Přehled domácích úkolů z NUM2

Tomáš Oberhuber

4. března 2025

1 Zdrojové kódy

K vypracování většiny domácích úkolů budete potřebovat již hotové zdrojové kódy, které budete pouze upravovat pro účely jednotlivých úloh. Tyto zdrojové kódy lze najít zde:

https://gitlab.com/oberhuber.tomas/fjfi-num-src.

Stahovat lze pomocí:

```
git clone https://gitlab.com/oberhuber.tomas/fjfi-num-src-python.git
```

nebo tlačítkem *Download*.
Rozbalíme příkazem

```
tar xvf fjfi-num-src-python*
cd fjfi-num-src-python*
```

Vysvětlení, jak s kódy pracovat, se dozvíte v průběhu cvičení.

2 Přehled domácích úkolů

Následuje přehled všech úkolů za celý semestr. Každý úkol je uveden a vysvětlen v prezentacích a doplňujících videích. Tento dokument slouží spíše pro snazší celkový přehled, jaké úkoly máte vypracovat.

2.1 Konečné diference

Cílem tohoto úkolu je připomenout si konečné diference z přednášky z numeriky ve druhém ročníku a pocvičit si jejich odvození. To je velice důležité např. pro metodu konečných diferencí.

Pomocí Taylorových rozvojů dokažte **jedno** z následujících tvrzení:

1. Nechť $f \in C^{(5)}(x_{-l_1}, x_{l_2})$. Pak

$$\frac{1}{12h}(-f_2 + 8f_1 - 8f_{-1} + f_{-2}) = f'(x_0) + O(h^4).$$

2. Nechť $f \in C^{(5)}(x_{-l_1}, x_{l_2})$. Pak

$$\frac{1}{2h^3}(f_2 - 2f_1 + 2f_{-1} - f_{-2}) = f^{(3)}(x_0) + O(h^2).$$

3. Nechť $f \in C^{(6)}(x_{-l_1}, x_{l_2})$. Pak

$$\frac{1}{h^4}(f_2 - 4f_1 + 6f_0 - 4f_{-1} + f_{-2}) = f^{(4)}(x_0) + O(h^2).$$

4. Nechť $f \in C^{(3)}(x_{-l_1}, x_{l_2})$. Pak

$$\frac{-1}{2h}(f_2 - 4f_1 + 3f_0) = f'(x_0) + O(h^2).$$

2.2 Implementace Rungovy-Kuttovy metody

Cílem tohoto úkolu je ozkoušet si přechod od numerické metody zapsané na papíře k její implementaci v C++.

Proveďte implementaci Rungovy-Kuttovy metody druhého řádu:

1. do souboru `ODE/RungeKutta.py` dopište kód pro Rungovu-Kuttovu metodu druhého řádu
2. zdrojové kódy vytisknete a odevzdejte

2.3 Numerické řešení Riccatiho rovnice

Cíle tohoto úkolu je ozkoušet si některé potíže, které vznikají při numerickém řešení obyčejných diferenciálních rovnic. Zároveň se také naučit odvozovat experimentální řád konvergence - EOC. Nakonec je také cílem naučit se napsat report s výsledky a přesným popisem nastavení, za jakého byly tyto výsledky obdrženy.

Implementujte řešič pro Riccatiho rovnici a proveďte porovnání s analytickým řešením:

1. zvolte vhodný interval, kde je možné napočítat numericky řešení Ricattiho rovnice
2. napočítejte *experimentální řád konvergence - EOC* pro Eulerův a vámi implementovaný Rungův-Kuttův řešič
3. napište report k numerické analýze
    - řešená rovnice

- interval, na kterém ji řešíme
- počáteční podmínka
- ~~parametry úlohy~~
- zvolená numerická metoda
- parametry numerické metody - integrační časový krok,...
- tabulka EOC

2.4 Numerické řešení Volterrovy-Lotkovy úlohy

Cílem tohoto úkolu je řešení úlohy, které může vykazovat různé chování pro různé nastavení parametrů. Pak je často užitečné udělat si analýzu takové rovnice a ozkoušet si, jak se chová pro různá nastavení.

Proveďte numerickou analýzu, tj. pokuste se najít zajímavou kombinaci parametrů a, b, c a d ve Volterrově-Lotkově úloze z prezentace a napište opět report doplněný obrázky.

1. řešená rovnice
2. interval, na kterém ji řešíme
3. počáteční podmínka
4. parametry úlohy
5. zvolená numerická metoda
6. parametry numerické metody
7. obrázky s výsledky

2.5 Numerické řešení Lorentzova atraktoru

Cílem tohoto úkolu je osahat si řešení známé úlohy z teorie chaosu.

Proveďte numerickou studii Lorenzových rovnic a pokuste se najít Lorenzův atraktor. Napište report podobně jako v předchozích úkolech.

2.6 Numerické řešení epidemiologického SIR modelu

Cílem tohoto úkolu je ozkoušet si kompletní implementaci a výpočetní studii na úloze SIR modelu, který je zřejmě nejoblíbenějším epidemiologickým modelem.

Implementujte za pomocí existujících zdrojových kódů řešič pro SIR model.

1. implementujte tento model
2. proveďte výpočetní studii, tj. různá nastavení parametrů
3. (na internetu najděte parametry pro reálné epidemie a porovnejte s výpočtem)

2.7 Numerické řešení Poissonovy úlohy ve 2D

Cílem tohoto úkolu je pocvičit si řešení eliptických parciálních diferenciálních rovnice. Zejména jde o indexování uzlů numerické sítě ve 2D, sestavení matice soustavy lineárních rovnic a její následné řešení pomocí SOR metody. Cílem je také ozkoušet si konvergenci SOR metody pro různé hodnoty jejího relaxačního parametru ω.

Implementujte numerický řešič pro Poissonovu úlohu ve 2D.

1. do souboru `DE/Poisson2D.py` doplňte potřebný kód pro vyřešení Poissonovy úlohy ve 2D
2. proveďte výpočetní studii pro různé okrajové podmínky a pravé strany f
3. porovnejte rychlost konvergence různých stacionárních metod
4. najděte nejvhodnější parametr ω pro SOR metodu
5. zmenšujte h a sledujte počet iterací nutných pro vyřešení lineární soustavy rovnic
6. napište report a odevzdejte spolu se zdrojovým kódem

2.8 Numerické řešení rovnice vedení tepla ve 2D

Cílem tohoto úkolu je ozkoušet si řešení základní parabolické parciální diferenciální rovnice ve 2D.

Implementujte metodu přímek pro rovnici vedení tepla ve 2D. Použijte nachystané soubor `PDE/HeatEquation2D.py`.

Zvolte si vhodnou počáteční podmínku např. $sign(x^2+y^2-c^2)$ nebo náhodně generované hodnoty a sledujte, jak se řešení vyvíjí v čase.

Dále pomocí metody `readPGM` třídy `Vector` v `Vector.h` načtěte jako počáteční podmínku obrázek `data/motyl.pgm`:

```
int width, height;
Vector v;
// readPGM ulozi do width a height rozmery obrazku
v.readPGM( "../data/motyl.pgm", width, height );
double* u = v.getData();
// provedte vypocet reseni rovnice vedeni tepla na u
....
// zapiste vysledek do obrazku motyl.pgm
v.writePGM( "motyl.pgm", width, height );
```

Na obrázek aplikujte rovnici vedení tepla pro různé finální časy, tj. `finalTime`.

Nakonec implementujte implicitní řešič:

- implementujte implicitní řešič pro rovnici vedení tepla ve 2D do souboru

  `implicit-heat-equation-2d.py`

- proveďte stejné výpočty jako s pomocí metody přímek, ale s větším časovým krokem
- pokuste se najít optimální parametr ω pro SOR metodu
- porovnejte napočítané výsledky a CPU časy

Napište report se získanými výsledky.

2.9 Aplikace Peronovy-Malikovy úlohy ve zpracování obrazu - dobrovolný úkol

Implementujte ve 2D explicitní nebo semi-implicitní řešič Peronovy-Malikovy úlohy a aplikujte na odstranění šumu v obrázku `data/motyl-sum.pgm`.