(Trefethen-Bau, Lekcia 31)

Výpočet SVD rozkladu ľubovoľnej matice sa dá zredukovať na výpočet spektrálneho rozkladu štvorcovej hermitovskej matice, ale najviac nákladný sa spôsob nie je stabilný. Treba použiť inú redukciu na SVD, (implicitne) čo vedie k tzv. unitárnej bidiagonalizácii v prvej fáze.

SVD matice A a vl. hodnoty A^*A

Singulárny rozklad matice $A_{m\times n} = U\Sigma V^*$ $(m \geq n)$

súvisí so spektrálnym rozkladom matice A^*A

$$A^*A = V\Sigma^*\Sigma V^*$$

z toho by sme (matematicky) vedeli vypočítať SVD nasledovne:

1. vytvoriť A^*A
2. nájsť spektrálny rozklad $A^*A = V\Delta V^*$
3. Σ_1 je $n\times n$ nezáporná diagonálna odmocnina z Δ.
4. nájsť riešenie $U\Sigma_1 = AV$, kde U je unitárna.

Takýto algoritmus sa používa (aj ja sám pri výpočte SVD rôznych pre vedecké odhady, ...), ale ako píšu v knihe — používajú ho "hlúpi ľudia čo si samo-objavili SVD (resp. na písomke MC) pre seba".

Matica A^*A - kovariančná matica A, resp. Gramova matica má známe využitie v štatistike a inde.

Takýto algoritmus je však nestabilný, lebo

pridružený problém nl. hodnôt je citlivý na perturbácie.

Problém je v nasledovnom:

Ak perturbujeme hermitovskú A^*A o δB, potom

$$\left| \lambda_k(A^*A + \delta B) - \lambda_k(A^*A) \right| \leq \|\delta B\|_2$$

Ako vidíme neskôr, podobný vzťah platí aj pre singulárne hodnoty

$$\left| \sigma_k(A + \delta A) - \sigma_k(A) \right| \leq \|\delta A\|_2 .$$

Späte stabilný algoritmus pre singulárne hodnoty by dával:

$$\tilde{\sigma}_k = \sigma_k(A + \delta A) \quad \text{pre} \quad \frac{\|\delta A\|}{\|A\|} = O(\varepsilon_{machine})$$

teda $\quad |\tilde{\sigma}_k - \sigma_k| = O(\varepsilon_{machine} \|A\|)$.

Ak by sme však počítali (stabilne) $\lambda_k(A^*A)$, máme:

$$|\tilde{\lambda}_k - \lambda_k| = O(\varepsilon_{machine} \|A^*A\|) = O(\varepsilon_{machine} \|A\|^2)$$

keďže $\sigma_k = \sqrt{\lambda_k}$, tak

$$|\tilde{\sigma}_k - \sigma_k| = O\left(|\tilde{\lambda}_k - \lambda_k| / \sqrt{\lambda_k}\right) = O\left(\varepsilon_{machine} \|A\|^2 / \sigma_k\right)$$

čiže vzdiel je $O(\|A\| / \sigma_k)$.

Pre veľké σ_k $(\approx \|A\|)$ to problém nie je,

pre $\sigma_k \ll \|A\|$ však áno. Pre najmenšiu σ_n máme

totiž $\frac{1}{\sigma_n} = \|A^{-1}\|$, teda $\|A\|^2 / \sigma_n$ bude $\|A\| \cdot \|A^{-1}\| = \kappa(A)$

– teda máme stratu presnosti na úrovni $\kappa(A)$.

Čo zodpovedá "štvorcu čísla podmienenosti" pre normálne rovnice pri najmenších štvorcoch

$$Ax = b \quad \sim \kappa(A)$$
$$A^*Ax = A^*b \quad \sim \kappa(A)^2 .$$

Iná redukcia na hermitovskú maticu

Existuje však iný, stabilnejší, spôsob, ako previesť SVD rozklad na problém spektrálneho rozkladu:
pre jednoduchosť, nech $m = n$ a A je štvorcová.
(to nebude problém, ako uvidíme neskôr).

Uvažujme $2m \times 2m$ hermitovskú maticu:

$$H = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}.$$

Keďže $A = U\Sigma V^*$, tak $AV = U\Sigma$ a aj $A^*U = V\Sigma^* = V\Sigma$,

Teda

$$\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}\begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix}\begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}.$$

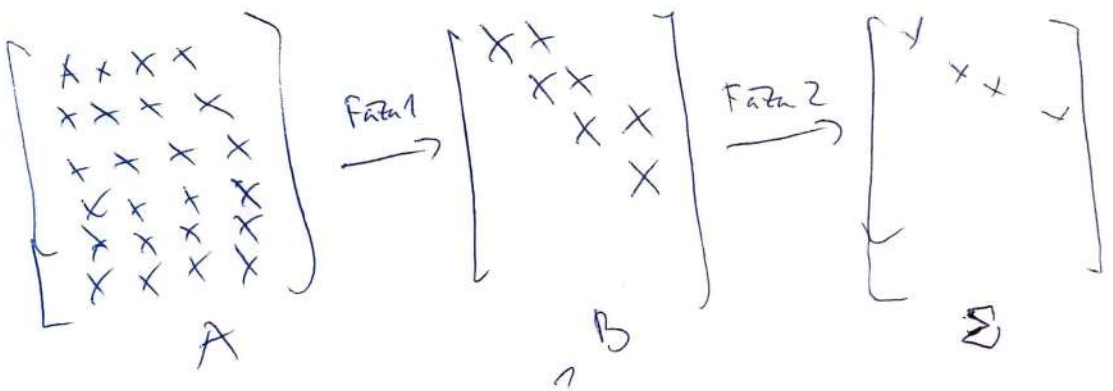
Lenže to je spektrálny rozklad matice H.
→ singulárne hodnoty A sú teda abs. hodnoty matice H, sčítky.
vektory A sa dajú zistiť z vl. ~~vektorov~~ vektorov pre H.

SVD rozklad A teda vieme získať prechodom k matici H a hľadaním jej spektrálneho rozkladu. Toto, na rozdiel od počítania s A^*A bude stabilné.
Štandardné SVD algoritmy však s maticou H pracujú len implicitne — čiže žiadne matice typu $(m+n)\times(m+n)$ sa nevoria.
Proces sa chce vykonať rýchlo → vo fáze 1 sa spraví redukcia na bidiagonálny tvar.

Dve fázy

podobne ako pre problém vl. hodnôt hermitovskej matice sme mali dvojfázový výpočet, (redukcia na tri-diag. tvar, diagonalizácia tridiagonálnej).

Od 60-tych rokov (Golub, Kahan, etc) sa podobný dvoj-fázový prístup používa aj pre SVD.



Fáza 1 si vyžaduje konečný počet operácií — $O(mn^2)$ flopov.

Fáza 2 si (teoreticky) vyžaduje nekonečný počet operácií, ale vďaka superlineárnej konvergencii stačí

$O(n \log(|\log(\varepsilon_{\text{machine}})|))$ iterácií, t.j. $O(n)$

ak $\varepsilon_{\text{machine}}$ berieme ako konštantu, na účtovaní presnosti $\varepsilon_{\text{machine}}$.

vďaka bidiagonálnosti v každej iterácii potrebujeme len $O(n)$ flopov

– spolu vo fáze 2 $O(n^2)$ flopov, (ak sa počítajú len singulárne ~~vektory~~ hodnoty a nie vektory)

Teda Fáza 1 je konečná, fáza 2 v princípe nekonečná, ale v praxi typicky ovela lacnejšia, podobne ako pre problém vl. hodnôt v symetrickom prípade.

Golub-Kahan bidiagonalizácia

Vo fáze 1 sa A prevedie na bidiagonálnu maticu pomocou vhodných unitárnych operácií zľava a sprava.

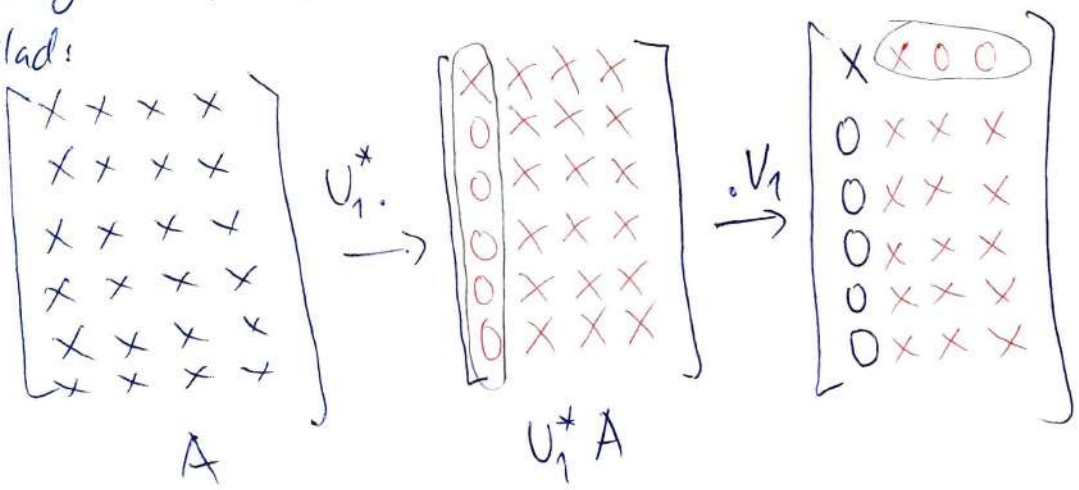
Na rozdiel od výpočtu vl. hodnôt, kde sme potrebovali rovnaké unitárne operácie z oboch strán, tu stačia rôzne.

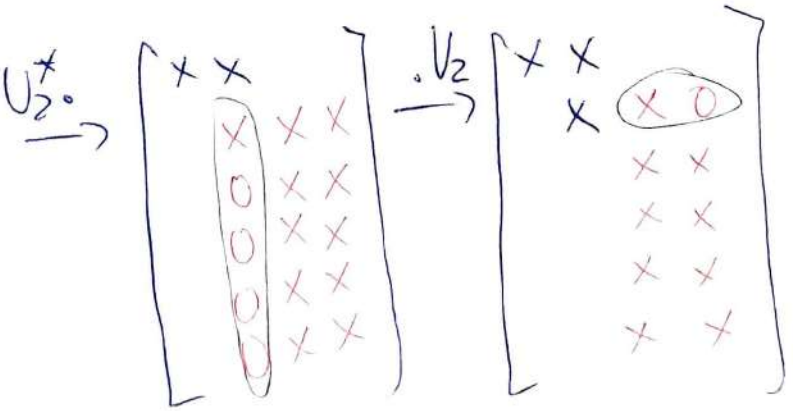
Takto vieme dostať nuly nielen na spodnej vedľajšej diagonále, ale aj nad prvou hornou vedľajšou.

Najjednoduchšie to ide pomocou Golub-Kahan bidiagonalizácie:

- Householderove reflektory sa striedavo používajú sprava a zľava. Každý ľavý reflektor vyrobí nuly pod diagonálou, zatiaľ čo pravé vyrobia nuly napravo od prvej hornej vedľajšej diagonály (ľubovoľnej).

Príklad:





Ľavé násobenie U_1^* mení riadky 1 až m, vyrobí nuly v prvom stĺpci pod diagonálou.

pravé násobenie V_1 mení stĺpce $2 \dots n$, vyrobí nuly v riadku 1, za bez toho aby pokazilo prvý stĺpec s nulami.

Atď - riadky $2 \dots m$, stĺpce $3 \dots n$, ...

V rámci tohto procesu sa použije n reflektorov zľava a $n-2$ sprava.

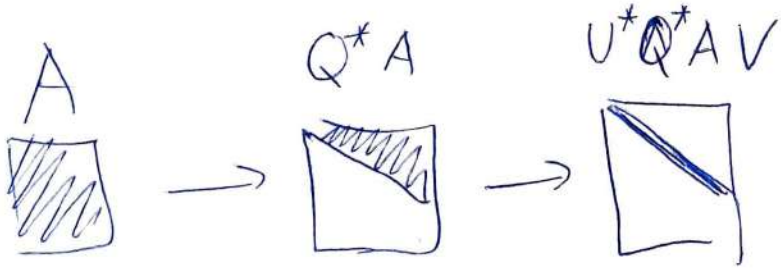
Pri analýze počtu flopov to teda vyzerá tak, akoby sme robili dve preplietené Householderovské QR redukcie na matici A typu $m \times n$ zľava a na matici $n \times m$ A^* (sprava)

$\rightarrow$ Lekcia 10 $\rightarrow$
počet flopov pre Golub-Kahan $\sim 4mn^2 - \frac{4}{3}n^3$ flopov

---

Fáza 1 sa dá urýchliť ...

Ak $m \gg n$, toto je možno príliš veľa. Väčšina núl sa totiž nachádza pod diagonálou a tie vieme docieliť QR rozkladom matice A.
(Lawson-Hanson + Chan): LHC - bidiagonalizácia



začne sa s QR rozkladom $A = QR$, a potom sa spraví Golub-Kahan bidiagonalizácia matice R, $B = U^*RV$.
- QR potrebuje $2mn^2 - \frac{2}{3}n^3$ flopov,
Golub-Kahan na $n \times n$ matici potom potrebuje $\frac{8}{3}n^3$ flopov.

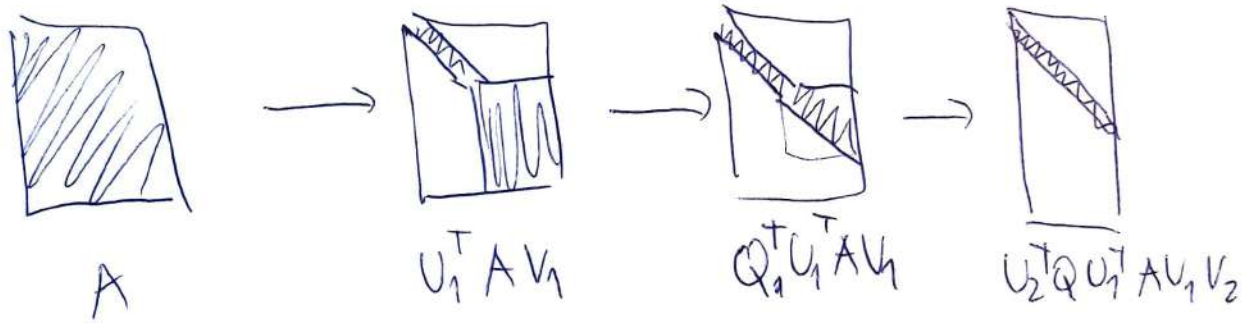
$\sim 2mn^2 + 2n^3$ ↑ LHC

Toto bude efektívnejšie ako Golub-Kahan ak $m > \frac{5}{3}n$.

Zapamätajme si, že LHC vyrobí nuly (pod diagonálou) a potom ich pokazí, keď vyrába nuly napravo od nej.

LHC sa oplatí iba ak $m > \frac{5}{3}n$, ale myšlienka sa dá použiť aj pre $m > n$. Trik spočíva v tom, že QR rozklad použijeme až v neskoršej fáze.

Začneme s Golub-Kahan, ktorý z $m \times n$ matice vyrobí $(m-k) \times (n-k)$ podmatica so správnym pomerom, potom sa prepneme do módu LHC



→ počet operácií bude minimálny pri pomere $\frac{m-k}{n-k} = 2$, potom táto tri-diagonalizácia bude stáť

$$\sim 4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m-n)^3 \text{ flopov.}$$

– mierne zlepšenie pre $n \leq m < 2n$.

Fáza 2

Tu sa robí iteratívny výpočet SVD pre bidiagonálnu maticu, podobný ako QR algoritmus (60-te – 90-te roky).

Neskôr nastupuje divide-and-conquer, ktorý pravdepodobne pre väčšie matice bude štandardný.. Bez detailov.

(Trefethen-Bau, Lekcia 32)

Prehľad iteratívnych metód

- v tejto kapitole sa presunieme tématicky od priamych metód výpočtov (ktoré sú pomerne ustálené) k iteratívnym metódam, kde je to ešte stále "divoké". Napriek tomu, toto je budúcnosť ...

Prečo iterovať?

Hlavný dôvod prečo uprednostniť iteratívne metódy je zložitosť. Neiteratívne, priame metódy typicky stoja $O(m^3)$ operácií. To je však pre matice (s $O(m^2)$ dlahým vstupom) možno privela.

Nehovoriac o tom, že m^3 je obrovské pre veľké m.

História "obrovskosti"

1950	$m = 20$	- toto sú príklady z dobrej literatúry pre "veľmi veľké" husté matice pre priame výpočty.
1965	$m = 200$	
1980	$m = 2000$	
1995	$m = 20000$	
2010 (?)	$m = ?$	

Ako sme už spomenuli pred pár týždňami, za 45 rokov sa rýchlosť zvýšila 10^3 krát, ale kapacita počítačov až 10^9 krát.

- čo ukazuje na "bottleneck" - úzky profil zložitosti $O(m^3)$ (10^{12}?) priamych výpočtov.

Ak by sme vedeli efektívne veci počítať $O(m^2)$, tak zväčšenie môžeme zrýchliť až 100 krát.

To, či sa dá $O(n^3)$ zložitosť porazit nie je jasné.
Pre náhodný maticový problém to asi nepôjde.
Treba sa však zamyslieť nad tým odkiaľ pochádzajú veľké maticové výpočty.

"malé matice", 3 až 30 môžu pochádzať z lokálnych interakcií 3 síl, či 30-tich subpopulácií – ťažko môžeme mať husto vyplnené bez nejakej štruktúry.

Veľké matice sa však vyskytujú nepliance – ako diskretizácia spojitého problému (PDR alebo integrálne rovnice)

– ak je n veľké, ide o aproximáciu nekonečna.
V tom prípade majú veľké matice jednoduchšiu štruktúru, ktorá sa dá využiť.

Zjavná štruktúra je <u>riedkosť</u> (opak <u>hustej</u>)
Napr. konečná diskretizácia pomocou diferencií pre PDR môže viesť k matici s $n = 10^5$ ale iba $\nu = 10$ nenulových zložiek v riadku:

To sa dá využiť pri iteratívnych metódach, ktoré využívajú maticové násobenie ako "čiernu skrinku"

$$x \longrightarrow \boxed{\begin{matrix}\text{BLACK}\\ \text{BOX}\end{matrix}} \longrightarrow Ax .$$

Teda oddelíme iteratívny algoritmus od násobenia maticou, ktoré bude riadené samostatnou procedúrou, ktorej detaily nemusíme poznať.

(niekedy treba poznať aj súčin A^*x)

Napr. pre riedku maticu sa dá nájsť metóda výpočtu Ax, ktorá potrebuje len $O(m)$ operácií namiesto $O(m^2)$.

Toto je veľký rozdiel oproti priamym metódam (Gaussova eliminácia / LU-rozklad, Householderova triangularizácia / QR-rozk.) ktoré pracujú priamo so zložkami matice A.
manipulujú

– fielo vyrábajú nuly, ale môžu pokaziť riedkosť.

Historicky, riedkosť bola dominantný druh štruktúry.

Pri súčasnom masívnom zbere dát sa však môže objavovať iný typ – ~~riedky~~ matice budú husté, ale dáta v nich budú mať repetetívny charakter – teda budú v sebe obsahovať istú regularitu.
Tá sa dá zachytiť pomocou Fourierovej transformácie, wavelet expansions či "multipole" methods.
-> potom "black box" metódy môžu byť dosť sofistikované...

Projekcie na Krylovove podpriestory

Myšlienka iteratívnych metód je založená na projekcii m-rozmerného problému na nižšerozmerný krylovov priestor.
Pre maticu A a vektor b bude postupnosť $b, Ab, A^2b, A^3b, \ldots$ ktorú vieme dostať pomocou cieľovej skúšky ako $b, A(b), A(Ab), A(A(Ab)), \ldots$
krylovova postupnosť.

Príslušné Krylovove podpriestory sú generované prvými k-členmi tejto postupnosti.

Budeme sa venovať týmto algoritmom:

$Ax = b$ $\quad$ $Ax = \lambda x$

V každej z týchto metód budeme pôvodný problém redukovať na maticové problémy dimenzií $n = 1, 2, 3, \ldots$

Pre Hermitovské matice A budeme mať redukované matice tridiagonálne, pre nehermitovské budú v Hessenbergovom tvare.

- Arnoldiho algoritmus počíta vl. hodnoty pomocou vl. hodnôt redukovanej matice vyšších Hessenbergových matíc väčších dimenzií.

Počet krokov, počet operácií v kroku, predpríprava (preconditioning)

Gaussova eliminácia, QR rozklad a iné algoritmy "hustej" lineárnej algebry majú zložitosť $O(n^3)$ — majú $O(n)$ krokov, každý s $O(n^2)$ veľa práce - operáciami.

Pre iteratívne metódy toto bude pravda, ale iba v najhorších prípadoch. Keď sú iteratívne metódy rýchlejšie, budeme vedieť redukovať obe tieto zložky.

- Počet krokov potrebných na konvergenciu bude, v širšom zmysle slova, závisieť od spektrálnych vlastností A

Napr. metóda združených gradientov pre riešenie kl. definitného hermitovského systému $Ax=b$ bude konvergovať rýchlo, ak sú vl. hodnoty zhrčené ďaleko od 0. (sú reálne, kladné...)

Podobne Lanczosov algoritmus dá niektoré vl. hodnoty reálnej hermitovskej matice, ak sú dobre oddelené od zvyšku spektra. (+ štartovací vektor je dostatočne generický).

-> uvidíme súvislosť S aproximáciami funkcií $f(z)$ polynómami $p(z)$ na častiach reálnej osi alebo komplexnej roviny.

Množstvo práce v jednom kroku závisí od štruktúry matice a ako vie túto štruktúru využiť čierna skrinka násobiaca $x \longmapsto Ax$.

Ideálna metóda vedúca počet krokov z $O(m)$ na $O(1)$ a prácu v krokoch z $O(m^2)$ na $O(m)$ (dĺžka vektora) - čím zníži celkovú náročnosť z $O(m^3)$ na $O(m)$.

Takéto zrýchlenie sa niekedy naozaj dá docieliť, ale typicky sa dosahuje zlepšenie $O(m^3) \longrightarrow O(m^2)$.

V 90-tych rokoch to znamenalo 10-násobné zrýchlenie oproti priamym algoritmom.

Pri zväčšovaní m však tento rozdiel bude narastať a význam iteratívnych algoritmov vzrastie:

Základný zákon Computer Science:

> čím rýchlejší počítač, tým väčší význam/dôležitosť rýchlosti algoritmu

Presné vs. približné riešenie

Iteratívne metódy sú približné v tom zmysle, že v princípe by na presný výsledok potrebovali nekonečne veľa krokov v presnej aritmetike. (bez zaokrúhľovacích chýb).

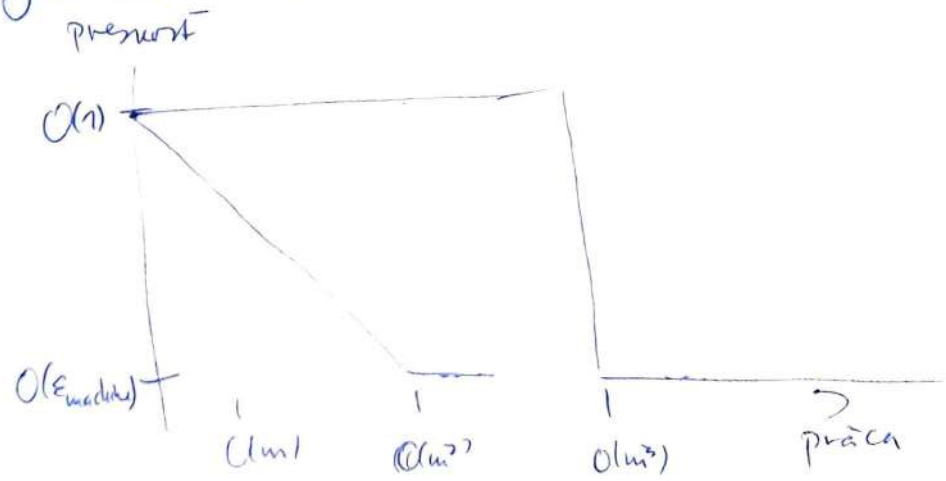
Toto zvyknú "začiatočníkov" rozladiť... lebo majú pocit, že iterovanie je len takáto inžinierska bez elegancie pochybnej spoľahlivosti. Chce to však 'prísť na chuť'!

Koniec-koncov, vo floating point aritmetike sú aj priame metódy len približné a najviac v čo najmenšie dúfať je strojová presnosť.

Keďže iteratívne metódy vedia takúto presnosť dosiahnuť, nebrelo sa robiť zbytočné stavosti.

A elegancia bude tiež ---

Obr



Konvergencia priamych a iteratívnych metód
– geometrická konvergencia pod presnosť ε-machine
vs. O(n³) počet krokov, keby výpočet dá niečo celočíselné.

Priame metódy lepšie ako $O(n^3)$

Treba povedať, že existujú priame algoritmy (konečné, v princípe presné), ktoré riešia $Ax=b$ a pritom robia problémy na menej ako $O(n^3)$ operácií.

Príklad: násobenie dvoch 2x2 matíc (typicky 8 operácií násobení)

ktoré potrebuje len 7 – 1969 Volker Strassen.

→ toto viedlo k zníženiu exponentu z 3 na $\log_2(7) \approx 2.81$

Následne: (Coppersmith-Winograd ≈ 2.376)

→ dôležité tieto algoritmy nie sú praktické, lebo buď rozdiel 3 a 2,81 je malý alebo veľkosť n, keď sa oplatí prejsť z $O(n^3)$ na $O(n^{2.376})$ je astronomická pre súčasné počítače.

Budúcnosť? Násobenie matíc za $O(n^2)$?

alebo riešenie $Ax=b$ za $n^2 \log(n)$?

"Fast inverse"?