Úvod do numerické matematiky

Martin Kuba, Matěj Janouch

Upraveno: 2. 6. 2025

Tento text vznikl čistě na základě materiálů, které vypracoval a odpřednášel Mgr. Michal Outrata, Ph.D.

Tímto mu srdečně děkujeme.

Obsah

1 Interpolace

1.1 Úvod

Začneme formulací problému v 1D.

Definice 1.1. *Mějme interval (a, b), body $x_0, \ldots, x_n \in (a, b)$ a jejich funkční hodnoty $f_0, \ldots, f_n \in \mathbb{R}$. řekneme, že funkce $P_f(x) : (a, b) \to \mathbb{R}$ je interpolující funkce, pokud platí*

$$\forall i : P_f(x_i) = f_i.$$

Poznámka. Rozšíření do 2D či 3D jde různě, např. $P_f(x, y) = P_f^{(x)}(x)P_f^{(y)}(y)$.

V této kapitole se budeme věnovat klasické interpolaci, tj. polynomiální interpolaci tvaru $P_f(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_p x^p$

Jednu metodu polynomiální aproximace už známe - Taylorův polynom. To by ale rovnost $P_f(x_i) = f_i$ byla splněna pouze pro jeden bod.

V praxi se často setkáme i s jinými typy interpolací, např. místo x^i můžeme brát třeba $\cos(i\pi x)$, nebo $\sin(i\pi x)$.

1.2 Naivní formulace

Rozepíšeme si interpolační podmínky:

$$\forall i = 1, \ldots, n : \alpha_0 + \alpha_1 x_i + \alpha_2 (x_i)^2 + \cdots + \alpha_n (x_i)^n = f_i$$

$$\iff$$

$$\forall i = 1, \ldots, n : \begin{bmatrix} 1 & x_i & (x_i)^2 & \ldots & (x_i)^n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = f_i$$

$$\iff$$

$$\underbrace{\begin{bmatrix} 1 & x_0 & \ldots & (x_0)^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & (x_n)^n \end{bmatrix}}_{\text{tzv. Vandermondova matice}} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

Vyřešením této lineární soustavy rovnic dostaneme $P_f(x)$.

1.3 Lagrangeova interpolace

V minulé sekci jsme hledali $P_f(x)$ jako lineární kombinaci monických polynomů x^i. Tento postup ale není ideální. Zkusme na to jít jinak.

Mějme polynomy tvaru $l_i(x)$, také, že $l_i(x_j) = \delta_{i,j},\ i, j = 1, \ldots, n$. Zkusme vyjádřit následovně

$$P_f(x) = \alpha_0 l_0(x) + \cdots + \alpha_n l_n(x).$$

Zjevně platí

$$P_f(x_i) = \alpha_i,$$

a odtud odvodíme

$$\alpha_i = f(x_i).$$

Maticový zápis

$$\begin{bmatrix} l_0(x_0) & l_1(x_0) & \dots & l_n(x_0) \\ l_0(x_1) & l_1(x_1) & \dots & l_n(x_1) \\ \vdots & & \ddots & \vdots \\ l_0(x_n) & l_1(x_n) & \dots & l_n(x_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

Konečně dostáváme vztah

$$f(x) \approx \sum_{i=0}^{n} l_i(x) f(x_i).$$

Jde o změnu báze prostoru polynomů, ve které hledáme naši aproximaci.
Z toho plyne rovnost

$$l_i(x) = \prod_{k \neq i} \frac{x - x_k}{x_i - x_k}.$$

1.4 Hermit

Co když chceme, aby $\frac{d}{dx} P_f(x_i) = f_i'$?
Lze to analogicky dosáhnout přes systém lineárních rovnic

$$P_f(x) = \sum_{i=0}^{n} \alpha_i x^i \implies P_f(x_i) = f_i,$$

$$\frac{d}{dx} P_f(x) = \sum_{i=0}^{n-1} (i+1)\alpha_{i+1} x^i \implies \frac{d}{dx} P_f(x_i) = f_i'.$$

Je zjevné, že jde o systém $2(n+1)$ rovnic.
Analogicky jako u Lagrangeových polynomů

$$h_i(x) := (1 - 2(x - x_i) l_i'(x_i)) l_i^2(x),\ h_i(x_j) = \delta_{i,j},\ \frac{d}{dx} h_i(x_j) = 0,$$

$$g_i(x) := (x - x_i) l_i^2(x),\ g_i(x_j) = 0,\ \frac{d}{dx} g_i(x_j) = \delta_{i,j},$$

$$P_f(x) := \sum_{i=0}^{n} \alpha_i h_i(x) + \sum_{i=0}^{n} \beta_i g_i(x).$$

1.5 Souhrn interpolačních podmínek

- Vandermond: $P_n = \sum_{i=0}^{n} \alpha_i x^i$
- Lagrange: $P_n = \sum_{i=0}^{n} \tilde{\alpha}_i l_i(x),\ l_i(x_j) = \delta_{i,j}$
- Hermit: $P_{2n+1}(x_i) = f_i,\ P_{2n+1}'(x_i) = f_i'$

1.6 Odhad chyby

Tvrzení 1.2. *Nechť $f \in \mathcal{C}^{n+1}((a,b)), x_0 \leq x_1 \leq \dots \leq x_n \in (a,b)$. Mějme $f_i := f(x_i)$ a $P_f(x)$ jako u Lagrangeovy interpolace. Pak*

$$\forall x \in (a,b)\ \exists \xi \in (a,b) : f(x) - P_f(x) = (x - x_0)(x - x_1) \dots (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!},$$

a tedy

$$\max_{x\in(a,b)} |f(x) - P_f(x)| \leq \frac{\max_{\xi\in(a,b)} |f^{(n+1)}(\xi)|}{(n+1)!} |(x-x_0)\dots(x-x_n)|.$$

Téměř v žádné aplikaci nelze změnit či ovlivnit $f(x)$, tudíž člen $\max_{\xi\in(a,b)} |f^{(n+1)}(\xi)|$ je mimo náš dosah.

Člen $1/(n+1)!$ je fajn, ten nám pomáhá a ukazuje, že chyba aproximace může klesat až exponenciálně.

Člen $|(x-x_0)\dots(x-x_n)|$ je někdy pod naší kontrolou. U některých aplikací si můžeme vybrat body měření/pozorování. Tento člen jde minimalizovat pomocí tzv. Chebyshevových bodů. Jsou to kořeny tzv. Chebyshevových polynomů. Jde je popsat jako "husté u krajů, řídké uprostřed".

Věta 1.3. *Mějme $f(x) \in \mathcal{C}^1([a,b])$ a $P_f(x)$ odpovídající Lagrangeově interpolaci v Chebyshevových bodech. Pak*

$$\max_{x\in(a,b)} |f(x) - P_f(x)| \xrightarrow{deg(P_f)\to\infty} 0.$$

Všimneme si, že problémy při interpolaci se objevují pro velké n. Co kdybychom měli n "malé, ale hodněkrát"? Tato úvaha nás dostává do poslední podkapitoly o interpolaci.

1.7 Spliny

Po částech polynomiální aproximace (např. přeložíme přímkami). V praxi se často používají adaptivní spliny, které lokálně přidají více interpolačních bodů v kritických místech. Ovšem zachování polynomů stejného stupně je výhodné. Víme totiž lépe určit $\max_{x\in(a,b)} |s_f(x) - f(x)|$.

Věta 1.4. *Mějme $(a,b) \subset \mathbb{R}$, dělení $x_0 = a, x_1 = a+h, x_2 = a+2h, \dots, x_k = b = a + k\cdot h$. Pak platí*

- *$f \in \mathcal{C}^2$ a s_f je lineární spline* $\implies \max_{x\in(a,b)} |s_f(x) - f(x)| \leq \frac{1}{8} h^2 max_{\xi\in(a,b)} |f^{(2)}(\xi)|$,
- *$f \in \mathcal{C}^4$ a s_f je kubický spline* $\implies \max_{x\in(a,b)} |s_f(x) - f(x)| \leq \frac{3}{8} h^4 max_{\xi\in(a,b)} |f^{(4)}(\xi)|$.

2 Podmíněnost a stabilita

2.1 Úvod

Od roku 1985 existuje standardizace (IEEE754) - ukládání dat do 0 a 1. Není však realistické uložit čísla jako $\pi, \sqrt{2}, e$, ale ani "jednodušší", jako například 1,3.

2.2 Reprezentace čísel v PC

Mějme $x \in \mathbb{R}$ jako číslo, jež chceme reprezentovat v PC a $x_{\mathrm{FPA}} \in \mathbb{R}$, jež představuje skutečnou reprezentaci čísla v PC. (FPA = floating-point precision arithmetic/finite precision arithmetic)

Poznámka. Předpokládáme, že $x = x_{\mathrm{FPA}}$ nebo $x \approx x_{\mathrm{FPA}}$.

Standard ve všech počítačích je v současné době tzv. double precision - fp64 formát čísel (čísla se ukládají do 64 bitů), kde
1 bit je určen pro znaménko,
11 bitů je pro exponent,
52 bitů zbývá pro binární rozvoj.

Příklad 2.1. *Mějme $x = 61$*
Dekadický zápis:
$x = 6 \cdot 10^1 + 1 \cdot 10^0$.
Binární zápis:
$x = 32 + 16 + 8 + 4 + 1 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$,
$x_{FPA} = 111101$,
$x = x_{FPA}$, *protože ho umíme přesně vyjádřit.*

▲

Příklad 2.2. *Mějme $x = 0{,}15625$.*
Dekadický zápis:
$x = (+1) \cdot 10^{-1} \cdot 1{,}5625 = (+1) \cdot 10^{-1} \cdot (1 \cdot 10^1 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2} + 2 \cdot 10^{-3} + 5 \cdot 10^{-4})$.
Binární zápis:
$x = (+1) \cdot 2^{-3} \cdot 1{,}25 = (+1) \cdot 2^{-3} \cdot (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}) = (+1) \cdot 2^{-3} \cdot [101] = (+1) \cdot 2^{[11]} \cdot [101]$,
x_{FPA} = *{1},-{11},{101};* x_{FPA} *umíme tedy vyjádřit přesně a* $x = x_{FPA}$.

▲

Příklad 2.3. *Mějme $x = 1{,}3$.*
Dekadický zápis:
$x = (+1) \cdot (1 \cdot 10^0 + 3 \cdot 10^{-1})$.
Binární zápis:
$x = (+1) \cdot 2^0 \cdot (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-1} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 0 \cdot 2^{-6} + 0 \cdot 2^{-7} + 1 \cdot 2^{-8} + 1 \cdot 2^{-9} + 0 \cdot 2^{-10} + \dots) = (+1) \cdot 2^0 \cdot [10100110011\dots]$.
Počítač má konečnou paměť, tudíž nekonečný rozvoj musíme prakticky někde zastavit; tedy $x \neq x_{FPA}$, *ale* $x \approx x_{FPA}$.

▲

Všechny operace v PC se provádějí s x_{FPA} (tedy s $\{0{,}1\}^{64}$) a tedy všechna data a naše manipulace/výpočty s nimi jsou vždy (i když třeba jen nepatrně) nepřesné!

Tvrzení 2.4. *Platí, že* $\forall x \in (-10^{308}, -10^{-38}) \cup (10^{-38}, 10^{308})\ \exists x_{FPA} : \frac{|x - x_{FPA}|}{|x|} \leq 2 \cdot 10^{-16}$.

2.2.1 Efekt motýlích křídel

Existují matematické problémy, pro které i malá změna vstupních dat/parametrů (malá perturbace) může vést k velké změně v přesném řešení daného problému = *efekt motýlích křídel*. Takové problémy se nazývají "špatně podmíněné"(opak "dobře podmíněné").

Podmíněnost je vlastnost problému - pro extrémně špatně podmíněné problémy je pouze malá naděje na numerický výpočet řešení (správného).

Vlastnost efektu motýlích křídel může mít i námi navržený numerický algoritmus - zaokrouhlovací chyby se mohou akumulovat a i pro dobře podmíněný problém dostaneme zkreslenou odpověď. Obecně mluvíme o stabilitě algoritmu.

2.2.2 Obecná strategie pro analyzování

***KROK* 1**

Analyzujeme matematický problém a zjistíme jeho podmíněnost ≡ citlivost řešení na změnu vstupních dat/parametrů.

***KROK* 2**

Analyzujeme náš algoritmus jako "přesný výpočet pro nepřesná data".

Příklad 2.5. *Algoritmus pro sčítání dvou skalárů* $\alpha, \beta \in \mathbb{R}$.
Tužka a papír: $\alpha + \beta = \gamma$.
Algoritmus: $\alpha +_{PC} \beta = \tilde{\gamma}$.
Analýza stability: dokážeme existenci $\tilde{\alpha}$,$\tilde{\beta} \in \mathbb{R}$, *takových, že* $\tilde{\alpha} \approx \alpha, \tilde{\beta} \approx \beta$ *a* $\tilde{\alpha} + \tilde{\beta} = \tilde{\gamma}$.
Pak je algoritmus stabilní (zpětně stabilní), pokud platí

$$\frac{max\{|\tilde{\alpha} - \alpha|,|\tilde{\beta} - \beta|\}}{max\{|\alpha|,|\beta|\}} \leq C \cdot \varepsilon_{mach},$$

kde C *značí malou konstantu, a* $\varepsilon_{mach} = 10^{-16}$ *značí počítačový limit.*

▲

***KROK* 3**

x_{PC} = výstup ze stabilního algoritmu pro dobře podmíněný problém $\xrightarrow{\text{stabilní algoritmus}}$ výstup z výpočtu "tužka a papír" pro "trochu změněná" vstupní data $\xrightarrow{\text{dobře podmíněný problém}} x_{\text{PC}} \approx x_{\text{tužka a papír}}$

2.3 Podmíněnost a stabilita

Problém $\mathcal{P}$: input $\rightarrow$ output.
Například se může jednat o

$$(\alpha, \beta) \rightarrow \alpha + \beta,$$

nebo

$$A, \vec{b} \rightarrow \vec{x}; A\vec{x} + \vec{b} = 0.$$

2.3.1 Podmíněnost

Definice 2.6. *Problém* $\mathcal{P}$ *má podmíněnost* $\kappa > 0$, *pokud platí*

$$\forall\, input\ \forall \varepsilon > 0\ (perturbace) : \frac{||\mathcal{P}(input + \varepsilon) - \mathcal{P}(input)||}{||\mathcal{P}(input)||} = \kappa \frac{||input + \varepsilon - input||}{||input||} + \omega(||\varepsilon||).$$

Problém $\mathcal{P}$ má podmíněnost κ, pokud platí pro malé perturbace vstupních dat, že výstup se změní nejvýše κ-krát více než je velikost perturbace

$$\frac{||\mathcal{P}(input+\varepsilon)-\mathcal{P}(input)||}{||\varepsilon||} = \kappa\frac{||\mathcal{P}(input)||}{||input||} + \omega(||\varepsilon||).$$

Pozorný čtenář nahlédne, že definice připomíná derivaci - tedy pro $||\varepsilon|| \to 0$

$$||\mathcal{P}'(input)|| = \kappa\frac{||\mathcal{P}(input)||}{||input||}.$$

2.3.2 Stabilita

Definice 2.7. *Řekněme, že algoritmus $\mathcal{A} : input \to output$ je stabilní (zpětně stabilní), pokud* $\exists\, C > 0$, $\forall\, input$, $\exists\, input_p$, *takové, že splňuje*

- $\mathcal{A}(input) = \mathcal{P}(input_p)$,
- $\frac{||input-input_p||}{||input||} \leq C \cdot \varepsilon_{mach}$.

2.3.3 Důvěryhodnost výpočtů

$$\frac{||\mathcal{P}(input)-\mathcal{A}(input)||}{||\mathcal{P}(input)||} \overset{\text{stabilní } \mathcal{A}}{=} \frac{||\mathcal{P}(input)-\mathcal{P}(input_p)||}{||\mathcal{P}(input)||} \overset{\text{dobře podm.}}{=} \kappa\frac{||input_p-input||}{||input||} \overset{\omega}{\leq} C\cdot\kappa\cdot\varepsilon_{mach}.$$

Výpočtům můžeme tedy věřit, pokud $C \cdot \kappa \ll \varepsilon_{mach}$.

2.3.4 Polynomy v monické bázi a jejich kořeny

Vezmeme si tzv. Wilkinsonův polynom

$$P_\delta(x) := (x-1)\cdot\cdots\cdot(x-20) + \delta x^{19} \equiv x^{20} + (\alpha_{19}+\delta)x^{19} + \alpha_{18}x^{18} + \cdots + \alpha_0$$

s kořeny $r_1 \equiv r_1(\delta), r_2 \equiv r_2(\delta), \ldots, r_{20} \equiv r_{20}(\delta)$ (přirozeně tedy závislé na δ).

Pro $\delta = 0$ tedy máme $r_1(0) = 1, \ldots, r_{20} = 20$.

Pokud chceme s $P_{\delta=0}(x)$ počítat v PC, je možné, že některé z koeficientů α_i budou nepřesné. Pak δ použijeme jako simulaci této nepřesnosti (místo α_{19} máme $\alpha_{19}+\delta$).

Jak velký vliv má malá nepřesnost ($\delta = 10^{-7}$) na $r_1(\delta), \ldots, r_{20}(\delta)$ (Input je δ a outputy jsou r_i)? Ptáme se tedy, jaká je podmíněnost kořenů $P_{\delta=0}(x)$ vzhledem k δ.

$$\frac{|r_i(\delta)-r_i(0)|}{|r_i(0)|} = \kappa_i\delta + \mathcal{O}(\delta) \to \kappa_i \approx |r_i(0)|\frac{d}{d\delta}(r_i(\delta)|_{\delta=0}).$$

Zjevně každý kořen $r_i(\delta)$ může být jinak citlivý na změny α_{19} (a i jiných α_j) - místo slova podmíněnost se někdy používá slovo citlivost.

Otázka je tedy, jak velká derivace $\frac{d}{d\delta}(r_i(\delta)|_{\delta=0})$ pro $i = 1,2,\ldots,20$

$$\frac{d}{d\delta}(r_i(\delta)|_{\delta=0}) = \frac{-r_i(0)^{19}}{P_0'(r_i(0))} = \underbrace{\frac{-i^{19}}{\prod_{j\neq i}(i-j)}}_{\text{může být} \gg 1}.$$

Velmi malá změna δ může výrazně změnit $r_{20}, \ldots$, ale ne r_1.

2.3.5 Polynomiální interpolace

Máme data, která chceme interpolovat, tj. body a jejich funkční hodnoty $(x_0,f_0),\ldots,(x_n,f_n)$.
Jak už víme, Lagrangeova interpolace je tvaru $p_f(x) := \sum_{i=0}^{n} l_i(x) f_i$.
Co když jsme ale naměřili nepřesná data a skutečná (přesná) data a f_i jsou trochu jiná? Označme si je $(x_0,g_0),\ldots,(x_n,g_n)$.
Přesná Lagrangeova interpolace pak tedy bude vypadat jako $p_g(x) := \sum_{i=0}^{n} l_i(x) g_i$.
Chyba pouze v měření vyjádříme jako

$$p_f(x) - p_g(x) = \sum_{i=0}^{n} (f_i - g_i),$$

$$\max_{x\in[a,b]} |p_f(x) - p_g(x)| \leq \max_i |f_i - g_i| \cdot \underbrace{\max_{x\in[a,b]} |\sum_{i=0}^{n} l_i(x)|}_{=:\Lambda_n(x_0,\ldots,x_n)}.$$

Λ_n je tzv. Lebesgueova konstanta. Bez odvození, pro $(a,b) \equiv (-1,1)$

- $\Lambda_n(ekvidistantní) \approx \frac{2^{n+1}}{n \cdot e \cdot log(n)}$,
- $\Lambda_n(Chebyshev) \approx \frac{2log(n+1)}{\pi}$.

Platí

$$\frac{\max_{x\in[a,b]} |p_f(x) - p_g(x)|}{\max_{x\in[a,b]} |p_g(x)|} \leq \Lambda_n(x_0,\ldots,x_n) \frac{\max_i |f_i - g_i|}{\max_i |g_i|}.$$

- pro Chebyshevovy body: $n \leq 20000 \to \Lambda_n \ \frac{2\cdot 10}{\pi}\ 64 \to$ dobře podmíněný problém,
- pro ekvidistantní body $n \geq 60 \to \Lambda_n \geq \frac{1}{60 log(60)} 10^{16} \to$ špatně podmíněný problém.

2.3.6 Lineární soustavy rovnic

Věta 2.8 (Cramerovo pravidlo)**.** *Nechť $A \in \mathbb{R}^{n\times n}$ je regulární matice, $j \in \{1,2,\ldots,n\}$. Pak j-tá složka vektoru řešení $\vec{x}$ soustavy $A\vec{x} = \vec{b}$ je*

$$x_j = \frac{det(A_j)}{det(A)},$$

kde A_j je matice, která vznikne z A nahrazením j-tého sloupce vektorem $\vec{b}$.

Příklad 2.9. *Mějme problém*

$$\begin{bmatrix} -1/10 & 3/10 & 5/10 \\ 1/2 & 3/2 & 5/2 \\ 1/100 & 3/100 & 5/100 - 10^{-13} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7/10 \\ 9/2 \\ 9/100 - 10^{-13} \end{bmatrix}.$$

Přesný výpočet nám dá

$$\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Využitím Cramerova pravidla na PC dostaneme

$$\vec{x}_{CP} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} 0{,}99991905\ldots \\ 0{,}99977449\ldots \\ 1{,}00009252\ldots \end{bmatrix}.$$

Pokud si zadefinujeme

$$\hat{b} := A\vec{x}_{CP},$$

pak lze Cramerovo pravidlo vnímat jako přesný řešič, ale pro perturbovaný problém. Tato perturbace je velikosti $\approx 10^{-4}$*. Cramerovo pravidlo bychom proto nenazvali stabilní.*

▲

3 Kvadratura

V této sekci se budeme zabývat kvadraturou, jinak řečeno numerickou integrací, tj. mnohorozměrným integrálem tvaru

$$\int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} f(\vec{x}) d\vec{x},\ \vec{x} = (x_1, \ldots, x_n),\ n \in \mathbb{N}.$$

3.1 Typy kvadratur

3.1.1 Newton-Cotes

Mějme interval $(a, b) \subseteq \mathbb{R},\ x_0, x_1, \ldots, x_n \in (a, b)$ ekvidistantní body (tj. $x_i = (\frac{b-a}{n})i$), $f_0 = f(x_0), \ldots, f_n = f(x_n)$ funkční hodnoty. Pak platí

$$P_f(x) = \sum_{i=0}^{n} f(x_i) l_i(x),$$

$$\int_a^b P_f(x) dx = \sum_{i=0}^{n} f(x_i) \cdot \underbrace{\int_a^b l_i(x) dx}_{=:\, w_i},$$

kde w_i jsou tzv. kvadraturní váhy, a máme

$$\int_a^b f(x) dx \approx \sum_{i=0}^{n} f(x_i) w_i.$$

3.1.2 Gauss

Vezmeme $x_0, \ldots, x_n$ jako Legendrovy body na (a, b).

3.1.3 Clenshow-Curtis

Založené na $x_0, \ldots x_n$ jako Chebyshevovy body na (a, b).

3.2 Odhad chyby

Věta 3.1. *Nechť $q(x)$ je polynom, $n \in \mathbb{N}$ řád kvadratury. Pak platí*

- *$\forall q(x)$ polynom stupně $\leq n$: Newton-Cotes zintegruje $q(x)$ přesně,*
- *$\forall q(x)$ polynom stupně $\leq 2n + 1$: Gauss zintegruje $q(x)$ přesně,*
- *$\forall q(x)$ polynom stupně $\leq n$: Clenshaw-Curtis zintegruje $q(x)$ přesně.*

Nyní odvodíme jednoduchý vztah pomocí aritmetiky integrálu

$$\left| \int_a^b f(x) dx - \int_a^b P_f(x) dx \right| \leq \int_a^b |f(x) - P_f(x)|\, dx \leq \frac{\max_{\xi \in (a,b)} \left| f^{(n+1)}(\xi) \right|}{(n+1)!} \cdot \int_a^b \left| \prod_{i=0}^{n} (x - x_i) \right| dx,$$

kde část

$$\int_a^b \left| \prod_{i=0}^{n} (x - x_i) \right| dx$$

je malá pro Chebyshevovy body a velká pro ekvidistantní. Na základě tohoto pozorování můžeme prohlásit, že obecně kvadratury typu Gauss a Clenshaw-Curtis jsou lepší, neboť zmenšují chybu aproximace.

3.3 Kompozitní kvadratury

Pojďme si nyní zformulovat kvadraturu pro funkci, kterou po částech interpolujeme (tj. pomocí splinů). Mějme

$$f(x) \approx s_f(x) = P^{(k)}(x) \equiv \sum_{i=0}^{N-1} l_i^{(k)}(x) f(x_i^{(k)}),\ x \in (c_k, c_{k+1}),$$

kde (c_k, c_{k+1}) je dělení intervalu $(a, b) \subset \mathbb{R}$ na $N-1$ podintervalů, $k = 0, \ldots, N-1$. Pak

$$\int_a^b f(x)dx \approx \int_a^b s_f(x)dx = \sum_{k=0}^{N-1} \int_{c_k}^{c_{k+1}} P^{(k)}(x)dx = \sum_{k=0}^{N-1} \sum_{i=0}^{s} w_i^{(k)} f(x_i^{(k)}).$$

Odhad chyby této kvadratury se odvodí stejně jako výše - na základě odhadu chyby interpolace, např. pro $s_f(x)$ kubický spline na (a, b) v bodech $x_0, \ldots, x_n$ máme

$$\left| \int_a^b f(x)dx - \int_a^b s_f(x)dx \right| \leq C \cdot \max_{\xi \in (a,b)} |f^{(4)}(\xi)| \cdot h^4 \cdot |b - a|,$$

kde

$$h = \frac{b-a}{n}.$$

3.4 Nedostatky kvadratur

Pokud máme funkci $f : [a_1, b_1] \times \cdots \times [a_{50}, b_{50}] \longrightarrow \mathbb{R}$, pak lze všechny postupy ukázané výše přirozeně zobecnit. Potřebujeme však kvadraturní body

$$x_1^{(1)}, \ldots, x_n^{(n)} \in [a_1, b_1],$$

$$x_1^{(2)}, \ldots, x_n^{(2)} \in [a_2, b_2],$$

$$\vdots$$

$$x_1^{(50)}, \ldots, x_n^{(50)} \in [a_{50}, b_{50}].$$

Pro $n = 3$ bychom měli $3^{50} \approx 8 \cdot 10^{23}$ bodů. Jen pro porovnání, všech atomů na Zemi je přibližně 10^{50}. Je zjevné, že potřebujeme efektivnější postup.

4 Monte Carlo

4.1 Úvod

V problematice kvadratur jsme zjistili, že pokud chceme zachovat konstantní počet kvadratických bodů v každé dimenzi, pak nám exponenciálně roste počet bodů, ve kterých musíme f vyhodnotit v závislosti na dimenzi $f : \mathbb{R}^d \to \mathbb{R}$. To je ale problém, protože v praxi se běžně dostaneme k $d \approx 10^2$, a i hrubé odhady ukazují, že tohle nejde upočítat.

Řešení je prosté - musíme se vzdát přístupu "jedna dimenze po druhé".

Začneme vztahem, který známe již z kapitoly o kvadraturách

$$\int_M f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i).$$

Pro rovnoměrně rozdělené náhodné body chceme integrovat přesně konstantní funkce, tj.

$$f(x) = C \in \mathbb{R},$$

$$\implies \int_M f(x)dx = vol(M)C,\ \sum_{i=1}^{N} w_i f(x_i) = C\sum_{i=1}^{N} w_i.$$

Abychom integrovali přesně konstanty, musí platit

$$vol(M)C = CNw,$$

tj.

$$w = (w_1, \ldots, w_N) = \frac{vol(M)}{N},$$
$$\implies vol(M) = Nw.$$

Dohromady tedy dostaneme

$$\int_M f(x)dx \approx \frac{vol(M)}{N}\sum_{i=1}^{N} f(x_i),$$

$$\underbrace{\int_M 1dx}_{vol(M)} = \sum_{i=1}^{N} w_i 1.$$

4.2 Opakování z teorie pravděpodobnosti

Mějme X náhodnou veličinu na M spojenou s pravděpodobností $\mathbb{P}$

$$\mathbb{P}(|Z - Z_0| \le \delta) = \int_M \chi_{|Z-Z_0|\le\delta}(z)d\mathbb{P}z = \int_M \chi_{|Z-Z_0|\le\delta}(z)w(z)dz.$$

Pro střední hodnotu a rozptyl platí

$$\mathbb{E}(Z) = \int_M z d\mathbb{P}z = \int_M zw(z)dz,$$

$$Var(Z) = \mathbb{E}((Z - \mathbb{E}(Z))^2) = \mathbb{E}(Z^2) - \mathbb{E}(Z)^2,$$

kde $w(z)$ je tzv. probability density function (pdf) naší pravděpodobnostní míry $\mathbb{P}$ (např. pro rovnoměrné rozdělení máme $w(z) = 1/vol(M)$).

Platí tzv. Chebyshevovo lemma:

$$\mathbb{P}(|Z - \mathbb{E}(Z)| \ge \delta) \le \frac{Var(Z)}{\delta^2}.$$

4.3 Naše využití

Nechť X je náhodná veličina s hodnotami v M s rovnoměrným rozdělením, $(X_1, \ldots, X_N)$ je N náhodných, nezávislých veličin (také rovnoměrně rozdělených). Pak definujeme náhodnou veličinu

$$Y_N := \frac{vol(M)}{N} \sum_{i=1}^{N} f(X_i).$$

Pro tuto náhodnou veličinu platí

$$\mathbb{E}(Y_N) = vol(M)\mathbb{E}(f(X)) = \int_M f(x)dx,$$

$$Var(Y_N) = \frac{vol(M)^2}{N^2} NVar(f(X)) = vol(M)^2 \frac{Var(f(X))}{N}$$

.

Z Chebyshevova lemmatu máme

$$\mathbb{P}(|Y_N - \mathbb{E}(Y_N)| \geq \delta) \leq vol(M)^2 \frac{1}{\delta^2} \frac{Var(f(X))}{N}.$$

Nyní tuto rovnost přepíšeme úhledněji. Zafixujeme $\varepsilon > 0$ a zadefinujeme si $\delta > 0$ jako

$$\delta := \sqrt{vol(M) \frac{Var(f(X))}{\varepsilon N}}.$$

Pak platí

$$\mathbb{P}\left(\left|Y_N - \int_M f(x)dx\right| \geq \delta\right) \leq \varepsilon$$

a odtud dostáváme

$$\mathbb{P}\left(\left|Y_N - \int_M f(x)dx\right| \leq \sqrt{vol(M) \frac{Var(f(X))}{\varepsilon N}}\right) > 1 - \varepsilon.$$

Tato rovnost se dá interpretovat následovně - pokud samplujeme iid náhodné veličiny, máme $\varepsilon > 0$ a $\delta > 0$ velmi malé, pak nám metoda Monte Carlo dá s vysokou pravděpodobností velmi dobrou aproximaci integrálu.

Pro iid $(X_i)_{i=1}^N$ s rovnoměrným rozdělením a pro fixní $\varepsilon > 0$ chyba klesá (s pravděpodobností $1 - \varepsilon$) asymptoticky jako $\mathcal{O}(1/\sqrt{N})$, tj. velmi pomalu.

Jde tuto konvergenci chyby nějak zrychlit? Co když vybereme i jiná rozdělení než rovnoměrná? Tyto otázky nás dostávají do dalších podkapitol.

4.4 Importance sampling

Pro libovolné funkce $f(x)$ a $w(x)$, kde $w(x)$ je pdf, platí

$$\int_M f(x)dx = \int_M \frac{f(x)}{w(x)} w(x)dx \equiv \int_M \frac{f(\tilde{x})}{w(\tilde{x})} dW\tilde{x},$$

kde W je pravděpodobnostní míra odpovídající $w(x)$.

Pak pro iid náhodné veličiny $(X_1, \ldots, X_N)$ a rozdělení daného $W(X)$ platí

$$\int_M f(x)dx = \int_M \frac{f(x)}{w(x)} dW(x) = \mathbb{E}_W\left(\frac{f(X)}{w(X)}\right) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{w(X_i)}.$$

Označme

$$Y_N := \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{w(X_i)},$$

a máme

$$\mathbb{E}_W(Y_N) = \frac{1}{N}\sum_{i=1}^{N}\int_M \frac{f(x)}{w(x)}dW(x) = \int_M f(x)dx,$$

$$Var_W(Y_N) = \frac{Var_W(\frac{f(X)}{w(X)})}{N} = \mathbb{E}_W\left((\frac{f(X)}{w(X)})^2\right) - \underbrace{\mathbb{E}_W\left((\frac{f(X)}{w(X)})\right)^2}_{=\mathbb{E}(f(X))^2}.$$

Pak nám Chebyshevovo lemma dává

$$\mathbb{P}\left(\left|\int_M f(x)dx - \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{w(X_i)}\right| < \sqrt{\frac{Var_W(\frac{f(X)}{w(X)})}{\varepsilon N}}\right) > 1-\varepsilon.$$

Jde o přímé a úplné zobecnění Monte Carla pro libovolně rozdělené náhodné body.

Poznámka. Lze psát

$$\int_M f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i),$$

kde

$$w_i = \frac{1}{Nw(x_i)}.$$

Zjevně tedy stále nejde o klasickou kvadraturu. Je to pořád lineární kombinace funkčních hodnot v různých bodech, kde koeficienty nám zohledňují, jak moc je výskyt daného bodu pravděpodobný - je-li x_i dvakrát více pravděpodobné, pak má poloviční váhu. Tudíž název "importance sampling".

4.5 Další variace Monte Carla

4.5.1 Quasi Monte Carlo

Pokud si necháme vygenerovat 10^4 bodů v $(0,1)^2$, nedostaneme rovnoměrné pokrytí. Místo toho pozorujeme něco jako "lokální clusterování". To lze spravit tím, že nepoužijeme pseudo-random generátory (např. NumPy, SciPy, MATLAB, ...), ale tzv. quasi-random generátory.

Ty jsou navrženy tak, aby uniformně pokryly $(0,1)^d$. V principu negenerujeme náhodné body, ale deterministické.

Lze ukázat, že místo chyby $\mathcal{O}(N^{-1/2})$ dosáhneme $\mathcal{O}(\log^d(N)N^{-1})$.

4.5.2 Stratifikované Monte Carlo

Rozdělíme prostor Ω na podoblasti a aplikujeme Monte Carlo nezávisle.

4.5.3 Adaptivní Monte Carlo

Uděláme stratifikované Monte Carlo a na podoblastech s nejvyšší variací přidáme další samples - řešíme problém tam, kde je.

4.5.4 Antithetic Monte Carlo

Využijeme symetrii Ω, pokud nějaká je.
Když $X \in (0,1)$, pak pro $Y := 1 - X$ také platí, že $Y \in (0,1)$.
Vzorec pro Antithetic Monte Carlo

$$\frac{1}{2N} \sum_{i=1}^{N} f(X_i) + f(Y_i).$$

Platí

$$\mathbb{E}(\text{antithetic MC}) = \int_0^1 f(x)dx,$$

$$Var(\text{antithetic MC}) = \frac{Var(f(X))}{2n}(1+\rho),$$

kde $\rho = cor(X,Y)$.

Pokud umíme najít jednoduchou transformaci $X \to Y$, která bude mít malou korelaci, pak "zadarmo"získáme dvojnásobek sample bodů bez zvětšení variance.

5 ODR

V této sekci budeme pracovat s rovnostmi tvaru

$$y'(t) = f(t, y(t)),\ y(t_0) = y_0.$$

Budeme zkoumat rovnici tvaru

$$y'(t+\tau) =?,\ \tau > 0.$$

5.1 Úvod

Jevy ve světe obvykle umíme popsat pouze na základě jejich změn či vývoje v čase. Tyto změny představují derivace.

Obyčejné diferenciální rovnice, s kterými budeme v této kapitole pracovat, jsou nejméně použitelné na modelovaní složitých jevů. Ovšem jsou nejlepší pro budování intuice.

5.2 Příklady modelů

Neznámá je funkce jedné proměnné $y(t)$.

- Harmonický oscilátor

$$y''(t) + \omega^2 y(t) = \cos(t),\ y(0) = 1,\ y'(0) = 0$$

- Lotka-Voltera

$$\begin{cases} x'(t) = \alpha x(t) + x(t)y(t),\ x(0) = x_0 \\ y'(t) = -\gamma y(t) + \delta x(t)y(t),\ y(0) = y_0 \end{cases}$$

- SIR model

$$\begin{cases} S(t) = -\beta I(t)S(t),\ S(0) = S_0 \\ I(t) = \beta I(t)S(t) - \gamma I(t),\ I(0) = I_0 \\ R(t) = \gamma I(t),\ R(0) = R_0 \end{cases}$$

- Lorenzův model

$$\begin{cases} x'(t) = \sigma(y - x),\ x(0) = x_0 \\ y'(t) = x(\rho - z) - y,\ y(0) = y_0 \\ z'(t) = xy - \beta z,\ z(0) = z_0 \end{cases}$$

5.3 Eulerovy metody

Začneme se skalárním příkladem $y'(t) = f(t, y(t))$.

5.3.1 Odvození přes limitu

Platí

$$\lim_{\tau \to 0} \frac{y(t+\tau) - y(t)}{\tau} = f(t, y(t)) = \lim_{\tau \to 0} f(t+\tau, y(t+\tau)).$$

Tuto rovnost umíme pro malé $\tau > 0$ aproximovat dvěma způsoby, a to

- Eulerova explicitní metoda

$$y(t+\tau) = y(t) + \tau f(t, y(t)),$$

- Eulerova implicitní metoda

$$y(t+\tau) = y(t) + \tau f(t+\tau, y(t+\tau)).$$

5.3.2 Odvození přes integrál

$$\int_t^{t+\tau} y'(\sigma)d\sigma = \int_t^{t+\tau} f(\sigma, y(\sigma))d\sigma \implies y(t+\tau) = y(t) + \int_t^{t+\tau} f(\sigma, y(\sigma))d\sigma$$

Nyní využijeme kvadraturu

- Eulerova explicitní metoda

$$\int_\alpha^\beta g(x)dx \approx (\beta-\alpha)g(\alpha) \longrightarrow y(t+\tau) \approx y(t) + \tau f(t, y(t)),$$

- Eulerova implicitní metoda

$$\int_\alpha^\beta g(x)dx \approx (\beta-\alpha)g(\beta) \longrightarrow y(t+\tau) \approx y(t) + \tau f(t+\tau, y(t+\tau)),$$

- Rungeho metoda

$$\int_\alpha^\beta g(x)dx \approx (\beta-\alpha)g(\frac{\beta+\alpha}{2}) \longrightarrow y(t+\tau) \approx y(t)+\tau f(t+\frac{\tau}{2}, y(t+\frac{\tau}{2})) \approx y(t)+\tau f(t+\frac{\tau}{2}, y(t)+\frac{\tau}{2}f(t, y(t))).$$

5.3.3 Odvození přes Taylorův rozvoj

$$y(t+\tau) = y(t) + y'(t)(t+\tau-t) + \frac{y^{(2)}(t)}{2}\tau^2 + \frac{y^{(3)}(t)}{3!}\tau^3 + \ldots,$$

kde člen

$$\frac{y^{(2)}(t)}{2}\tau^2$$

jde rozvíjet následovně

$$y^{(2)}(t) = \frac{d}{dt}y'(t) = \frac{d}{dt}f(t, y(t)) = \frac{\partial}{\partial t}f(t, y(t)) + \frac{\partial}{\partial y}f(t, y(t))y'(t) = \frac{\partial}{\partial t}f(t, y(t)) + \frac{\partial}{\partial y}f(t, y(t))f(t, y(t))$$

a všechny další členy jdou rozvíjet analogicky.

Příklad 5.1 (Explicitní Euler)**.** *Mějme rovnici*

$$y'(t) = -y(t) + \cos(t),\ y(0) = y_0.$$

Pak platí

$$y(\tau) \approx y_1 = y_0 + \tau(y_0 + \cos(0)) = y_0 + \tau y_0 + \tau,$$
$$y(2\tau) \approx y_2 = y_1 + \tau(y_1 + \cos(\tau)).$$

▲

Příklad 5.2 (Implicitní Euler)**.** *Mějme opět tu samou rovnici*

$$y'(t) = -y(t) + \cos(t),\ y(0) = y_0.$$

Pak platí

$$y^{(\tau)} \approx y_1,$$
$$y_1 = y_0 + \tau(y_1 + \cos(0)),$$

a potřebujeme vyřešit

$$y_1 - \tau y_1 = y_0 + \tau,$$
$$y_1 = \frac{y_0 + \tau}{1 - \tau}.$$

V druhém kroku máme

$$y^{(2\tau)} \approx y_2,$$
$$y_2 = y_1 + \tau(y_2 + \cos(\tau)),$$

a potřebujeme vyřešit

$$y_2 = \frac{y_1 + \tau\cos(\tau)}{1 - \tau}.$$

▲

V praxi implicitní metody vyžadují spuštění nějakého řešiče pro rovnice

$$y_{k+1} = y_k + \tau f(t_k + \tau, y_{k+1}),$$

ale jsou zpravidla výrazně stabilnější.

Jak jsme viděli, Eulerovy metody jsou odvozeny z mid-point kvadratury, která je přesná pouze pro polynomy do 1. stupně. To pro nás znamená, že pokud tyto metody konvergují, tak pomalu.

Důsledek 5.3. *Pokud je naše pravá strana $f(t, y)$ hladká funkce, lze ukázat pro $\tau \to 0$*

$$\max_n |y_n - y(t_0 + n\tau)| = \mathcal{O}(\tau) \equiv \mathcal{O}(\tau^1)$$

a tedy říkáme, že Eulerovy metody jsou 1. řádu.

Jak zísáme metody vyšších řádů? Tato otázka nás dovede na celkem očekávanou odpověď - použijeme kvadratury vyšších řádů, čímž se dostáváme do další podkapitoly.

5.4 Metody typu Runge-Kutta

5.4.1 Odvození

Začneme s naší klasickou rovnicí

$$y'(t) = f(t, y(t)),$$
$$\int_t^{t+\tau} y'(\sigma)d\sigma = \int_t^{t+\tau} f(\sigma, y(\sigma))d\sigma,$$
$$y(t + \tau) = y(t) + \int_t^{t+\tau} f(\sigma, y(\sigma))d\sigma.$$

Jak nyní numericky aproximovat? Použijeme kvadraturu

$$y(t + \tau) = y(t) + \sum_{i=1}^{s} w_i f(\sigma_i, y(\sigma_i)),$$

pro nějaké body $\sigma_1, \ldots, \sigma_s \in (t, t + \tau)$ a nějaké váhy $w_1, \ldots, w_s \in \mathbb{R}$.

Analýza chyby se transformuje na analýzu chyby kvadratury. Jediný problém nastane u prvků $y(\sigma_1), \ldots, y(\sigma_s)$, které neznáme. Musíme je nejprve nějak odvodit.

5.4.2 Obecné vzorce

Nechť $b_1, \dots, b_s \in \mathbb{R}, c_1, \dots, c_s \in [0,1], a_{i,j} \in \mathbb{R}$. Pak umíme n-tou iteraci aproximovat následovně

- Explicitní Runge-Kutta

$$\begin{aligned} y(t+\tau) &= y(t) + \tau \sum_{i=1}^{s} b_i k_i, \\ k_1 &= f(t, y(t)), \\ k_2 &= f(t + c_2\tau, y(t) + \tau a_{2,1} k_1), \\ &\vdots \\ k_s &= f(t + c_s\tau, y(t) + \tau(a_{s,1}k_1 + \cdots + a_{s,s-1}k_{s-1})), \end{aligned}$$

- Implicitní Runge-Kutta

$$\begin{aligned} y(t+\tau) &= y(t) + \tau \sum_{i=1}^{s} b_i k_i, \\ k_1 &= f(t + c_1\tau, y(t) + \tau(a_{1,1}k_1 + \cdots + a_{1,s}k_s)), \\ k_2 &= f(t + c_2\tau, y(t) + \tau(a_{2,1}k_1 + \cdots + a_{2,s}k_s)), \\ &\vdots \\ k_s &= f(t + c_s\tau, y(t) + \tau(a_{s,1}k_1 + \cdots + a_{s,s}k_s)). \end{aligned}$$

Tohle jsou tzv. obecné s-stupňové Runge-Kutta metody, protože byla použita s-stupňová kvadratura.

Koeficienty se klasicky zapisují do tzv. Butcher table

$$\begin{array}{c|cccc} c_1 & 0 & 0 & \dots & 0 \\ c_2 & a_{2,1} & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ c_s & a_{s,1} & \dots & a_{s-1,s} & 0 \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

nebo zkráceně

$$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b} \end{array}$$

Důsledek 5.4. *Pokud je naše pravá strana $f(t,y)$ v explicitní Runge-Kutta metodě hladká funkce, lze ukázat pro $\tau \to 0$*

$$\max_n |y_n - y(t_0 + n\tau)| \equiv \mathcal{O}(\tau^p),$$

pro nějaké $p \in \mathbb{N}, p \leq s$, neboli explicitní Runge-Kutta metody jsou řádu nejvýše $p \leq s$, a jsou přinejlepším podmíněně stabilní.

Pro implicitní Runge-Kutta metody však dokážeme dosáhnout až stupně $p \leq 2s$. Tyto metody jsou nepodmíněně stabilní.

Tedy za práci navíc v podobě řešení soustavy rovnic pro získání $k_1, \dots, k_s$ jsme odměněni stabilitou a až dvojnásobným řádem konvergence.

5.4.3 Dahlquistova rovnice

V této části se podíváme na jeden z nejjednodušších netriviálních ODR. Tento příklad bude demonstrovat potřebu implicitních metod.

Příklad 5.5 (pomocí explicitního Eulera)**.** *Mějme rovnici*

$$y'(t) = \lambda y(t),$$

jejíž řešení je

$$y(t) = e^{\lambda t}.$$

Z formulky pro explicitního Eulera máme

$$y(t+\tau) \approx y(t) + \tau\lambda y(t) = (1+\tau\lambda)y(t),$$

a z linearity plyne

$$y(t+n\tau) = (1+\tau\lambda)^n y(t).$$

Pokud

- $\lambda = 0 \implies$ *přesné řešení je* $y(t) = y_0$*, tedy explicitní Euler bude přesný.*
- $\lambda > 0 \implies$ *přesné řešení je* $y(t) = y_0 e^{\lambda t}$*, roste exponenciálně rychle v čase a explicitní Euler to bude správně mimikovat a to jako* $y(t_0 + n\tau) \approx (1+\tau\lambda)^n y(t_0)$*.*
- $\lambda < 0 \implies$ *řešení bude oscilovat, pokud* $1 + \tau\lambda < 0$*. Jestli kromě záporné lambdy platí i* $|1+\tau\lambda| > 1$*, pak řešení nejen osciluje, ale i diverguje v absolutní hodnote do nekonečna, což je ještě horší.*

Takovému řešení se říká nestabilní.

▲

Příklad 5.6 (pomocí explicitní Runge-Kutta metody)**.** *Mějme situaci*

$$y(t+\tau) = R(\tau\lambda)y(t).$$

Pro studium stability pak stačí zkoumat, jak vypadá množina

$$\{z : |R(z)| \leq 1\}.$$

Protože $z = \tau\lambda$ *a* $\forall\lambda \in \mathbb{C}^-$*, máme přesné řešení* $y(t) = e^{\lambda t}$ *omezené, tj.*

$$\lambda : Re(\lambda) \leq 0.$$

Metoda muže být stabilní pouze pokud

$$\forall z \in \mathbb{C}^- : |R(z)| \leq 1,$$

tj.

$$\mathbb{C}^- \subseteq \{z : |R(z)| \leq 1\}.$$

Lze ukázat, že toho nejde dosáhnout pro explicitní metody. Jsou proto přinejlepším podmíněně stabilní.

▲

Příklad 5.7 (Pomocí implicitního Eulera)**.** *Máme*

$$y^{(t+\tau)} = y^{(t)} + \tau\lambda y^{(t+\tau)} = \frac{1}{1-\tau\lambda} y^{(t)},$$

z čeho plyne

$$y^{(t+n\tau)} = \left(\frac{1}{1-\lambda\tau}\right)^n y(t).$$

Pokud $\lambda \in \mathbb{C}^-$*, pak*

$$\left|\frac{1}{1-\tau\lambda}\right| < 1,$$

a tedy dostáváme nepodmíněně stabilní metodu.

▲

Věta 5.8. *Označne* $S := \{z \in \mathbb{C} : |R(z)| \leq 1\}$*. Řekneme, že metoda je A-stabilní (absolutne stabilní), právě tehdy když* $\mathbb{C}^-, \mathbb{R}^- \subset S$ ($\mathbb{C}^-$ *uvažujeme ve smyslu* $Re(z) \leq 0$)*.*

Poznámka. Implicitní Euler je A-stabilní, explicitní není.

To, že v implicitním Eulerovi máme rovnici a ne předpis, nám pomáhá se stabilitou. Dává smysl zkusit tento model adaptovat pro implicitní Runge-Kutta metody.

Lze ukázat, že pro správnou volbu butcher table dostaneme nepodmíněně stabilní algoritmy, které mají navíc vyšší řád konvergence (více viz podkapitola 5.4.2).

5.4.4 Runge-Kutta v praxi

V praxi musíme nejdříve zjistit, jestli naše ODR produkuje oscilace (tzv. Stiff problems). Pokud ne, vystačíme si s explicitní metodou. Pokud ano, bude potřeba použít implicitní metodu.

Zároveň se často používá tzv. adaptivní τ, tj. zkusíme brát τ co největší, a podle nějakých ukazatelů odhadujeme, jestli je potřeba ho zmenšit.

6 Nelineární algebraické rovnice

6.1 Úvod

U implicitní Runge-Kutta metody jsme viděli, že je potřeba řešit soustavu rovnic

$$\begin{bmatrix} k_1 \\ \vdots \\ k_s \end{bmatrix} = \begin{bmatrix} f(t + c_1\tau, y(t) + \tau \sum_{j=1}^{s} a_{1,j} k_j) \\ \vdots \\ f(t + c_s\tau, y(t) + \tau \sum_{j=1}^{s} a_{s,j} k_j) \end{bmatrix},$$

po úpravě

$$\vec{k} = G(\vec{k}),$$
$$F(\vec{k}) = \vec{b}.$$

V praxi není mnoho typů rovnic, které umíme vyřešit. Proto se je pokusíme aproximovat lineárním systémem, který následně spočteme.

Má to však jeden problém - lineární aproximace (tj. aproximace funkce pomocí přímky) je většinou dost nepřesná.

Tuto metodu však umíme trošku vylepšit, a to tím, že budeme postupovat iteračně. Tato idea nás posouvá k další podkapitole.

6.2 Newtonova metoda

Situace je následující - chceme aproximovat kořen $F(x)$ okolo nějakého bodu (našeho počátečního odhadu) x_0 jako

$$a_0 x + b_0,$$

tj. lineární aproximace, a najdeme aproximaci kořene $F(x) = 0$ jako

$$x_1 := \frac{-b_0}{a_0}.$$

Pak aproximujeme $F(x)$ okolo bodu x_1 jako

$$a_1 x + b_1,$$

najdeme druhou aproximaci kořene $F(x) = 0$ jako

$$x_2 := \frac{-b_1}{a_1},$$

atd.

6.2.1 Matematické odvození

Z Taylorova rozvoje máme

$$F(\vec{x}) = F(\vec{x}_0) + \left(\frac{d}{d\vec{x}} F(\vec{x}_0)\right)(\vec{x} - \vec{x}_0) + \mathcal{O}(||\vec{x} - \vec{x}_0||^2)_{x \to x_0}.$$

Dále máme

$$F : \mathbb{R}^d \to \mathbb{R}^d, F(\vec{x}) = \begin{bmatrix} F_1(\vec{x}) \\ F_2(\vec{x}) \\ \vdots \\ F_d(\vec{x}) \end{bmatrix},$$

a proto platí

$$\left(\frac{d}{d\vec{x}}F(\vec{x})\right) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(\vec{x}) & \dots & \frac{\partial F_1}{\partial x_d}(\vec{x}) \\ \vdots & & \vdots \\ \frac{\partial F_d}{\partial x_1}(\vec{x}) & \dots & \frac{\partial F_d}{\partial x_d}(\vec{x}) \end{bmatrix},$$

což je Jacobiho matice F (=: $J(\vec{x}) \equiv J_x$).

Místo $\vec{F}(\vec{x}) = 0$ budeme řešit

$$\vec{F}(\vec{x}_0) + J_{x_0}(\vec{x} - \vec{x}_0) = 0$$

$$J_{x_0}\vec{x} = J_{x_0}\vec{x_0} - \vec{F}(\vec{x}_0)$$

$$J_{x_0}\vec{x} = \vec{b}_0,$$

což je soustava lineárních rovnic - řešení označíme x_1. Pak platí

$$\vec{x}_1 = \vec{x}_0 + v$$

a $\vec{v}$ je řešení

$$J_{x_0}\vec{v} = -\vec{F}(\vec{x}_0).$$

Z tohoto postupu dokážeme odvodit obecný vzorec

$$\vec{x}_{k+1} := \vec{x}_k + \vec{v},$$

kde $\vec{v}$ je řešení

$$J_{x_k}\vec{v} = -F(\vec{x}_k),$$

neboli

$$J_{x_k}\vec{x}_{k+1} = J_{x_k}\vec{x}_k - \vec{F}(\vec{x}_k),$$

$$\vec{x}_{k+1} = \vec{x}_k - J_{x_k}^{-1}\vec{F}(\vec{x}_k),$$

$$J_{x_k}^{-1}\vec{F}(\vec{x}_k) = \vec{x}_k - \vec{x}_{k+1}.$$

6.2.2 Analýza konvergence

Mějme kořen $\vec{x}^*$, tj. $\vec{F}(x^*) = 0$. Pak v k-tém kroku Newtonovy metody platí

$$0 = F(\vec{x}^*) = F(\vec{x}_k) + J_{x_k}(\vec{x}^* - \vec{x}_k) + \mathcal{O}(||x^* - x_k||^2),$$

$$-J_{x_k}(\vec{x}^* - \vec{x}_k) = F(\vec{x}_k) + \mathcal{O}(||x^* - x_k||^2),$$

$$J_{x_k}(\vec{x}_k - \vec{x}^*) = F(\vec{x}_k) + \mathcal{O}(||x^* - x_k||^2),$$

$$\vec{x}_k - \vec{x}^* = J_{x_k}^{-1}F(\vec{x}_k) + \mathcal{O}(||x^* - x_k||^2),$$

$$\vec{x}_k - \vec{x}^* = \vec{x}_k - \vec{x}_{k+1} + \mathcal{O}(||x^* - x_k||^2),$$

$$||x^* - x_{k+1}|| = \mathcal{O}(||x^* - x_k||^2)_{x_k \to x^*},$$

a platí

$$\exists C \neq 0 : \lim_{k \to +\infty} \frac{||x^* - x_{k+1}||}{||x^* - x_k||^2} = C,$$

což je tzv. kvadratická konvergence (jedna iterace zmenší chybu kvadraticky).

Celá tato metoda může však selhat, pokud J_{x_k} nebude regulární - x_{k+1} nemusí existovat. Stejně tak se může stát, že dostaneme posloupnost $x_0, x_1, x_2, \dots$, která nekonverguje.

6.3 Banachova věta

Jak jsme viděli, Newtonova metoda - jelikož je odvozena pomocí Taylora - funguje jen lokálně, tj. můžeme zaručit konvergenci pouze pokud náš počáteční bod x_0 bude dostatečně blízko přesnému řešení x^*.

Náš úkol je tedy jasný - vymyslet způsob, jak se dostat dostatečně blízko k řešení, a následně budeme moci spustit Newtona. Přeformulujeme náš problém na hledání tzv. pevného bodu

$$\tilde{x} \text{ je pevný bod } F(x) \overset{\text{def}}{\Leftrightarrow} F(\tilde{x}) = \tilde{x}.$$

Na hledání pevných bodů máme hezkou větu.

Věta 6.1 (Banachova věta o pevném bodě)**.** *Nechť $M \subseteq \mathbb{R}^n$ je uzavřená a $G : M \to M$ je zobrazení, pro které platí*

$$\exists q \in [0,1) \; \forall x, y \in M : ||G(x) - G(y)|| \leq q||x - y||$$

(říkáme, že G je tzv. kontrakce na M). Pak

$$\exists! \tilde{x} : G(\tilde{x}) = \tilde{x},$$

a navíc

$$\forall x_0 \in M : x_0, G(x_0), G(G(x_0)), G(G(G(x_0))), \ldots$$

konverguje k $\tilde{x}$ lineárně s rychlostí q, tj.

$$\lim_{k \to +\infty} \frac{||x_{k+1} - \tilde{x}||}{||x_k - \tilde{x}||} = q.$$

Důkaz. Mějme posloupnost bodů $x_0, x_1, \ldots$ tak, že $G(x_{k+1}) = x_k$. Pak platí

$$||x_{k+1} - x_k|| = ||G(x_k) - G(x_{k-1})|| \leq L||x_k - x_{k-1}|| \leq L^k||G(x_0) - x_0||.$$

□

Poznámka. Pokud

$$\forall a \in M : ||\frac{d}{dx}G(a)|| \leq q < 1,$$

pak platí i Banachova věta.

6.4 Směr Newtonovy metody

Klasický problém je následující situace - začali jsme se Newtonovou metodou přibližovat ke kořenu, ale pak jsme ho přestřelili, a kvůli tomu jsme zkonvergovali k jinému kořenu.

Místo toho jde postupovat úměrněji. Spočítáme řešení

$$J_{x_k} d_k = F(x_k)$$

a položíme

$$x_{k+1} = x_k + \tau_k d_k,$$

kde d_k je tzv. směr Newtonovy metody a τ_k je délka kroku ve směru d_k.

6.4.1 Vyřešení Jacobiho matice

V praxi u hodně problémů nemáme k dispozici explicitní formulku pro F, máme pouze program, který spočítá $x \to F(x)$ (a i toto vyhodnocování může trvat dlouho a být náročné).

Je zjevné, že stejně tak nemáme přístup k $\partial F_i / \partial x_j$. Musíme ho aproximovat následovně.

$$\frac{\partial}{\partial x_j} F_i(a) = \lim_{h \to 0} \frac{F_i(a_1, \ldots, a_{j-1}, a_j + h, a_{j+1}, \ldots, a_n) - F_i(a_1, \ldots, a_{j-1}, a_j - h, a_{j+1}, \ldots, a_n)}{2h} \approx$$

$$\approx \frac{F_i(a_1, \ldots, a_{j-1}, a_j + h, a_{j+1}, \ldots, a_n) - F_i(a_1, \ldots, a_{j-1}, a_j - h, a_{j+1}, \ldots, a_n)}{2h} =: FD_j(F_i, a, h),$$

což je tzv. Finite difference (konečná diference) pro funkci $F_i(x)$ v bodě a a v j-té proměnné.

Matici J_{x_k} lze aproximovat maticí $\tilde{J}_{x_k}$ definovanou jako

$$[\tilde{J}_{x_k}]_{i,j} := FD_j(F_i, x_k, h), \ h > 0.$$

K sestavení $\tilde{J}_{x_k}$ potřebujeme $2n$ vyčíslení $F(x)$ na jeden krok. To může být příliš výpočtově obtížné. Existují další aproximační metody, které za cenu více aproximací sníží počet vyčíslení $F(x)$ pro jednu iteraci. Jde o tzv. Quasi-Newton metody.

7 Lineární algebraické rovnice

7.1 Úvod

7.1.1 Motivace

- normální rozdělení více proměnných $\vec{x} \sim N(\vec{\mu}, \Sigma)$ (tedy $\mathbb{E}(\vec{x}) = \vec{\mu}$ a $cov(x_i,x_j) = \Sigma_{i,j}) \to \Sigma$ je tedy symetrická pozitivně definitní matice $\implies$ hustota rozdělení je $\frac{exp(\frac{-1}{2} \cdot (x-\mu)^T \Sigma^{-1} (x-\mu))}{\sqrt{(2\pi)^n \cdot \det(\Sigma)}} \to$ výpočet $z := \Sigma^{-1}(x-\mu)$ odpovídá řešení soustavy $\Sigma \cdot z = x - \mu$
- systémy ODR: nepodmíněně stabilní metody $\equiv$ každý časový krok vyžaduje řešení soustavy algebraických rovnic
- nelineární soustavy algebraických rovnic: 1 krok Newtonovy metody $\equiv$ 1 soustava algebraických rovnic
- parciální diferenciální rovnice (vedení tepla, proudění tekutin, šíření vln, vývoj cen opcí, ...)

7.1.2 Black-Scholes model

Black-Scholes model je jeden z nejznámějších modelů pro oceňování opcí. Mějme následující proměnné:

- t je čas; $t \in (0,T)$
- $S(t)$ značí cenu konkrétní komodity
- r je bezriziková úroková míra
- $V(t,S(t))$ je cena opce založená na $S(t)$
- σ je volatilita komodity $S(t)$

Pak platí

$$\frac{\partial}{\partial t} V(t,S(t)) + \frac{(\sigma S)^2}{2} \cdot \frac{\partial^2}{(\partial S(t))^2} V(t,S(t)) = rV(t,S(t)) - rS(t) \cdot \frac{\partial}{\partial S(t)} V(t,S(t)),$$

spolu s podmínkami BC (boundary conditions) a IC (initial conditions).

Po vhodné transformaci proměnných dostaneme

$$\frac{\partial}{\partial t} u(t,x) + \frac{\sigma^2}{2} \cdot \frac{\partial^2}{\partial x^2} u(t,x) = 0,$$

kde x je transformované $S(t)$ (transformovaná cena komodity) ... $x = ln(\frac{S}{K(S)}) + (T-t)(r - \frac{\sigma^2}{2})$; $x \in (0,L)$; $K(S)$ je "výplata" opce na základě ceny komodity a $u(t,x)$ je transformované $V(t,S(t))$ (tranformovaná cena opce) ... $u(t,x) = V(t,S(t)) \cdot e^{r \cdot (T-t)}$.

My si tento model (nerealisticky) zjednodušíme a položíme $r = \frac{\sigma^2}{2} = \alpha$ a dostaneme

$$\frac{\partial}{\partial t} u(t,x) = \frac{-\sigma^2}{2} \cdot \frac{\partial^2}{\partial x^2} u(t,x) \tag{7.1}$$

$$u(0,x) = u^0(x) \text{ [počáteční podmínka]},$$

$$u(t,0) = g_0(t) \wedge u(t,L) = g_L(t) \text{ [okrajové podmínky]}.$$

Poznámka. Shodou okolností se tato rovnice používá i k modelování spousty jiných fenoménů.

Jak získat (aproximaci) $u(t.x)$?

Podobně jako u ODR se spokojíme s aproximací u pouze v jistých bodech. Konkrétně, v bodech $x_i \in (0,L)$ rovnoměrně pokrývajících interval $(0,L)$, tj.

$$x_i = i \cdot h$$

a v každém bodě x_i budeme aproximovat řešení $u(t,x_i)$ pomocí funkce $u_i(t)$.

***KROK* 1** (Aproximace pravé strany)

$$\frac{\partial^2}{\partial x^2}u(t,x_i) = \lim_{h\to 0}\frac{\frac{\partial}{\partial x}u(t,x_{i+1}) - \frac{\partial}{\partial x}u(t,x_i)}{h} \approx \frac{\frac{\partial}{\partial x}u(t,x_{i+1}) - \frac{\partial}{\partial x}u(t,x_i)}{h} =$$

$$= \frac{\lim_{h\to 0}\frac{u(t,x_{i+1})-u(t,x_i)}{h} - \lim_{h\to 0}\frac{u(t,x_i)-u(t,x_{i-1})}{h}}{h} \approx \frac{u(t,x_{i+1} - 2u(t,x_i) + u(t,x_{i-1})}{h^2}.$$

Poznámka. U obou "≈"používáme aproximaci derivace pomocí diference, tzv. metoda konečných diferencí. V našem modelu toto můžeme snadno odvodit pomocí Taylorova rozvoje

$$\frac{\partial^2}{\partial x^2}u(t,x_i) = \frac{u(t,x_{i+1} - 2u(t,x_i) + u(t,x_{i-1})}{h^2} + \mathcal{O}(h^2).$$

Pravá strana může být tedy aproximována lineární rekurencí s chybou (tzv. diskretizační chyba) řádu $\mathcal{O}(h^2)$.

***KROK* 2** (ODR zápis)

Nyní aproximujeme (7.1) systémem ODR pro funkce $u_1(t),\ldots,u_{n-1}(t)$:

$$\begin{array}{c} u_1'(t) = \frac{\sigma^2}{2h^2}(u_2(t) - 2u_1(t) + u_0(t)) \\ u_2'(t) = \frac{\sigma^2}{2h^2}(u_3(t) - 2u_2(t) + u_1(t)) \\ \vdots \\ u_{n-2}'(t) = \frac{\sigma^2}{2h^2}(u_{n-1}(t) - 2u_{n-2}(t) + u_{n-3}(t)) \\ u_{n-1}'(t) = \frac{\sigma^2}{2h^2}(u_n(t) - 2u_{n-1}(t) + u_{n-2}(t)) \\ u_i(0) = u^0(x_i) \end{array} \tag{7.2}$$

Platí zde, že $u_0(t) \equiv g_0(t)$ a $u_n(t) \equiv g_L(t)$ - známe z (7.1)

Vektorový zápis:

$$\vec{u}(t) \equiv \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{n-1}(t) \end{bmatrix}; L = \frac{\sigma^2}{2h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \ldots & 0 \\ 1 & -2 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & -2 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 & -2 & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 & -2 & 1 \\ 0 & 0 & \ldots & 0 & 0 & 1 & -2 \end{bmatrix}; \vec{g}(t) = \frac{\sigma^2}{2h^2}\begin{bmatrix} g_1(t) \\ g_2(t) \\ \vdots \\ g_L(t) \end{bmatrix}$$

A tedy (7.2) platí, pokud:

$$\vec{u}'(t) = L \cdot \vec{u}(t) + \vec{g}(t),$$

$$\vec{u}(0) = \begin{bmatrix} u^0(x_0) \\ \vdots \\ u^0(x_n) \end{bmatrix}.$$

***KROK* 3**

Chceme předpovědět vývoj ceny. Použijeme implicitního Eulera

- $\vec{u}(\tau) \approx \vec{u}_1$ & $\vec{u}_1 = \vec{u}_0 + \tau(L \cdot \vec{u}_1 + \vec{g}(\tau)) \iff (I - \tau L)\vec{u}_1 = \vec{u}_0 + \tau \cdot \vec{g}(\tau)$
- $\vec{u}(2\tau) \approx \vec{u}_2$ & $\vec{u}_2 = \vec{u}_1 + \tau(L \cdot \vec{u}_2 + \vec{g}(2\tau)) \iff (I - \tau L)\vec{u}_2 = \vec{u}_1 + \tau \cdot \vec{g}(2\tau)$
- $\vec{u}(3\tau) \approx \vec{u}_3$ & $\vec{u}_3 = \vec{u}_2 + \tau(L \cdot \vec{u}_3 + \vec{g}(3\tau)) \iff (I - \tau L)\vec{u}_3 = \vec{u}_2 + \tau \cdot \vec{g}(3\tau)$
- ...

Dostáváme aproximaci cen v závislosti na čase a hodnotě komodity S.

Poznámka. $(I - \tau L)$ je vždy matice a $\vec{u}_k + \tau \cdot \vec{g}(k\tau)$ je vždy pravá strana - dostáváme tedy v každém bodě systém lineárních algebraických rovnic.

***KROK* 4**

V některých aplikacích nám stačí spočítat (aproximovat) tzv. steady state - ekvilibrium, ke kterému $\vec{u}(t)$ směřuje a tedy stav, kdy se $\vec{u}(t)$ už s časem nemění. Taková situace nastane, právě když

$$\vec{u}'(t) = 0,$$

což je právě tehdy, když

$$-L\vec{u}(t) = \vec{g}(t).$$

7.1.3 Typy metod k nalezení řešení

Z kurzu lineární algebry již víme, že když matice A je čtvercová a regulární, pak existuje právě jedno $\vec{x} \in \mathbb{R}^n$ takové, že $A\vec{x} = \vec{b}$; $\vec{b} \in \mathbb{R}^n$.

Jaké metody tedy známe k nalezení hledaného x? Dělíme je do dvou hlavních kategorií podle principů, na kterých fungují.

- Přímé metody
  - založené na Gaussově eliminaci a LU (Choleského) rozkladu
  - řešení x (jeho aproximaci) získám až na konci výpočtu
  - není lehké volit požadovanou přesnost řešení
  - vyžadují přístup ke (skoro) celé matici A
- Iterační metody
  - různé přístupy ke generování stále se zlepšujících aproximací
  - ne vždy je třeba mít přístup k A (stačí část nebo jen informace o $\vec{v} \to A\vec{v}$)
  - snadnější volba přesnosti řešení
  - menší záruky na výpočetní čas
  - možná závislost rychlosti konvergence na konkrétní pravé straně b

7.2 Gaussova eliminace a LU rozklad

7.2.1 Gaussova eliminace

Idea za touto metodou je, že vyřešit matici soustavy rovnic $A\vec{x} = \vec{b}$ je obtížné, pokud je A obecná čtvercová matice. Mnohem jednodušší je takovou soustavu řešit, máme-li A v odstupňovaném tvaru. My se tedy v této metodě budeme zejména snažit převést všechny matice do odstupňovaného tvaru.

Poznámka. Uvažujeme zde soustavy rovnic $A\vec{x} = \vec{b}$; $\vec{b} \in \mathbb{R}^n$; $\vec{x} \in \mathbb{R}^n$. Řádky a sloupce budeme značit "pythonovsky", tedy i-tý řádek matice budeme značit jako $A_{i,:}$ a j-tý sloupec matice budeme značit jako $A_{:,j}$.

Vandermondovská matice

V této podkapitole se budeme zabývat příkladem Vandermondovské matice - řešíme interpolaci hodnot $\{1, -1, 1\}$ v bodech $\{1,2,3\}$, což odpovídá následujícímu řešení soustavy $A\vec{x} = \vec{b}$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Prvními algoritmickými úpravami z této soustavy získáme soustavu $A^1\vec{x} = \vec{b}^1$, která bude vypadat následovně:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}.$$

Druhým krokem Gaussovy eliminace již získáme soustavu $A^2\vec{x} = \vec{b}^2$, tedy

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix}.$$

Poznámka. Horní indexy jsou zde opravdu indexy, ne mocniny matic nebo vektorů!

Tvrzení 7.1. *Všechny ekvivalentní úpravy matic si umíme vyjádřit jako matici M^n, pro kterou platí, že, pokud s ní vynásobíme matici A^{n-1} zleva, pak získáme matici A^n.*

Pojďme si tedy nyní každý krok rozebrat trochu podrobněji.

***KROK* 1**

Podle předchozího tvrzení by tedy měla existovat matice M^1 taková, že

$$A^1 = M^1 A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} A,$$

protože

$$\begin{aligned} A^1_{1,:} &= 1 \cdot A_{1,:} + 0 \cdot A_{2,:} + 0 \cdot A_{3,:} \\ A^1_{2,:} &= -1 \cdot A_{1,:} + 1 \cdot A_{2,:} + 0 \cdot A_{3,:} \\ A^1_{3,:} &= -1 \cdot A_{1,:} + 0 \cdot A_{2,:} + 1 \cdot A_{3,:}. \end{aligned} \tag{7.3}$$

Poznámka. Pro obecnou matici $A_{i,j}$ bychom získali následující matici

$$M^1 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{-a_{2,1}}{a_{1,1}} & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \frac{-a_{n-1,1}}{a_{1,1}} & 0 & \cdots & 1 & 0 \\ \frac{-a_{n,1}}{a_{1,1}} & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Odečteme-li tuto matici od jednotkové (se kterou se vždycky dobře počítá), dostaneme

$$M^1 = I - \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \frac{a_{2,1}}{a_{1,1}} & 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \frac{a_{n-1,1}}{a_{1,1}} & 0 & \cdots & 0 & 0 \\ \frac{a_{n,1}}{a_{1,1}} & 0 & \cdots & 0 & 0 \end{bmatrix}$$

a toto můžeme ještě upravit na

$$M^1 = I - \begin{bmatrix} 0 \\ \frac{a_{2,1}}{a_{1,1}} \\ \vdots \\ \frac{a_{n,1}}{a_{1,1}} \end{bmatrix} [1\ 0 \dots 0],$$

co lze přepsat do kompaktnějšího zápisu jako

$$M^1 = I - \begin{bmatrix} 0 \\ \frac{A_{2:n,1}}{a_{1,1}} \end{bmatrix} \cdot \vec{e}_1.$$

KROK 2

$$A^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} A^1,$$

protože

$$\begin{aligned} A^2_{1,:} &= 1 \cdot A^1_{1,:} + 0 \cdot A^1_{2,:} + 0 \cdot A^1_{3,:} \\ A^2_{2,:} &= 0 \cdot A^1_{1,:} + 1 \cdot A^1_{2,:} + 0 \cdot A^1_{3,:} \\ A^2_{3,:} &= 0 \cdot A^1_{1,:} + -2 \cdot A^1_{2,:} + 1 \cdot A^1_{3,:} \end{aligned} \tag{7.4}$$

Také ovšem platí, že

$$A^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} A,$$

tedy

$$A^2 = M^2 M^1 A.$$

Poznámka. Pro obecnou matici $A_{i,j}$ bychom obdobným odvozením jako v předešlé poznámce získali následující matici

$$M^2 = I - \begin{bmatrix} 0 \\ 0 \\ \frac{a_{3,2}}{a_{2,2}} \\ \vdots \\ \frac{a_{n,2}}{a_{2,2}} \end{bmatrix} [0\ 1\ 0 \dots 0] = I - \begin{bmatrix} 0 \\ 0 \\ \frac{A_{3:n,2}}{a_{2,2}} \end{bmatrix} \cdot \vec{e}_2.$$

V našem případě jsme po 2 krocích hotoví. Místo těžkého systému $A\vec{x} = \vec{b}$ máme ekvivalentní lehký systém $A^2\vec{x} = \vec{b}^2$, přičemž A^2 je horní trojúhelníková matice.

7.2.2 LU rozklad

Pro obecnou matici A skončíme až v kroku (n-1) a budeme mít A^{n-1} horní trojúhelníkovou matici

$$A^{n-1} = M^{n-1}M^{n-2}\dots M^1 A,$$

kde

$$M^k = I - \vec{m}^k \cdot \vec{e}_k^T,$$

$$\vec{m}^k = \frac{1}{a_{k,k}^{k-1}} \cdot \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_{k+1,k}^{k-1} \\ a_{k+2,k}^{k-1} \\ \vdots \\ a_{n,k}^{k-1} \end{bmatrix}.$$

Označme si

$$M := M^{n-1}M^{n-2}\dots M^1,$$

$$U := A^{n-1},$$

pak

$$U = MA,$$

$$A = M^{-1}U,$$

$$A = (M^1)^{-1}(M^2)^{-1}\dots(M^{n-1})^{-1}U.$$

Navíc platí, že

$$(M^k)^{-1} = (I - \vec{m}^k \cdot \vec{e}_k^T)^{-1} = I + \vec{m}^k \cdot \vec{e}_k^T =: L_k$$

Poznámka. Pro každé k platí, že L_k je dolní trojúhelníková matice.

Dostáváme, že

$$A = L_1 L_2 \dots L_{n-1} U$$

a když zavedeme, že

$$L := L_1 L_2 \dots L_{n-1},$$

dostáváme

$$A = LU.$$

Tímto jsme tedy získali LU rozklad matice A.
Gaussova eliminace pro $A\vec{x} = \vec{b}$ pak tedy odpovídá:

$$A\vec{x} = \vec{b} \to U\vec{x} = L^{-1}\vec{b}$$

a tuto soustavu vyřešíme (trojúhelníková matice - lehké).

LU řešič pro $A\vec{x} = \vec{b}$ pak vypadá následovně:

- spočteme L, U takové, že $A = LU$ jako výše,
- spočteme $\vec{y} = L^{-1}\vec{b}$ - to odpovídá řešení soustavy $Ly = b$ (tady máme trojúhelníkovou matici - lehké),
- vyřešíme soustavu $U\vec{x} = \vec{y}$ (tady máme také trojúhelníkovou matici - lehké).

Poznámka. Výpočet LU faktorizace probíhá stejně jako Gaussova eliminace výše, budeme si pouze ukládat i ty vektory $\vec{m}^k$, které definují M^k. Máme totiž

$$L := L_1 L_2 \dots L_{n-1} = (I + \vec{m}^1 \cdot \vec{e}_1^T)(I + \vec{m}^2 \cdot \vec{e}_2^T) \dots (I + \vec{m}^{n-1} \cdot \vec{e}_{n-1}^T) =$$

$$= I + \vec{m}^1 \cdot \vec{e}_1^T + \vec{m}^2 \cdot \vec{e}_2^T + \dots + \vec{m}^{n-1} \cdot \vec{e}_{n-1}^T = I + [\vec{m}^1 | \vec{m}^2 | \dots | \vec{m}^{n-1} | \vec{0}]$$

(rozumíme tím matici, která má ve sloupcích vektory $\vec{m}^j$ a na diagonále 1).

7.2.3 Pivotace

Při použití LU rozkladu vlastně "navíc" sestavujeme $L = M^{-1}$ místo toho, abychom matici M rovnou aplikovali na vektor b.

Poznámka. Jedná se opravdu jen o sestavování, jelikož umíme vyjádřit matici $(M^k)^{-1}$.

LU rozklady matice jsou tedy užitečné, pokud budeme řešit mnoho systémů $A\vec{x} = \vec{b}_l$ pro různé vektory $\vec{b}_1, \vec{b}_2, \dots$, které ještě ale neznáme (jako u implicitního Eulera/Runge-Kutta metody). Rozklad nám tedy umožňuje používat faktory opakovaně.

Pokud máme pouze jeden systém nebo posloupnost, kde se mění i matice A, je rychlejší nepočítat L, ale rovnou využít Gaussovu eliminaci.

Takto můžeme ale ovšem počítat pouze pokud platí, že všechny pivoty jsou nenulové, což platí, když

$$a_{k,k}^{k-1} \neq 0, \ \forall k \in \{1,2,\dots,n-1\} \ (\Leftrightarrow a_{1,1}^0 \equiv a_{1,1}).$$

Takovou vlastnost mají silně regulární matice.

Tvrzení 7.2. *Matice je silně regulární právě tehdy, když platí, že všechny diagonální podmatice jsou regulární.*

Co když ale máme pouze regulární matici, ale $\exists\, k \leq n-1 : a_{k,k}^{k-1} = 0$ (případně ≈ 0)?

Mějme tedy takové k a rozepíšeme-li si A^k uvidíme, že k-tý sloupec matice bude vypadat

$$[\underbrace{0, \dots, 0}_{\text{k-1}}, \quad \underbrace{v_k, \dots, v_n}_{\text{vektor } v \in \mathbb{R}^{n-k+1}} \quad]^T.$$

Jelikož A je regulární, nemůže se stát, že

$$\vec{v} = [0, \dots, 0]^T \implies \exists\, l \in \{k, \dots, n\} : v_l \neq 0.$$

Vezměme si $l \leq n$, pak dostaneme, že

$$|v_l| = \max_{i \leq n} |v_i|$$

a teď, když prohodíme řádky k a l v matici A^k, dostaneme

$$\tilde{A}^k = P_k A^k.$$

Algoritmicky můžeme tedy dále počítat s $\tilde{A}^k$ jako s A^k. Tento algoritmus nazýváme částečná pivotace (partial pivoting).

Obecně tedy máme

$$A^k = M^k \tilde{A}^{k-1} = M^k P_{k-1} A^{k-1}.$$

Poznámka. P_{k-1} je identita, pokud řádky není potřeba prohodit, a permutační matice, pokud řádky prohazujeme.

Lze ukázat, že

$$A^{n-1} = U = M^{n-1}P_{n-2}\dots P_2M^2P_1M^1P_0A$$

a také platí

$$P_kM^k = P_k(I - \vec{m}^k \cdot \vec{e}_k^T) = P_k - P_k\vec{m}^k\vec{e}_k^T \overset{P_k = P_k^T}{=} P_k - \vec{m}^k\vec{e}_k^TP_k = (I - \vec{m}^k\vec{e}_k^T)P_k,$$

a tedy

$$A^{n-1} = U = M^{n-1}\tilde{M}^{n-2}\dots\tilde{M}^1\underbrace{P_{n-2}P_{n-1}\dots P_1P_0}_{:=P}A,$$

takže získáme

$$PA = LU.$$

LU je zde faktorizace s částečnou pivotizací.
Soustavu $A\vec{x} = \vec{b};\ PA = LU$ budeme řešit následovně:

$$L\vec{y} = P^T\vec{b},$$

$$U\vec{x} = \vec{y}.$$

Poznámka. Existují i jiné pivotace, ty ale v tomto kurzu vynecháme.

Důsledek 7.3. *Pokud je A regulární, existují matice P (permutační), L (dolní trojúhelníková) a U (horní trojúhelníková) takové, že PA = LU.*

Jednoznačnost zde závisí na volbě pivotační strategie. Pro částečnou pivotaci jsou P,L,U jednoznačné.

Věta 7.4 (Wilkinsonova)**.** *Nechť A je regulární matice a nechť má LU faktorizaci s částečnou pivotací PA = LU. Označme si $\tilde{P}$,$\tilde{U}$,$\tilde{L}$ matice získané počítačovým výpočtem v konečné aritmetice s machine precision ε_{mach}. Pak platí, že*

$$\frac{max_{i,j}|(\tilde{L}\tilde{U})_{i,j} - (\tilde{P}A)_{i,j}|}{max_{i,j}|(\tilde{P}A)_{i,j}|} \leq c\frac{max_{i,j,k}|\tilde{a}_{i,j}^k|}{max_{i,j}|(\tilde{P}A)_{i,j}|}n \cdot \varepsilon_{mach} + \mathcal{O}(\varepsilon_{mach}^2).$$

Poznámka. Zlomek na pravé straně nerovnosti je tzv. růstový faktor - umíme ho během výpočtu monitorovat a pokud nebude příliš velký, tak víme, že se zaokrouhlovací chyby příliš neakumulují.

$$\frac{max_{i,j,k}|\tilde{a}_{i,j}^k|}{max_{i,j}|(\tilde{P}A)_{i,j}|} \leq \frac{max_{i,j}|\tilde{L}_{i,j}| \cdot max_{i,j}|\tilde{U}_{i,j}|}{max_{i,j}|(\tilde{P}A)_{i,j}|}.$$

Z této věty nám plyne, že Gaussova eliminace s částečnou pivotací není vždy stabilní (říkáme, že je podmíněně stabilní), protože pro některé matice můžou limity PC aritmetiky (např. zaokrouhlování) zcela dominovat výpočtu.

V praxi se ale GE používá běžně a pro mnoho konkrétních problémů lze stabilitu zaručit za využití dalších vlastností A.

Pokud je A symetrická pozitivně definitní matice (SPD), lze ukázat, že $a_{k,k}^{k-1} \neq 0$ a tedy není třeba pivotace.

7.2.4 Choleského faktorizace

Když si vezmeme LU faktorizaci $A = LU$, pak $A = A^T$. Samotná faktorizace však symetrická není ($L \neq U^T$).

Pokud však položíme

$$A = L \cdot diag(U) \cdot \underbrace{diag(U)^{-1}U}_{R} =: LDR,$$

pak lze snadno ukázat, že $R^T = L$ a $D > 0$.

Pro A SPD tedy můžeme provést LU faktorizaci symetricky. Když získáme $A^{n-1} = U$, pak automaticky $L = (diag(U)^{-1}U)^T$.

Klasicky se místo toho ale volí $L = diag(U)^{-1/2}U$ a dostáváme tzv. Choleského faktorizaci: $A = LL^T$.

Pro výpočet Choleského faktorizace umíme ukázat

$$\frac{max_{i,j}|(\tilde{L}\tilde{L}^T)_{i,j} - a_{i,j}|}{max_{i,j}|a_{i,j}|} \leq 2n\varepsilon_{mach} + \mathcal{O}(\varepsilon_{mach}^2)$$

$\implies$ pro A SPD je Gaussova eliminace stabilní.

Poznámka. Růstový faktor zde lze odhadnout jako 2 - zaokrouhlovací chyby se neakumulují.

7.2.5 Podmíněnost problému soustavy lineárních rovnic

Úmluva: $x, \tilde{x}, b, \tilde{b}$ jsou v této podkapitole vektory $\in \mathbb{R}^n$

Uvažujme malou perturbaci systému $Ax = b : \tilde{A}x = \tilde{b}$, kde

$$\frac{||\tilde{A} - A||}{||A||} \leq \varepsilon_A; \frac{||\tilde{b} - b||}{||b||} \leq \varepsilon_b$$

nám značí relativní velikost perturbací ($\ll 1$).

Pak máme

$$\tilde{x} - x = A^{-1}(A\tilde{x} - Ax) = A^{-1}(A\tilde{x} - \tilde{A}\tilde{x} + \tilde{A}\tilde{x} - Ax) = A^{-1}[(A - \tilde{A})\tilde{x} + \tilde{b} - b]$$

a odsud dostáváme

$$||\tilde{x} - x|| \leq ||A^{-1}|| \cdot (\varepsilon_A \cdot ||A|| \cdot ||\tilde{x}|| + \varepsilon_b \cdot ||b||) \leq ||A^{-1}|| \cdot (\varepsilon_A \cdot ||A|| \cdot ||\tilde{x}|| + \varepsilon_b \cdot ||A|| \cdot ||x||) \leq$$

$$\leq ||A^{-1}|| \cdot ||A||(\varepsilon_A \cdot ||\tilde{x} - x + x|| + \varepsilon_b||x||) \leq ||A^{-1}|| \cdot ||A||[\varepsilon_A \cdot ||\tilde{x} - x|| + (\varepsilon_A + \varepsilon_b)||x||],$$

takže celkem

$$||\tilde{x} - x||(1 - ||A^{-1}|| \cdot ||A|| \cdot \varepsilon_A) \leq ||A^{-1}|| \cdot ||A||(\varepsilon_A + \varepsilon_b)||x||.$$

Dohromady platí, že

$$\underbrace{\frac{||\tilde{x} - x||}{||x||}}_{\substack{velikost\ změny\ řeš.}} \leq \underbrace{\frac{||A^{-1}|| \cdot ||A||}{1 - ||A^{-1}|| \cdot ||A|| \cdot \varepsilon_A}}_{podmíněnost\ problému} \cdot \underbrace{(\varepsilon_A + \varepsilon_b)}_{velikost\ perturbace}$$

Číslo $||A^{-1}|| \cdot ||A||$ je číslo podmíněnosti matice A (klasické značení je $\kappa(A) := ||A^{-1}|| \cdot ||A||$).

Vidíme, že pro obecnou perturbaci dat o velikosti ε lze matematicky očekávat přesnost řešení přinejlepším řádu $\mathcal{O}(\kappa(A) \cdot \varepsilon_{mach})$ - i pro zcela stabilní algoritmy máme chyby $\mathcal{O}(\kappa(A) \cdot \varepsilon_{mach})$.

7.2.6 Efektivita

Viděli jsme, že pro některé aplikace máme A tzv. řídkou (sparse). To znamená, že většina prvků se rovná 0 a ty nenulové prvky jsou "řídké", např. aproximace druhé derivace na matici, kde z n^2 prvků bylo pouze $3n-2$ nenulových.

Podobně je tomu pro většinu problémů, které v nějakém smyslu aproximují nebo pracují s derivacemi.

Abychom měli efektivní výpočty, stačí nám pracovat pouze s těmi nenulovými prvky - tzv. sparse matrix algorithms. Jejich odvozování a implementace je technická a pokročilá, avšak jejich používání je snadné.

Nestabilita nám zatím vznikala tak, že jsme disproporčně dělili velkým nebo malým číslem. V řeči matic matice M^k velmi výrazně měnily normu sloupců a řádků matice A^{k-1}. Chtěli bychom tedy navrhnout algoritmy/metody, které převedou A na horní trojúhelníkovou matici, ale za použití unitárních maticových transformací. (tj. tak abychom jejich aplikací neměnili normu - nebo alespoň ne tolik).

7.3 QR faktorizace

Z kurzu lineární algebry víme, že

$$\forall A \in \mathbb{R}^{n\times n}\ \exists Q \text{ (unitární) } \exists R \text{ (horní trojúhelníková)} : A = QR.$$

Navíc, Q,R je možné spočítat Gramovou-Schmidtovou ortogonalizací sloupců matice A. Pak

$$A\vec{x} = \vec{b} \iff QR\vec{x} = \vec{b} \iff R\vec{x} = Q^T\vec{b} \quad \& \quad Q \text{ je unitární.}$$

Umíme tedy přesně to, co jsme potřebovali. Zbývá ujasnit dvě otázky: Umíme spočítat/aplikovat Q,R stabilně?, Lze Q,R rovnou aplikovat? (Ve stejném smyslu jako "GE je přímá aplikace LU").

7.3.1 Gramova-Schmidtova ortogonalizace

***KROK* 1**

$$\vec{q}_1 = \frac{\vec{A}_{:,1}}{||\vec{A}_{:,1}||_2}$$

$$r_{1,1} := ||\vec{A}_{:,1}||_2$$

***KROK* 2**

$$\vec{v}_2 = \vec{A}_{:,2} - \langle\vec{q}_1,\vec{A}_{:,2}\rangle \cdot \vec{q}_1 = (I - \vec{q}_1\vec{q}_1^T)\cdot\vec{A}_{:,2}$$

$$\vec{q}_2 := \frac{\vec{v}_2}{||\vec{v}_2||_2}$$

$$r_{1,2} = \langle\vec{q}_1, \vec{A}_{:,2}\rangle \quad \& \quad r_{2,2} = ||\vec{v}_2||_2$$

Z toho tedy plyne

$$[\vec{A}_{:,1},\vec{A}_{:,2}] = [\vec{q}_1,\vec{q}_2]\cdot\begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2}\end{bmatrix}.$$

***KROK* 3**

$$\vec{v}_3 = \vec{A}_{:,3} - \langle\vec{q}_1,\vec{A}_{:,3}\rangle\cdot\vec{q}_1 - \langle\vec{q}_2,\vec{A}_{:,3}\rangle\cdot\vec{q}_2 = (I - \vec{q}_1\vec{q}_1^T - \vec{q}_2\vec{q}_2^T)\cdot\vec{A}_{:,3} \overset{\vec{q}_1^T\vec{q}_2=0}{\underset{*}{=}} (I-\vec{q}_1\vec{q}_1^T)(I-\vec{q}_2\vec{q}_2^T)\cdot\vec{A}_{:,3}$$

$$\vec{q}_3 = \frac{\vec{v}_3}{||\vec{v}_3||_2}$$

$$r_{1,3} = \langle \vec{q}_1, \vec{A}_{:,3} \rangle \quad \& \quad r_{2,3} = \langle \vec{q}_2, \vec{A}_{:,3} \rangle \quad \& \quad r_{3,3} = ||\vec{v}_3||_2$$

A opět z toho získáme

$$[\vec{A}_{:,1}, \vec{A}_{:,2}, \vec{A}_{:,3}] = [\vec{q}_1, \vec{q}_2, \vec{q}_3] \cdot \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ 0 & r_{2,2} & r_{2,3} \\ 0 & 0 & r_{3,3} \end{bmatrix}.$$

Poznámka. Stejně jako v rovnosti * platí, že

$$r_{2,3} = \langle \vec{q}_2, \vec{A}_{:,3} \rangle = \langle \vec{q}_2, (I - \vec{q}_1 \vec{q}_1^T) \vec{A}_{:,3} \rangle,$$

obdobně pak i pro $r_{2,4}, r_{3,4}, r_{2,5}, \ldots, r_{4,5}, \ldots$.

KROK n

$$v_n := (I - \sum_{i=1}^{n-1} \vec{q}_i \vec{q}_i^T) \vec{A}_{:,n} = (\prod_{i=1}^{n-1} I - \vec{q}_i \vec{q}_i^T) \vec{A}_{:,n}$$

$$\vec{q}_n = \frac{\vec{v}_n}{||\vec{v}_n||_2}$$

$$r_{i,n} = \langle \vec{q}_i, \vec{A}_{:,i} \rangle \quad \& \quad r_{n,n} = ||\vec{v}_n||_2$$

a získáme

$$\underbrace{[\vec{A}_{:,1}, \ldots, \vec{A}_{:,n}]}_{A} = \underbrace{[\vec{q}_1, \ldots, \vec{q}_n]}_{Q} \cdot \underbrace{\begin{bmatrix} r_{1,1} & & \cdots & & r_{1,n} \\ 0 & \ddots & & & r_{2,n} \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & & \ddots & \ddots & r_{n-1,n} \\ 0 & & \cdots & 0 & r_{n,n} \end{bmatrix}}_{R}.$$

Přirozeně vidíme tedy dva způsoby výpočtu:

$$r_{i,j} = \langle \vec{A}_{:,j}, \vec{q}_i \rangle \text{ - tzv. klasický a}$$

$$r_{i,j} = \langle \vec{q}_i, (I - \vec{q}_{i-1} \vec{q}_{i-1}^T) \ldots (I - \vec{q}_1 \vec{q}_1^T) \vec{A}_{:,j} \rangle \text{ - tzv. modifikovaný.}$$

Na papíře jsou ekvivalentní, ale naprogramované už nikoliv.

7.3.2 Stabilita výpočtu

Mějme A jako vstupní data a $\tilde{Q}, \tilde{R}$ produkty algoritmů výše, pak

$$||A - \tilde{Q}\tilde{R}|| \leq c \cdot ||A|| \cdot \varepsilon_{mach}$$

pro oba způsoby.

Uvažujeme-li však už $||I - \tilde{Q}\tilde{Q}^T||$, platí, že pro klasický způsob

$$||I - \tilde{Q}\tilde{Q}^T|| \leq c \cdot \kappa(A)^2 \cdot \varepsilon_{mach}$$

a pro modifikovaný

$$||I - \tilde{Q}\tilde{Q}^T|| \leq c \cdot \kappa(A) \cdot \varepsilon_{mach}.$$

Kdybychom počítali přesně, pak by chyba byla 0, tedy $\tilde{Q}$ by měla mít ortonormální sloupce.

Toto byla motivace, proč QR rozklad dělat. Platí to i pro PC?

Obě verze Gram-Schmidta spočítají rozklad "stabilně" ve smyslu relativního rezidua, tedy

$$\frac{||A - \tilde{Q}\tilde{R}||}{||A||} \leq c \cdot \varepsilon_{mach}.$$

Obě verze ale trpí numerickou ztrátou ortogonality (pro špatně podmíněné matice). Kvůli tomu se obecně považují tyto algoritmy za nestabilní.

Navíc matice $(I - \vec{q}_k\vec{q}_k^T)\dots(I - \vec{q}_1\vec{q}_1^T)$, které aplikujeme na sloupce A nejsou unitární a tedy zde potenciálně vzniká stejný problém jako u GE/LU. Ideálně chceme QR rozklad, který bude stabilní a používá unitární transformace. Existují i tedy lepší způsoby, jak k problému přistoupit.

Příklad 7.5. *Začněme v $\mathbb{R}^{2\times 2}$ s obecnou maticí A (s prvky $a_{i,j}$). Chceme najít unitární matici $U \in \mathbb{R}^{2\times 2}$ takovou, že*

$$UA = \begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix}$$

pro nějaké $r_{i,j} \in \mathbb{R}$. Tímto v podstatě už máme QR rozklad matice A:

$$Q = U^T; R = \begin{bmatrix} r_{1,1} & r_{1,2} \\ 0 & r_{2,2} \end{bmatrix}.$$

Otázka, která nám tedy vyplývá je, jak "vynulovat" vektor pod prvním složkou a nezměnit jeho normu. Zobrazení, která mají takovou vlastnost, jsou rotace a reflexe.

ROTACE
Budeme uvažovat rotaci o úhel α. Chceme, aby platilo

$$U \cdot \vec{A}_{:,1} = ||\vec{A}_{:,1}|| \cdot \vec{e}_1.$$

Matice rotace tedy bude vypadat

$$U = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}.$$

Hledejme tedy

$$U = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}.$$

Pak tedy dostaneme

$$U \cdot \vec{A}_{:,1} = \begin{bmatrix} c \cdot a_{1,1} - s \cdot a_{2,1} \\ s \cdot a_{1,1} + c \cdot a_{2,1} \end{bmatrix}.$$

Jelikož chceme druhou složku vektoru vynulovat a první složku mít stejné velikosti jako normu původního vektoru ($\sqrt{a_{1,1}^2 + a_{2,1}^2}$), vypočtením soustavy rovnic dostaneme

$$c = \frac{a_{1,1}}{\sqrt{a_{1,1}^2 + a_{2,1}^2}}; s = \frac{-a_{1,2}}{\sqrt{a_{1,1}^2 + a_{2,1}^2}}.$$

REFLEXE
Uvažujme reflexi podél přímky s normálovým vektorem q. Chceme, aby platilo

$$U \cdot \vec{A}_{:,1} = ||\vec{A}_{:,1}|| \cdot \vec{e}_1$$

$$\vec{e}_1 \cdot ||\vec{A}_{:,1}|| = \vec{A}_{:,1} - 2\vec{v},$$

kde v je projekce $\vec{A}_{:,1}$ *na směr q*

$$\vec{q} \cdot \langle \vec{q}, \vec{A}_{:,1} \rangle = \vec{q} \cdot \vec{q}^T \cdot \vec{A}_{:,1}.$$

Z toho tedy plyne, že

$$U \cdot \vec{A}_{:,1} = \vec{A}_{:,1} - 2\vec{q}\vec{q}^T \vec{A}_{:,1} \implies U = I - 2\vec{q}\vec{q}^T$$

Jak tedy vypadá q?

$$\vec{q} = \frac{\vec{A}_{:,1} \pm \vec{e}_1 ||\vec{A}_{:,1}||}{||\vec{A}_{:,1} \pm \vec{e}_1 ||\vec{A}_{:,1}||\,||} \rightarrow \vec{q} = \frac{\vec{A}_{:,1} + sgn(a_{1,1}) \cdot \vec{e}_1 ||\vec{A}_{:,1}||}{||\vec{A}_{:,1} + sgn(a_{1,1}) \cdot \vec{e}_1 ||\vec{A}_{:,1}||\,||}$$

Poznámka. Vždycky můžeme dělat reflexi podle dvou různých přímek (které jsou na sebe kolmé). Jak jsme tedy vybrali znaménko? Jde nám o stabilitu - nechceme tedy dělit něčím, co je ≈ 0. Jde nám tedy o co největší normu ve jmenovateli zlomku.

▲

7.3.3 Givensovy rotace

Givensovy rotace jsou metodou, jak rotace v předchozím příkladu rozšířit na $\mathbb{R}^{n \times n}$.
Chceme $A = QR$, kde Q je unitární a R horní trojúhelníková.

***KROK* 1**
Převedeme unitární transformací $\vec{A}_{:,1}$ na $\vec{e}_1 ||\vec{A}_{:,1}||$. Budeme to dělat souřadnici po souřadnici.

$$\underbrace{\begin{bmatrix} c_1 & -s_1 & 0 & \cdots & 0 \\ s_1 & c_1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{G_1^1; c_1, s_1 \, jako \, a_{1,1}, a_{2,1} \, v \, příkladu} \cdot A_{:,1} = \begin{bmatrix} \alpha_1 \\ 0 \\ a_{3,1} \\ \vdots \\ a_{n,1} \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} c_2 & 0 & -s_2 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ s_2 & 0 & c_2 & 0 & & 0 \\ 0 & 0 & 0 & 1 & & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{G_1^2; c_2, s_2 \, jako \, \alpha_1, a_{3,1} \, v \, příkladu} \cdot \begin{bmatrix} \alpha_1 \\ 0 \\ a_{3,1} \\ \vdots \\ \vdots \\ a_{n,1} \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ 0 \\ 0 \\ a_{4,1} \\ \vdots \\ a_{n,1} \end{bmatrix}$$

$$\vdots$$

$$\underbrace{\begin{bmatrix} c_{n-1} & 0 & \dots & 0 & -s_{n-1} \\ 0 & 1 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & 1 & 0 \\ s_{n-1} & 0 & \dots & 0 & c_{n-1} \end{bmatrix}}_{G_1^{n-1}} \cdot \begin{bmatrix} \alpha_{n-2} \\ 0 \\ \vdots \\ 0 \\ a_{n,1} \end{bmatrix} = \begin{bmatrix} \alpha_{n-1} \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

Celkem tedy

$$\underbrace{G_1^{n-1} \cdot \ldots \cdot G_1^1}_{G_1} \cdot A = B; A = G_1^T B,$$

kde

$$B = \begin{bmatrix} \alpha_{n-1} & \beta_1 & \ldots & \beta_{n-1} \\ 0 & & & \\ \vdots & & \tilde{A} & \\ 0 & & & \end{bmatrix}.$$

KROK 2

Použijeme krok 1 na matici $\tilde{A}$ (neměníme tedy 1. sloupec a 1. řádek) a dostaneme matice $G_2^{n-2} \cdot \ldots \cdot G_2^1$ takové, že

$$\underbrace{G_2^{n-2} \cdot \ldots \cdot G_2^1}_{G_2} \cdot \tilde{A} = \begin{bmatrix} \tilde{\alpha}_{n-2} & \tilde{\beta}_1 & \ldots & \tilde{\beta}_{n-2} \\ 0 & & & \\ \vdots & & \tilde{A}^2 & \\ 0 & & & \end{bmatrix}.$$

Pak tedy platí, že

$$\begin{bmatrix} 1 & 0 \\ 0 & G_2 \end{bmatrix} \cdot G_1 \cdot A = \begin{bmatrix} \alpha_{n-1} & \beta_1 & \ldots & & \beta_{n-1} \\ 0 & \tilde{\alpha}_{n-2} & \tilde{\beta}_1 & \ldots & \tilde{\beta}_{n-2} \\ \vdots & 0 & & & \\ & \vdots & & \tilde{A}^2 & \\ 0 & 0 & & & \end{bmatrix}.$$

KROK 3

Použijeme krok 1 na $\tilde{A}^2$ a dostaneme

$$G_3 \cdot \tilde{A}^2 = \begin{bmatrix} \tilde{\alpha}_{n-3}^2 & \tilde{\beta}_1^2 & \ldots & \tilde{\beta}_{n-3}^2 \\ 0 & & & \\ \vdots & & \tilde{A}^3 & \\ 0 & & & \end{bmatrix}$$

$$\begin{bmatrix} I_2 & 0 \\ 0 & G_3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & G_2 \end{bmatrix} \cdot G_1 \cdot A = \ldots.$$

KROK n − 1

Použijeme krok 1 na $\tilde{A}^{n-2}$ a dostaneme

$$G_{n-1} A^{n-2} = \begin{bmatrix} \tilde{\alpha}_1^{n-2} & \tilde{\beta}_1^{n-2} \\ 0 & \gamma \end{bmatrix}.$$

Pak po n-1 krocích tedy dostaneme

$$\underbrace{\begin{bmatrix} I_{n-2} & 0 \\ 0 & G_{n-1} \end{bmatrix} \cdot \ldots \cdot \begin{bmatrix} 1 & 0 \\ 0 & G_2 \end{bmatrix} \cdot G_1}_{U} \cdot A = R(\text{horní trojúhelníková matice}),$$

U je unitární matice, jelikož je to součin unitárních matic (každá jednotlivá matice je pouze maticí rotace - tedy unitární). Položme nyní $Q = U^T$. Q je nyní také unitární matice. Celkem tedy máme

$$A = QR.$$

Tento algoritmus je stabilní, protože platí

$$||A - \tilde{Q}\tilde{R}|| \leq c \cdot \varepsilon_{mach} \cdot ||A||; ||I - \tilde{Q}^T\tilde{Q}|| \leq c \cdot \varepsilon_{mach}.$$

Pokud $a_{i,j} \neq 0$, cena výpočtu je $2n^3 + \mathcal{O}(n^2)$. Pokud je ale matice A řídká (nebo alespoň její část pod diagonálou), nemusí algoritmus dělat všechny operace a může počítat jen pro nenulové prvky.

7.3.4 Householderovy reflexe

Householderovy reflexe jsou metodou, jak reflexe v předchozím příkladu rozšířit na $\mathbb{R}^{n\times n}$.
Chceme $A = QR$, kde Q je unitární a R horní trojúhelníková.

***KROK* 1**
Převedeme unitární transformací $\vec{A}_{:,1}$ na $\vec{e}_1||\vec{A}_{:,1}||$.
Položíme

$$\vec{q}_1 = \frac{\vec{A}_{:,1} + sgn(a_{1,1}) \cdot \vec{e}_1||\vec{A}_{:,1}||}{||\vec{A}_{:,1} + sgn(a_{1,1}) \cdot \vec{e}_1||\vec{A}_{:,1}||\,||}; H_1 := I - 2\vec{q}_1\vec{q}_1^T$$

Pak

$$H_1 A = \begin{bmatrix} \phi_{n-1} & \chi_1 & \cdots & \chi_{n-1} \\ 0 & & & \\ \vdots & & \tilde{A} & \\ 0 & & & \end{bmatrix}$$

***KROK* 2**
Použijeme krok 1 na $\tilde{A}$:

$$\vec{q}_2 = \frac{\tilde{A}_{:,1} + sgn(\tilde{a}_{1,1}) \cdot \vec{e}_1||\tilde{A}_{:,1}||}{||\tilde{A}_{:,1} + sgn(\tilde{a}_{1,1}) \cdot \vec{e}_1||\tilde{A}_{:,1}||\,||}; H_2 := I - 2\vec{q}_2\vec{q}_2^T.$$

Pak

$$H_2\tilde{A} = \begin{bmatrix} \tilde{\phi}_{n-2} & \tilde{\chi}_1 & \cdots & \tilde{\chi}_{n-2} \\ 0 & & & \\ \vdots & & \tilde{A}^2 & \\ 0 & & & \end{bmatrix}.$$

a

$$\begin{bmatrix} 1 & 0 \\ 0 & H_2 \end{bmatrix} \cdot H_1 \cdot A = \begin{bmatrix} \phi_{n-1} & \chi_1 & \cdots & & \chi_{n-1} \\ 0 & \tilde{\phi}_{n-2} & \tilde{\chi}_1 & \cdots & \tilde{\chi}_{n-2} \\ \vdots & 0 & & & \\ & \vdots & & \tilde{A}^2 & \\ 0 & 0 & & & \end{bmatrix}.$$

***KROK* $n-1$**

Použijeme krok 1 na $\tilde{A}^{n-2}$ a dostaneme

$$H_{n-1}\tilde{A}^{n-2} = \begin{bmatrix} \tilde{\phi}_1^{n-2} & \tilde{\chi}_1^{n-2} \\ 0 & \psi \end{bmatrix}.$$

Pak po n-1 krocích tedy dostaneme

$$\underbrace{\begin{bmatrix} I_{n-2} & 0 \\ 0 & H_{n-1} \end{bmatrix} \cdot \ldots \cdot \begin{bmatrix} 1 & 0 \\ 0 & H_2 \end{bmatrix} \cdot H_1}_{U} \cdot A = R(\text{horní trojúhelníková matice}),$$

U je unitární matice, jelikož je to součin unitárních matic (každá jednotlivá matice je pouze maticí reflexe - tedy unitární). Položme nyní $Q = U^T$. Q je nyní také unitární matice. Celkem tedy máme

$$A = QR.$$

Tento algoritmus je stabilní, protože platí

$$||A - \tilde{Q}\tilde{R}|| \leq c \cdot \varepsilon_{mach} \cdot ||A||; ||I - \tilde{Q}^T\tilde{Q}|| \leq c \cdot \varepsilon_{mach}.$$

Pokud $a_{i,j} \neq 0$, cena výpočtu je $\frac{2n^2}{3} + \mathcal{O}(n^2)$.

7.3.5 Aplikace unitárních QR rozkladů

Prakticky máme dvě možnosti, jak se s problémem vypořádat.

Buď samostatně spočítáme $A = QR$ a poté řešíme $Rx = Q^T\vec{b}$,

nebo při výpočtu $A = QR$ aplikujeme matice G_i nebo H_i zároveň i na příslušnou část pravé strany $\vec{b}$. To v podstatě odpovídá aplikaci G_i nebo H_i na $[A|\vec{b}]$ z čehož dostaneme $[R|\vec{c}]$ a poté řešíme soustavu $R\vec{x} = \vec{c}$.

Je to tedy analogické výpočtu LU faktorizace vs. Gaussově eliminaci coby aplikaci LU.

7.4 Iterační metody

Rádi bychom měli stabilnější algoritmy a pokud možno i iterační algoritmy.

Iterační algoritmy jsou takové algoritmy, které produkují sekvenci aproximací x_1,x_2,... přesného řešení x. Například abychom mohli kontrolovat požadovanou přesnost řešení a/nebo její trade-off s již uběhlým časem výpočtu.

Je dobré mít oba nástroje - iterační i přímé.

Zatím jsme v principu viděli pouze 1 způsob, jak vytvořit iterační metodu - reformulace na hledání pevného bodu + Banachova věta o kontrakci.

Idea této metody je následující - na začátku "rozštěpíme" matici A, například $A = D + N$, kde $D = diag(A), N = A - diag(A)$ (tzv. štěpení matice). Máme

$$A\vec{x} = \vec{b},$$

$$(D + N)\vec{x} = \vec{b},$$

$$D\vec{x} = \vec{b} - N\vec{x},$$

$$\vec{x} = D^{-1}(\vec{b} - N\vec{x}),$$

kde $\vec{x}$ tedy musí být pevný bod funkce $F : v \to D^{-1}(b - Nx)$ a tedy platí

$$\vec{x}_{k+1} = D^{-1}(\vec{b} - N\vec{x}_k) = D^{-1}(\vec{b} - A\vec{x}_k + D\vec{x}_k) = \vec{x}_k + D^{-1}(\vec{b} - A\vec{x}_k).$$

Vezměme si x^* jako přesné řešení - platí tedy $A\vec{x}^* = \vec{b}$ (jde o rovnost, ne rovnici).

$$\vec{x}_{k+1} - \vec{x}^* = \vec{x}_k - \vec{x}^* + D^{-1}(\vec{b} - A\vec{x}_k) = \vec{x}_k - \vec{x}^* + D^{-1}(A\vec{x}^* - A\vec{x}_k) = (I - D^{-1}A)(\vec{x}_k - \vec{x}^*) =$$

$$= (I - D^{-1}A)^2(\vec{x}_{k-1} - \vec{x}^*) = \cdots = (I - D^{-1}A)^{k+1}(\vec{x}_0 - \vec{x}^*)$$

Vidíme zde tedy stejný postup jako u Banachovy věty o kontrakci. Naše metoda tedy konverguje

$$\forall \vec{x}_0 \in \mathbb{R}^n \iff (I - D^{-1}A)^k \overset{k\to\infty}{\longrightarrow} 0$$

Mějme $T = I - D^{-1}A$ diagonalizovatelnou. Pro T tedy platí $T = S\Lambda S$, kde S je matice vlastních vektorů T a $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ (matice vlastních čísel). Pak

$$T^k = S\Lambda S^{-1} S\Lambda S^{-1} \ldots S\Lambda S^{-1} = S\Lambda^k S^{-1}$$

a tedy

$$T^k \overset{k\to\infty}{\longrightarrow} 0 \iff \underbrace{\max_{i=1,\ldots,n} |\lambda_i|}_{=:\rho(T)} < 1,$$

kde $\rho(T)$ je tzv. spektrální poloměr A.

Pro k blížící se do nekonečna tedy konverguje

$$\forall \vec{x}_0 \in \mathbb{R}^n \iff \rho(I - D^{-1}A) < 1.$$

Poznámka. Tento výpočet funguje i pro matici v Jordanově tvaru, výpočet ale není tak názorný.

7.4.1 Štěpící metody

V této části se budeme bavit o obecném štěpení matice A, tedy $A = M - (M - A) = M - N$.

Pro libovolnou regulární matici M správných rozměrů lze napsat iterační metodu $\vec{x}_{k+1} = \vec{x}_k + M^{-1} \underbrace{(\vec{b} - A\vec{x}_k)}_{=:\vec{r}_k \text{(reziduum)}}$

- Richardsonova metoda: $M = I$
- Jacobiho metoda: $M = diag(A)$
- Gaussova-Seidlova metoda: $M = upper\text{-}triangle(A)$

Pro Richardsonovu metodu platí

$$A\vec{x} = \vec{b},$$
$$(M - N)\vec{x} = \vec{b},$$
$$M\vec{x} = \vec{b} + N\vec{x},$$
$$\vec{x} = M^{-1}(\vec{b} + N\vec{x}).$$

Stejně jak dříve, pro přesné řešení $\vec{x}^*$ platí $A\vec{x}^* = \vec{b}$. Pak

$$\vec{x}^* - \vec{x}_{k+1} = \vec{x}^* - M^{-1}(\vec{b} + N\vec{x}_k) = \vec{x}^* - M^{-1}(A\vec{x}^* + N\vec{x}_k) = \vec{x}^* - M^{-1}((M - N)\vec{x}^* + N\vec{x}_k) =$$

$$= \vec{x}^* - (M^{-1}M\vec{x}^* - M^{-1}N\vec{x}^* + M^{-1}N\vec{x}_k) = M^{-1}N\vec{x}^* - M^{-1}N\vec{x}_k =$$

$$= M^{-1}N(\vec{x}^* - \vec{x}_k) = (M^{-1}N)^2(\vec{x}^* - \vec{x}_{k-1}) = \cdots = (M^{-1}N)^{k+1}(\vec{x}^* - \vec{x}_0).$$

Obecně máme vztah

$$\vec{x}_{k+1} = M^{-1}(b + N\vec{x}_k).$$

Jelikož platí $M = I$, pak máme

$$\vec{x}_{k+1} = I(b + (I - A)\vec{x}_k) = b - A\vec{x}_k + \vec{x}_k.$$

Dále platí

$$\begin{aligned}
\vec{x}_{k+1} &= \vec{x}_k + \vec{b} - A\vec{x}_k, \\
\vec{x}^* - \vec{x}_{k+1} &= \vec{x}^* - \vec{x}_k - A\vec{x}^* + A\vec{x}_k, \\
\vec{x}^* - \vec{x}_{k+1} &= \vec{x}^* - \vec{x}_k - A(\vec{x}^* - \vec{x}_k), \\
\vec{x}^* - \vec{x}_{k+1} &= (I - A)(\vec{x}^* - \vec{x}_k) \\
A(\vec{x}^* - \vec{x}_{k+1}) &= (I - A)A(\vec{x}^* - \vec{x}_k) \\
\underbrace{b - A\vec{x}_{k+1}}_{=\vec{r}_{k+1}} &= (I - A)^{k+1} \underbrace{(b - A\vec{x}_0)}_{=\vec{r}_0}.
\end{aligned}$$

Člen $(I - A)^{k+1}$ můžeme rozepsat jako

$$(I - A)^{k+1} = \alpha_0 I + \alpha_1 A + \cdots + \alpha_k A^k + \alpha_{k+1} A^{k+1} =: p^{Rich}(A),$$

což je polynom v A, kde platí

$$p^{Rich}(t) = \sum_{i=0}^{k+1} \alpha_i t^i = (1 - t)^{k+1}$$

(místo skaláru zde ovšem mocníme matici). Celkem tedy dostáváme, že

$$\vec{r}_k = p^{Rich}(A) \cdot \vec{r}_0.$$

V praxi používáme $\vec{r}_k$ jako ukazatel konvergence. Pro $\vec{x}_k = \vec{x}^*$ máme $\vec{r}_k = \vec{0}$ a tedy $\vec{r}_k \approx \vec{0}$ často indikuje $\vec{x}_k \approx \vec{x}^*$.

Poznámka. Jelikož $\vec{r}_k = A(\vec{x}^* - \vec{x}_k)$, tak už víme (podle podkapitoly 7.2.5), že malé reziduum neznamená vždy přesnou aproximaci - záleží i na podmíněnosti matice A.

7.4.2 GMRES

Jedna možnost, jak zlepšit Richardsonovu metodu, je najít polynom $p_k^{opt}(t)$ takový, že

$$||p_k^{opt}(t)\vec{r}_0|| = \min_{p(t) \text{ st.} \leq k} ||p(A)\vec{r}_0||. \tag{7.5}$$

Jedná se o tzv. Generalized minimal residual method (GMRES).
Porovnání s Richardsonem:

- Není jasné, jak z (7.5) dostat předpis pro $\vec{x}_k$.
- Richarson používá v každé iteraci stejnou formulku, GMRES každou iteraci konstruuje polynom.
- Richardson velmi často nekonverguje.
- Podle Caleyho-Hamiltonovy věty($\forall A \in \mathbb{R}^{n\times n}\ \exists q(t)$ - polynom stupně n takový, že $A^{-1} = q(A)$) na papíře dostaneme $p_n^{opt}(t) = q(t)$ - na papíře GMRES zkonverguje v n krocích.

Metody typu GMRES (založené na optimálních polynomech) se nazývají Krylovovské metody a jsou prakticky nejpoužívanější.

8 Problém nejmenších čtverců

8.1 Maximum likelihood estimate (MLE)

Předpokládejme, že máme model předpovídající jistý fenomén nebo fyzikální jev/děj. Pak model, v závislosti na vstupních datech a kalibraci, vrací aproximaci/predikci výsledkového stavu:

$$\vec{x}, \vec{\theta} \longrightarrow F(\vec{x},\vec{\theta}) \approx \vec{y},$$

kde $\vec{x}$ představuje vstupní data, $\vec{\theta}$ parametry modelu, $F(\vec{x},\vec{\theta})$ predikci modelu a $\vec{y}$ skutečné řešení.

Poznámka. Výběr modelu je problém pro odborníky dané oblasti.

Volba parametrů (kalibrace modelu) na základě dat je problém matematiky/data-science.

Například, zajímá-li nás pohyb planet, je modelem pohyb po elipse (Keplerův zákon). Otázkou je po které a problém spočívá v nalezení správných parametrů $\vec{\theta}$.

Volba modelu (tj. funkce $F(\vec{x},\vec{\theta})$) je u každého problému jiná, ale volbu $\vec{\theta}$ lze analyzovat relativně obecně.

Předpokládejme, že máme „dokonalý model“, tedy že existuje $\vec{\theta}$ taková, že pro libovolné přesné $\vec{x}$ existuje přesné $\vec{y}$ takové, že $\vec{y} = F(\vec{x},\vec{\theta})$. I za tohoto předpokladu budou ale naše pozorování, měření nebo data nepřesná. Pokud jsme pro inputy $x_1, x_2, \ldots, x_m$ naměřili outputy $z_1,z_2,\ldots,z_m$, pak bohužel $\delta z_i \equiv z_i - F(x_i,\vec{\theta})$.

Standardně modelujeme měřená data jako náhodnou veličinu z normálního rozdělení vycentrovaného okolo y_i (tj. okolo „přesného měření“) $\Rightarrow Z_i \sim \mathcal{N}(y_i,\sigma_i^2)$, kde rozptyl σ_i je často odhadnutelný podle typu/kvality měření. Z předpokladu $y_i \equiv F(x_i,\vec{\theta})$ víme, že y_i existují, ale zjevně je neznáme. Tudíž i chyba $\delta Z_i := Z_i - F(x_i,\vec{\theta})$ je náhodná veličina, zjevně $\delta Z_i \sim \mathcal{N}(0,\sigma_i^2)$, jejíž realizace jsou hodnoty $\delta z_i,\ldots,\delta z_m$

Máme-li $(x_1,z_1),\ldots,(x_m,z_m)$, kde $Z_i \sim \mathcal{N}(y_i,\sigma_i^2)$ je stejné jako výše, jaké musely být parametry $\vec{\theta}$ modelu $F(\cdot,\cdot)$?

Musely být takové, pro které $\vec{\theta}$ je největší šance, že jsme pak pozorovali $(x_1,z_1),\ldots,(x_m,z_m)$, tzv. maximum likelihood estimate parametrů $\vec{\theta}$.

$$\mathbb{P}(z_i-\varepsilon \le Z_i < z_i+\varepsilon) = \int_{z_i-\varepsilon}^{z_i+\varepsilon} \frac{1}{\sigma_i\cdot\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{\zeta - F(x_i,\vec{\theta})}{\sigma_i})^2}\, d\zeta \approx 2\varepsilon \cdot \frac{1}{\sigma_i\cdot\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{z_i - F(x_i,\vec{\theta})}{\sigma_i})^2}$$

Předpokládáme-li, že Z_i jsou *iid* (independent & identically distributed), pak

$$\mathbb{P}(z_i-\varepsilon \le Z_i < z_i+\varepsilon, \forall i) = \prod_{i=1}^{m} \mathbb{P}(z_i-\varepsilon \le Z_i < z_i+\varepsilon)$$

a tedy

$$\mathbb{P}(z_i-\varepsilon \le Z_i < z_i+\varepsilon, \forall i) = \left(\frac{2\varepsilon}{\sqrt{2\pi}}\right)^m \cdot \prod_{i=1}^{m} \frac{1}{\sigma_i} \cdot e^{-\frac{1}{2}\cdot\sum_{i=1}^{m}[\frac{z_i}{\sigma_i} - \frac{1}{\sigma_i}F(x_i,\theta)]^2}.$$

Platí tedy, že čím menší

$$\sum_{i=1}^{m}\left[\frac{z_i}{\sigma_i} - \frac{1}{\sigma_i}F(x_i,\vec{\theta})\right]^2,$$

tím větší pravděpodobnost, že pro tyto parametry $\vec{\theta}$ jsme mohli získat tato měření.

To znamená, že ten MLE odhad dopadne tak, že $\vec{\theta}$ volíme jako

$$\arg\min_i \|[\frac{1}{\sigma_i}F(x_i,\vec{\theta}) - \frac{z_i}{\sigma_i}]_{i=1,\ldots,m}\|^2 \implies \min_{\vec{\theta}} \|[\frac{1}{\sigma_i}F(x_i,\theta) - b_i]_{i=1,\ldots,m}\|^2,$$

příčemž implikace platí, neboť $b_i = \frac{z_i}{\sigma_i}$.

Problém, který jsme dostali, se nazývá problém nejmenších čtverců.

Ačkoliv $Z_i \sim \mathcal{N}(y_i,\sigma_i^2)$ není vždy oprávněný předpoklad, na základě CLT (central limit theorem) lze analogicky postupovat i v případě, kdy δz_i jsou „pouze“ iid (independent & identically distributed).

Problém nejmenších čtverců (least squares problem - LS) může být lineární ($f(x_i,\vec{\theta}) = a_i^T \cdot \vec{\theta} + c_i$ pro nějaké a_i,c_i) nebo nelineární (nejsou-li lineární).

Obecné řešení vypadá tak, že najdeme „kandidáty“, tj. $\vec{\theta}_l, l = 1,2,\ldots,$ pro které (uvažujeme-li stále $b_i = \frac{z_i}{\sigma_i}$)

$$\nabla_{\vec{\theta}} \left(\| [\frac{1}{\sigma_i} F(x_i,\vec{\theta}) - b_i]_{i=1,\ldots,m} \|^2 \right)_{\vec{\theta}=\vec{\theta}_l} = 0.$$

Pro lineární LS lze „explicitně“ spočítat. Pro nelineární LS musíme použít numerické metody k aproximaci takových bodů. Některé numerické metody ale s $b_i = \frac{z_i}{\sigma_i}$ vůbec nepracují a uplatňují jiné postupy.

8.2 Lineární LS

$$F(x_i,\vec{\theta}) = a_i \cdot \vec{\theta} + c$$

pro všechna $i \in 1,\ldots,m$ bude tedy soustava vypadat následovně:

$$\begin{bmatrix} F(x_1,\vec{\theta}) \\ \vdots \\ F(x_m,\vec{\theta}) \end{bmatrix} = \begin{bmatrix} a_1^T \cdot \vec{\theta} + c_1 \\ \vdots \\ a_m^T \cdot \vec{\theta} + c_m \end{bmatrix} = A \cdot \vec{\theta} + c$$

a z toho dostáváme

$$\| [\frac{1}{\sigma_i} F(x_i,\vec{\theta}) - b_i]_{i=1,\ldots,m} \|^2 = \| \frac{1}{\sigma_i} A \cdot \vec{\theta} + \vec{c} - \vec{b} \|^2,$$

kde A je matice, která má jako řádky vektory $\vec{a}_i$.

Přeznačíme $A := \frac{1}{\sigma_i} A$, $\vec{b} := \vec{c} - \vec{b}$.

Máme tedy lineární LS: $\min \|A\vec{\theta} - \vec{b}\|^2$, kde $A \in \mathbb{R}^{m \times n}$, $\vec{b} \in \mathbb{R}^m$ (m je počet měření a n je počet parametrů). Máme tedy $m > n$ nebo dokonce $m \gg n$ a tedy A je obdélníková.

Pozorování 8.1. *$\|A\vec{\theta} - \vec{b}\| \geq 0$, $\forall \vec{\theta}$. Hledáme tedy $\vec{\theta}$ takové, že je řešením lineární soustavy rovnic $A\vec{\theta} = \vec{b}$ s obdélníkovou maticí.*

Lze převést na systém se čtvercovou maticí?

Lemma 8.2. *$\nabla_\theta(\|\vec{b} - A\vec{\theta}\|) = 2 \cdot A^T(A\vec{\theta} - \vec{b}) \implies [\nabla_{\vec{\theta}}(\|\vec{b} - A\vec{\theta}\|^2) = 0 \iff A^T A\vec{\theta} = A^T\vec{b}]$, kde $A^T A\vec{\theta} = A^T\vec{b}$ je tzv. systém normálových rovnic.*

Pozorování 8.3. *Nakolik z Problému normálových rovnic máme*

$$||A^T A|| = ||A||^2,$$

pak platí

$$\kappa(A^T A) = \kappa(A)^2,$$

neboli nejen, že je tento postup výpočetně náročný sestavit, navíc nám výrazně zhorší přesnost řešení.

V praxi máme velmi často mnoho sloupců A "skoro" lineárně závislých. Tento jev může odpovídat příliš mnoha parametrům pro vysvětlení dat nebo nevhodně zvolené parametrizaci či modelu.

Často platí $\kappa(A) \gg 1$, co znamená, že náš problém je citlivý na perturbace dat.

Stejně jako u systému lineárních rovnic

$$A\vec{x} = b, A \in \mathbb{R}^{n \times n}$$

nemůžeme očekávat lepší přesnost než

$$C \cdot \kappa(A) \cdot \varepsilon_{mach},$$

bez ohledu na volbu algoritmu.

My ale chceme algoritmus, který alespoň sám o sobě tuto chybu nezvětší. Tedy klasicky volíme řešiče založené na unitárních transformacích. My zatím známe pouze QR-faktorizaci (ale existují i iterační alternativy).

8.2.1 Předpočítaná QR-faktorizace

***KROK* 1**
Spočítáme QR-faktorizaci $A = QR$ a uložíme si faktory Q, R.

***KROK* 2**
Spočítáme $\vec{c} = Q^T\vec{b}$.

***KROK* 3**
"Vyřešíme" $R\vec{\theta} = \vec{c}$.

8.2.2 Aplikovaná QR-faktorizace

***KROK* 1**
Počítáme QR-faktorizaci A a všechny transformační matice aplikované na sloupce A aplikujeme i na vektor $\vec{b}$. To odpovídá výpočtu QR-faktorizace prvních n sloupců rozšířené matice

$$\left[\begin{array}{c|c} A & b \end{array}\right].$$

Tímhle postupem dostaneme $R, \vec{c}$.

***KROK* 2**
"Vyřešíme" $R\vec{\theta} = \vec{c}$.

Poznámka. Slovo "vyřešíme" jsme schválně nechali v uvozovkách, protože ne vždy naši soustavu vyřešit lze. Platí totiž, že pokud

$$\vec{c} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \\ \vdots \\ c_m \end{bmatrix},$$

pak

$$\exists \text{ řešení} \iff c_{n+1} = \cdots = c_m = 0,$$

neboli ne vždy existuje řešení.

8.2.3 Existence a jednoznačnost

Mějme proměnné

$$A \in \mathbb{R}^{m \times n}, m > n, \vec{b} \in \mathbb{R}^m.$$

Z lineární algebry máme následující znalosti:

$$Im(A) = span\{\vec{A}_{:,1}, \ldots, \vec{A}_{:,n}\},$$

$$Ker(A^T) = \{\vec{v} : A^T\vec{v} = \vec{0}\},$$

$$\mathbb{R}^m = Im(A) \oplus Ker(A^T),$$

$$Im(A) \perp Ker(A^T).$$

Jinými slovy

$$\forall \vec{b},\ \exists! \vec{v} \in Im(A),\ \exists! \vec{w} \in Ker(A^T) : \vec{b} = \vec{v} + \vec{w}$$

a nutně

$$||\vec{b}||^2 = ||\vec{v}||^2 + ||\vec{w}||^2.$$

Pozorování 8.4. *Platí*

$$\exists \vec{\theta} : A\vec{\theta} = \vec{b} \iff \vec{b} \in Im(A).$$

Dále platí implikace

$$\vec{b} \neq Im(A) \implies \exists! \vec{v}, \vec{w} : \vec{b} = \vec{v} + \vec{w},\ \vec{v} \in Im(A),\ \vec{w} \in Ker(A^T).$$

Z toho plyne, že LS problém

$$\min_{\vec{\theta}} ||\vec{b} - A\vec{\theta}||^2$$

se dá zapsat jako

$$\min_{\vec{\theta}} ||\vec{w} + (\vec{v} - A\vec{\theta})||^2 = \min_{\vec{\theta}} ||w||^2 + ||\vec{v} - A\vec{\theta}||^2 = ||w||^2 + \min_{\vec{\theta}} ||\vec{v} - A\vec{\theta}||^2,$$

kde $(\vec{v} - A\vec{\theta}) \in Im(A)$. *Čili pokud* $\nexists \vec{\theta} : A\vec{\theta} = \vec{b}$, *pak řešení LS problému odpovídá řešení rovnice*

$$A\vec{\theta} = \vec{v},$$

kde $\vec{v}$ *je projekce* $\vec{b}$ *na* $Im(A)$. *To je přesně to samé, co jsme psali výše - Q je báze* $Im(A)$ *a* $\vec{c} = Q^T\vec{b} = \vec{v}$.

Pozorování 8.5. *Nechť*

$$\vec{\xi} \in Ker(A) : A\vec{\theta} = \vec{b} \iff A(\vec{\theta} + \vec{\xi}) = \vec{b}.$$

Pak řešení je jednoznačné právě tehdy, když sloupce matice A jsou lineárně nezávislé, což je právě tehdy, když $Rank(A) = n$.

Pokud $Rank(A) < n$ a $\vec{b} \in Im(A)$, pak existuje nekonečně mnoho $\vec{\theta} : A\vec{\theta} = \vec{b}$. Ovšem analogicky postupu výše

$$\vec{\theta} \in \mathbb{R}^n \implies \exists! \vec{p} \in Ker(A),\ \exists! \vec{q} \in Im(A^T) : \vec{\theta} = \vec{p} + \vec{q}$$

a platí

$$||\vec{\theta}||^2 = ||\vec{p}||^2 + ||\vec{q}||^2.$$

Z toho plyne, že $\exists! \vec{\theta}$ s minimální normou (tj. $\vec{\theta} \in Im(A^T) \perp Ker(A)$), které bychom chtěli spočítat.

Pokud $Rank(A) =: r < n$, pak existuje permutační matice $P \in \mathbb{R}^{n \times n}$ taková, že

$$AP = \left[\begin{array}{c|c} A_r & \tilde{A} \end{array} \right],$$

kde $A_r \in \mathbb{R}^{m \times r}$ má lineárně nezávislé sloupce a každý sloupec $\tilde{A} \in \mathbb{R}^{m \times (n-r)}$ lze zapsat jako lineární kombinace sloupců A_r.

Pak QR-faktorizace AP nám dá

$$AP = \left[\begin{array}{c|c} A_r & \tilde{A} \end{array} \right] = Q_r \left[\begin{array}{c|c} R_r & \tilde{R} \\ 0 & 0 \end{array} \right].$$

Sérií úprav jde ukázat

$$\begin{gathered} A\vec{\theta} = \vec{b} \\ \Longleftrightarrow \\ APP^T\vec{\theta} = \vec{b} \\ \Longleftrightarrow \\ Q_r \left[\begin{array}{c|c} R_r & \tilde{R} \\ 0 & 0 \end{array} \right] \vec{v} = \vec{b}, \\ \vec{\theta} = P\vec{v} \\ \Longleftrightarrow \\ \left[\begin{array}{c|c} R_r & \tilde{R} \\ 0 & 0 \end{array} \right] \begin{bmatrix} \vec{v} \\ \vec{v} \end{bmatrix} = Q_r^T \vec{b} \end{gathered}$$

Lze ukázat, že počítání permutační matice P můžeme dělat za pochodu - zcela analogicky pivotaci u GE nebo LU-faktorizaci.

Dále lze ukázat, že řešení $\vec{\theta} = P\vec{v}_r$ je to hledané, tj. s minimální normou.

8.3 Nelineární LS a minimalizace

Potřebujeme metody pro numerickou minimalizaci - chceme řešit $\min_\theta F(\theta)$.

My si tu představíme jen hlavní myšlenky: všechny minimalizační metody jsou iterativní - produkují tedy posloupnost $\vec{\theta_0}$,$\vec{\theta_1}$,$\ldots$,$\vec{\theta_N}$, myšlenky jak získat $\vec{\theta_n}$ z $\vec{\theta_{n-1}}$,$\vec{\theta_{n-2}}$,$\ldots$ se ale liší.

V praxi můžeme mít i tzv. constrained minimum problémy, kde parametry $\vec{\theta}$ jsou uvažovány pouze v nějaké množině, například $\vec{\theta} \in \{||\vec{\theta}||_2 \leq 1\}; \max_{i=1,\ldots,n} |\vec{\theta_i}| \leq \delta$ nebo $\{\vec{\theta} \in \mathbb{R}^n | C \cdot \vec{\theta} = \vec{d}\}$.

8.3.1 Derivative-free metody

Jedná se o metody, kde se odvozuje pouze z vyhodnocování $F(\vec{\theta})$ v několika bodech a iterativním procesu založeném na jejich hodnotách. Používáme interpolace/heuristiky.

Příkladem derivative-free metody je Nelder-Mead.

Máme-li

- $\vec{\theta} \in \mathbb{R}^2$ - pracujeme s trojúhelníky (určeno 2+1 body v $\mathbb{R}^2$)
- $\vec{\theta} \in \mathbb{R}^3$ - pracujeme s čtyřstěny (určeno 3+1 body v $\mathbb{R}^3$)
- $\vec{\theta} \in \mathbb{R}^n$ - pracujeme se simplexy v $\mathbb{R}^n$

Definice 8.6. *Mějme* $\vec{x_1},\ldots,\vec{x}_{n+1} \in \mathbb{R}^n$ *a* $\vec{x_2}-\vec{x_1},\ldots,\vec{x}_{n+1}-\vec{x_1}$ *jsou lineárně nezávislé. Simplex je pak*

$$S_{x_1,\ldots,x_{n+1}} := \{\sum_{i=1}^{n+1} \lambda_i \cdot \vec{x_i},\ \lambda_i \in (0,1)\ \forall i;\ \lambda_1+\cdots+\lambda_{n+1}=1\}.$$

Předpokládejme, že máme body $\vec{\theta_1},\ldots,\vec{\theta}_{n+1}$ a chceme přidat bod $\vec{\theta}_{n+2}$. (přechod od simplexu $S_{\theta_1,\ldots,\theta_{n+1}}$ k novému, lepšímu S^{new} výměnou jednoho bodu $\vec{\theta_1},\ldots,\vec{\theta}_{n+1}$)

***KROK* 1**

Spočteme $F(\vec{\theta}_1),\ldots,F(\vec{\theta}_{n+1})$,

vyměníme $\vec{\theta}_{max}$ za $\vec{\theta}_i := \arg\max_i |F(\vec{\theta}_i)|$,

zachováme $\vec{\theta}_1,\ldots,\vec{\theta}_{i-1},\vec{\theta}_{i+1},\ldots,\vec{\theta}_{n+1}$ - n bodů, které určují "protější" stranu (tedy protější stranu simplexu $S_{\theta_1,\ldots,\theta_{n+1}}$ k bodu $\vec{\theta}_i$) a označíme $\vec{\theta}_c$ jako centrum této strany (v $\mathbb{R}^2$ bude centrem střed protější strany trojúhelníku).

***KROK* 2**

Za co $\vec{\theta}_i$ vyměnit?

Heuristicky, $\vec{\theta_i}$ je nejhorší - vezmeme tedy $\vec{\theta}_{n+2}$ jako nějaký bod na přímce dané body $\vec{\theta_1}$,$\vec{\theta_c}$ (pokud nejsou všechny horší).

Záruky konvergence "téměř" neexistují, ale v praxi to často funguje (speciálně v nižších dimenzích), je robustní a výpočetně nenáročný (1 iterace $\leq$ 2 vyhodnocení funkce F).

8.3.2 Derivative-based metody

Jedná se o metody založené na používání (parciálních) derivací $F(\vec{\theta}_{n-1})$. Často založeno na hledání bodů $\vec{\theta}_{kandidát} : \nabla F \cdot \vec{\theta}_{kandidát} = \vec{0}$. Pomocí nějaké iterační metody. My se budeme zabývat dvěma metodami - First-order DBM (využívají se pouze gradienty) a second-order DBM(využívají se gradienty a Hessova matice).

Poznámka. Téměř všechny základní myšlenky kopírují hledání minima funkce $\mathbb{R}\to\mathbb{R}$. Například $f'(x)<0 \to f$ je klesající v bodě x nebo nutná podmínka optimality: $f'(x_{opt})=0$

First-order metody

Založené na principu, že $F:\mathbb{R}^n\to\mathbb{R}$ nejrychleji klesá ve směru $-\nabla F(\vec{\theta}_0)$ ($\vec{\theta}_0\in\mathbb{R}^n$).

Jedná se například o Steepest descent metody.

Idea za těmito metodami je, že máme $\vec{\theta}_k$ a chceme dostat $\vec{\theta}_k + update$, kde $update := \alpha_k\vec{d}_k$ (kde $\vec{d}_k$ má normu jedna - určuje směr a α_k určuje, jak daleko daným směrem pokračujeme).

$\vec{d}_k$ vybíráme jako směr, ve kterém nám funkce nejprudčeji klesá, tedy $\frac{-\nabla F(\vec{\theta}_0))}{||\nabla F(\vec{\theta})_0||}$ (d_k musí mít normu 1).

V praxi buď musí rutinu pro výpočet $\vec{\theta}\to\nabla F(\vec{\theta})$ poskytnout uživatel, nebo se používají aproximace

$$[\nabla F(\vec{\theta}_0)]_i = \lim_{h\to 0}\frac{F(\vec{\theta}+h\cdot\vec{e_i})-F(\vec{\theta})}{h} \approx \frac{F(\vec{\theta}+h\cdot\vec{e_i})-F(\vec{\theta})}{h}.$$

α_k volíme podle konkrétního typu metody. Nalezení optimálního α_k už je samo o sobě pouze 1D minimalizace ($\arg\min_{\alpha\in\mathbb{R}} F(\vec{\theta}_k-\alpha_k\vec{d}_k)$) - tzv. line-search. V praxi používáme spoustu různých heuristik (Armijo rule, trust region, ...). V principu lze použít i například Nelder-Mead (v $\mathbb{R}$) na nalezení optimálního α_k.

Second-order metody

Pokud $\vec{\theta}_{opt} \in \mathbb{R}^d$ je lokální extrém F a $(\nabla_0 F)(\vec{\theta}_{opt})$ existuje, pak $(\nabla_0 F)(\vec{\theta}_{opt}) = \vec{0}$.

Jako příklad metody uvedeme quasi-Newton metody.

Idea za těmito metodami je, že chceme najít body $\vec{\theta}^1_{kand}$,$\vec{\theta}^2_{kand}$, $\dots$,$\vec{\theta}^d_{kand}$, pro které $\nabla F(\vec{\theta}^i_{kand}) = \vec{0}$. Chci tedy vyřešit (nenelineární) soustavu rovnic $\rightarrow$ Newtonova metoda.

Budeme potřebovat aproximovat naší nelineární funkci ∇F pomocí lineární - Jacobiho matici funkce ∇F - dostaneme tedy Hessovu matici funkce F:

$$\vec{\theta} \rightarrow \begin{bmatrix} \frac{\partial^2 f}{\partial \theta_1^2} & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 f}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 f}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 f}{\partial \theta_2^2} & \cdots & \frac{\partial^2 f}{\partial \theta_2 \partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 f}{\partial \theta_n \partial \theta_2} & \cdots & \frac{\partial^2 f}{\partial \theta_n^2} \end{bmatrix} =: H_{\vec{\theta}}.$$

Z Newtonovy metody tedy máme

$$\vec{\theta}_{k+1} = \vec{\theta}_k - \vec{v}_k,$$

kde

$$H_{\vec{\theta}_k} \vec{v_k} = \nabla F(\theta_k).$$

Jak získat $\nabla_{\vec{\theta}_k} \vec{F}(\vec{\theta}_k)$ a $H_{\vec{\theta}_k}$? - Buď nám uživatel poskytne rutiny na výpočet, nebo aproximací.

Aproximace ve stylu "diferencí":

$$\nabla F(\vec{\theta}_k)_i = \lim_{h \to 0} \frac{F(\vec{\theta} + h \cdot \vec{e}_i) - F(\vec{\theta})}{h} \approx \frac{F(\vec{\theta} + h \cdot \vec{e}_i) - F(\vec{\theta})}{h}.$$

Aproximace ve stylu "update":

tzv. quasi-Newton metody, protože nepočítáme "správnou Hessovu matici", ale místo toho položíme

$$H_{\vec{\theta}_{k+1}} = H_{\vec{\theta}_k} + \text{update},$$

kde update může být například $rank - 1 : \beta_k \cdot \vec{w}_k \cdot \vec{w}_k^T$ a volíme $\beta_k, \vec{w}_k$ "optimálně".

9 Singulární rozklad a vlastní čísla

9.1 Motivace

Představme si následující situaci - máme dataset (uložený obrázek, data z burzy, ...). Problém - máme hodně dat, nevíme, jak je analyzovat, a nelze je prakticky dlouho skladovat.

Potřebujeme kompresi dat, která zachová důležité rysy a analýzu dat, tj. co z nich jde vyčíst.

Ve spoustě případech máme data v tabulce, neboli v matici, a potřebujeme umět komprimovat velikost matice a analyzovat jí.

Hlavní nástroj pro nás bude singulární rozklad (SVD) pro $A \in \mathbb{R}^{m\times n}, m > n$.

Věta 9.1 (O singulárním rozkladu)**.** *Nechť $A \in \mathbb{R}^{m\times n}$ je matice. Pak*

$$\exists U \in \mathbb{R}^{m\times m}, \exists V \in \mathbb{R}^{n\times n}, \exists \Sigma \in \mathbb{R}^{m\times n} : A = U\Sigma V^T,$$

kde U, V jsou unitární a

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_n & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \end{bmatrix}, \ \sigma_1 \geq \sigma_1 \geq \cdots \geq \sigma_n \geq 0.$$

9.2 Komprese dat

Mějme $A = U\Sigma V^T, U = \begin{bmatrix}\vec{u}_1 & \dots & \vec{u}_m\end{bmatrix}, V = \begin{bmatrix}\vec{v}_1 & \dots & \vec{v}_n\end{bmatrix}$. Pak umíme vyjádřit matici A pomocí tzv. dyadického rozvoje

$$A = \sigma_1\vec{u}_1\vec{v}_1^T + \sigma_2\vec{u}_2\vec{v}_2^T + \cdots + \sigma_n\vec{u}_n\vec{v}_n^T.$$

Idea aproximace matice A

$$A \approx \sigma_1\vec{u}_1\vec{v}_1^T + \sigma_2\vec{u}_2\vec{v}_2^T + \cdots + \sigma_r\vec{u}_r\vec{v}_r^T =: A_r,$$

kde

$$A_r = U_{:,1:r}\Sigma_{1:r,1:r}\left(V_{:,1:r}\right)^T.$$

Memory cost naší původní matice $A \sim mn$, zatímco $A_r \sim r(m+n+1)$.

Věta 9.2 (Eckart-Young-Mirsky)**.** *Mějme $A \in \mathbb{R}^{m\times n}, r \leq n \leq m$. Nechť $A = U\Sigma V^T$ je singulární rozklad. Pak nejpřesnější aproximace A maticí hodnosti r je právě matice A_r daná prvními r členy dyadického rozvoje A.*

Poznámka. Předchozí větu jde taky zapsat jako

$$\forall X \in \mathbb{R}^{m\times r}, \forall Y \in \mathbb{R}^{n\times r} : ||A - XY^T|| \geq ||A - A_r||.$$

Navíc platí

$$||A - A_r||_2 = \sigma_{r+1},$$
$$||A - A_r||_F = \sqrt{\sigma_{r+1}^2 + \sigma_{r+2}^2 + \cdots + \sigma_{\min\{m,n\}}^2}.$$

Spočtením singulárního rozkladu jsme schopni najít nejpřesnější kompresi dat v Eukleidovské normě, a to konkrétně z mn dat na $r(m+n)$ dat.

9.3 Analýza dat

Nechť moje data představují řádky matice A, čili $\vec{a}_i \in \mathbb{R}^n$, a máme jich m, neboli $i = 1, \ldots, m$.

Zafixujme $n = 3$.

Když budou $\vec{a}_1, \ldots, \vec{a}_m$ všechny ležet na jedné přímce, směrový vektor $\vec{s}$ té přímky nám o našich datech hodně vypovídá.

Máme

$$\vec{a}_i = \alpha_i \vec{s}.$$

V maticovém zápisu

$$A = \begin{bmatrix} \vec{a}_1^T \\ \vdots \\ \vec{a}_m^T \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} \cdot \vec{s}^T$$

a platí $Rank(A) = 1$.

Slovy řečeno, všechna data mají stejný poměr mezi hodnotami - liší se jen škálováním, neboli rozptyl v datech lze vysvětlit pozorováním pouze tohoto škálování.

Když budou $\vec{a}_1, \ldots, \vec{a}_m$ ležet okolo přímky, všechny rovnosti výše lze nahradit "$\approx$"a dostaneme

$$A = \begin{bmatrix} \vec{a}_1^T \\ \vdots \\ \vec{a}_m^T \end{bmatrix} \approx \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} \cdot \vec{s}^T.$$

Opět lze tento jev popsat slovy tak, že data mají podobný poměr mezi hodnotami, a tedy rozptyl v datech odpovídá pouze škálování tohoto poměru.

Pozorování 9.3. *Pokud by data ležela přibližně v rovině dané vektory $\vec{s}_1, \vec{s}_2$, pak máme*

$$A = \begin{bmatrix} \vec{a}_1^T \\ \vdots \\ \vec{a}_m^T \end{bmatrix} \approx \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \vdots & \vdots \\ \alpha_{m,1} & \alpha_{m,2} \end{bmatrix} \cdot \begin{bmatrix} \vec{s}_1^T \\ \vec{s}_2^T \end{bmatrix}$$

a platí $Rank(A) = 2$. Analogicky lze odvodit vztahy v $\mathbb{R}^n$.

Analýza dat se tedy skládá z úkolu najít co nejmenší počet vektorů $\vec{s}_1, \ldots, \vec{s}_r$, které nám odvozeným způsobem vysvětlí naše data. Těmto vektorům se říká principal components a celé této metodě se proto říká Principal component analysis (PCA).

Nalezením těchto vektorů odpovídá nalezení nejlepší $Rank - r$ aproximaci matice A. K tomu stačí spočíst SVD.

9.4 Vlastní čísla

9.4.1 Spektrální rozklad

Nechť $A = U\Sigma V^T$. Pak platí

$$A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V,$$

kde V je unitární a $\Sigma^T\Sigma = diag(\sigma_1^2, \ldots, \sigma_n^2)$. $A^T A = V\Sigma^T\Sigma V$ je spektrální rozklad (tj. diagonalizace) symetrické, pozitivně semi-definitní matice $A^T A$. Všechna její vlastní čísla (tj. prvky na diagonále $\Sigma^T\Sigma$) jsou nezáporná.

Výpočet SVD se standardně převádí na výpočet spektrálního rozkladu (EVD), tj. pro danou matici M nalezení jejich vlastních čísel a vlastních vektorů (protože $A^T A$ je symetrická, její EVD vždy existuje).

9.4.2 Motivace pro výpočet vlastních čísel

Příklad 9.4 (Algoritmus vyhledávače Google)**.** *Mějme seznam všech webstránek L délky $n \in \mathbb{N}$.*

***KROK* 1**
Sestrojíme si matici $A \in \mathbb{R}^{n\times n}$ jako

$$A_{i,j} = \begin{cases} 0, \text{ pokud se webstránka } L(i) \text{ nezmiňuje o webstránce } L(j), \\ \frac{1}{k}, \text{ pokud se webstránka } L(i) \text{ zmiňuje o webstránce } L(j), \end{cases}$$

kde k označuje počet stránek, které webstránka $L(i)$ zmiňuje.

***KROK* 2**
Spočteme dominantní vlastní vektor matice A, tj. $\vec{v} \in \mathbb{R}^n$. Lze ukázat, že všechny jeho složky jsou ≥ 0. Toto zjistění můžeme interpretovat jako ekvilibrium popularity stránek v L, tj. $(\vec{V})_i \equiv v_i > v_j$, neboli stránka $L(i)$ bude podle tohoto modelu populárnější než $L(j)$.

***KROK* 3**
Na outputu seřadíme webstránky z listu L zestupně podle velikosti složek vektoru $\vec{v} \in \mathbb{R}^n$.

▲

9.5 Mocninná metoda

Předpokládejme, že máme

$$A = S\Lambda S^{-1},$$

kde S je matice vlastních vektorů, $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ matice vlastních čísel.

Pak platí

$$A^k = S\Lambda^k S^{-1} = S \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{bmatrix} S^{-1},$$

a tedy

$$\forall i > 1 : |\lambda_1| > |\lambda_i|,$$
$$\forall i > 1 : |\lambda_1^k| \gg |\lambda_i^k|,$$

tj. dominantní vlastní pár je ještě více dominantní.

Vezměme si vektor

$$\vec{w} = c_1\vec{s}_1 + \cdots + c_n\vec{s}_n = S\vec{c},$$

neboli $\vec{w}$ je ve vlastní bázi A pouze $\vec{c}$. Pak

$$A^k\vec{w} = S\Lambda^k S^{-1}S\vec{c} = S\Lambda^k\vec{c} = S \begin{bmatrix} \lambda_1^k c_1 \\ \vdots \\ \lambda_n^k c_n \end{bmatrix} = \lambda_1^k c_1\vec{s}_1 + \cdots + \lambda_n^k c_n\vec{s}_n =$$

$$= \lambda_1^k \left(c_1\vec{s}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \vec{s}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \vec{s}_n \right)$$

a platí implikace

$$\forall i > 1 : |\lambda_1| > |\lambda_i| \implies \left(\frac{\lambda_i}{\lambda_1}\right)^k \xrightarrow{k\to+\infty} 0.$$

Navíc pokud $c_1 \neq 0$, pak

$$A^k \vec{w} \to \lambda_1^k c_1 \vec{s}_1.$$

Tomuto vyjádření říkáme vektor ve směru dominantního vlastního vektoru matice A. Jde o základní myšlenku tzv. Mocninné metody.

V praxi je potřeba normalizace, tj. nepočítáme $AA \dots A\vec{w}$, ale

$$\vec{v}_0 \to \vec{w}_1 = A\vec{v}_0,$$

$$\vec{v}_1 = \frac{\vec{w}_1}{||\vec{w}_1||} \to \vec{w}_2 = A\vec{v}_1,$$

$$\vec{v}_2 = \frac{\vec{w}_2}{||\vec{w}_2||} \to \dots \to \vec{w}_k = A\vec{v}_{k-1},\ \vec{v}_k = \frac{\vec{w}_k}{||\vec{w}_k||}.$$

9.5.1 První zobecnění

Co když chceme nejmenší vlastní pár? Platí λ_{min} matice A je λ_{max} matice A^{-1}, protože

$$\forall i < n, \lambda_i \neq 0 : |\lambda_n| < |\lambda_i| \implies \left|\frac{1}{\lambda_n}\right| > \left|\frac{1}{\lambda_i}\right|.$$

Nyní nám stačí předešlý vztah přepsat pro inverzní matice

$$\vec{v}_0 \to \vec{w}_1 = A^{-1}\vec{v}_0,$$

$$\vec{v}_1 = \frac{\vec{w}_1}{||\vec{w}_1||} \to \vec{w}_2 = A^{-1}\vec{v}_1,$$

$$\vec{v}_2 = \frac{\vec{w}_2}{||\vec{w}_2||} \to \dots \to \vec{w}_k = A^{-1}\vec{v}_{k-1},\ \vec{v}_k = \frac{\vec{w}_k}{||\vec{w}_k||}.$$

Místo výpočtu $A^{-1}\vec{v}_{i-1}$ budeme řešit $A\vec{w}_i = \vec{v}_{i-1}$.

Co kdybychom nechtěli nejmenší či největší vlastní pár, ale nejbližší číslu $\alpha \in \mathbb{C}$? Pak platí stejná úvaha, místo násobení A budeme násobit $(A - \alpha I)^{-1}$.

9.5.2 Druhé zobecnění

Co když nechceme jen jeden vlastní pár, ale všechny?

Vybereme si rutinu na QR-faktorizaci, a to jako funkci

$$qr : M \to Q_M, R_M \text{ tak, že } M = Q_M R_M.$$

Chceme všechny vlastní páry. Musíme proto iterovat s n vektory. Máme

$$A \equiv A_0 \to Q_0 R_0 := qr(A_0),\ A_1 := R_0 Q_0.$$

Pak jistě

$$A_0 = Q_0 R_0 = Q_0 A_1 Q_0^T \to$$
$$\to Q_1 R_1 := qr(A_1),\ A_2 := R_1 Q_1.$$

Pak opět

$$A_1 = Q_1 R_1 = Q_1 A_2 Q_1^T \to \dots \to Q_k R_k := qr(A_k),\ A_{k+1} := R_k Q_k.$$

Ve všech rovnostech platí, že

$$\{\text{vlastní čísla } A_i\} = \{\text{vlastní čísla } A_{i-1}\}.$$

Tenhle postup se nazývá tzv. QR-algoritmus.

9.5.3 Podmíněnost

Jak moc se liší vlastní páry matice $A + E$ od vlastních párů matice A, pro $||E||$ malé.
Je-li A symetrická, pak platí

$$|\lambda_i(A) - \lambda_i(A + E)| \leq ||E||_2.$$

Pokud je však A obecná, nelze nic zaručit. Matematicky toto odpovídá podmíněnosti kořenů polynomu v závislosti na změně koeficientu v neortogonální bázi polynomu (viz. Wilkinsonův polynom).

Můžeme tedy prohlásit, že problém nalezení SVD matice je dobře podmíněný.

9.5.4 Stabilita

Jelikož používáme unitární transformaci, která zachovává normu matice, platí, že QR-algoritmus je stabilní.

Co se týče Mocninné metody a variace, záleží na řešiči.

Souhrn otázek

Interpolace

- Na základě interpolačních podmínek $p(x_i) = f_i$ pro $i = 0, 1, \ldots, n$ odvoďte lineární systém algebraických rovnic pro koeficienty polynomu $p(x)$. Šel by tento postup zobecnit pro případ, kdy máme dané i hodnoty f_i' pro první derivace polynomu $p(x)$ v bodech $x_0, \ldots, x_n$? Pokud ano, vysvětlete ideu, pokud ne, vysvětlete proč. Jaké jsou výhody a nevýhody výpočtu $p(x)$ tímto způsobem?

  Definujte alternativní přístup k nalezení $p(x)$, který je explicitní, tj. nevyžaduje řešení soustav rovnic.

- Odvoďte Lagrangeův interpolační polynom $p(x)$ pro dané interpolační podmínky $p(x_i) = f_i$ pro $i = 0, 1, \ldots, n$.

  Na základě znalosti chyby interpolace pro hladké funkce porovnejte použití ekvidistantních, Čebyševových a Legendrových bodů pro polynomiální interpolaci. Jaké body byste nazvali optimálními pro 1D interpolaci a proč?

  Odvoďte odhad podmíněnosti problému interpolace v daných bodech $x_0, \ldots, x_n$ na základě Lebesgueovy konstanty.

- Vysvětlete pojem *spline*. Definujte lineární spline pro interpolační podmínky $p(x_i) = f_i$ pro $i = 0, 1, \ldots, n$ a odvoďte explicitní vyjádření jeho hodnoty v libovolném bodě x z intervalu (x_0, x_n).

  Okomentujte výhody a nevýhody lineárního splinu a definujte kubický spline.

  Porovnejte vývoj interpolační chyby pro polynomiální interpolaci, lineární spline a kubický spline pro rostoucí počet bodů, tj. asymptoticky pro $n \to +\infty$.

Podmíněnost a stabilita

- Vysvětlete rozdíl mezi výpočtem na papíře a na počítači. Uveďte jednoduchý konkrétní příklad. Čím je tento problém způsoben? Jak tento rozdíl ovlivňuje používání teoreticky odvozených výsledků v praxi? Podpořte vaši odpověď alespoň jedním příkladem.

  Vysvětlete pojem stabilního algoritmu, uveďte definici. Uveďte příklad stabilního algoritmu.

- Vysvětlete pojem čísla podmíněnosti matematického problému. U praktických výpočtů se téměř vždy musí počítat s chybou v datech - uveďte dva různé důvody proč (odkud tyto chyby často přicházejí). Jak toto pozorování souvisí s vaší definicí a co z toho vyplývá o nutnosti/zbytečnosti čísla podmíněnosti problemu pro praxi?

  Uveďte jeden příklad matematického problému a odvoďte jeho číslo podmíněnosti.

Kvadratura

- Pro
$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} f(x_i) \cdot w_i$$
pro dané body $x_0, \ldots, x_n$ v intervalu (a, b) odvoďte hodnoty kvadraturních vah w_i založených na polynomiální interpolaci v těchto bodech.

  Pro jakou volbu bodů $x_0, \ldots, x_n$ získáme tzv. Newton-Cotesovu kvadraturu a kterým bodům odpovídá tzv. Gaussova (resp. Gauss–Legendrova) kvadratura?

Definujte pojem *řád kvadratury*.

Definujte pojem *kompozitní kvadratura*.

Co lze říci o chybě kvadraturních pravidel založených na interpolaci?

Pro jaké případy je vhodné používat kvadraturní pravidla a kdy (a proč) je nutné použít jiné metody pro numerickou integraci?

Monte Carlo

- Formulujte metodu *Monte Carlo importance sampling* pro aproximaci integrálu.

  Uveďte klasický odhad chyby, platný s pravděpodobností $1 - \varepsilon$.

  Jaké jsou hlavní výhody metody Monte Carlo a jaké jsou naopak její hlavní nevýhody?

  Jak lze tuto metodu adaptovat, abychom minimalizovali dopad těchto nevýhod?

ODR

- Odvoďte explicitní a implicitní Eulerovu metodu (nikoliv pouze napište) pro numerické řešení problému
$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0.$$

  Pro danou rovnici a danou velikost kroku τ nakreslete dva kroky explicitní a implicitní Eulerovy metody.

  Definujte řád jednokrokové metody tvaru

$$y_{n+1} = y_n + \tau\, \Psi(n, \tau, y_n, y_{n+1})$$

  a uveďte řád explicitní a implicitní Eulerovy metody.

  K čemu je pojem řád metody dobrý a co nám o dané metodě říká?

  Vysvětlete, jak lze odvodit explicitní a implicitní metody vyšších řádů, například metody typu Runge–Kutta.

  Porovnejte silné a slabé stránky jednokrokových implicitních a explicitních metod.

- Odvoďte explicitní a implicitní Eulerovu metodu (nikoliv pouze napište) pro numerické řešení problému
$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0.$$

  Pro danou rovnici a danou velikost kroku τ nakreslete dva kroky explicitní a implicitní Eulerovy metody.

  Definujte A-stabilitu pro jednokrokovou metodu pro testovací rovnici

$$y'(t) = \lambda y(t), \quad y(0) = y_0.$$

  Odvoďte oblast stability pro explicitní a implicitní Eulerovu metodu.

  Porovnejte silné a slabé stránky jednokrokových implicitních a explicitních metod.

Nelineární algebraické rovnice

- Pro $F : \mathbb{R}^n \to \mathbb{R}^n$ odvoďte Newtonovu metodu (nikoliv pouze napište) pro aproximaci kořene x^* (tj. pro x^* takové, že $F(x^*) = 0$).

  Jaké podmínky musí platit, abychom mohli tuto metodu použít?

  Graficky znázorněte tuto metodu pro speciální případ $n = 1$, tj. $f : \mathbb{R} \to \mathbb{R}$, danou na obrázku.

  Odvoďte řád konvergence Newtonovy metody pro $f : \mathbb{R} \to \mathbb{R}$ a okomentujte, za jakých podmínek tuto konvergenci očekáváte.

  Co lze říci o chování Newtonovy metody, pokud tyto podmínky nejsou splněny?

- Zformulujte Banachovu větu o kontrakci a vysvětlete její využití k aproximaci kořene x^* funkce $F : \mathbb{R}^n \to \mathbb{R}^n$ (tj. k aproximaci x^* takového, že $F(x^*) = 0$).

  Pro konkrétní dané funkce $f, g : \mathbb{R} \to \mathbb{R}$ rozhodněte, zda lze použít tuto větu.

  Zformulujte Newtonovu metodu pro hledání aproximace kořene $F : \mathbb{R}^n \to \mathbb{R}^n$ (tj. k aproximaci x^* takového, že $F(x^*) = 0$).

  Porovnejte výhody a nevýhody použití Banachovy věty vs použití Newtonovy metody pro aproximaci kořene.

Lineární algebraické rovnice

Gaussova eliminace a LU faktorizace

- Pro regulární matici A velikosti $n \times n$ odvoďte její LU rozklad (nikoliv pouze napište).

  Za jakých podmínek může tento postup selhat? Jak byste v takovém případě postupovali?

  Detailně odvoďte nový, vylepšený postup v maticovém zápisu.

  Vysvětlete souvislost LU rozkladu matice A a Gaussovou eliminaci pro soustavu lineárních algebraických rovnic $A\mathbf{x} = \mathbf{b}$ (pro daný n-dimenzionální vektor $\mathbf{b}$).

  Definujte pojem *růstový faktor* a to, jak souvisí se stabilitou těchto algoritmů (není třeba psát vzorečky, ale pak je třeba správně popsat situaci slovy. Vzorečky s komentářem jsou samozřejmě také přípustné).

  Co lze obecně říci o velikosti růstového faktoru?

- Pro regulární matici A velikosti $n \times n$ odvoďte její LU rozklad (nikoliv pouze napište).

  Za jakých podmínek může tento postup selhat? Jak byste v takovém případě postupovali?

  Popište nový, vylepšený postup (stačí popsat změnu oproti tomu, co jste odvodili výše).

  Vysvětlete souvislost LU rozkladu matice A s Gaussovou eliminací pro soustavu lineárních algebraických rovnic $A\mathbf{x} = \mathbf{b}$ (pro daný n-dimenzionální vektor $\mathbf{b}$).

  Definujte číslo podmíněnosti matice A a napište set-up pro studium čísla podmíněnosti problému $A\mathbf{x} = \mathbf{b}$.

  Náčrtněte odvození čísla podmíněnosti problému $A\mathbf{x} = \mathbf{b}$.

Ortogonalizace a QR faktorizace

- Pro regulární matici A velikosti 4×4 odvoďte její QR rozklad pomocí Gram-Schmidtova ortogonalizačního procesu.

  U tohoto procesu existují dvě varianty - tzv. klasická a modifikovaná. Uveďte, kterou jste použili a vysvětlete, jak by se postup lišil pro tu druhou variantu.

Porovnejte stabilitu těchto algoritmů - jak přesně se spočítá QR rozklad matice A? Označili byste tyto algoritmy za stabilní?

Porovnejte je s ostatními algoritmy pro výpočet QR rozkladu, které jsme na přednášce viděli.

- Pro regulární matici A velikosti 3×3 odvoďte její QR rozklad pomocí Householderových reflexí a následně pomocí Givensových rotací.

  Porovnejte stabilitu těchto algoritmů – jak přesně se spočítá QR rozklad matice A? Definujte veličinu určující „ztrátu ortogonality“ a uveďte její vývoj vzhledem k číslu podmíněnosti matice A pro každý z těchto algoritmů.

  Porovnejte tento výpočet QR rozkladu s ostatními alternativami, které jsme na přednášce viděli.

  Uveďte dva příklady, jeden pro který bude nejvýhodnější využít Householderovy reflexe a druhý, pro který byste jako nejvhodnější metodu výpočtu QR rozkladu zvolili Givensovy rotace.

Iterační řešiče

- Pro soustavu lineárních algebraických rovnic $Ax = b$ se čtvercovou regulární maticí A odvoďte Richardsonovu stacionární iterativní metodu pro aproximaci řešení x.

  Na základě odvození ukažte nutnou podmínku pro konvergenci této metody pro libovolný vektor b a libovolný počáteční vektor x_0.

  Vysvětlete dvě různá zobecnění Richardsonovy metody – obecné stacionární metody založené na stupni matice A a metody Krylovových podprostorů.

  Porovnejte výhody a nevýhody použití iterativních a přímých metod pro řešení $Ax = b$.

Problém nejmenších čtverců

Lineární

- Mějme lineární model F určený parametry θ (vektor délky n), který mapuje input x na output $F(x,\theta)$.

  Předpokládejme, že proces, který náš model F aproximuje, můžeme pozorovat a máme tedy k dispozici naměřené outputy z pro inputy x (kde x, z jsou vektory délky m). Formulujte tzv. MLE odhad (maximum likelihood estimate) pro parametry θ pro dané F, x a z.

  Předpokládejte, že chyba měření $z - F(x,\theta)$ odpovídá normálnímu rozdělení a odvoďte z MLE formulace matematickou formulaci problému nejmenších čtverců. Na základě linearity odvoďte jeho maticovou formulaci a uveďte dva různé způsoby řešení tohoto problému. U každého ze způsobů vyzdvihněte jeho klady a zápory.

- Mějme problém nejmenších čtverců zapsaný maticovou formulací $A\theta = b$. Odvoďte, za jakých podmínek existuje přesné řešení tohoto problému.

  Je určeno jednoznačně? Pokud ano, dokažte; pokud ne, formulujte dodatečné podmínky, které tuto jednoznačnost zaručí.

  Může se v praxi stát, že přesné řešení neexistuje (tj. že vaše podmínky výše nejsou splněny)? Vysvětlete svou odpověď na konkrétním příkladu. Pokud by se to stát mohlo, popište, jak problém změnit na jiný, řešitelný problém. Vysvětlete rozdíl mezi původním a pozměněným problémem a důvod, proč jste se rozhodli původní problém změnit právě takto.

Nelineární a optimalizace

- Mějme model F určený parametry θ (vektor délky n), který mapuje vstup x na výstup $F(x, \theta)$. Předpokládejme, že proces, který náš model F aproximuje, můžeme pozorovat a máme tedy k dispozici naměřené výstupy z pro vstupy x (kde x, z jsou vektory délky m). Formulujte tzv. MLE odhad (maximum likelihood estimate) pro parametry θ pro dané F, x a z.

  Předpokládejte, že chyba měření $z - F(x, \theta)$ odpovídá normálnímu rozdělení a odvoďte z MLE formulace matematickou formulaci problému nejmenších čtverců jako minimalizačního (optimalizačního) problému. Uveďte tři základní skupiny numerických metod pro řešení minimalizačních problémů. U každé vysvětlete základní myšlenku, na které je založena, a uveďte jednu konkrétní metodu.

Singulární rozklad a vlastní čísla

- Napište set-up pro problém vlastních čísel pro čtvercovou matici A a odvoďte mocninovou metodu bez normalizace. K čemu má tato metoda konvergovat?

  Na základě odvození ukažte, jaké jsou nutné podmínky pro konvergenci této metody a vysvětlete, jak, kde a proč se v praxi přidává normalizace. Jak byste tuto metodu upravili, aby aproximovala vlastní pár nejbližší danému komplexnímu číslu $z = \alpha + i\beta$?

  Uveďte větu, která charakterizuje nejlepší aproximaci dané matice A (rozměru $m \times n$) maticí hodnosti r.

  Vysvětlete alespoň jeden praktický příklad, ve kterém se tento výsledek běžně používá, a přesně popište matematický zápis tohoto problému a jak se na tento matematický problém použije vámi formulovaná věta.