Kapitola 1

Zadání

Tento dokument obsahuje zadání pro semestrální programy z PAA. Vypracování alespoň jedné úlohy je nutnou podmínkou pro úspěšné složení zkoušky resp. získaní (klasifikovaného) zápočtu (viz. bílá kniha). Student musí osobně předvést vypracovanou úlohu podle níže uvedených zadání. To mimo jiné znamená, že student musí umět detailně vysvětlit svůj kód. Ten by měl být maximálně jednoduchý a přehledný. Vyučující si vyhrazuje právo neuznat vypracovanou úlohu, pokud kód obsahuje hrubé programátorské chyby a to i v případě, že kód je funkční. Není-li uvedeno jinak, lze programovat pouze v jazycích C/C++ nebo Fortran.

1.1 Paralelizace explicitního řešiče pro difuzní rovnici

- podle kapitoly 2. nastudujte explicitní schéma pro řešení difůzní rovnice
- proveďte implementaci sekvenčního řešiče Mersonovy metody popsané právě v kapitole 2.
- pomocí vhodné počáteční podmínky (např. $u_{ini}(\mathbf{x}) = sign(-|\mathbf{x}| + 0.1)$) proveďte porovnání s pseudoanalytickým řešením - tj. napočítejte příslušnou konvoluci pro určité časové hladiny funkce $u_{ij}(t)$ pro např. $t = 0.1, 0.2, \cdots 1$ a zjistěte zda se výsledky shodují. Můžete také naměřit CPU čas nutný pro získání výsledku pomocí obou metod a určit, která je rychlejší.
- proveďte paralelizaci sekvenčího řešiče pomocí standardu MPI, kdy síť $\overline{\omega}_h$ rozdělíte na $m_1 \times m_2$ stejných podoblastí $m_1 > 1$ $m_2 > 1$ a v každou podoblast budete mapovat na jeden proces. Nezapomeňte, že pro každé napočítání pravé strany rovnice tj. funkce $f\left(t, u^h\right)$ je nutné provést synchronizaci hodnot funkce u^h na hranicích jednotlivých podoblastí.

- porovnejte, zda paralelní řešič dává shodné výsledky jako sekvenční verze

Pro uznání semestrálního programu je nutné vypracovat všechny výše uvedené body a osobně je předvést tzn. předvedení sekvenčního řešiče, srovnání s konvoluční metodou a předvedení paraleního řešiče. Pro vizualizaci výsledků doporučuji použít program Gnuplot, který dokáže číst a zobrazovat síťové funkce zapsané do obyčejného textového souboru ve formátu trojice

```
x y f(x,y)
```

na každém řádku.

1.2 Paralelizace implicitního řešiče pro difuzní rovnici

- podle kapitoly 2. nastudujte implicitní schéma pro řešení difůzní rovnice
- proveďte implementaci sekvenčního řešiče metody konjugovaných gradientů popsané právě v kapitole 2.
- pomocí vhodné počáteční podmínky (např. $u_{ini}(\mathbf{x}) = sign(-|\mathbf{x}| + 0.1)$) proveďte porovnání s pseudoanalytickým řešením - tj. napočítejte příslušnou konvoluci pro určité časové hladiny funkce $u_{ij}(t)$ pro např. $t = 0.1, 0.2, \cdots 1$ a zjistěte zda se výsledky shodují. Můžete také naměřit CPU čas nutný pro získání výsledku pomocí obou metod a určit, která je rychlejší.
- proveďte paralelizaci sekvenčího řešiče pomocí standardu OpenMP
- porovnejte, zda paralelní řešič dává shodné výsledky jako sekvenční verze

Pro uznání semestrálního programu je nutné vypracovat všechny výše uvedené body a osobně je předvést tzn. předvedení sekvenčního řešiče, srovnání s konvoluční metodou a předvedení paraleního řešiče. Pro vizualizaci výsledků doporučuji použít program Gnuplot, který dokáže číst a zobrazovat síťové funkce zapsané do obyčejného textového souboru ve formátu trojice

```
x y f(x,y)
```

na každém řádku.

Kapitola 2

Difuzní rovnice

2.1 Tvar rovnice a její pseudoanalytické řešení

Pro oblast $\Omega \subset \mathbb{R}^2$ hledáme funkci $u = u(\mathbf{x}, t)$ jako řešení parciální diferenciální rovnice

$$\begin{aligned} \frac{\partial u(\mathbf{x},t)}{\partial t} - \Delta u(\mathbf{x},t) &= 0 \text{ na } \Omega \times (0,T\rangle\,, \\ u(\mathbf{x},0) &= u_0(\mathbf{x}) \text{ na } \Omega, \\ u(\mathbf{x},t) &= g(\mathbf{x},t) \text{ na } \partial\Omega \times \langle 0,T\rangle\,, \end{aligned} \tag{2.1}$$

kde Laplacův operátor Δ je definován jako

$$\Delta u(\mathbf{x}) = \frac{\partial^2 u(\mathbf{x})}{\partial x^2} + \frac{\partial^2 u(\mathbf{x})}{\partial y^2},$$

a $\partial\Omega$ značí hranici oblasti Ω. Funkce $u_0(\mathbf{x})$ je daná počáteční podmínka a funkce $g(\mathbf{x}, t)$ je také daná a jde o okrajovou podmínku Dirichletova typu. Pseudoanalytické řešení této rovnice pro $\Omega \equiv \mathbb{R}^2$ má tvar

$$u(\mathbf{x},t) = \int_{\mathbb{R}^2} G_{\sqrt{2t}}(\mathbf{x}-\mathbf{y})\, u_0(\mathbf{y})\, \mathrm{d}y = \left(G_{\sqrt{2t}} * u_0\right)(\mathbf{x}), \tag{2.2}$$

kde

$$G_\sigma = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|\mathbf{x}|^2}{2\sigma^2}\right),$$

označuje Gaussovo jádro.

2.2 Prostorová diskretizace pomocí metody konečných diferencí

Pro jednoduchost nyní předpokládáme, že $\Omega \equiv \langle 0,1\rangle \times \langle 0,1\rangle$. Za účelem prostorové diskretizace volíme numerickou síť o rozměrech $N_1 \times N_2$, pro $N_1, N_2 \in \mathbb{N}^+$

danou jako

$$\omega_h \equiv \{x_{ij} = (ih_1, jh_2) \mid i = 1, \cdots N_1 - 1, j = 1, \cdots N_2 - 1\}, \tag{2.3}$$

její uzávěr

$$\bar{\omega}_h \equiv \{x_{ij} = (ih_1, jh_2) \mid i = 0, \cdots N_1, j = 0, \cdots N_2\}, \tag{2.4}$$

a hranici

$$\partial\omega_h = \bar{\omega}_h - \omega_h,$$

pro $h_1 = \frac{1}{N_1}$ a $h_2 = \frac{1}{N_2}$. Prostorovou diskretizace rovnice (2.2) lze provést pomocí **metody konečných diferencí**, **metody konečných objemů** nebo **metody konečných prvků**. V následujícím textu se budeme zabývat metodou konečných diferencí. Nejprve zavedeme síťovou funkci u_{ij}^h vztahem

$$u_{ij}^h = u\left(x_{ij}\right),$$

a následně aproximujeme Laplaceův operátor v rovnici (2.2) vztahem

$$\Delta u\left(x_{ij}\right) = \Delta_h u_{ij}^h \approx \frac{u_{i+1,j}^h - 2u_{ij}^h + u_{i-1,j}^h}{h_1^2} + \frac{u_{i,j+1}^h - 2u_{ij}^h + u_{i,j-1}^h}{h_2^2}, \tag{2.5}$$

čímž můžeme rovnici (2.2) převést do semidiskrétního tvaru

$$\frac{\mathrm{d}u_{ij}^h}{\mathrm{d}t} = -\Delta_h u_{ij}^h \text{ na } \omega_h. \tag{2.6}$$

Okrajové podmínky diskretizujeme snadno jako

$$u_{ij}^h = g\left(x_{ij}\right) \text{ na } \partial\omega_h. \tag{2.7}$$

K úplnému numerickému řešení rovnice (2.2) nyní zbývá jen provést časovou diskretizaci. Zde se nabízejí dvě možnosti:

- explicitní schéma pomocí metody přímek
- implicitní schéma pomocí dopředné diskretizace v čase.

2.3 Metoda přímek pro difuzní rovnici

Z přednášek z numerické matematiky víme, že obyčejné diferenciální rovnice tvaru $\dot{y}\left(t\right) = f\left(t, y\right)$ lze řešit pomocí **Rungových-Kuttových** vzorců. V našem případě neřešíme jednu obyčejnou diferenciální rovnici, ale celou soustavu (2.5), tj.

$$\dot{u^h}_{ij} = \Delta_h u_{ij}^h = f\left(t, u^h\right)_{ij} \text{ na } \omega_h, \tag{2.8}$$

$$\dot{u^h}_{ij} = \dot{g}\left(x_{ij}\right) = f\left(t, u^h\right)_{ij} \text{ na } \partial\omega_h, \tag{2.9}$$

$$\tag{2.10}$$

Snadno vidíme, že jsme pouze zaměnili u_{ij} za y a pravá strana f v našem případě vlastně nezávisí na t, ale pouze na u^h. Záměrně píšeme, že f závisí na u^h a nikoliv jen na u^h_{ij} neboť k napočítání aproximace Laplaceove operátoru potřebujeme kromě uzlové hodnoty u_{ij} i čtyři další sousedy.

V praxi je velice šikovné volit integrační metodu s automatickou vobou časového kroku τ. Jednou z nich je tzv. **Mersonova metoda**. Jde o jakousi modifikaci základních Rungových-Kuttových vzorců a nabízí integraci s přesností čtvrtého řádu v čase. Metoda spočívá v napočítání následujících síťových funkcí definovaných na $\overline{\omega}_h$:

$$\begin{aligned}
k^1_{ij} &:= \tau f\left(t, u^h\right)_{ij} \\
k^2_{ij} &:= \tau f\left(t + \frac{1}{3}\tau, u^h + \frac{1}{3}k^1\right)_{ij} \\
k^3_{ij} &:= \tau f\left(t + \frac{1}{2}\tau, u^h + \frac{1}{6}k^1 + \frac{1}{6}k^2\right)_{ij} \\
k^4_{ij} &:= \tau f\left(t + \frac{1}{2}\tau, u^h + \frac{1}{8}k^1 + \frac{3}{8}k^3\right)_{ij} \\
k^5_{ij} &:= \tau f\left(t + \tau, u^h + \frac{1}{2}k^1 - \frac{3}{2}k^3 + 2k^4\right)_{ij}.
\end{aligned}$$

Pozor! Jelikož pravá strana f závisí na celé síťové funkci, není možné začít počítat k^2, dokud neznáme k^1 na celém $\overline{\omega}_h$ a podobně pro ostatní k^l pro $l = 3, 4, 5$!

Následně počítáme odhad chyby, které se dopouštíme při integraci s časovým korkem τ

$$E = \max_{\overline{\omega}_h} \frac{1}{3}\left|\frac{1}{5}k^1_{ij} - \frac{9}{10}k^3_{ij} + \frac{4}{5}k^4_{ij} - \frac{1}{10}k^5_{ij}\right|.$$

Je-li tato chyba menší než-li zadaná tolerance ϵ, napočítáme nové u^h pomocí

$$u^h_{ij} := u^h_{ij} + \frac{1}{6}\left(k^1_{ij} + 4k^4_{ij} + k^5_{ij}\right), \tag{2.11}$$

a nastavíme $t := t + \tau$. V každém případě pak upravíme časový krok τ jako

$$\tau := \min\left\{\tau \cdot \omega\left(\frac{\epsilon}{E}\right)^{\frac{1}{5}}, T - t\right\}. \tag{2.12}$$

Pokud bylo $E < \epsilon$, dojde ke zvětšení τ, jinak se časový kork zmenší. Celý výpočet pak opakujeme s novým τ. Pro parametr ω by mělo platit $\omega \in \langle 0.8, 0.9\rangle$ a $\epsilon \approx 0.0001$.

2.4 Implicitní schéma

Narozdíl od metody přímek se v této části omezíme na pevný časový krok τ, který ale může být zvolen výrazně větší, než u explicitního schématu, proto bývá

implicitní schéma efektivnější z hlediska nároků na čas CPU. Diskretizaci časové derivace provedeme následujícím způsobem

$$\frac{\mathrm{d}u\left(x_{ij}\right)}{\mathrm{d}t} \approx \frac{u_{ij}^{n} - u_{ij}^{n-1}}{\tau} = \Delta_h u_{ij}^{n}, \tag{2.13}$$

kde platí

$$u_{ij}^{n} = u\left(x_{ij}, n\tau\right).$$

Dostáváme tak vztah (pro jednoduchost nyní předpokládáme $h_1 = h_2 = h$ a $N_1 = N_2 = N$)

$$\frac{u_{ij}^{n} - u_{ij}^{n-1}}{\tau} = \frac{u_{i+1,j}^{n} - 2u_{ij}^{n} + u_{i-1,j}^{n}}{h^2} + \frac{u_{i,j+1}^{n} - 2u_{ij}^{n} + u_{i,j-1}^{n}}{h^2} \tag{2.14}$$

neboli

$$u_{ij}^{n} - \frac{\tau}{h^2}\left(u_{i+1,j}^{n} + u_{i,j+1}^{n} + u_{i-1,j}^{n} + u_{i,j-1}^{n} - 4u_{ij}^{n}\right) = u_{ij}^{n-1} \tag{2.15}$$

pro $i, j = 1, \cdots N-1$. Jelikož neznámé jsou zde $u_{i+1,j}^{n}$, $u_{i,j+1}^{n}$, $u_{i-1,j}^{n}$, $u_{i,j-1}^{n}$ a u_{ij}^{n}, jde vlastně o soustavu $(N-1)^2$ rovnic. Okrajové podmínky mají tvar

$$u_{ij}^{n} = g\left(x_{ij}, n\tau\right) \text{ na } \partial\omega_h. \tag{2.16}$$

Přepíšeme-li nyní (2.15) na tvar

$$\left(1 + 4\lambda\right) u_{ij}^{n} - \lambda u_{i+1,j}^{n} - \lambda u_{i,j+1}^{n} - \lambda u_{i-1,j}^{n} - \lambda u_{i,j-1}^{n} = u_{ij}^{n-1}, \tag{2.17}$$

můžeme celou úlohu převést do maticového tvaru

$$\mathbb{A}\mathbf{x} = \mathbf{b}. \tag{2.18}$$

kde uzlové proměnné $u_i j$ odpovídá $(iN+j)$-tý sloupec resp. řádek. Jelikož matice $\mathbb{A}$ je symetrická a pro dostatečně malé λ resp. τ i pozitivně definitní, lze k řešení použít metodu **konjugovaných gradientů**. Ta má následujíc podobu

1. $\mathbf{r}_0 := \mathbf{b} - \mathbb{A}\mathbf{x}_0;\ \mathbf{p}_0 = \mathbf{r}_0$
2. pro $j = 0, 1, \cdots$ až do konvergence počítej
3. $\alpha_j := \left(\mathbf{r}_j, \mathbf{r}_j\right) / \left(\mathbb{A}\mathbf{p}_j, \mathbf{p}_j\right)$
4. $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5. $\mathbf{r}_{j+1} := \mathbf{r}_j - \alpha_j \mathbb{A}\mathbf{p}_j$
6. $\beta_j := \left(\mathbf{r}_{j+1}, \mathbf{r}_{j+1}\right) / \left(\mathbf{r}_j, \mathbf{r}_j\right)$
7. $p_{j+1} := \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$.

Matice $\mathbb{A}$ je řídká, proto do paměti ukládáme jen její nenulové prvky. Jelikož ty nezávisí na u_{ij} je také možné napsat přímo funkci, která pro zadaný vektor napočítá jeho součin s maticí.