

PA081: Programování numerických výpočtů

7. Systémy lineárních rovnic

Aleš Křenek

jaro 2011

Motivace

- ▶ hledáme řešení systému

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1N}x_N = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2N}x_N = b_2$$

...

$$a_{M1}x_1 + a_{M2}x_2 + \dots + a_{MN}x_N = b_M$$

- ▶ maticové vyjádření

$$\mathbf{Ax} = \mathbf{b}$$

- ▶ součást metod řešení nelineárních rovnic, optimalizací atd.
- ▶ diferenciální rovnice
 - ▶ simulace dynamických systémů
 - ▶ metoda konečných prvků
- ▶ realistické osvětlení scény (radiosita)
- ▶ a mnoho dalších ...

Motivace

Přehled metod

Gaussova
eliminaceLU
dekompoziceIterační
zpřesnění

Řídké matice

Iterační
metody

- ▶ založena na Newtonově zákoně $F = ma$
- ▶ vede na systém diferenciálních rovnic (13 proměnných)

$$\frac{d\mathbf{y}(t)}{dt} = \begin{pmatrix} d\mathbf{x}/dt \\ d\mathbf{q}/dt \\ d\mathbf{P}/dt \\ d\mathbf{L}/dt \end{pmatrix} = \begin{pmatrix} \frac{1}{m}\mathbf{P} \\ \frac{1}{2}\omega_q\mathbf{q} \\ \mathbf{F} \\ \mathbf{T} \end{pmatrix}$$

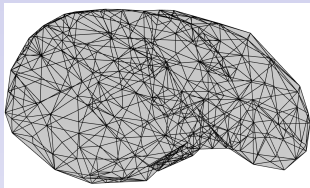
- ▶ pro simulaci potřebujeme opakovaně řešit

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t \left. \frac{d\mathbf{y}}{dt} \right|_{\mathbf{y}_n}$$

- ▶ triková aproximace - zanedbání vyšších členů Taylorova rozvoje dy/dt jako funkce \mathbf{y}
- ▶ konečný krok od \mathbf{y}_{n-1} k \mathbf{y}_n znamená řešení systému 13 lineárních rovnic

- ▶ Interakce s měkkými tkáněmi
 - ▶ trénink chirurgů - vytvořit simulaci chování tkání
 - ▶ potřebujeme vytvořit matematický model tkáně
 - ▶ s touto tkání pak můžeme interagovat (např. hapticky)
 - ▶ je třeba simulovat chování tkáně s dostatečnou přesností
- ▶ použité matematické modely
 - ▶ deformovatelná tělesa
 - ▶ chování je popsáno parciálními diferenciálními rovnicemi, zpravidla neznáme analytické řešení
 - ▶ řešíme numericky pomocí diskretizace metodou konečných prvků (FEM)

- ▶ simulovaná tkáň je geometricky aproximována meshem



- ▶ deformace popisuje teorie elasticity
- ▶ geometricky lineární model – systém lineárních rovnic
- ▶ geometricky nelineární model – systém nelineárních rovnic
 - ▶ v každém kroku iterační metody systém lineárních rovnic
- ▶ systémy rovnic jsou řídké (ovlivňují se jen přímo spojené uzly)

- ▶ pro vývoj filtrů rekonstruujících data z ultrazvuku je výhodné mít počítačový model přístroje a vyšetřovaného objektu
- ▶ snáze pak ladíme metody rekonstrukce, nevnášíme nepřesnost danou nedokonalým modelem vyšetřovaného objektu či nedokonalým přístrojem
- ▶ pro výpočet šíření vln v objektu použijeme FEM
 - ▶ na vlnovou délku ultrazvuku potřebujeme několik lineárních elementů
 - ▶ velikost elementu je pak v desetinách mm
 - ▶ vyšetřovaný objekt má velikost v jednotkách až desítkách centimetrech
- ▶ systémy o jednotkách až desítkách miliónů rovnic
 - ▶ vysoké nároky na výpočetní kapacitu
 - ▶ vysoké paměťové nároky

Klasifikace problémů

- ▶ $M = N$ - dobře podmíněné systémy
 - ▶ naděje na jednoznačné řešení (je-li A regulární)
- ▶ $M < N$ - nedostatečně podmíněné systémy
 - ▶ žádné řešení
 - ▶ nekonečně mnoho řešení ($N - M$ -rozměrný prostor)
- ▶ $M > N$ - příliš podmíněné systémy
 - ▶ přesné řešení zpravidla neexistuje
 - ▶ hledáme „nejlepší kompromis“

Klasifikace problémů

- ▶ $M = N$ - dobře podmíněné systémy
 - ▶ naděje na jednoznačné řešení (je-li A regulární)
- ▶ $M < N$ - nedostatečně podmíněné systémy
 - ▶ žádné řešení
 - ▶ nekonečně mnoho řešení ($N - M$ -rozměrný prostor)
- ▶ $M > N$ - příliš podmíněné systémy
 - ▶ přesné řešení zpravidla neexistuje
 - ▶ hledáme „nejlepší kompromis“
- ▶ husté vs. řídké
 - ▶ $O(N^2)$ vs. $O(N)$ nenulových prvků

Klasifikace problémů

- ▶ $M = N$ - dobře podmíněné systémy
 - ▶ naděje na jednoznačné řešení (je-li A regulární)
- ▶ $M < N$ - nedostatečně podmíněné systémy
 - ▶ žádné řešení
 - ▶ nekonečně mnoho řešení ($N - M$ -rozměrný prostor)
- ▶ $M > N$ - příliš podmíněné systémy
 - ▶ přesné řešení zpravidla neexistuje
 - ▶ hledáme „nejlepší kompromis“
- ▶ husté vs. řídké
 - ▶ $O(N^2)$ vs. $O(N)$ nenulových prvků
- ▶ velikost - dopady kumulace chyby
 - ▶ $N \sim 50$ - zpravidla stačí `float`
 - ▶ $N \sim 200-500$ - `double`
 - ▶ větší - vyžadují speciální metody

- ▶ přímé
 - ▶ daný algoritmus, vždy stejný počet operací
 - ▶ nemusí fungovat dobře pro „skoro singulární“ nebo příliš velké systémy
 - ▶ preferované pro „normální“ problémy
 - ▶ Gaussova eliminace, LU dekompozice, ...
- ▶ iterační
 - ▶ postupně vylepšované řešení
 - ▶ dosažení kritéria konvergence
 - ▶ různá náročnost pro různé vstupy
 - ▶ Jacobi, Gauss-Seidel, sdružené směry ...

- ▶ specializované metody pro řídké matice
 - ▶ pro různé vzory řídkých matic
 - ▶ výrazně efektivnější, jediná možnost řešení velkých systémů
 - ▶ přímé i iterační
- ▶ metody pro singulární systémy
 - ▶ analyticky i numericky („skoro“) singulární
 - ▶ nalezení celého prostoru řešení
 - ▶ standardní metody zhavarují
 - ▶ dekompozice na singulární hodnoty (v příští přednášce)
- ▶ řešení příliš podmíněných systémů
 - ▶ specializované metody nejmenších čtverců

- ▶ zpravidla sáhneme k hotové knihovně
 - ▶ nejsme první, kdo tento problém řeší
 - ▶ implementace optimalizované pro různé případy
 - ▶ k volbě vhodné metody je třeba rámcově rozumět jejich principům
- ▶ Numerical Recipes in C
 - ▶ hlavní zdroj pro tuto přednášku
 - ▶ jednoduché přehledné implementace
- ▶ LAPACK <http://www.netlib.org/lapack/>
 - ▶ plnohodnotná volně dostupná knihovna
 - ▶ využívá nízkoúrovňové knihovny BLAS, zpravidla implementované strojově závisle
 - ▶ obecné a specializované funkce pro různé typické případy
- ▶ NAG <http://www.nag.co.uk/>
 - ▶ rozsáhlá komerční numerická knihovna
- ▶ a další ...

Motivace

Přehled metod

Gaussova
eliminace

LU
dekompozice

Iterační
zpřesnění

Řídké matice

Iterační
metody

Gaussova eliminace

Základní postup

- ▶ standardní „školní“ metoda
 - ▶ řešení systému se nezmění, nahradíme-li libovolný řádek \mathbf{A} a odpovídající prvek \mathbf{b} lineární kombinací tohoto řádku s libovolným dalším
- ▶ vynulujeme a_{21}, \dots, a_{M1} odečtením $\frac{a_{i1}}{a_{11}}$ násobku prvního řádku
- ▶ obdobně pokračujeme druhým sloupcem od a_{32}
- ▶ výsledkem je matice s vynulovanými prvky pod diagonálou
- ▶ řešení systému získáme **zpětnou substitucí**
 - ▶ poslední rovnice $a_{MM}x_M = b_M$ je triviální
 - ▶ do předposlední dosadíme získanou hodnotu x_M atd.

- ▶ diagonální prvek a_{kk} je 0 nebo příliš malý
 - ▶ v k -té iteraci se jím dělí
 - ▶ dojde k přetečení
- ▶ při odečítání dochází k přílišné ztrátě platných cifer

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{bmatrix}$$

zaokrouhlení může vést až k $a_{22} = -\frac{1}{\epsilon}$, to odpovídá původní matici

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}$$

- ▶ metoda v této podobě je numericky nestabilní

Gaussova eliminace

Pivoting

- ▶ další tvrzení o systému
 - ▶ řešení systému se nezmění výměnou libovolné dvojice řádků
 - ▶ řešení systému se nezmění výměnou libovolné dvojice sloupců, zaměníme-li zároveň příslušné komponenty vektoru \mathbf{x}
- ▶ **pivot** je prvek, který po aplikaci těchto kroků použijeme k dělení při eliminaci k -tého sloupce
- ▶ **částečný pivoting** (parciální)
 - ▶ v iteraci k vybíráme z a_{jk} pro $j \geq k$, tj. jen z nulovaného sloupce
- ▶ **plný pivoting**
 - ▶ vybíráme z a_{ij} pro $i, j \geq k$, tj. z celé podmatice doprava a dolů
 - ▶ vyžaduje udržovat vznikající permutaci proměnných

Motivace

Přehled metod

Gaussova
eliminaceLU
dekompoziceIterační
zpřesnění

Řídké matice

Iterační
metody

Gaussova eliminace

Pivoting

- ▶ za pivota volíme maximum z kandidátů
 - ▶ pro všechny faktory v eliminačních krocích platí $|\frac{a_{ik}}{a_{kk}}| \leq 1$
- ▶ parciální i plný pivoting jsou pak numericky stabilní
- ▶ přínos plného pivotingu je jen okrajový

- ▶ za pivota volíme maximum z kandidátů
 - ▶ pro všechny faktory v eliminačních krocích platí $|\frac{a_{ik}}{a_{kk}}| \leq 1$
- ▶ parciální i plný pivoting jsou pak numericky stabilní
- ▶ přínos plného pivotingu je jen okrajový
- ▶ volba pivota závisí na řádu koeficientů původního systému
 - ▶ vynásobení jedné rovnice faktorem 10^{10} ...
 - ▶ za pivota lze volit koeficient, který by byl největší, kdyby byl celý původní systém normalizovaný, tj. největší koeficient všech rovnic roven 1
 - ▶ vyžaduje dodatečnou údržbu faktorů škálování, může přinést větší robustnost

Gaussova eliminace

Gauss-Jordanova eliminace

- ▶ Gaussovu metodu lze použít pro více pravých stran současně
- ▶ speciálně pro N bázových vektorů dostaneme výpočet matice \mathbf{A}^{-1}
- ▶ rozšíření – Gauss-Jordanova eliminace
 - ▶ v každé iteraci eliminujeme prvky nad i pod diagonálou \mathbf{A}
 - ▶ výsledkem je jednotková matice
 - ▶ řešení systému je přímo modifikovaný vektor \mathbf{b}
 - ▶ resp. dostáváme \mathbf{A}^{-1}
- ▶ srovnatelné se základní implementací
 - ▶ reálně provádíme tytéž operace

Gaussova eliminace

Výhody a nevýhody

- ▶ jednoduchá, dobře pochopitelná a stabilní metoda
- ▶ je třeba znát pravé strany dopředu
 - ▶ jsou součástí celého výpočtu
- ▶ výpočet \mathbf{A}^{-1} je zatížen poměrně velkou numerickou chybou
 - ▶ přímý výpočet řešení systému $\mathbf{Ax} = \mathbf{b}$ je přesnější než výpočet \mathbf{A}^{-1} a následné $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

LU dekompozice

Princip

- ▶ předpokládejme, že dokážeme rozložit $\mathbf{A} = \mathbf{LU}$ tak, že
 - ▶ \mathbf{L} je spodní trojúhelníková matice (prvky nad diagonálou jsou nulové)
 - ▶ \mathbf{U} je horní trojúhelníková matice (prvky pod diagonálou jsou nulové)
- ▶ potom lze psát

$$\mathbf{Ax} = (\mathbf{LU})\mathbf{x} = \mathbf{L}(\mathbf{Ux}) = \mathbf{b}$$

- ▶ původní systém lze vyřešit postupným řešením

$$\mathbf{Ly} = \mathbf{b} \quad \text{a} \quad \mathbf{Ux} = \mathbf{y}$$

- ▶ to je triviální dopřednou a zpětnou substitucí
 - ▶ viz Gaussova metoda

- ▶ prvky rozkladu $\mathbf{A} = \mathbf{LU}$ lze, s ohledem na nuly psát

$$i < j: a_{ij} = u_{i1}l_{1j} + u_{i2}l_{2j} + \dots + u_{ii}l_{ij}$$

$$i = j: a_{ij} = u_{i1}l_{1j} + u_{i2}l_{2j} + \dots + u_{ii}l_{jj}$$

$$i > j: a_{ij} = u_{i1}l_{1j} + u_{i2}l_{2j} + \dots + u_{ij}l_{jj}$$

- ▶ je to systém N^2 rovnic v $N^2 + N$ neznámých
 - ▶ diagonála je pokryta u i l
 - ▶ můžeme tedy volit $l_{ii} = 1$

LU dekompozice

Croutův algoritmus

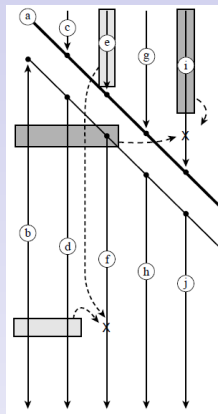
- ▶ postupně pro $j = 1, 2, \dots, N$:
 - ▶ pro $i = 1, 2, \dots, j$ spočítáme na základě uvedených rovnic

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$$

- ▶ pro $i = j + 1, j + 2, \dots, N$

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right)$$

- ▶ postup vždy využívá dříve spočtené prvky
- ▶ k numerické stabilitě je třeba navíc dodat pivoting l_{jj}



Motivace

Přehled metod

Gaussova
eliminace

LU
dekompozice

Iterační
zpřesnění

Řídké matice

Iterační
metody

- ▶ řešení lineárních rovnic pro libovolný počet \mathbf{b}
 - ▶ Croutův algoritmus $O(N^3)$
 - ▶ dopředná a zpětná substituce $O(N^2)$
 - ▶ všechna \mathbf{b} není třeba znát dopředu - hlavní přednost metody
- ▶ výpočet inverzní matice
 - ▶ řešení systému pro $\mathbf{b} =$ bázové vektory
 - ▶ potřebujeme-li počítat $\mathbf{A}^{-1}\mathbf{B}$, je lepší přímo pro sloupce \mathbf{B} než explicitní vyjádření \mathbf{A}^{-1}

Iterační zpřesnění

- ▶ i přímé metody řešení lineárních rovnic ztrácí přesnost
 - ▶ v závislosti na velikosti systému a poměru koeficientů
 - ▶ kumulace chyb pocházejících ze sčítání/odčítání
 - ▶ v optimistickém případě 2-3 platná místa
 - ▶ u „skoro singulárních“ matic podstatně horší
- ▶ lze vylepšit iteračním procesem
- ▶ \mathbf{x} je přesné řešení $\mathbf{Ax} = \mathbf{b}$, známe ale jen $\mathbf{x} + \delta\mathbf{x}$; položíme

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$$

odečtením dostaneme

$$\mathbf{A}\delta\mathbf{x} = \delta\mathbf{b} = \mathbf{A}(\mathbf{x} + \delta\mathbf{x}) - \mathbf{b}$$

pravou stranu umíme spočítat, řešíme nový systém rovnic pro $\delta\mathbf{x}$

- ▶ pro výpočet pravé strany je nutná **vyšší přesnost** odčítání
- ▶ s výhodou využijeme recyklaci LU dekompozice
- ▶ výsledným $\delta\mathbf{x}$ korigujeme původní \mathbf{x}

- ▶ typicky rozsáhlé problémy (miliony rovnic)
 - ▶ ale počet nenulových koeficientů je malý, zpravidla $O(N)$
- ▶ v extrémním případě neřešitelné standardními metodami
 - ▶ příliš velká paměťová a časová náročnost
 - ▶ de-facto nepřekonatelné problémy s přesností výpočtu
- ▶ specializované metody
 - ▶ využívají nulových koeficientů
 - ▶ vyžadují jen $O(N)$ operací i paměti
 - ▶ závislé na konkrétním vzoru nenulových prvků
 - ▶ např. LU dekompozice tridiagonální matice – jen N prvkový vektor navíc a 2 cykly o $N - 1$ iteracích
- ▶ v některých případech aproximační
 - ▶ nenulové prvky se během řešení „množí“ nad míru
 - ▶ je třeba některé zanedbávat

Motivace

Přehled metod

Gaussova
eliminace

LU
dekompozice

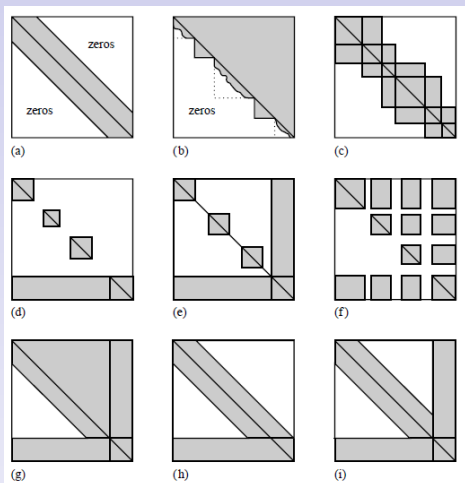
Iterační
zpřesnění

Řídké matice

Iterační
metody

Řídké matice

Některé speciální vzory



Motivace

Přehled metod

Gaussova
eliminace

LU
dekompozice

Iterační
zpřesnění

Řídké matice

Iterační
metody

Řídké matice

Uložení v paměti

- ▶ problematická není jen výpočetní náročnost, ale zejména nároky na paměť
 - ▶ $N = 10^6$ by vyžadovalo ve floatech 4 TB
- ▶ existují různá schémata, např.
 - ▶ pole hodnot `float val[]` a indexů `int idx[]`
 - ▶ prvních N prvků `val[]` jsou všechny diagonální prvky, zpravidla bývají nenulové
 - ▶ prvních N prvků `idx[]` jsou indexy ve `val[]`, kde jsou uloženy první nenulové nediagonální prvky jednotlivých řádků matice
 - ▶ `idx[N + 1]` ukazuje za poslední prvek `val[]`
 - ▶ další prvky `val[]` jsou nenulové nediagonální prvky matice uspořádané po řádcích a sloupcích
 - ▶ další prvky `idx[]` jsou indexy sloupců odpovídajících prvků `val[]`

Řídké matice

Uložení v paměti

- ▶ původní matice

$$\begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 7 & 5 & 9 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 6 & 5 \end{bmatrix}$$

- ▶ je reprezentována

	1	2	3	4	5	6	7	8	9	10	11
<code>idx[]</code>	7	8	8	10	11	12	3	2	4	5	4
<code>val[]</code>	3	4	5	0	5	n/a	1	7	9	2	6

Řídké matice

Sherman-Morrisonův postup

- ▶ řídké matice, které se „trochu liší“ od známého vzoru
 - ▶ navíc řádek, sloupec apod.
 - ▶ obecně $\mathbf{A}' = \mathbf{A} + (\mathbf{u} \otimes \mathbf{v})$
- ▶ lze ukázat

$$(\mathbf{A}')^{-1} = (\mathbf{A} + (\mathbf{u} \otimes \mathbf{v}))^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{u}) \otimes (\mathbf{v}\mathbf{A}^{-1})}{1 + \mathbf{v}\mathbf{A}^{-1}\mathbf{u}}$$

- ▶ některou z hotových metod vypočteme \mathbf{A}^{-1}
- ▶ aplikací vzorce získáme $(\mathbf{A}')^{-1}$
 - ▶ je-li \mathbf{A}^{-1} řídká v řádu $O(N)$, náročnost celé metody je $O(N)$
- ▶ postup lze opakovat pro různá \mathbf{u}, \mathbf{v}
- ▶ existují další zobecnění (Woodbury, viz literatura)

- ▶ obecně méně přesné než přímé metody
- ▶ řešení řídkých systémů
 - ▶ neexistuje přímá metoda pro konkrétní vzor
 - ▶ výpočet iteračního kroju je zpravidla $O(N)$
 - ▶ nevyžaduje další paměť
- ▶ téměř singulární systémy
 - ▶ přímé metody ztrácí přesnost kumulací chyby

Iterační metody

Prostá iterace

- ▶ rovnici $\mathbf{Ax} = \mathbf{b}$ převedeme na tvar $\mathbf{x} = \mathbf{A}'\mathbf{x} + \mathbf{b}'$
- ▶ opakovaně počítáme iterační krok
- ▶ kritérium konvergence
 - ▶ zobrazení $\mathbf{x} \mapsto \mathbf{A}'\mathbf{x} + \mathbf{b}'$ musí být kontrakce
 - ▶ stejné jako u nelineárních rovnic
- ▶ naplnění předpokladů věty o pevném bodě
 - ▶ $\lim_{k \rightarrow \infty} |(\mathbf{A}')^k| = 0$ pro nějakou maticovou normu
 - ▶ Frobeniova norma

$$|\mathbf{A}| = \sqrt{\sum_{i,j} a_{ij}^2}$$

- ▶ spektrální norma

$$|\mathbf{A}| = \rho(\mathbf{A}^T \mathbf{A})$$

kde $\rho()$ je maximální absolutní hodnota vlastních hodnot matice

- ▶ maximální součet sloupce nebo řádku

Motivace

Přehled metod

Gaussova
eliminaceLU
dekompoziceIterační
zpřesnění

Řídké matice

Iterační
metody

Iterační metody

Prostá iterace - příklady konkrétních metod

- ▶ Jacobi: z rozkladu $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$ dostaneme $\mathbf{x} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}$, tj.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right)$$

- ▶ Gauss-Seidel: $\mathbf{x} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}\mathbf{x} + (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}\mathbf{b}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$$

Ize recyklovat uložení \mathbf{x}

- ▶ konvergence není zaručena automaticky
 - ▶ pro konkrétní \mathbf{A} některé metody konvergují, jiné ne
 - ▶ specifická kritéria viz literatura

Motivace

Přehled metod

Gaussova
eliminaceLU
dekompoziceIterační
zpřesnění

Řídké matice

Iterační
metody

Iterační metody

Metoda konjugovaných směrů

- ▶ lze využít i metody řešení nelineárních rovnic
 - ▶ smysl to má jen ve velmi specifických případech
 - ▶ výjimkou je metoda konjugovaných gradientů
- ▶ minimalizujeme funkci

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x} \mathbf{A} \mathbf{x} - \mathbf{b} \mathbf{x}$$

- ▶ v minimu je její gradient nulový, tj.

$$0 = \nabla f = \mathbf{A} \mathbf{x} - \mathbf{b}$$

tedy minimum f přesně odpovídá řešení $\mathbf{A} \mathbf{x} = \mathbf{b}$

- ▶ metoda tedy najde minimum po N iteracích
 - ▶ f je kvadratická forma, viz minulá přednáška
- ▶ takto jednoduše funguje jen pro pozitivně definitní \mathbf{A} , lze rozšířit
- ▶ při výpočtu je třeba jen násobit $\mathbf{A} \mathbf{x}$
 - ▶ to lze u řídkých matic velmi efektivně