

Moderní metody počítačové fyziky II (NEVF161)

MFF UK



Částicové modely plazmatu

Výpočet síly



Částicové modelování - úvod

Modelování pohybu N_p částic se započtením vzájemného silového působení

Lze interpretovat jako Monte-Carlo řešení Boltzmannovy rovnice - náhodné vzorkování rozdělovací funkce pomocí "částic,,

$$f(\mathbf{r}, \mathbf{v}, t) \rightarrow \sum_{i=1}^{N_p} \delta(\mathbf{r} - \mathbf{r}_i) \delta(\mathbf{v} - \mathbf{v}_i)$$

- ▶ Astrofyzika - pohyb hvězd v galaxii - gravitace
- ▶ Molekulární dynamika - atomy v molekulách / molekuly v látce – Lennard Jones a složitější potenciály
- ▶ Modelování plazmatu - nabitě částice v plazmatu - elektrostatičká interakce



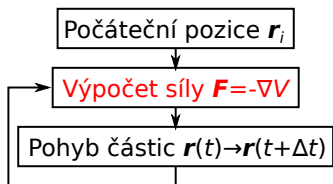
Metody výpočtu síly

Ekvivalentní výpočtu potenciálu: $\mathbf{F} = -\nabla V$

- ▶ Typicky dvoučásticový potenciál

$$V(\mathbf{r}) = \sum_{i=1}^{N_p} V_2(\mathbf{r} - \mathbf{r}_i)$$

- ▶ Možné i vícečásticové interakce - polarizace
- ▶ Obvykle mnoho částic (N_s) v jedné superčástici



- ▶ Speciálně pro elstat interakci, $V = q\phi$

$$\Delta\phi = -\rho/\epsilon_0 \quad (1)$$

- ▶ Cílem je efektivní řešení Poissonovy rovnice (1).



Klasifikace metod I

PP – particle-particle

- ▶ Přímý výpočet potenciálu resp. síly

$$\mathbf{F}_j = \sum_{i \neq j} \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ji}^2} \frac{\mathbf{r}_{ji}}{|\mathbf{r}_{ji}|}; \quad \mathbf{r}_{ji} = \mathbf{r}_i - \mathbf{r}_j$$

- ▶ Ekvivalentní řešení Poissonovy rce. metodou Greenových funkcí

$$\mathbf{F} = -q\nabla\phi; \quad \phi(\mathbf{r}) = \int G(\mathbf{r} - \mathbf{r}')\rho(\mathbf{r}')d\mathbf{r}'$$

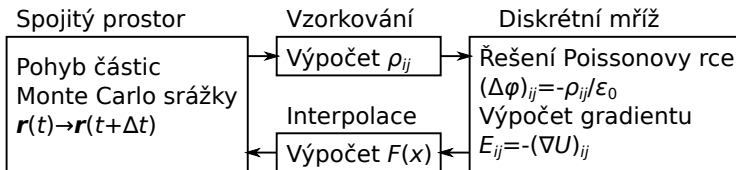
- ▶ Výpočetně náročné – $O(N_p^2)$
- ▶ Vhodné pro krátkodosahové síly $O(N_p)$



Klasifikace metod II

PM – particle-mesh

- ▶ Výpočet potenciálu na mříži metodami pro řešení PDE
- ▶ Metoda PIC – Particle-In-Cell



- ▶ Prostorové rozlišení určené váhovými funkcemi W
- ▶ Konvoluční metody



Klasifikace metod III

P3M – particle-particle—particle-mesh

- ▶ Ewaldova sumace
- ▶ Rozdělení interakce na krátkodosahovou a dalekodosahovou “hladkou,, složku

$$\mathbf{f} = \mathbf{f}^{\text{sr}} + \mathbf{f}^{\text{lr}}$$

- ▶ Krátkodosahová síla - stačí sečíst příspěvky částic v okolí
- ▶ Dalekodosahová síla - stačí sečíst nízkofrekvenční členy Fourierovy řady - hrubá mříž
- ▶ Příklad: Částice konečné velikosti



Klasifikace metod IV

Stromové (hierarchické) algoritmy

Barnes-Hut a rychlá multipólová metoda (FMM)

- ▶ Částice uloženy v listech stromové struktury
- ▶ Blízké částice jsou započítány přesně
- ▶ Vzdáledné částice jen přibližně ve skupinách - nadřazené uzly stromu
- ▶ Barnes-Hut - každý uzel má jenom "těžiště", $O(N_p \log(N_p))$
- ▶ FMM - multipólový rozvoj potenciálu v každém bodě $O(N_p)$



Stromový algoritmus Barnes-Hut



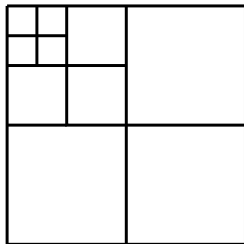
Barnes-Hut

Datová struktura

1D Binární strom

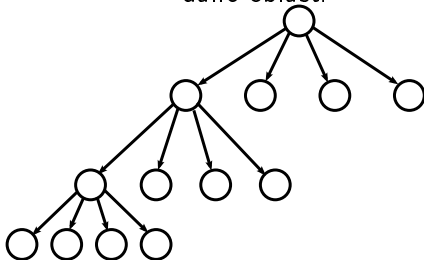
2D Quadtree

3D Octree



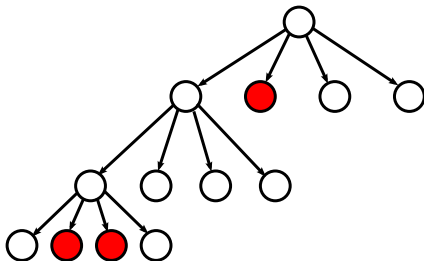
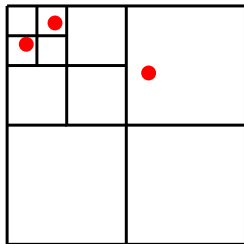
Příklad: 2D Barnes-Hut

- ▶ Kořen stromu - pracovní oblast
- ▶ 4 potomci uzlu - 4 kvadranty dané oblasti



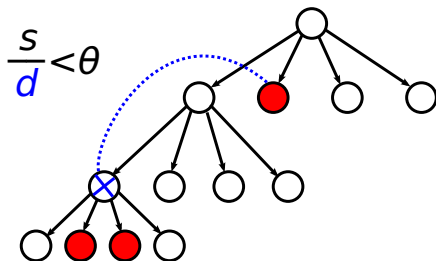
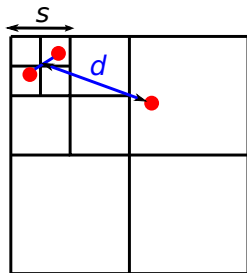
Konstrukce stromu

- ▶ Tvar stromu určen rozložením částic
- ▶ Každý list obsahuje maximálně jednu částici
- ▶ Každý uzel obsahuje informaci o své hmotnosti a poloze svého těžiště



Výpočet síly I

- ▶ Začíná se od kořenu
- ▶ Pokud Rozměr uzlu s / vzdálenost uzlu $d > \theta \Rightarrow$
 - Vypočti síly od jeho potomků
- ▶ Jinak započti sílu od těžiště uzlu



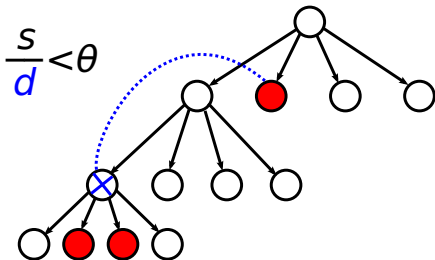
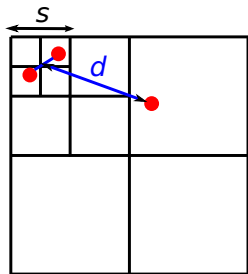
Výpočet síly II

Přesnost určena parametrem θ (opening angle):

- ▶ $\theta = 1$ originální Barnes-Hut
- ▶ $\theta \rightarrow 0$ konverguje k přímému výpočtu **PP** algoritmem

Při použití makročástic se používá “změkčení,, silové interakce - vyhlazování: 1 částice \sim hvězdokupa

- ▶ Potenciál $1/r \rightarrow 1/\sqrt{r^2 + \epsilon^2}$, částice o poloměru ϵ
- ▶ Zamezuje vzniku nefyzikálních “dvojhvězd,,
- ▶ Snižuje nefyzikální ohřev



- ▶ Přirozené jsou “otevřené,, okrajové podmínky: $\phi(\mathbf{x}) \xrightarrow{|\mathbf{x}| \rightarrow \infty} 0$
- ▶ Jiné okrajové podmínky - komplikované, ale možné s využitím fiktivních nábojů a dipólů na okrajích - boundary integral
- ▶ Výpočetní náročnost $O(N_p \log(N_p))$
- ▶ Rychlá multipólová metoda uchovává v každém uzlu víc informace - multipólový rozvoj potenciálu - náročnost $O(N_p)$



PM particle-mesh algorithmy (PIC)



Terminologie

- ▶ Dimenze mříže d
- ▶ Počet uzlů mříže $N_g = n^d$
- ▶ Vzdálenost uzlů h
- ▶ Jednotková matice I



Příklad: 1D PIC algoritmus pro výpočet síly

- ▶ Vzorkování $\rho(x) \rightarrow \rho_i$

$$\rho_i^n = \frac{1}{h} \sum_{k=1}^{N_p} q_k W_{\text{samp}}(x_k^n - x_i)$$

- ▶ Poissonova rovnice

$$\frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{h^2} = -\frac{\rho_i^n}{\epsilon_0}$$

$$E_i^n = -\frac{\phi_{i+1}^n - \phi_{i-1}^n}{h}$$

- ▶ Interpolace

$$F(x_k^n) = q_k \sum_{i=1}^{N_g} W_{\text{inter}}(x_k^n - x_i) E_i^n$$



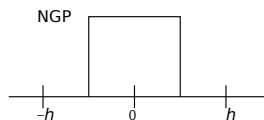
PM algoritmus - Váhové funkce

Hierarchie váhových funkcí $W = \Pi(x/h) * S(x)$

Obdélníková funkce $\Pi(x) = 1$ pro $x \in [-1/2, 1/2]$, jinak 0

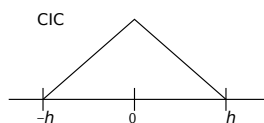
- ▶ Nearest grid point (NGP)

$$W(x) = \Pi\left(\frac{x}{h}\right)$$



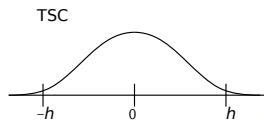
- ▶ Cloud in cell (CIC)

$$W(x) = \frac{1}{h} \Pi\left(\frac{x}{h}\right) * \Pi\left(\frac{x}{h}\right)$$



- ▶ Triangular shaped cloud (TSC)

$$W(x) = \frac{1}{h^2} \Pi\left(\frac{x}{h}\right) * \Pi\left(\frac{x}{h}\right) * \Pi\left(\frac{x}{h}\right)$$



PM algoritmus - Volba váhové funkce

Požadujeme zachování hybnosti: $F_{\text{self}} = 0$, $F_{12} = -F_{21} \Rightarrow$

- ▶ Stejné schéma pro interpolaci a vzorkování $W_{\text{samp}} = W_{\text{inter}}$
- ▶ Symetrické diferenční schéma pro aproximaci derivací
- ▶ Funkce vyšších řádů mají nižší rozlišení a menší šum

Alternativně - metoda konečných prvků (Minimalizace akce)

- ▶ Funkce vyššího řádu pro vzorkování než pro interpolaci
- ▶ Exaktní zachování energie, ale nezachovává hybnost
- ▶ Částice působí sama na sebe



Řešení Poissonovy rovnice

- ▶ Uvažujeme metodu konečných diferencí

$$\frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{h^2} = -\frac{\rho_i}{\epsilon_0}$$

- ▶ Lze zapsat jako $KU = F$

$$\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} = -\frac{h^2}{\epsilon_0} \begin{bmatrix} \rho_1 - \phi_0 \\ \rho_2 \\ \vdots \\ \rho_{N-1} \\ \rho_N - \phi_{N+1} \end{bmatrix}$$

- ▶ Ve 2D bloková matice K_{2D}

$$\begin{bmatrix} K-2I & I & & & \\ I & K-2I & I & & \\ & & \ddots & \ddots & \\ & & & I & K-2I & I \\ & & & I & K-2I \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}$$



Obecné metody:

Gaussova eliminace

- ▶ Použitelné pouze v 1D - Thomasův algoritmus $O(N_g)$
- ▶ Ve 2D - náročnost $O(N_g^2)$ - nevhodné

Relaxační metody

- ▶ Počáteční odhad ϕ^0
- ▶ Jacobiho metoda (J)

$$\phi_i^{n+1} = \frac{\phi_{i-1}^n + \phi_{i+1}^n}{2} + \frac{h^2 \rho_i}{2\epsilon_0}$$

- ▶ Gauss Seidelova metoda (GS)

$$\phi_i^{n+1} = \frac{\phi_{i-1}^{n+1} + \phi_{i+1}^n}{2} + \frac{h^2 \rho_i}{2\epsilon_0}$$



Princip relaxační metody

- ▶ Systém $KU = F$, zbytek $r_k = F - AU_k$, preconditioner $P \approx A$
- ▶ P takové, aby šlo invertovat, Jacobi - diagonála, GS - dolní trojúhelník

$$PU = (P - K)U + F$$

- ▶ Iterace (iterační matice $M = P^{-1}(P - K)$)

$$PU_{k+1} = (P - K)U_k + F$$

- ▶ Redukce chyby $e_k = U - U_k$:

$$Pe_{k+1} = (P - K)e_k$$

- ▶ Metoda konverguje, pokud spektrální poloměr $\rho(M) < 1$



Superrelaxační metoda (SOR) I

- ▶ Jacobi i GS jsou pomalé - počet iterací $\sim N_g^{2/d}$
- ▶ Successive OverRelaxation - SOR
- ▶ Lze urychlit relaxaci zvětšením relaxačního kroku faktorem ω

$$r = \frac{\phi_{i-1}^n - 2\phi_i^n + \phi_{i+1}^n}{2} + h^2 \frac{\rho_i}{2\epsilon_0}$$

$$\phi_i^{n+1} = \phi_i^n - \omega r$$

- ▶ Volba ω - Čebyševovo urychlení

$$\omega^{(0)} = 1$$

$$\omega^{(1/2)} = 1/(1 - \rho^2/2)$$

$$\omega^{(t+1/2)} = 1/(1 - \rho^2\omega^{(t)}/4)$$

$$\omega^{(\infty)} \rightarrow 2/(1 + \sqrt{1 - \rho^2})$$

- ▶ Šachovnicový vzor (odd-even ordering)
- ▶ Spektrální poloměr (J)
 $\rho = \cos(\pi/n)$
- ▶ Zabraňuje poč. překmitnutí



Superrelaxační metoda (SOR) II

- ▶ Potřebný počet iterací $\sim N_g^{1/d}$
- ▶ *By overcorrecting from x_k to x_{k+1} , hand calculators noticed that they could finish in a few weeks.*
- ▶ Stále pomalé, ale již použitelné pro 2D PIC, kde je dobrý poč. odhad.

Blokové metody

- ▶ Successive Line OverRelaxation - SLOR
- ▶ Namísto bodu řeší celý řádek (i několik řádků) najednou (Thomasův algoritmus)
- ▶ Urychlení konvergence o malý konstantní faktor ~ 2



Cyklická redukce

- ▶ Na rozdíl od předchozích metod využívá speciálních vlastností Poissonovy rovnice
- ▶ 1D příklad - rovnice pro 3 sousední body

$$\phi_{i-2} - 2\phi_{i-1} + \phi_i = F_{i-1} \quad (\text{a})$$

$$\phi_{i-1} - 2\phi_i + \phi_{i+1} = F_i \quad (\text{b})$$

$$\phi_i - 2\phi_{i+1} + \phi_{i+2} = F_{i+1} \quad (\text{c})$$

- ▶ Sečtením řádků (a) + 2 × (b) + (c) dostaneme

$$\phi_{i-2} - 2\phi_i + \phi_{i+2} = F_{i-1} + 2F_i + F_{i+1}$$

- ▶ Redukujeme množství rovnic na polovinu
- ▶ Redukované rovnice mají stejný tvar a redukci lze opakovat
- ▶ Náročnost $O(N_g \log(N_g))$



Cyklická redukce 2D

- ▶ Stejný postup lze aplikovat na blokovou matici ve 2D

$$\begin{bmatrix} A & I & & & \\ I & A & & & \\ & & \ddots & & \\ & & & A & I \\ & & & I & A \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix}; \quad A = K - 2I$$

- ▶ Sečtením řádků (a) $- A \cdot$ (b) + (c) dostaneme

$$U_{i-2} - (A^2 - 2I)U_i + U_{i+2} = F_{i-1} + AF_i + F_{i+1}$$

- ▶ Matice $(A^2 - 2I)$ už je 5-diagonální a dalšími iteracemi roste zaplnění
- ▶ Lze faktorizovat $(A^2 - 2I) = (A - \sqrt{2}I)(A + \sqrt{2}I)$



Odbočka - vlastní čísla matice K

- ▶ Analogicky ke spojitému případu v 1D
- ▶ 0 Dirichletovy o.p. vlastní funkce jsou \sin
- ▶ Okrajové podmínky splňují: $z_k(p) = \sin\left(\frac{pk\pi}{N+1}\right)$
- ▶ Lze ověřit a získat vlastní čísla dosazením do diferenčního vztahu:

$$z_k(p-1) - 2z_k(p) + z_k(p+1) = \sin\left(\frac{pk\pi}{N+1}\right) \left[2 \cos\left(\frac{k\pi}{N+1}\right) - 2 \right]$$

- ▶ Potom vlastní čísla

$$\lambda_k = 2 \cos\left(\frac{k\pi}{N+1}\right) - 2$$

- ▶ Normované vlastní vektory

$$z_k(p) = \sqrt{\frac{2}{N+1}} \sin\left(\frac{pk\pi}{N+1}\right)$$



Rozšíření do 2D

- ▶ Kroneckerův součin $A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix}$
- ▶ Matici 2. derivací ve 2D (na čtverci) lze zapsat jako $K_{2D} = K \otimes I + I \otimes K$
- ▶ Potom vlastní čísla

$$\lambda_{kl} = \lambda_k + \lambda_l$$

- ▶ Vlastní vektory

$$z_{kl}(p, q) = z_k(p)z_l(q)$$



Metoda Fourierovy transformace I

Ve spojitém případě:

$$\blacktriangleright \hat{f}(k) = \int_{\mathbb{R}} f(x) e^{-2\pi i k x} dx, \quad \frac{df}{dx} \rightarrow 2\pi i k \hat{f}(k)$$

\blacktriangleright Poissonova rce:

$$-4\pi^2(k_x^2 + k_y^2)\hat{\phi} = -\frac{\hat{\rho}}{\epsilon_0}$$

Funguje to i v diskrétním případě? DFT

\blacktriangleright Potenciál lze rozložit do Fourierovy ř.:

$$\phi_p = \frac{1}{N} \sum_{k=1}^n \hat{\phi}_k e^{2\pi i p k / N}$$

\blacktriangleright Speciálně pro 0 Dirichletovy o.p. lze použít DST:

$$\phi_p = \frac{2}{N+1} \sum_{k=1}^N \hat{\phi}_k \sin\left(\frac{pk\pi}{N+1}\right)$$



Metoda Fourierovy transformace II

- ▶ DST ve vektorovém zápisu s užitím báзовých vektorů DST:

$$\phi = \frac{2}{N+1} \sum_{k=1}^N \hat{\phi}_k s_k ; \quad s_{k,p} = \sin\left(\frac{pk\pi}{N+1}\right)$$

- ▶ Vidíme, že báзовé funkce DST s_k jsou shodné s vlastními vektory λ_k odpovídající matice druhých derivací K . Proto:

$$K\phi = \frac{2}{N+1} \sum_{k=1}^N \hat{\phi}_k K s_k = \frac{2}{N+1} \sum_{k=1}^N \hat{\phi}_k \lambda_k s_k$$

- ▶ Fourierova transformace matice K je matice vlastních čísel Λ



Metoda Fourierovy transformace III

- ▶ Fourierova transformace matice K je matice vlastních čísel Λ
- ▶ Diskrétní Poissonovu rovnici lze transformovat:

$$K\phi = F \xrightarrow{\text{FT}} \hat{K}\hat{\phi} = \hat{F} \rightarrow \Lambda\hat{\phi} = \hat{F}$$

- ▶ Inverze Λ je triviální a dostáváme řešení v k -prostoru

$$\hat{\phi}_k = \hat{\rho}_k / \left[2 \cos\left(\frac{k\pi}{N+1}\right) - 2 \right]$$



Metoda Fourierovy transformace IV

- ▶ Výpočetní náročnost odpovídá náročnosti FFT: $O(N \log(N))$
- ▶ Snadné zobecnění do 2D a 3D
- ▶ Jde o speciální případ spektrální metody
 - ▶ Transformace soustavy do báze vlastních vektorů—obecně pomalé, ale FFT je rychlé



FACR

- ▶ Fourierovu transformaci lze kombinovat s cyklickou redukcí
- ▶ I kroků cyklické redukce následovaných FT – metoda FACR(I)
- ▶ V 1D má redukovaný systém stejnou strukturu → jednoduché
- ▶ Ve 2D, např. po jedné iteraci je rovnice pro x řádek mříže

$$U_{i-2} - (A^2 - 2I)U_i + U_{i+2} = F'_i$$

- ▶ Matice $L = (A^2 - 2I)$ má diagonální FT → lze převést na

$$\hat{U}_{i-2} - \hat{L}\hat{U}_i + \hat{U}_{i+2} = \hat{F}'_i$$

- ▶ Jde o nezávislý tridiagonální systém pro každou frekvenci ve směru y
- ▶ Rychlé řešení Thomasovým algoritmem



FACR(*)

- ▶ Existuje ideální volba počtu iterací CR – minimalizace počtu operací
- ▶ Přibližně $r \approx \log_2 \log_2 n$
- ▶ Počet operací

$$4.5 N_g \log_2 \log_2 N_g$$

- ▶ Značení FACR(*)



Multigridy – konvergence Jacobiho metody

Problém Jacobiho metody – pomalá konvergence. Např v 1D:

- ▶ Preconditioner – diagonální matice $P = -2I$
- ▶ Iterační matice $M = P^{-1}(P - K) = I + K/2$
- ▶ Připomínka – vlastní čísla K : $\lambda_k(K) = 2 \cos\left(\frac{k\pi}{N+1}\right) - 2$
- ▶ Vl. č. iterační matice

$$\lambda_k(M) = \cos\left(\frac{k\pi}{N+1}\right)$$

- ▶ Nízkofrekvenční, ale i vysokofrekvenční chyby konvergují pomalu



Multigridy – vyhlazování

- ▶ Vyhlazování - smoothed Jacobi (SJ),
 $M = I - \omega D^{-1}K = I - \omega K/2, \quad 0 < \omega < 1$
- ▶ Vl. č. (SJ) (obrázek)

$$\lambda_k(M) = -1 + \omega - \omega \cos\left(\frac{k\pi}{N+1}\right)$$

- ▶ Pro obvyklou volbu $\omega = 2/3$, vyhlazovací faktor $1/3$

$$\phi_i^{n+1} = \frac{\phi_{i-1}^n + \phi_i^n + \phi_{i+1}^n}{3} + \frac{h^2 \rho_i}{3\epsilon_0}$$

- ▶ Kvalitní vyhlazení je zabraňuje aliasingu (!)



Multigridy – princip

- ▶ Konvergenci lze urychlit přechodem na hrubší mříž
 - ▶ Vyskofrekvenční chyby se vyhlazují na jemné mříži
 - ▶ Nízkofrekvenční chyby se odstraňují na hrubé mříži
- ▶ Přejchody s pomocí operátorů restrikce R a prolongace P
- ▶ Hierarchie mříží, vzorkování $h, 2h, 4h, \dots$

Příklad v 1D

$$P_{2h}^h = \frac{1}{2} \begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & 1 & \\ & & 2 & \\ & & 1 & 1 \\ & & & 2 \\ & & & 1 \end{bmatrix} ; \quad R_h^{2h} = \frac{1}{2} (P_{2h}^h)^T = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & & & & & \\ & 2 & 1 & 2 & 1 & & & \\ & & 1 & 2 & 1 & & & \\ & & & 2 & 1 & 2 & 1 & \\ & & & & 1 & 2 & 1 & \\ & & & & & 2 & 1 & \\ & & & & & & 2 & \\ & & & & & & & 2 \end{bmatrix}$$



Multigridy – v-cyklus

Postup řešení soustavy $Au = b$

1. Několik iterací vyhlazování (proti aliasingu)
2. Restrikce residua $r_h = b - Au_h$ na mříž $2h$ podle $r_{2h} = R_h^{2h} r_h$
3. Výpočet chyby v potenciálu $A_{2h} E_{2h} = r_{2h}$ (přímý nebo iterační)
4. Prolongace $E_h = P_{2h}^h$ a korekce potenciálu $u_h \rightarrow u_h + E_h$
5. Vyhlazování

Redukovaná matice

$$A_{2h} = R_h^{2h} A_h P_{2h}^h$$



Multigridy

- ▶ Složitější cykly - V -cyklus, W -cyklus, FMG
- ▶ Pracuje se s korekcemi E namísto potenciálu u
- ▶ Lze ukázat redukci chyby v jedné iteraci $\approx 1/10$
 - ▶ Nezávisle na velikosti mříže
 - ▶ Iterace na hrubých mřížích jsou levné
 - ▶ Náročnost $O(N)$ pro dosažení dané chyby
- ▶ Důležitou součástí MG je přímý řešič na hrubé mříži
- ▶ Použitelné pro obecné operátory a libovolné geometrie



Metody pro řídké matice

- ▶ Obecné maticové metody
- ▶ Vhodné jako přímý solver pro multigridy (také zvládají obecné geometrie a typy rovnic)
- ▶ Minimum degree algoritmy - hledají permutaci A tak, aby počet nenulových prvků pod diagonálou byl minimální

Knihovna UMFPACK

- ▶ Transformuje matici $A' = PRAQ$, P, Q : permutační matice, R : diagonální škálovací matice
- ▶ Minimalizuje počet nenulových prvků v $LU = A'$



Postup řešení

1. Analýza struktury matice, P, R, Q
2. Výpočet rozkladu LU
3. Vyřešení rovnice $Ax = b$ – rychlé

Datové struktury – row-indexed storage (Yale format)

- ▶ Ax : hodnoty nenulových prvků
- ▶ Ai : řádkové indexy i
- ▶ $Ap[j]$: počet nenulových prvků v prvních j řádcích – index j -tého řádku v polích Ax, Ai



Příklad 2D Poissonova rovnice v UMFPACKu

```
l = 0;
Ap[0] = 0;
for(i=0; i<param.x_sampl; i++)                //cislo radku
for(j=0; j<param.z_sampl; j++)                //cislo sloupce
{
    k = j + param.z_sampl*i;
    if(grid.mask[i][j]==FIXED || grid.mask[i][j]==FIXED_RF)
    // elektrody a okraje
    {
        Ax[l] = 1;
        Ai[l] = k;
        Ap[k+1] = Ap[k]+1;
        l++;
    }
    else
    {
        Ax[l] = Ax[l+1] = Ax[l+3] = Ax[l+4] = 1.0;
        Ax[l+2] = -4;
        Ai[l] = k-param.z_sampl;
        Ai[l+1] = k-1;
        Ai[l+2] = k;
        Ai[l+3] = k+1;
        Ai[l+4] = k+param.z_sampl;
        Ap[k+1] = Ap[k]+5;
        l += 5;
    }
}
(void) umfpack_di_symbolic (n, n, Ap, Ai, Ax, &Symbolic, NULL, NULL) ;
(void) umfpack_di_numeric (Ap, Ai, Ax, Symbolic, &Numeric, NULL, NULL) ;
umfpack_di_free_symbolic (&Symbolic) ;
```

