FAKULTA
STROJNÍ
ČVUT V PRAZE

**ZÁKLADY PROGRAMOVÁNÍ
A ALGORITMIZACE
(2012036)**

SEMESTRÁLNÍ PRÁCE

Gaussova eliminace a Taylorův polynom v Matlabu/Pythonu

Martin Lukášek

(lukasma9@cvut.cz)

Datum:

23. října 2025

Cvičení:

C4

Cvičící:

Ing. Ondřej Krejčí

1 Úvod

1.1 Úkoly:

1.1.1 Gaussova eliminace:

1. Vytvořte vlastní implementaci Gaussovy eliminace v rámci MATLABu i Pythonu bez užití jiných vestavěných solverů.
2. Otestujte řešení pro čtvercové, regulární matice různých velikostí:
   n = {10, 100, 500}.
3. Pro každé n změřte výpočetní čas Vaší vlastní implementace v MATLABU a v Pythonu. Vyneste do grafu závislost výpočetní čas x velikost matice. Porovnejte v grafu s výpočetními časy vestavěných metod.

1.1.2 Taylorův polynom:

4. Vyberte si reálnou funkci f(x), např. sin(x), volbu ale nechám na Vás.
5. Implementujte výpočet Taylorova polynomu n-tého řádu na zvoleném intervalu. (Vyberte si zda MATLAB nebo Python).
6. Graficky porovnejte původní funkci f(x) a její aproximace Taylor. polynomem různého řádu n = 2,3,4,6,8 na vhodném intervalu.
7. Porovnejte s vestavěnými funkcemi sympy.series().
8. Diskutujte přesnost aproximace v závislosti na řádu polynomu.

1.2 Výstup práce:

- Zdrojový kód
- Stručná zpráva (max 3 strany)
- Prezentace - (max 10 minut)

2 Vypracování

2.1 Gaussova eliminace (MATLAB & Python)

Cílem bylo implementovat řešení soustavy lineárních rovnic $A\mathbf{x} = \mathbf{b}$ vlastní verzí Gaussovy eliminace se zpětnou substitucí a volitelným částečným pivotováním. Implementace v Pythonu poskytuje funkci:

`gauss_solve(A,b,pivot,return_intermediate,verbose)`,

která provádí dopřednou eliminaci (včetně prohazování řádků podle *max pivotu* v aktuálním sloupci) a následnou zpětnou substituci; při volbě:

`return_intermediate=True`

navíc vrací horní trojúhelníkovou matici U a upravenou pravou stranu $\mathbf{b}$.

Algoritmus. Dopředná fáze prochází sloupce $k = 1, \dots, n-1$, v každém z nich volitelně vybere pivot s největší absolutní hodnotou a provede elementární řádkové operace, aby vynulovala prvky pod diagonálou. Pro $i > k$ se používá multiplikátor $m_{ik} = U_{ik}/U_{kk}$ a aktualizace $U_{i,k:n} \leftarrow U_{i,k:n} - m_{ik}\, U_{k,k:n},\; b_i \leftarrow b_i - m_{ik}\, b_k$. Získanou horní trojúhelníkovou soustavu $U\mathbf{x} = \mathbf{b}$ řeší zpětná substituce. Časová složitost je $\mathcal{O}(n^3)$, paměť $\mathcal{O}(n^2)$. Částečné pivotování výrazně zlepšuje numerickou stabilitu (omezí růstový faktor) a snižuje riziko dělení malými pivoty.

Testy a měření. Pro měření výkonu se generují dobře podmíněné náhodné systémy $\big(A, b\big)$ s diagonální dominancí $A \leftarrow A + n * E$ a fixním seedem (reprodukovatelnost). Měří se průměrné časy přes několik běhů pro velikosti $n \in \{10, 100, 300, 500\}$ a porovnává se s `A\b` (MATLAB) a `np.linalg.solve` (Python). Součástí jsou grafy závislosti času na n a textový výpis testu na systému 3×3 (matice, upravená pravá strana, řešení).

Ošetření chyb. Kód detekuje singulární či téměř singulární situace (pivot ≈ 0) a ukončí běh s chybou. Doporučeným doplněním je kontrola rezidua $\|A\mathbf{x} - \mathbf{b}\|_2$ a podmíněnosti $\kappa_2(A)$ pro hlubší diagnostiku (volitelné).

MATLAB přepis. MATLAB skript `gauss_solve_benchmark.m` je věrný přepis Python verze: obsahuje funkci `gauss_solve` (s přepínači `pivot`, `return_intermediate`, `verbose`), ukázkový test 3×3 a benchmark proti `A\b`. Grafy se vykreslují přes `plot`.

Očekávané chování výsledků. Výpočetní čas roste zhruba jako $\mathcal{O}(n^3)$. Vestavěné `A\b`/`np.linalg.solve` jsou rychlejší (využívají BLAS/LAPACK). Má „učebnicová“ verze je vhodná na ukázku kroků eliminace. S pivotováním je přesnost obvykle srovnatelná; bez pivotu může být řešení nestabilní nebo selhat.

Rozdíly v přesnosti, výkonu a vhodnosti metod. *Přesnost:* S `pivot=True` dosahujeme podobné přesnosti jako vestavěné solvery; bez pivotu hrozí velké chyby u špatně podmíněných matic. Doporučujeme kontrolovat reziduum $\|A\mathbf{x} - \mathbf{b}\|$.

Výkon: Vše má složitost $\mathcal{O}(n^3)$, ale BLAS/LAPACK jsou díky optimalizacím a paralelizaci výrazně rychlejší.

Vhodnost: Má implementace: výuka, menší až střední úlohy, plná kontrola nad postupem. Vestavěné solvery: praxe a velké úlohy; pro špatně podmíněné problémy raději QR/SVD, pro řídké matice řídké metody.

2.2 Taylorův polynom pro $\sin(x)$ (Python)

Druhá část implementuje Maclaurinův polynom $\sin(x)$ do stupně n (tj. rozvoj v bodě $a = 0$):

$$T_n(x) = \sum_{k=0}^{\lfloor n/2 \rfloor} (-1)^k \frac{x^{2k+1}}{(2k+1)!},$$

tj. využívají se pouze liché mocniny. Funkce `taylor_sin_maclaurin(x,n)` vrací vektor aproximací pro dané x. Kód dále vykreslí porovnání $\sin(x)$ a $T_n(x)$ pro $n \in \{2, 3, 4, 6, 8\}$ na intervalu $[-\pi, \pi]$, graf absolutní chyby a “zoom” v okolí nuly.

Porovnání se `sympy.series()`. Pro kontrolu správnosti se symbolicky vygeneruje rozvoj pomocí `sympy.series` (odstraní se $O(x^{n+1})$) a numericky se porovnají hodnoty na sadě testovacích bodů; reportuje se maximální odchylka. Součástí je i tabulka chyb (max a RMS) na plném intervalu $[-\pi, \pi]$ a na zúženém $[-1, 1]$.

Diskuse přesnosti. Taylorův polynom má nejlepší shodu v okolí bodu rozvoje ($x = 0$). Se zvyšujícím se n klesá chyba zejména u malých $|x|$; pro vzdálenější body je nutné vyšší řád, případně volba jiného bodu rozvoje a (např. centrování v oblasti zájmu). Na rozdíl od polynomiální interpolace na širších intervalech se zde neprojevuje Rungeho jev; konvergence je dána vlastnostmi řady a chováním derivací funkce.

Přesnost aproximace vs. řád polynomu. Pro $\sin x$ (rozvoj v $a = 0$) platí Lagrangeův zbytek

$$R_{2m+1}(x) = \sin x - T_{2m+1}(x) = \frac{f^{(2m+2)}(\xi)}{(2m+2)!}\, x^{2m+2}, \quad \text{kde } |f^{(k)}(\xi)| \le 1.$$

Tím pádem

$$|R_{2m+1}(x)| \le \frac{|x|^{2m+2}}{(2m+2)!}.$$

Z toho plyne: (i) přibývají pouze *liché* členy, takže sudý řád n nedává nový polynom oproti $n-1$; (ii) blízko nuly klesá chyba velmi rychle (faktoriál ve jmenovateli). Pro $|x| \le 1$ vychází hrubé horní meze

$$n{=}3: \ \tfrac{1}{5!} \approx 8.3{\times}10^{-3}, \quad n{=}5: \ \tfrac{1}{7!} \approx 2.0{\times}10^{-4}, \quad n{=}7: \ \tfrac{1}{9!} \approx 2.8{\times}10^{-6}.$$

Na širším intervalu (např. $[-\pi, \pi]$) je „další zanedbaný člen“ větší (např. obsahuje π^{n+1}), a proto je pro stejnou přesnost třeba vyšší n nebo volit jiný střed rozvoje a blíže oblasti zájmu. Prakticky pozorujeme, že absolutní chyba se chová zhruba jako první vynechaný člen řady a monotonně klesá s rostoucím lichým n; relativní chyba může být větší v okolí nulových průchodů, což je u trigonometrických funkcí běžné. Numerické vyhodnocení je vhodné provádět Hornerovým schématem kvůli menší kumulaci zaokrouhlovacích chyb.

Poznámky k implementaci. Výpočet využívá vektorové operace NumPy a faktoriály ze standardní knihovny; grafy jsou vykresleny pomocí `matplotlib`. Symbolická verifikace probíhá přes `sympy` a numerické vyhodnocení přes `lambdify`.

3 Závěr

Byly vyvinuty a otestovány dvě samostatné úlohy: (i) vlastní implementace Gaussovy eliminace se zpětnou substitucí a částečným pivotováním v MATLABu/Pythonu a (ii) aproximace $\sin(x)$ Maclaurinovými polynomy vybraných řádů v Pythonu.

U Gaussovy eliminace se prakticky potvrdila teorie: časová složitost $\mathcal{O}(n^3)$ a význam pivotování pro numerickou stabilitu. Vestavěné rutiny `A\b` / `np.linalg.solve` jsou rychlejší díky optimalizacím (BLAS/LAPACK), má varianta je však vhodná pro didaktické objasnění kroků algoritmu a sledování dopředné eliminace (matice U) i zpětné substituce (výpočet $\mathbf{x}$). Doporučeným rozšířením je automatická kontrola rezidua $\|A\mathbf{x} - \mathbf{b}\|$ a analýza podmíněnosti A.

U Taylorových polynomů se ukázalo, že zvýšení řádu zlepšuje přesnost především v okolí rozvojového bodu; na širších intervalech je vhodné zvýšit n nebo zvolit jiný střed rozvoje. Symbolická kontrola přes `sympy.series` potvrdila správnost koeficientů a numerické konzistence na testovacích bodech, stejně jako očekávané chování maximální a RMS chyby na intervalech $[-\pi, \pi]$ a $[-1, 1]$.

Shrnutí přínosů: (a) porozumění praktickým aspektům eliminace (pivotování, kumulace zaokrouhlovacích chyb, měření výkonu) a (b) procvičení práce s Taylorovými rozvoji včetně validace vůči symbolickým nástrojům. **Možné pokračování:** plné pivotování či LU/QR dekompozice, robustnější diagnostika přesnosti (rezidua, relativní chyby, `cond`), zobecnění Taylorova polynomu na libovolné $f(x)$ a libovolný bod a, případně porovnání s Chebyshevovskou aproximací na daném intervalu.

Použitá literatura

[1] K. Kuncová. „(9) Taylorův polynom.“ Slidy k předmětu Matematika B3, cit. 22. říj. 2025. URL: https://www.karlin.mff.cuni.cz/~kuncova/1920ZS_B3/09_Taylor_web.pdf.

[2] M. Tůma. „Numerická lineární algebra II — Kapitola 2: Gaussova eliminace.“ Poznámky k přednášce, cit. 22. říj. 2025. URL: https://www.karlin.mff.cuni.cz/~tuma/2003/NNLinalg2.pdf.