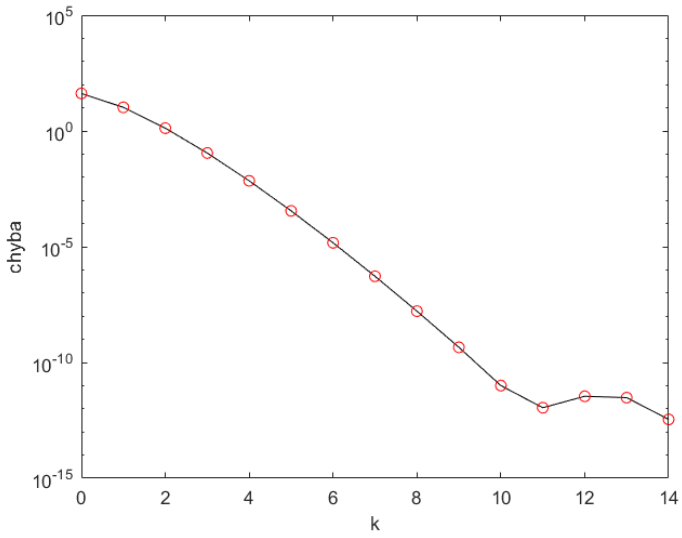
Obrázek 3.1: Konvergence algoritmu Parareal při řešení problému (3.1)

Algoritmus nám zaručeně konverguje k jemné aproximaci, která se téměř rovná přesnému řešení, ve 200 korekčních iteracích. Na Obrázku 3.1 ale vidíme, že už v 6. iteraci nám alogirtmus zkonvergoval k přesnému řešení velice blízko. Dále můžeme vidět na Obrázku 3.3, že se chyba snižuje až do jedenácté iterace. Zde už začne chyba být tak malá, že se začnou projevovat zaokrouhlovací chyby a chyba se zde už začne snižovat pomaleji, někdy i narůstat.

Počítejme tedy jedenáctou korekční iteraci jako bod, kdy jsme dokonvergovali k jemné aproximaci. Při každém výpočtu jsme sekvenčně vyhodnocovali funkci f 200krát a pokud předpokládáme, že máme dvě stě procesorů, na kterých můžeme provádět výpočet paralelně, pak funkci f vyhodnocujeme sekvenčně při každé korekční iteraci 1000krát. V jedenácté korekční iteraci jsme tedy provedli 11200 sekvenčních vyhodnocení funkce f, a kdybychom používali pouze jemný řešič, vyhodnocovali bychom funkci sekvenčně 80000krát. V tomto případě je tedy algoritmus Parareal zhruba $\frac{80000}{11200} \approx 7$krát rychlejší než jemné řešení.



Obrázek 3.2: Graf chyby $u(t_n) - u_n^k$ v 5. té a 6. té iteraci při řešení problému (3.1)



Obrázek 3.3: Velikost chyby v maximové normě algoritmu Parareal při řešení problému (3.1)

3.3 Řád metody Parareal

V této kapitole si ověříme, že algoritmus Parareal splňuje vlastnost zvyšování řádu metody při každé korekční iteraci. K tomu, abychom to mohli udělat, potřebujeme najít způsob, kterým jsme schopni změřit řád metody. Budeme ho měřit pomocí metody polovičního kroku, která funguje následovně:

Mějmě interval $(0,T]$ a mějme dvoje rovnoměrná dělení $\{t_n\}_{n=0}^{N}$ a $\{\tilde{t}_m\}_{m=0}^{2N-1}$, tedy pokud si krok prvního dělení označíme h, pak krok druhého dělení je rovný $h/2$. Použijeme-li na těchto dvou děleních numerickou metodu řádu p, pak dostaneme posloupnosti $\{u_n\}_{n=0}^{N}$ a $\{\tilde{u}_m\}_{m=0}^{2N-1}$. Označme si:

$$e^{(h)} = \max_{n=1,\dots,N} |u(t_n) - u_n|,$$
$$e^{(h/2)} = \max_{m=1,\dots,2N-1} |u(\tilde{t}_n) - \tilde{u}_n|.$$

Metoda je řádu p, tedy můžeme očekávat:

$$e^{(h)} \approx Ch^p,$$
$$e^{(h/2)} \approx C(h/2)^p.$$

Vydělíme-li tyto dvě aproximace, dostaneme následující vzorec:

$$\frac{e^{(h)}}{e^{(h/2)}} \approx \frac{Ch^p}{C(h/2)^p} = 2^p.$$

Pokud tuto aproximaci zlogaritmujeme, dostaneme:

$$p \approx \frac{log(e^{(h)}/e^{(h/2)})}{log(2)}. \tag{3.2}$$

My tedy provedeme aproximace řešení problému (2.1) u nichž budeme snižovat krok o polovinu, u nich spočteme maximální chybu metody a pak pomocí vzorce (3.2) spočteme koeficienty α, tj.

$$\alpha = \frac{log(e^{(h)}/e^{(h/2)})}{log(2)}.$$

Tyto koeficienty α by měly odpovídat řádu metody. To provedeme pro různé korekční iterace. Jelikož jsme používali jako hrubý řešič explicitní Eulerovu metodu, tak by se měl řád každou korekční iterací zvýšit o 1. Z Tabulky 3.1 můžeme vidět, že metoda Parareal tuto vlastnost skutečně splňuje. Jediné místo, kde tato vlastnost splněná není, je při velmi malém kroku u 7. a 8. korekční iterace, ale to je způsobeno zaokrouhlovacími chybami.

	Hrubá aproximace		1. korekční iterace		2. korekční iterace	
h	$max\|u(t_n) - u_n^0\|$	α	$max\|u(t_n) - u_n^1\|$	α	$max\|u(t_n) - u_n^2\|$	α
0.2	273.286		241.231		104.777	
0.1	115.879	1.24	55.1648	2.13	13.0842	3.00
0.05	41.6579	1.48	10.4746	2.40	1.32245	3.31
0.025	14.1927	1.55	1.87218	2.48	0.12460	3.41
0.0125	4.99658	1.50	0.34523	2.44	0.01205	3.37
	3. korekční iterace		4. korekční iterace		5. korekční iterace	
h	$max\|u(t_n) - u_n^3\|$	α	$max\|u(t_n) - u_n^4\|$	α	$max\|u(t_n) - u_n^5\|$	α
0.2	29.7852		6.23286		1.02376	
0.1	2.05568	3.86	0.24063	4.69	0.02337	5.52
0.05	0.11128	4.21	0.00702	5.10	0.00035	5.98
0.025	0.00554	4.33	0.00018	5.25	$4.9e-06$	6.16
0.0125	0.00028	4.30	$4.8e-06$	5.24	$6.8e-08$	6.18
	6. korekční iterace		7. korekční iterace		8. korekční iterace	
h	$max\|u(t_n) - u_n^6\|$	α	$max\|u(t_n) - u_n^7\|$	α	$max\|u(t_n) - u_n^8\|$	α
0.2	0.13742		0.01550		0.00150	
0.1	0.00172	6.32	0.00011	7.10	$6.8e-06$	7.87
0.05	$1.4e-05$	6.86	$5.2e-07$	7.74	$1.6e-08$	8.62
0.025	$1.1e-07$	7.08	$2.1e-09$	8.00	$2.9e-11$	9.09
0.0125	$8.3e-10$	7.03	$3.3e-11$	5.96	$3.4e-11$	-0.21

Tabulka 3.1: Konvergenční tabulka algoritmu Parareal při řešení problému (3.1)

3.4 Nelineární diferenciální rovnice

V minulé sekci jsme pomocí metody Parareal řešili problém (3.1), který je lineární a pro který vycházeli velmi pěkné výsledky. V této kapitole se podíváme, jestli metoda Parareal funguje dobře i pro nelineární problémy.

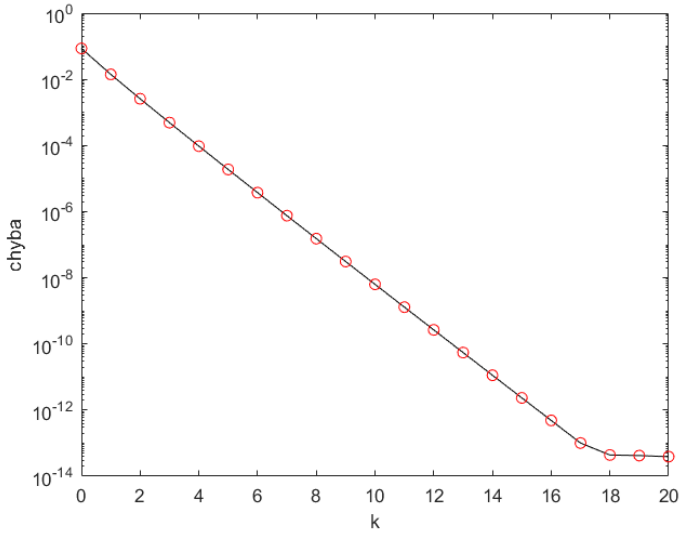
3.4.1 První diferenciální rovnice

Mějme následující nelineární diferenciální rovnici:

$$\begin{aligned} u'(t) &= sin(u(t)), \\ u(0) &= 1, \end{aligned} \tag{3.3}$$

Tento problém budeme aproximovat na intervalu (0,50] v 100 uzlech s hrubým krokem $h = 0.5$ a s jemným krokem $\tilde{h} = 0.005$. Chybu metody budeme počítat srovnáním s jemnou aproximací.

Na Obrázku 3.4 můžeme vidět, že pro tento problém funguje algoritmus Parareal opět velice dobře, každou korekční iterací se snižuje chyba exponenciálně, dokud nezačnou být chyby tak malé, že se začnou projevovat zaokrouhlovací chyby a algoritmus přestane konvergovat. V tomto případě dosahujeme zhruba čtyřnásobného zrychlení.



Obrázek 3.4: Velikost chyby v maximové normě algoritmu Parareal při řešení problému (3.3)

3.4.2 Druhá diferenciální rovnice

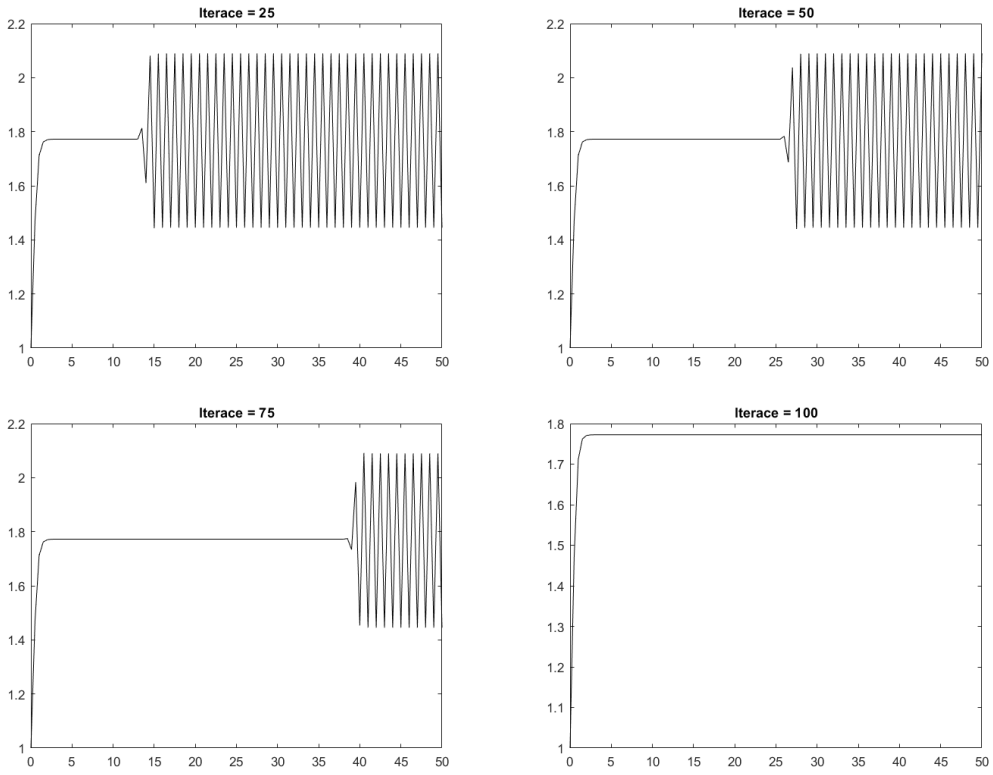
Pro předchozí diferenciální rovnici nám algortimus Parareal fungoval dobře, nyní si tedy uveďme následující diferenciální rovnici:

$$\begin{aligned} u'(t) &= sin(u(t)^2), \\ u(0) &= 1. \end{aligned} \tag{3.4}$$

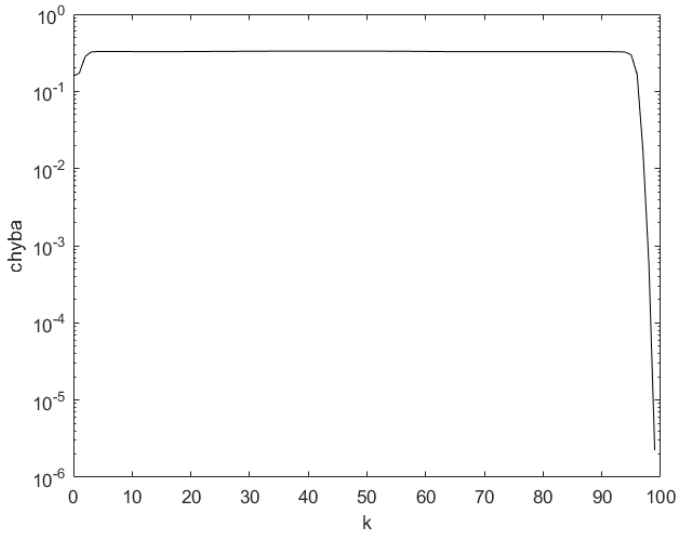
Opět budeme problém aproximovat na intervalu (0,50] ve 100 uzlech, s hrubým krokem $h = 0.5$ a jemným krokem $\tilde{h} = 0.005$. Chybu metody budeme opět počítat srovnáním s jemnou aproximací. Na Obrázku 3.5 můžeme vidět, jak se k jemné aproximaci přibližujeme pomocí algoritmu Parareal, ve sté korekční iteraci se rovná jemné aproximaci díky Větě 4.

Můžeme vidět, že zde začne aproximace oscilovat a k přesnému řešení se přibližujeme pouze tehdy, když máme zaručenou konvergenci díky Větě 4, což nevede k žádnému zrychlení. Na Obrázku 3.6 vidíme, že nám algortimus zkonverguje dostatečně blízko k jemné aproximaci až v posledních korekčních iteracích. Pro tento problém nám tedy algoritmus Parareal nefunguje.

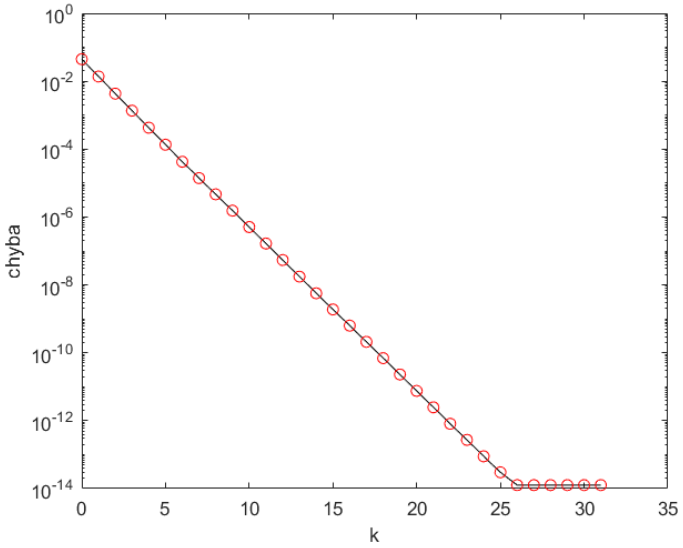
Tohoto chování se ale naštěstí můžeme zbavit zmenšením kroku h, tedy je toto chování pravděpodobně způsobeno nestabilitou explicitní Eulerovy metody. Pokud při aproximaci problému (3.4) zmenšíme hrubý krok na polovinu, tj. $h = 0.25$, algoritmus nám už začne fungovat dobře, jak můžeme vidět na Obrázku 3.7. Vidíme, že algoritmus se v tomto případě chová tak, jak bychom očekávali, zrychlení je zhruba pětinásobné.



Obrázek 3.5: Aproximace řešení problému (3.4) pomocí algoritmu Parareal v různých korekčních iteracích



Obrázek 3.6: Velikost chyby v maximové normě algoritmu Parareal při řešení problému (3.4) pro $h = 0.5$



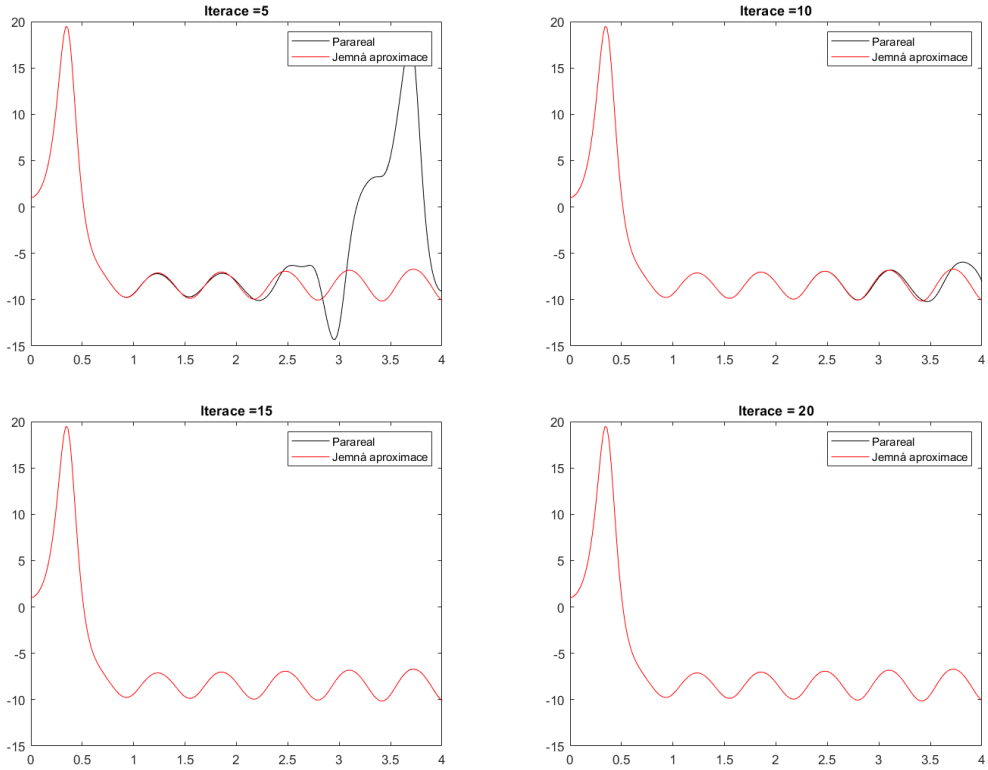
Obrázek 3.7: Velikost chyby v maximové normě algoritmu Parareal při řešení problému (3.4) pro $h = 0.25$

3.5 Soustava obyčejných diferenciálních rovnic

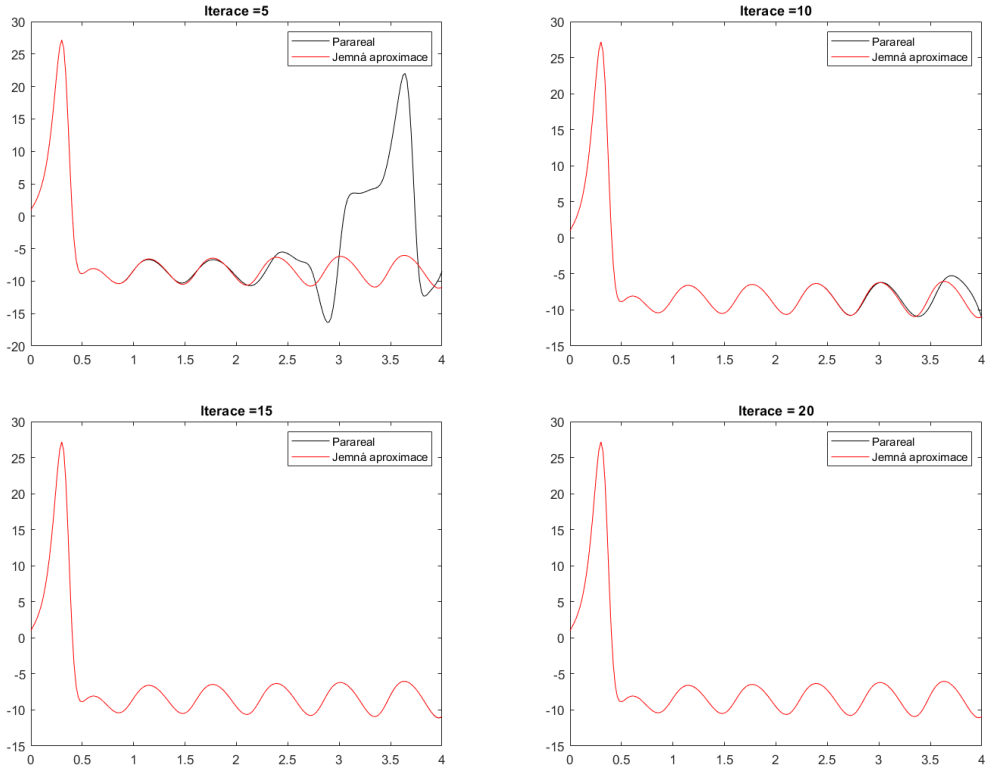
Nakonec se podíváme na aplikaci algoritmu Parareal na soustavu obyčejných diferenciálních rovnic. Algoritmus Parareal funguje na stejném principu pro soustavy obyčejných diferenciálních rovnic, jako pro jednu obyčejnou diferenciální rovnici. Budeme uvažovat Lorenzův systém (Lorenz (1969)) s parametry $\rho = 28$, $\sigma = 10$ a $\beta = \frac{8}{3}$:

$$\begin{aligned} u_1'(t) &= 10(u_2(t) - u_1(t)), \\ u_2'(t) &= u_1(t)(28 - u_3(t)) - u_2(t), \\ u_3'(t) &= u_1(t)u_2(t) - \tfrac{8}{3}u_3(t), \\ u_1(0) &= u_2(0) = u_3(0) = 1. \end{aligned} \tag{3.5}$$

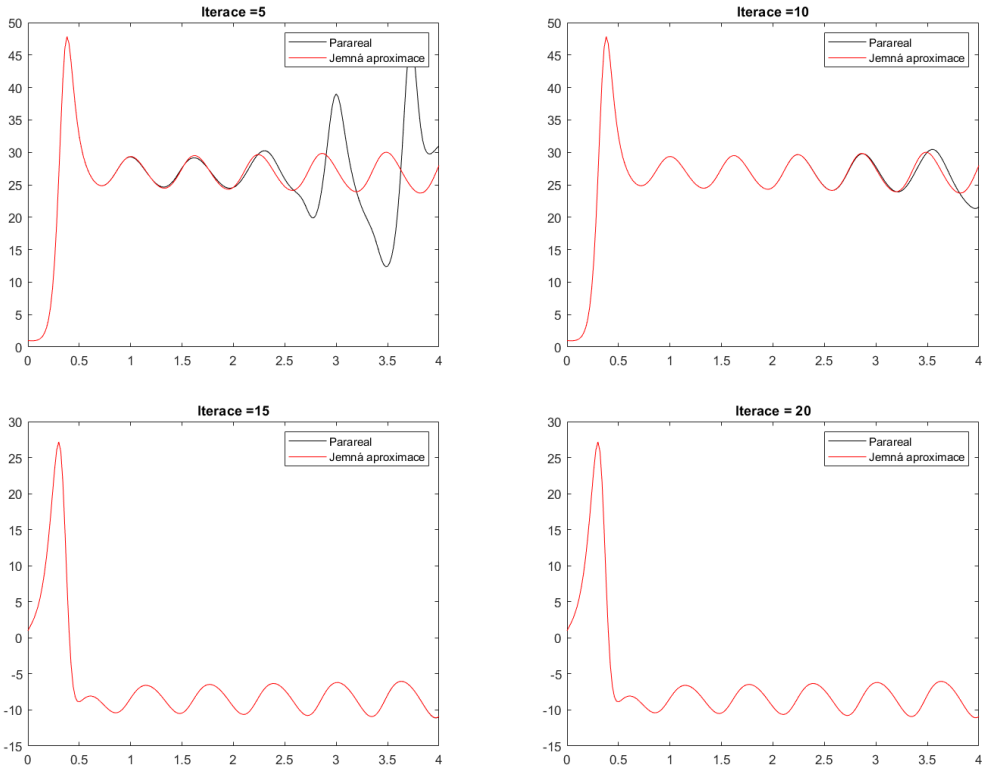
Systém budeme řešit na 200 uzlech s hrubým krokem $h = 0.02$ a jemným krokem $\tilde{h} = 0.0002$. Lorenzův systém je systém tří nelineárních diferenciálních rovnic, díky kterému se dá modelovat například atmosférický pohyb, a který je známý díky chaotickému chování řešení tohoto systému. Toto řešení nejsme schopni najít analyticky, budeme tedy porovnávat aproximaci dané algoritmem Parareal s jemnou aproximací. Na Obrázcích 3.8, 3.9, 3.10 můžeme vidět, že zde algoritmus Parareal opět funguje dobře a konverguje rychle k jemné aproximaci. Na Obrázku 3.11 můžeme vidět, že opět chybu snižujeme až do té doby, kdy se začnou projevovat zaokrouhlovací chyby, tedy algoritmus je zde opět efektivní, dosahujeme téměř pětinásobného zrychlení.



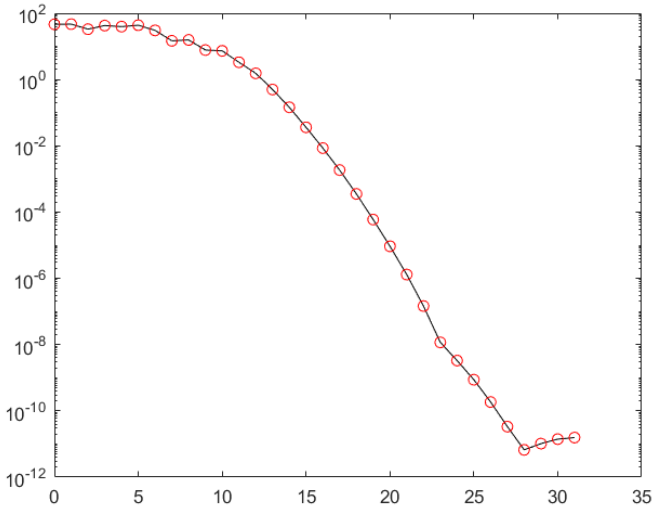
Obrázek 3.8: Aproximace $u_1(t)$ z problému (3.5) pomocí algoritmu Parareal



Obrázek 3.9: Aproximace $u_2(t)$ z problému (3.5) pomocí algoritmu Parareal



Obrázek 3.10: Aproximace $u_3(t)$ z problému (3.5) pomocí algoritmu Parareal



Obrázek 3.11: Chyba algoritmu Parareal v maximové normě při řešení problému (3.5)

Závěr

Ukázali jsme si, že je skutečně možné urychlit výpočet numerického řešení diferenciálních rovnic pomocí paralelních výpočtů. Algoritmus Parareal jsme si odvodili pomocí metody vícenásobné střelby, ale v minulosti byl odvozen téměř nezávisle na jakékoliv předchozí práci, a až posléze byly nalezeny souvislosti s ostatními metodami, viz (Gander (2013)). My jsme si ukázali na lineárních a nelineárních problémech, že doopravdy funguje, a že teoreticky dosahujeme až několikanásobného zrychlení. Prakticky dosahujeme zrychlení menšího, neboť vznikají další časové prodlevy, například při komunikaci mezi procesory. Algoritmus Parareal také není efektivní vzhledem k počtu vyhodnocení funkce f a při jeho používání provádíme obrovské množství výpočtů. Jejich množství je mnohonásobně vyšší než zrychlení, kterého dosahujeme. Důležité je pro nás ale to, že jsme schopni dosáhnout zrychlení v problému, u kterého se zdálo, že žádného zrychlení dosáhnout nelze. Toho můžeme využít pro případy, kdy je získání přesné aproximace omezené časem.

Seznam použité literatury

Bellen, A. a Zennaro, M. (1989). Parallel algorithms for initial-value problems for difference and differential equations. *J. Comput. Appl. Math.*, **25**, 341–350.

Butcher, J. C. (1996). A history of Runge-Kutta methods. *Applied numerical mathematics*, **20**(3), 247–260.

Chartier, P. a Phillipe, B. (1993). A parallel shooting technique for solving dissipative ODEs. *Computing*, **51**, 209–236.

Dickinson, R. P. a Gelinas, R. J. (1976). Sensitivity analysis of ordinary differential equation systems—a direct method. *Journal of Computational Physics*, **21**(2), 123–143.

Gander, M. J. (2013). 50 years of time parallel time integration. In Carraro, T., Geiger, M., Körkel, S. a Rannacher, R., editors, *Multiple Shooting and Time Domain Decomposition*, pages 69–114. Springer-Verlag.

Gander, M. J. a Hairer, E. (2007). Nonlinear convergence analysis for the Parareal algorithm. In Langer, U., Discacciati, M., Keyes, D. E., Widlund, O. B. a Zulehner, W., editors, *Domain Decomposition Methods in Science and Engineering XVII, Lecture Notes in Computational Science and Engineering 60*, pages 45–56. Springer-Verlag.

Gander, M. J. (2018). Time parallel time integration. *Avaivable online, https://www.unige.ch/~gander/poly.pdf, University of Geneva.*

Kučera, V. a Feistauer, M. (2014). *Základy numerické matematiky.* Matfyzpress, Praha.

Lions, J.-L., Maday, Y. a Turinici, G. (2001). A "parareal" in time discretization of PDEs. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, **332**, 661–668.

Lorenz, E. N. (1969). Atmospheric predictability as revealed by naturally occurring analogues. *Journal of Atmospheric Sciences*, **26**(4), 636–646.