TECHNISCHE UNIVERSITÄT DARMSTADT

A Tutorial on Elliptic PDE Solvers and Their Parallelization

von C. C. Douglas, G. Haase und U. Langer

Iterative Methoden – Roughers

Martin Lilienthal

Seminar Numerik und Wissenschaftliches Rechnen
WS 06/07

1. CG Verfahren
2. GMRES
3. BICGSTAB
4. Vorkonditionierung

Allgemeines zu Projektionsmethoden

Die hier beschriebenen Solver sind allesamt Projektionsmethoden zur Lösung von $A\mathbf{x} = \mathbf{b}$

Definition Projektionsmethode

Eine Projektionsmethode ist ein Verfahren zur Berechnung von Näherungslösungen $\mathbf{x}_m \in \mathbf{x}_0 + K_m$ mit $\mathbf{r}_m = (A\mathbf{x}_m - \mathbf{b}) \perp L_m$

Gilt $K_m = L_m$ liegt eine orthogonale Projektionsmethode vor.

Ansonsten handelt es sich um eine schiefe Projektionsmethode.

Definition Krylov-Unterraum-Methode

Projektionsmethode mit $K_m = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$

Allgemeines zu Projektionsmethoden

Beispiel

Wähle $K_m = L_m = \text{span}\{\mathbf{r}_{m-1}\}$. Man erhält das Gradientenverfahren.

Nachteil: Der m-te Residuumsvektor ist nicht orthogonal zum gesamten Unterraum $U=\text{span}\{\mathbf{r}_0,\ldots,\mathbf{r}_{m-1}\}$. Somit kann man die Lösung nicht als Linearkombination aus n Basisvektoren dargestellt werden. Es werden also mehr als n Iterationen benötigt.

Grundalgorithmus für Solver

Sei A $\in \mathbf{R}^{nxn}$ eine SPD Matrix und $\mathbf{x},\mathbf{b} \in \mathbf{R}^n$

Betrachte $F(x) = \frac{1}{2}(A\mathbf{x},\mathbf{x})_2 - (\mathbf{b},\mathbf{x})_2$

Dann gilt: $\overline{\mathbf{x}} = \arg\min F(\mathbf{x}) \iff \mathbf{A}\overline{\mathbf{x}} = \mathbf{b}$

Hier einfacher Grundalgorithmus:

$\mathbf{x_0} \in \mathbf{R^n}$

for m=0,1,...

$\mathbf{r_m} = \mathbf{b} - \mathbf{A}\mathbf{x_m}$

$\lambda_m = \frac{(\mathbf{r_m},\mathbf{s_m})_2}{(A\mathbf{s_m},\mathbf{s_m})_2}$

$\mathbf{x_{m+1}} = \mathbf{x_m} + \lambda_\mathbf{m}\mathbf{s_m}$

$\mathbf{x}_m$: Näherungslösung

$\mathbf{r}_m$: Residuenvektor

$\mathbf{s}_m$: Suchrichtung

λ_m: Schrittweite

CG Verfahren

Definition A-orthogonal

Sei $A \in \mathbf{R}^{nxn}$, dann heißen die Vektoren $\mathbf{s}_0,..,\mathbf{s}_m \in \mathbf{R}^n$ paarweise konjugiert oder A-orthogonal wenn

$(\mathbf{s_i}, \mathbf{s_j})_\mathbf{A} := (\mathbf{A s_i}, \mathbf{s_j})_\mathbf{2} = 0$ gilt.

Idee:

Wähle die Suchrichtungen wie folgt:

$$\mathbf{s_0} = \mathbf{r_0}$$

$$\mathbf{s_m} = \mathbf{r_m} + \sum_{\mathbf{j=0}}^{\mathbf{m-1}} \alpha_\mathbf{j} \mathbf{s_j}$$

Dabei sollen die Suchrichtungen konjugiert sein. Somit ergibt sich für *i=0,1,…,m-1:*

$$(\mathbf{s_m}, \mathbf{s_i})_\mathbf{A} = (\mathbf{r_m}, \mathbf{s_i})_\mathbf{A} + \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \alpha_\mathbf{j} (\mathbf{s_j}, \mathbf{s_i})_\mathbf{A} = 0$$

CG Verfahren

Aus $(\mathbf{s}_i,\mathbf{s}_j)_A = 0 \; \forall \; i,j \in \{0,1,\ldots,m-1\}, i \neq j$ ergibt sich:

$$\alpha_i = -\frac{(\mathbf{r_m},\mathbf{s_i})_\mathbf{A}}{(\mathbf{s_i},\mathbf{s_i})_\mathbf{A}}$$

Somit könnte man folgenden einfachen, aber leider ineffizienten Algorithmus verwenden:

Wähle $\mathbf{x_0} \in \mathbf{R^n}$

$\mathbf{s_0} := \mathbf{r_0} = \mathbf{b} - \mathbf{A x_0}$

for $\mathbf{m} = \mathbf{0}, \mathbf{1}, ..., \mathbf{n-1}$

$\lambda_\mathbf{m} = \frac{(\mathbf{r_m},\mathbf{s_m})_\mathbf{2}}{(\mathbf{s_m},\mathbf{s_m})_\mathbf{A}}$

$\mathbf{x_{m+1}} = \mathbf{x_m} + \lambda_\mathbf{m}\mathbf{s_m}$

$\mathbf{r_{m+1}} = \mathbf{r_m} - \lambda_\mathbf{m}\mathbf{s_m}$

$\mathbf{s_{m+1}} = \mathbf{r_{m+1}} - \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{m}} \frac{(\mathbf{r_{m+1}},\mathbf{s_j})_\mathbf{A}}{(\mathbf{s_j},\mathbf{s_j})_\mathbf{A}}\mathbf{s_j}$

Problem:

Um eine neue Suchrichtung $\mathbf{s}_{m+1}$ zu berechnen werden alle $\mathbf{s}_j$ mit j=0,1,…,m benötigt.
-> Unpraktikabel für große SPARSE-Matrizen, wie aus FE-Diskretisierung.

CG Verfahren

Bemerkungen:

- Es gilt $K_m = span\{\mathbf{s_0}, ..., \mathbf{s_{m-1}}\} = \mathbf{span}\{\mathbf{r_0}, ..., \mathbf{A^{m-1}r_0}\}$
  $\mathbf{x_m} \in \mathbf{x_0} + \mathbf{K_m}$
- Bei exakter Berechnung erhält man nach n-Iterationen die exakte Lösung.
- In der Realität treten jedoch Rundungsfehler auf, so dass eine nach n Schritten berechnete Lösung von der exakten abweicht.
- In der Praxis wird das Verfahren abgebrochen, wenn das Residuum klein genug ist.

Vorkonditioniertes CG Verfahren

Mit einigen Modifikationen und unter Verwendung eines Vorkonditionierers erhält man folgenden Algorithmus:

Choose $\underline{u}^0$

$\underline{r} := \underline{f} - K \cdot \underline{u}^0$

$\underline{w} := C^{-1} \cdot \underline{r}$

$\underline{s} := \underline{w}$

$\sigma := \sigma_{\text{old}} := \sigma_0 := (\underline{w}, \underline{r})$

repeat

$\underline{v} := K \cdot \underline{s}$

$\alpha := \sigma / (\underline{s}, \underline{v})$

$\underline{u} := \underline{u} + \alpha \cdot \underline{s}$

$\underline{r} := \underline{r} - \alpha \cdot \underline{v}$

$\underline{w} := C^{-1} \cdot \underline{r}$

$\sigma := (\underline{w}, \underline{r})$

$\beta := \sigma / \sigma_{\text{old}}$

$\sigma_{\text{old}} := \sigma$

$\underline{s} := \underline{w} + \beta \cdot \underline{s}$

until $\sqrt{\sigma / \sigma_0} < \text{tolerance}$

K: Systemsteifigkeitsmatrix
u: Näherungslösung
C: SPD Vorkonditionierer
α: Schrittweite
s: Suchrrichtung

CG Verfahren

Paralleler Algorithmus:

Choose $\underline{u}^0$

$\underline{r} := \underline{f} - K \cdot \underline{u}^0$

$\underline{w} := \sum_{j=1}^{P} A_j^T \underline{r}_j$

$\underline{s} := \underline{w}$

$\sigma := \sigma_{old} := \sigma_0 := (\underline{w}, \underline{r})$

repeat

$\underline{v} := K \cdot \underline{s}$

$\alpha := \sigma / (\underline{s}, \underline{v})$

$\underline{u} := \underline{u} + \alpha \cdot \underline{s}$

$\underline{r} := \underline{r} - \alpha \cdot \underline{v}$

$\underline{w} := \sum_{j=1}^{P} A_j^T \underline{r}_j$

$\sigma := (\underline{w}, \underline{r})$

$\beta := \sigma / \sigma_{old}$

$\sigma_{old} := \sigma$

$\underline{s} := \underline{w} + \beta \cdot \underline{s}$

until $\sqrt{\sigma / \sigma_0} < \text{tolerance}$

Strategie zur Parallelisierung

K: distributed-Matrix (Typ II)

$\mathfrak{u}, \mathfrak{s}$: accumulated-Vektoren (Typ I)

Also ergibt sich für die weiteren Vektoren:

$\mathfrak{w}$: accumulated-Vektor (Typ I)

v, r, f: distributed-Vektoren (Typ II)

CG Verfahren

Bemerkungen zur Parallelisierung:

- Die Wahl von C=I im parallelen Algorithmus führt zu einer Typumwandlung $\mathfrak{w} := r$, die eine Akkumulierung notwendig macht.
- Die so genannten DAXPY Operationen benötigen keinerlei Kommunikation.
- Die Skalarprodukte benötigen nur bei der Akkumulation der Teilskalarprodukte Kommunikation.

GMRES Allgemeines

- GMRES steht für *Generalized Minimal Residual*
- 1986 von Saad und Schulz vorgestellt.
- A muss regulär sein, sonst keine Anforderungen!
- Herleitung über Betrachtung als Krylov-Unterraum-Methode mit Petrov-Galerkin-Bedingung $L_m = A K_m$ möglich
- Oder durch Umformen des LGS in Minimierungsaufgabe mit: $\overline{\mathbf{x}} = argmin\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_\mathbf{2}^\mathbf{2}$
- GMRES basiert auf Orthonormalbasis $\{v_1,\ldots,v_m\}$ des Krylov-Raums K_m. Diese wird mit dem Arnoldi-Algorithmus berechnet.

Arnoldi-Algorithmus

Zur Herleitung wird von vorliegender Orthonormalbasis $\{v_1,\ldots,v_j\}$ des $K_j = \text{span}\{r_0,\ldots,A^{m-1}r_0\}$ ausgegangen.

Wegen $AK_m = \text{span}\{Ar_0,\ldots,A^m r_0\} \subset K_{m+1}$ kann man v_{m+1} wie folgt definieren:

$$\mathbf{v_{m+1}} = \mathbf{A v_m} + \xi \quad \text{mit} \quad \xi = -\sum_{\mathbf{j=1}}^{\mathbf{m}} \alpha_\mathbf{j} \mathbf{v_j} \in \mathbf{K_m}$$

Damit gilt: $(\mathbf{v_{m+1}}, \mathbf{v_j})_\mathbf{2} = (\mathbf{A v_m}, \mathbf{v_j})_\mathbf{2} - \alpha_\mathbf{j} (\mathbf{v_j}, \mathbf{v_j})_\mathbf{2}$

Somit kann unter Anwendung d. Orthogonalitätsbed.

$$\alpha_j = \frac{(\mathbf{A v_m}, \mathbf{v_j})_\mathbf{2}}{(\mathbf{v_j}, \mathbf{v_j})}$$

berechnet werden.

Arnoldi-Algorithmus

Arnoldi-Algorithmus:

$$\mathbf{v_1} := \frac{\mathbf{r_0}}{\|\mathbf{r_0}\|_2}$$
$$\text{for } \mathbf{j} = 1, \ldots, \mathbf{m}$$
$$\quad \text{for } \mathbf{i} = 1, \ldots, \mathbf{j}$$
$$\quad\quad \mathbf{h_{ij}} := (\mathbf{v_i}, \mathbf{A v_j})_2$$
$$\quad \mathbf{w_j} := \mathbf{A v_j} - \sum_{\mathbf{i}=1}^{\mathbf{j}} \mathbf{h_{ij} v_i}$$
$$\quad \mathbf{h_{j+1,j}} := \|\mathbf{w_j}\|_2$$
$$\quad \text{if } \mathbf{h_{j+1,j}} \neq \mathbf{0}$$
$$\quad \mathbf{v_{j+1}} = \frac{\mathbf{w_j}}{\mathbf{h_{j+1,j}}}$$
$$\quad \text{else}$$
$$\quad \mathbf{v_{j+1}} = \mathbf{0}$$
$$\quad \text{STOP}$$

Man erhält durch den Algorithmus eine obere Hessenbergmatrix $\mathbf{H}_m = \mathbf{V}_m^T \mathbf{A} \mathbf{V}_m$

mit $\mathbf{V}_m = (\mathbf{v}_1, \ldots, \mathbf{v}_m) \in \mathbf{R}^{nxm}$

Weiterhin gilt:

$$\mathbf{A V_m} = \mathbf{V_{m+1}} \overline{\mathbf{H}}_\mathbf{m}$$

$$\text{mit } \overline{\mathbf{H}}_\mathbf{m} = \begin{pmatrix} \mathbf{H_m} \\ 0 \cdots 0 \; h_{m+1,m} \end{pmatrix} \in \mathbf{R^{m+1xm}}$$

GMRES

Idee von GMRES: minimiere im m-ten Schritt die 2-Norm des Residuums über dem Raum $\mathbf{x}_0 + K_m$.

Mit $\mathbf{V}_m = (\mathbf{v}_1,\ldots,\mathbf{v}_m)$ kann $\mathbf{x}_m \in \mathbf{x}_0 + K_m$ als $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{z}_m$ dargestellt werden.

Minimierungsproblem:

$$
\begin{aligned}
\mathbf{z_m} &= arg\,min\|\mathbf{b} - \mathbf{A}\mathbf{x_m}\|_\mathbf{2} \\
&= arg\,min\|\mathbf{b} - \mathbf{A}(\mathbf{x_0} + \mathbf{V_m}\mathbf{z})\|_\mathbf{2} \\
&= arg\,min\|\|\mathbf{r_0}\|_\mathbf{2}\mathbf{v_1} - \mathbf{V_{m+1}}\overline{\mathbf{H}}_\mathbf{m}\mathbf{z}\|_\mathbf{2} \\
&= arg\,min\|\mathbf{V_{m+1}}\left(\|\mathbf{r_0}\|_\mathbf{2}\mathbf{e_1} - \overline{\mathbf{H}}_\mathbf{m}\mathbf{z}\right)\|_\mathbf{2} \\
&= arg\,min\|\|\mathbf{r_0}\|_\mathbf{2}\mathbf{e_1} - \overline{\mathbf{H}}_\mathbf{m}\mathbf{z}\|_\mathbf{2}
\end{aligned}
$$

GMRES

Transformiere nun die Matrix $\overline{\mathbf{H}}_m$ mit einer Orthogonalen Matrix $\mathbf{Q}_m \in \mathbf{R}^{(m+1)x(m+1)}$ auf eine obere Dreiecksmatrix mit angehängter „Null-Zeile“: $\mathbf{Q_m}\overline{\mathbf{H}}_\mathbf{m} = \overline{\mathbf{R}}_\mathbf{m}$

Die obere Dreiecksmatrix wird in der Regel mittels Givens-Rotationen berechnet. Es können jedoch auch andere Methoden zur QR-Zerlegung wie z.B. die Hausholder Transformation verwendet werden.

Dann kann das Residuum wie folgt umgeformt werden:

$$\begin{aligned} min\|\|\mathbf{r_0}\|_2\mathbf{e_1} - \overline{\mathbf{H}}_\mathbf{m}\mathbf{z}\|_2 &= min\|\mathbf{Q_m}\left(\|\mathbf{r_0}\|_2\mathbf{e_1} - \overline{\mathbf{H}}_\mathbf{m}\mathbf{z}\right)\|_2 \\ &= min\|\mathbf{Q_m}\|\mathbf{r_0}\|_2\mathbf{e_1} - \overline{\mathbf{R}}_\mathbf{m}\mathbf{z}\|_2 \end{aligned}$$

Durch die Gestalt von $\overline{\mathbf{R}}_m$ lassen sich der Vektor $\mathbf{z}_m$ einfach berechnen und damit schliesslich auch die (Näherungs)lösung $\mathbf{x}_m$.

GMRES

Choose $\underline{u}^0$

$\underline{r} := \underline{f} - K \cdot \underline{u}^0$

$\underline{w}^1 := C^{-1} \cdot \underline{r}$

$z_1 := (\underline{w}^1, \underline{r})$

$k := 0$

repeat $k := k + 1$

$\underline{r} := K \cdot \underline{w}^k$

for $i := 1$ to k do

$h_{i,k} := (\underline{w}^i, \underline{r})$

$\underline{r} := \underline{r} - h_{i,k} \cdot \underline{w}^i$

end

$\underline{w}^{k+1} := C^{-1} \cdot \underline{r}$

$h_{k+1,k} := (\underline{w}^{k+1}, \underline{r})$

$\underline{w}^{k+1} := \underline{w}^{k+1} / h_{k+1,k}$

for $i := 1$ to $k - 1$ do $\begin{pmatrix} h_{i,k} \\ h_{i+1,k} \end{pmatrix} := \begin{pmatrix} c_{i+1} & s_{i+1} \\ s_{i+1} & -c_{i+1} \end{pmatrix} \cdot \begin{pmatrix} h_{i,k} \\ h_{i+1,k} \end{pmatrix}$

$\alpha := \sqrt{h_{k,k}^2 + h_{k+1,k}^2}$

$s_{k+1} := h_{k+1,k}/\alpha \; ; \; c_{k+1} := h_{k,k}/\alpha \; ; \; h_{k,k} := \alpha$

$z_{k+1} := s_{k+1} z_k \; ; \; z_k := c_{k+1} z_k$

until $|z_{k+1}/z_1| <$ tolerance

$z_k := z_k / h_{k,k}$

for $i := k - 1$ down to 1 do $z_i := \left(z_i - \sum_{j=i+1}^{k} h_{i,j} z_j \right) / h_{i,i}$

$\underline{u}^k := \underline{u}^0 + \sum_{i=1}^{k} z_i \cdot \underline{w}^j$

K: Systemsteifigkeitsmatrix

u: Näherungslösung

C: SPD Vorkonditionierer

GMRES Parallelisiert

Choose $\underline{u}^0$

$\underline{r} := \underline{f} - K \cdot \underline{u}^0$

$\underline{w}^1 := \sum_{s=1}^{P} A_s^T \underline{r}_s$

$z_1 := (\underline{w}^1, \underline{r})$

$k := 0$

repeat $k := k + 1$

$\underline{r} := K \cdot \underline{w}^k$

for $i := 1$ to k do

$h_{i,k} := (\underline{w}^i, \underline{r})$

$\underline{r} := \underline{r} - h_{i,k} \cdot R^{-1} \cdot \underline{w}^i$

end

$\underline{w}^{k+1} := \sum_{s=1}^{P} A_s^T \underline{r}_s$

$h_{k+1,k} := (\underline{w}^{k+1}, \underline{r})$

$\underline{w}^{k+1} := \underline{w}^{k+1} / h_{k+1,k}$

for $i := 1$ to $k-1$ do $\begin{pmatrix} h_{i,k} \\ h_{i+1,k} \end{pmatrix} := \begin{pmatrix} c_{i+1} & s_{i+1} \\ s_{i+1} & -c_{i+1} \end{pmatrix} \cdot \begin{pmatrix} h_{i,k} \\ h_{i+1,k} \end{pmatrix}$

$\alpha := \sqrt{h_{k,k}^2 + h_{k+1,k}^2}$

$s_{k+1} := h_{k+1,k}/\alpha \; ; \; c_{k+1} := h_{k,k}/\alpha \; ; \; h_{k,k} := \alpha$

$z_{k+1} := s_{k+1} z_k \; ; \; z_k := c_{k+1} z_k$

until $|z_{k+1}/z_1| < \text{tolerance}$

$z_k := z_k / h_{k,k}$

for $i := k-1$ down to 1 do $z_i := \left(z_i - \sum_{j=i+1}^{k} h_{i,j} z_j \right) / h_{i,i}$

$\underline{u}^k := \underline{u}^0 + \sum_{i=1}^{k} z_i \cdot \underline{w}^j$

Speicherstrategie zur Parallelisierung

Matrizen u. Vektoren:

K: distributed-Matrix (Typ II)

$\mathfrak{u}, \mathfrak{w}$: accumulated-Vektor (Typ I)

r, f: distributed-Vektoren (Typ II)

Skalare:

z_j, s_j, c_j, $h_{i,j}$: redundant auf jedem Prozessor

GMRES

Bemerkungen

- Rechenaufwand zu Berechnung der Orthonormalbasis des Krylov-Raums steigt mit dessen Dimension
- Da die Basisvektoren benötigen viel Speicherplatz. Bei einer SPARSE-Matrix $A \in R^{nxn}$ muss schlimmstenfalls eine vollbesetzte Matrix $V_n \in R^{nxn}$ gespeichert werden.
- Deshalb in der Praxis meist Abbruch des Verfahrens nach m Schritten. Wird bei Abbruch nicht die gewünschte Genauigkeit erreicht, so wird das Verfahren mit der vorher berechneten Näherungslösung als Startwert neugestartet.
- Das Konvergenzverhalten ist stark vom Vorkonditionierer abhängig. Bewährt hat sich z.B. die unvollständige LU Zerlegung.

GMRES

Bemerkungen zur Parallelisierung

- Skalarprodukte benötigen nur geringen Kommunikationsaufwand
- Alle DAXPY-Operationen benötigen keinen Kommunikationsaufwand
- Schritt *k* benötigt *k+1* ALL_REDUCE-Operationen
- Wegen der Typumwandlung von w^i werden in jedem Schritt zusätzliche $k \cdot n$ Multiplikationen benötigt
- Die einmalige Typänderung von w^1 := r führt zu Kommunikation zwischen allen Prozessen

BICGSTAB

- Wurde 1992 von van der Vorst vorgestellt.
- Das BICGSTAB-Verfahren kann zum Lösen von Gleichungssystemen mit beliebiger regulärer Koeffizientenmatrix verwendet werden. Es basiert auf CGS-Verfahren.
- Ein Vorteil gegenüber GMRES liegt im wesentlich geringeren Speicherbedarf.

BICGSTAB

Sequentieller Algorithmus

Choose initial guess $\underline{u}$

$\underline{r} := \underline{f} - K \cdot \underline{u}$

Choose $\underline{q}$ such that $(\underline{q}, \underline{r}) \neq 0$

Set $\varrho := 1$, $\alpha := 1$, $\omega := 1$, $\underline{v} := 0$, $\underline{p} := 0$

while $(\underline{r}, \underline{r}) < \varepsilon^2$ do

$$
\begin{aligned}
\varrho_{old} &:= \varrho \\
\varrho &:= (\underline{q}, \underline{r}) \\
\beta &:= \varrho/\varrho_{old} \cdot \alpha/\omega \\
\underline{p} &:= \underline{r} + \beta \cdot (\underline{p} - \omega \cdot \underline{v}) \\
\underline{v} &:= K \cdot \underline{p} \\
\alpha &:= \varrho/(\underline{q}, \underline{v}) \\
\underline{s} &:= \underline{r} - \alpha \cdot \underline{v} \\
\underline{t} &:= K \cdot \underline{s} \\
\omega &:= (\underline{t}, \underline{s})/(\underline{t}, \underline{t}) \\
\underline{u} &:= \underline{u} + \alpha \cdot \underline{p} + \omega \cdot \underline{s} \\
\underline{r} &:= \underline{s} - \omega \cdot \underline{t}
\end{aligned}
$$

BICGSTAB

Paralleler Algorithmus

Choose initial guess $\underline{u}$

$$\underline{r} := \sum_{s=1}^{P} A_s^T \left(\underline{f}_s - K_s \cdot \underline{u}_s\right)$$

Choose $\underline{q}$ such that $(\underline{q}, \underline{r}) \neq 0$

Set $\varrho := 1,\ \alpha := 1,\ \omega := 1,\ \underline{v} := 0,\ \underline{p} := 0$

while $(\underline{r}, R^{-1}\underline{r}) < \varepsilon^2$ do

$$\begin{aligned}
\varrho_{old} &:= \varrho \\
\varrho &:= (\underline{q}, \underline{r}) \\
\beta &:= \varrho/\varrho_{old} \cdot \alpha/\omega \\
\underline{p} &:= \underline{r} + \beta \cdot (\underline{p} - \omega \cdot \underline{v}) \\
\underline{v} &:= \sum_{s=1}^{P} A_s^T K_s \cdot \underline{p}_s \\
\alpha &:= \varrho/(\underline{q}, \underline{v}) \\
\underline{s} &:= \underline{r} - \alpha \cdot \underline{v} \\
\underline{t} &:= K \cdot \underline{s} \\
\widehat{\underline{t}} &:= \sum_{s=1}^{P} A_s^T \underline{t}_s \\
\omega &:= (\underline{t}, \underline{s})/(\underline{t}, \widehat{\underline{t}}) \\
\underline{u} &:= \underline{u} + \alpha \cdot \underline{p} + \omega \cdot \underline{s} \\
\underline{r} &:= \underline{s} - \omega \cdot \widehat{\underline{t}}
\end{aligned}$$

Speicherstrategie zur Parallelisierung

Matrizen u. Vektoren:

K: distributed-Matrix (Typ II)

$\mathbf{r}, \mathbf{u}, \boldsymbol{\nu}, \mathbf{p}, \mathbf{s}, \mathbf{t}$: accumulated-Vektoren(Typ I)

q,t, f: distributed-Vektoren (Typ II)

BICGSTAB

Bemerkungen zur Parallelisierung

- Skalarprodukte benötigen nur geringen Kommunikationsaufwand
- Fast alle DAXPY-Operationen mit benötigen keine Kommunikation.
- Typkonvertierungen erfordern Kommunikation

Vorkonditionierung

Zur Vorkonditionierung können die parallelen Version der „klassischen“ iterativen Methoden, der ILU aber auch die symmetrische Mehrgittermethode verwendet werden.

Literatur

Douglas, C.C., Haase, G., Langer, U. - A Tutorial on Elliptic PDE Solvers and Their Parallelization *SIAM 2003*

Meister, A. – Numerik linearer Gleichungssysteme *Vieweg 2005*