

Gemsicht genaue CG-Löser mit a posteriori Fehlerschätzer

Tobias
Ehring
&
Daniel
Fink

SimTech-Projektarbeit

17. September 2018

Ians



Motivation

Das 2D-Poisson-Problem auf einem Einheitsquadrat $\Omega := [0, 1]^2$ mit homogenen Dirichlet-Randbedingungen:

$$-\Delta u = f \quad \text{in} \quad \Omega \quad \text{mit} \quad u = 0 \quad \text{auf} \quad \partial\Omega$$



Motivation

$$A\tilde{u} = \tilde{f}$$

mit $A = \frac{1}{h^2} \begin{bmatrix} T & -I \\ -I & \ddots \\ & \ddots & -I \\ & -I & T \end{bmatrix}$, wobei $T = \begin{bmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & \ddots & \ddots \\ & & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix}$.

$$A \in \mathbb{R}^{m^2 \times m^2} \text{ und } I, T \in \mathbb{R}^{m \times m}$$



Motivation

- Implementierung → Wahl der Datentypen
 - Double = 8 Byte
 - Single = 4 Byte
 - Single-Recheneinheiten sind kleiner → Mehr vorhanden auf Hardware
→ Single schneller als Double
- ABER:** geringere Genauigkeit

⇒ **Idee: gemischte Genauigkeit**

d.h. möglichst viele Rechnungen in Single durchführen – ohne Verlust der Genauigkeit

Übersicht – Theorie

Normwise Backward Error (NBE)	Gemischt genaue iterative Verfeinerung	IVNCG-Verfahren
Abbruchkriterium: $\frac{\ r_k\ }{\ A\ \ x_k\ + \ b\ }$	1. $r_m = b - Ax_m$ 2. $Ad_m = r_m$ 3. $x_{m+1} = x_m + d_m$	
NBE im CG-Verfahren		



Normwise Backward Error im CG-Verfahren



Normwise Backward Error

Rückwärtsfehler:

$$\min\{\epsilon : (A + \Delta A)x_m = b + \Delta b, \|\Delta A\| \leq \epsilon \|A\|, \|\Delta b\| \leq \epsilon \|b\|\}$$

wobei $\|\cdot\| := \|\cdot\|_2$

Ergebnisse des Rechnungssystems



Normwise Backward Error

Rückwärtsfehler:

$$\min\{\epsilon : (A + \Delta A)x_m = b + \Delta b, \|\Delta A\| \leq \epsilon \|A\|, \|\Delta b\| \leq \epsilon \|b\|\}$$

wobei $\|\cdot\| := \|\cdot\|_2$

Störungen des Gleichungssystems



Normwise Backward Error

Theorem 1. Sei $Ax = b$ ein Gleichungssystem mit $A \in \mathbb{R}^{n \times n}$ und $x, b \in \mathbb{R}^n$. Sei $x_m \in \mathbb{R}^n$ beliebig, dann ist

$$\min\{\epsilon : (A + \Delta A)x_m = b + \Delta b, \|\Delta A\| \leq \epsilon \|A\|, \|\Delta b\| \leq \epsilon \|b\|\} = \frac{\|r_m\|}{\|A\| \|x_m\| + \|b\|}$$

[Rigal, Gaches 1967]



Normwise Backward Error

$$\text{NBE}(x_k) = \frac{\|r_k\|}{\|A\|\|x_k\| + \|b\|} \leq \frac{\|r_k\|}{\|b\|} = \text{RR}(x_k)$$



Normwise Backward Error im CG-Verfahren

$$\Delta_k \leq \|A\|$$

$$\text{NBE}(x_k) = \frac{\|r_k\|}{\|A\|\|x_k\| + \|b\|} \leq \frac{\|r_k\|}{\Delta_k \|x_k\| + \|b\|}$$

[Tichy 2013]



A ist symmetrisch positiv definit:

$$\|A\| = \max \sigma(A)$$

Normwise Backward Error im CG-Verfahren

Lanczos-Algorithmus

Setze $\beta_0 = 0$, $v_0 = 0$

Berechne $v_1 = v / \|v\|$

Schleife über $k = 1, 2, \dots, n$

$$\alpha_k = \langle v_k, Av_k \rangle_2$$

$$w_k = Av_k - \alpha_k v_k - \beta_{k-1} v_{k-1}$$

$$\beta_k = \|w_k\|$$

$$v_{k+1} = w_k / \beta_k$$

Algorithmus berechnet
die Orthonormalbasis v_1, \dots, v_{k+1}
des Krylow-Unterraums

$$K_{k+1}(A, v) := \text{span}\{v, Av, A^2v, \dots, A^k v\}$$

mit $k + 1 \leq n$ und $A \in \mathbb{R}^{n \times n}$

Normwise Backward Error im CG-Verfahren

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 \\ \beta_1 & \ddots \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

Eigenschaften:

- T_k ist SPD $\Rightarrow \|T_k\| = \max \sigma(T_k)$
- $\|T_k\| \leq \|T_{k+1}\|$ für $k = 1 \dots n$
- $\|T_k\| \leq \|A\|$ für $k = 1 \dots n$
- $\|T_n\| = \|A\|$

Normwise Backward Error im CG-Verfahren

CG-Verfahren

Setze $r_0 = p_0 = b - Ax_0$ für eine Startlösung x_0

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

Lanczos mit $v = r_0$

$$\Rightarrow v_{k+1} = (-1)^k \frac{r_k}{\|r_k\|}$$

Normwise Backward Error im CG-Verfahren

CG-Verfahren

Setze $r_0 = p_0 = b - Ax_0$ für eine Startlösung x_0

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

Lanczos mit $v = r_0$

$$\Rightarrow v_{k+1} = (-1)^k \frac{r_k}{\|r_k\|}$$

Lanczos-Koeffizienten:

$$\beta_k = \frac{\sqrt{\delta_k}}{\gamma_{k-1}}$$

$$\alpha_k = \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}$$

Normwise Backward Error im CG-Verfahren

$$R_k = \begin{bmatrix} \frac{1}{\sqrt{\gamma_0}} & \sqrt{\frac{\delta_1}{\gamma_0}} & & \\ & \ddots & \ddots & \\ & & \ddots & \sqrt{\frac{\delta_{k-1}}{\gamma_{k-2}}} \\ & & & \frac{1}{\sqrt{\gamma_{k-1}}} \end{bmatrix}$$

$$T_k = R_k^\top R_k$$

$$\Delta_k := \|R_k\|^2 = \|T_k\| \leq \|A\|$$

Lanczos mit $v = r_0$
 $\Rightarrow v_{k+1} = (-1)^k \frac{r_k}{\|r_k\|}$

Lanczos-Koeffizienten:

$$\beta_k = \frac{\sqrt{\delta_k}}{\gamma_{k-1}}$$

$$\alpha_k = \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}$$

Normwise Backward Error im CG-Verfahren

$$\|R_{k-1}\|^2 = \sup_{\|z\|=1, z \in \mathbb{R}^{k-1}} \|R_{k-1}z\|^2 = \|R_{k-1}z_{k-1}\|^2$$

Normwise Backward Error im CG-Verfahren

$$\|R_k\|^2 = \sup_{s^2+c^2=1} \begin{pmatrix} sz_{k-1} & c \end{pmatrix} R_k^\top R_k \begin{pmatrix} sz_{k-1} \\ c \end{pmatrix} =$$

Normwise Backward Error im CG-Verfahren

$$\Delta_k = \|R_k\|^2 = \sup_{s^2+c^2=1} \begin{pmatrix} s & c \end{pmatrix} \begin{bmatrix} \Delta_{k-1} & c_{k-1}\beta_{k-1} \\ c_{k-1}\beta_{k-1} & \alpha_k \end{bmatrix} \begin{pmatrix} s \\ c \end{pmatrix}$$



Normwise Backward Error im CG-Verfahren

$$\lambda_{max} \geq \frac{\begin{pmatrix} s & c \end{pmatrix} B_k \begin{pmatrix} s \\ c \end{pmatrix}}{\begin{pmatrix} s & c \end{pmatrix} \begin{pmatrix} s \\ c \end{pmatrix}} = \begin{pmatrix} s & c \end{pmatrix} B_k \begin{pmatrix} s \\ c \end{pmatrix}$$

nach dem Satz von Courant-Fischer.



Normwise Backward Error im CG-Verfahren

Eigenwertproblem analytisch lösen:

$$\lambda_{max} = \Delta_k = \frac{1}{2}(\Delta_{k-1} + \alpha_k + \omega_{k-1}) \quad \text{mit} \quad \omega_{k-1} = \sqrt{(\Delta_{k-1} - \alpha_k)^2 + 4\beta_{k-1}^2 c_{k-1}^2}$$

zum Eigenvektor

$$u = \begin{pmatrix} \Delta_{k-1} - \alpha_k + \omega_{k-1} \\ 2\beta_{k-1} c_{k-1} \end{pmatrix}$$

Normwise Backward Error im CG-Verfahren

Es gilt also

$$\begin{pmatrix} s_k \\ c_k \end{pmatrix} = \frac{u}{\|u\|} = \frac{1}{\sqrt{2(\omega_{k-1}^2 + (\Delta_{k-1} - \alpha_k)\omega_{k-1})}} \begin{pmatrix} \Delta_{k-1} - \alpha_k + \omega_{k-1} \\ 2\beta_{k-1}c_{k-1} \end{pmatrix}$$

und somit

$$c_k^2 = \frac{1}{2} \left(1 - \frac{\Delta_{k-1} - \alpha_k}{\omega_{k-1}} \right).$$

Normwise Backward Error im CG-Verfahren

CG-Verfahren mit inkrementeller Normabschätzung

Setze $r_0 = p_0 = b - Ax_0$ für eine Startlösung x_0

Setze $\delta_0 = \gamma_{-1} = 0$

Schleife über $k = 1, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{k-1}, x_k, r_k, \delta_k, p_k$

$$\alpha_k = \frac{\delta_{k-1}}{\gamma_{k-2}} + \frac{1}{\gamma_{k-1}}$$

$$\beta_k^2 = \frac{\delta_k}{\gamma_{k-1}^2}$$

Falls $k = 1$, dann

$c_1^2 = \Delta_1 = \alpha_1$ und Konvergenzkontrolle

sonst

$$\omega_{k-1} = \sqrt{(\Delta_{k-1} - \alpha_k)^2 + 4\beta_{k-1}^2 c_{k-1}^2}$$

$$c_k^2 = \frac{1}{2} \left(1 - \frac{\Delta_{k-1} - \alpha_k}{\omega_{k-1}} \right)$$

$\Delta_k = \Delta_{k-1} + \omega_{k-1} c_k^2$ und Konvergenzkontrolle

Koeffizienten aus CG-Verfahren

Matrixkoeffizienten von T_k
für Matrix B_k

Eigenwertproblem der Matrix B_k

Normwise Backward Error im CG-Verfahren

$$\Delta_k = \|R_k\|^2 = \|T_k\| \leq \|A\|$$

$$\Rightarrow \frac{\|r_k\|}{\|A\| \|x_k\| + \|b\|} \leq \frac{\|r_k\|}{\Delta_k \|x_k\| + \|b\|}$$



Gemischt genaue iterative Verfeinerung



Gemischt genaue iterative Verfeinerung

Iterative Verfeinerung

Schleife über $m = 1, 2, \dots$

1. Berechnung des Residuums und Konvergenzkontrolle

$$r_m = b - Ax_m$$

2. Lösen des Gleichungssystems

$$Ad_m = r_m$$

3. Addition eines Korrekturterms

$$x_{m+1} = x_m + d_m$$



Gemischt genaue iterative Verfeinerung

Iterative Verfeinerung

Schleife über $m = 1, 2, \dots$

1. Berechnung des Residuums und Konvergenzkontrolle

$$r_m = b - Ax_m \text{ in Double}$$

2. Lösen des Gleichungssystems mit CG und NBE

$$Ad_m = r_m \text{ in Single}$$

3. Addition eines Korrekturterms

$$x_{m+1} = x_m + d_m \text{ in Double}$$

Gemischt genaue iterative Verfeinerung

F ist Rückwärtsfehler

⇒ Wird $Ad_m = r_m$ in endlicher Arithmetik gelöst, so gilt $(A + F)d_m = r_m$ im exakten Sinn.

1. $r_m = b - Ax_m$
2. $(A + F)d_m = r_m$
3. $x_{m+1} = x_m + d_m$

[Wilkinson 1963]

Gemischt genaue iterative Verfeinerung

$$\begin{aligned}x_{m+1} &= x_m + d_m \\ \iff x_{m+1} &= x_m + (A + F)^{-1} r_m \\ \iff x_{m+1} - x &= [I - (A + F)^{-1} A]^m (x_1 - x)\end{aligned}$$



Gemischt genaue iterative Verfeinerung

Um eine realistische Analyse betreiben zu können, müssen Korrekturterme in allen Schritten eingeführt werden:

1. $r_m = b - Ax_m + c_m$
2. $A(I + F_m)d_m = r_m + \Delta r_m$
3. $x_{m+1} = x_m + d_m + g_m$

mit $F_m = A^{-1}\Delta A$

Gemischt genaue iterative Verfeinerung

Um nun Konvergenzaussagen machen zu können, müssen Abschätzungen für c_m und g_m gefunden werden

$$3. \quad x_{m+1} = x_m + d_m + g_m$$

Gemischt genaue iterative Verfeinerung

Um nun Konvergenzaussagen machen zu können, müssen Abschätzungen für c_m und g_m gefunden werden

$$\begin{aligned} 3. \quad & x_{m+1} = x_m + d_m + g_m \\ \Rightarrow \quad & \|g_m\| \leq \epsilon_d \|x_m + d_m\| \end{aligned}$$

Gemischt genaue iterative Verfeinerung

Um nun Konvergenzaussagen machen zu können, müssen Abschätzungen für c_m und g_m gefunden werden

$$1. \quad r_m = b - Ax_m + c_m$$

Gemischt genaue iterative Verfeinerung

1. $r_m = b - Ax_m + c_m$
2. $(A + F_m)d_m = r_m + \Delta r_m$
3. $x_{m+1} = x_m + d_m + g_m$

Gemischt genaue iterative Verfeinerung

Theorem 2 (Relativer Vorwärtsfehler). *Sei x_0, x_1, \dots eine durch iterative Verfeinerung entstandene Folge und die Abbruchschranke ϵ_{NBE} des NBE sei konstant. Wenn $0 < \epsilon\kappa(A) < \frac{1}{2}$ gilt, so folgt für den relativen Vorwärtsfehler*

$$\frac{\|x - x_{m+1}\|}{\|x\|} \leq \alpha_V \frac{\|x - x_m\|}{\|x\|} + \beta_V$$

mit

$$\alpha_V \in O(\kappa(A)\epsilon_{NBE})$$

$$\beta_V \in O(\kappa(A)\epsilon_d)$$

Gemischt genaue iterative Verfeinerung

Korollar 3 (Limes relativer Vorwärtsfehler). *Unter den Annahmen von Theorem 2 und $\alpha_V < 1$, folgt*

$$\frac{\|x - x_\infty\|}{\|x\|} \leq \frac{\beta_V}{1 - \alpha_V}$$

mit $x_\infty = \lim_{m \rightarrow \infty} x_m$.

Gemischt genaue iterative Verfeinerung

n	α_V	$\frac{\beta_V}{1-\alpha_V}$
16	2,84E-06	7,62E-14
100	1,45E-05	2,11E-12
1.024	1,32E-04	1,92E-10
10.000	1,24E-03	1,76E-08
22.500	2,77E-03	8,84E-08

α_V und Limes des relativen Vorwärtsfehlers $\frac{\beta_V}{1-\alpha_V}$ abhängig von der Problemgröße n . Zur Be- rechnung der Konstanten wurde $\epsilon_{NBE} = 1E-07$ gesetzt und die Matrix aus dem Modellproblem verwendet.

Gemischt genaue iterative Verfeinerung

Theorem 5 (Normwise Backward Error). Sei $\|F_m\| \leq \kappa(A)\epsilon_{NBE} < 1$, $x_1 = 0$ und $\alpha_V + \frac{\beta_V}{1 - \alpha_V} < 1$, so gilt

$$\frac{\|r_m\|}{\|A\| \|x_{m+1}\| + \|b\|} \leq \alpha_{NBE} \frac{\|r_m\|}{\|A\| \|x_m\| + \|b\|} + \beta_{NBE}$$

mit

$$\alpha_{NBE} \in O(\kappa(A)\epsilon_{NBE})$$

$$\beta_{NBE} \in O(\epsilon_d).$$



Gemischt genaue iterative Verfeinerung

Korollar 6 (Limes Normwise Backward Error). *Unter der Annahme von Theorem 5 und $\alpha_{NBE} < 1$, folgt*

$$\frac{\|b - Ax_\infty\|}{\|A\| \|x_\infty\| + \|b\|} \leq \frac{\beta_{NBE}}{1 - \alpha_{NBE}}$$

mit $x_\infty = \lim_{m \rightarrow \infty} x_m$.

Gemischt genaue iterative Verfeinerung

n	α_{NBE}	$\frac{\beta_{NBE}}{1-\alpha_{NBE}}$
16	1,15E-06	1,55E-14
100	5,04E-06	8,67E-14
1.024	4,42E-05	8,70E-13
10.000	4,14E-04	8,49E-12
22.500	9,25E-04	1,91E-11

α_{NBE} und Limes des NBE $\frac{\beta_{NBE}}{1-\alpha_{NBE}}$ abhängig von der Problemgröße n . Zur Berechnung der Konstanten wurde $\epsilon_{NBE} = 1\text{E}-07$ gesetzt und die Matrix aus dem Modellproblem verwendet.



Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $r_0 = b - Ax_0$ für eine Startlösung x_0

Schleife über $m = 1, 2, \dots$

$r_m = b - Ax_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{out}$

Setze $\tilde{r}_{m,0} = p_{m,0} = \tilde{r}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{r}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2} \left(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}} \right)$$

$$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2 \text{ und Konvergenzkontrolle: } \text{NBE} < \epsilon_{in}$$

$$x_{m+1} = x_m + d_m$$



Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ für eine Startlösung \mathbf{x}_0

Schleife über $m = 1, 2, \dots$

$\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{\text{out}}$

Setze $\tilde{r}_{m,0} = p_{m,0} = \tilde{r}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{r}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2} \left(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}} \right)$$

$$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2 \text{ und Konvergenzkontrolle: } \text{NBE} < \epsilon_{in}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$$

Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ für eine Startlösung \mathbf{x}_0

Schleife über $m = 1, 2, \dots$

$\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{\text{out}}$

Setze $\tilde{r}_{m,0} = p_{m,0} = \tilde{r}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{r}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2}(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}})$$

$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2$ und Konvergenzkontrolle: $\text{NBE} < \epsilon_{in}$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$$

iterative Verfeinerung



Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ für eine Startlösung \mathbf{x}_0

Schleife über $m = 1, 2, \dots$

$\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{\text{out}}$

Setze $\tilde{\mathbf{r}}_{m,0} = p_{m,0} = \tilde{\mathbf{r}}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{\mathbf{r}}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2} \left(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}} \right)$$

$$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2 \text{ und Konvergenzkontrolle: } \text{NBE} < \epsilon_{in}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$$

iterative Verfeinerung

CG-Verfahren als innerer Löser



Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ für eine Startlösung \mathbf{x}_0

Schleife über $m = 1, 2, \dots$

$\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{\text{out}}$

Setze $\tilde{r}_{m,0} = p_{m,0} = \tilde{r}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{r}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2}(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}})$$

$$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2 \text{ und Konvergenzkontrolle: } \text{NBE} < \epsilon_{in}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$$

iterative Verfeinerung

CG-Verfahren als innerer Löser

Normabschätzung von A für NBE



Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

IVNCG-Verfahren

Setze $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ für eine Startlösung \mathbf{x}_0

Schleife über $m = 1, 2, \dots$

$\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ und Konvergenzkontrolle: $\text{RR} < \epsilon_{\text{out}}$

Setze $\tilde{\mathbf{r}}_{m,0} = p_{m,0} = \tilde{\mathbf{r}}_{m,0}$

Setze $\delta_{m,0} = \gamma_{m,-1} = 0$

Schleife über $k = 1, 2, \dots$

CG-Verfahren(k) $\rightarrow \gamma_{m,k-1}, d_{m,k}, \tilde{\mathbf{r}}_{m,k}, \delta_{m,k}, p_{m,k}$

$$\alpha_{m,k} = \frac{\delta_{m,k-1}}{\gamma_{m,k-2}} + \frac{1}{\gamma_{m,k-1}}$$

$$\beta_{m,k}^2 = \frac{\delta_{m,k}}{\gamma_{m,k-1}^2}$$

Falls $k = 1$, dann

$$c_{m,1}^2 = 1$$

$$\Delta_{m,1} = \alpha_{m,1}$$

sonst

$$\omega_{m,k-1} = \sqrt{(\Delta_{m,k-1} - \alpha_{m,k})^2 + 4\beta_{m,k-1}^2 c_{m,k-1}^2}$$

$$c_{m,k}^2 = \frac{1}{2}(1 - \frac{\Delta_{m,k-1} - \alpha_{m,k}}{\omega_{m,k-1}})$$

$$\Delta_{m,k} = \Delta_{m,k-1} + \omega_{m,k-1} c_{m,k}^2 \text{ und Konvergenzkontrolle: } \text{NBE} < \epsilon_{in}$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \mathbf{d}_m$$

iterative Verfeinerung

CG-Verfahren als innerer Löser

Normabschätzung von A für NBE

Wie wählt man ϵ_{in} ?

Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

Betrachte

1. $r_m = b - Ax_m$
2. $(A + \Delta A)d_m = r_m + \Delta r_m$
3. $x_{m+1} = x_m + d_m$

mit $\|\Delta A\| \leq \epsilon_{in} \|A\|$ und $\|\Delta r_m\| \leq \epsilon_{in} \|r_m\|$

Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

Betrachte

1. $r_m = b - Ax_m$
2. $(A + \Delta A)d_m = r_m + \Delta r_m$
3. $x_{m+1} = x_m + d_m$

mit $\|\Delta A\| \leq \epsilon_{in} \|A\|$ und $\|\Delta r_m\| \leq \epsilon_{in} \|r_m\|$

Gemischt genaue iterative Verfeinerung mit Normwise Backward Error

Nimmt man $x_0 = 0$ an, so gilt

$$\frac{\|r_m\|}{\|b\|} \leq [\epsilon_{in} \left(\frac{1 + \kappa(A)}{1 - \epsilon_{in} \kappa(A)} \right)]^m.$$



Implementierung und Analyse

Übersicht – Anwendung

Hardware

Mittelklasse



High-End



Abbruchkriterium

Relatives Residuum

$$\frac{\|r_k\|}{\|b\|}$$

Normwise Backward Error

$$\frac{\|r_k\|}{\|A\| \|x_k\| + \|b\|}$$

Verfahren

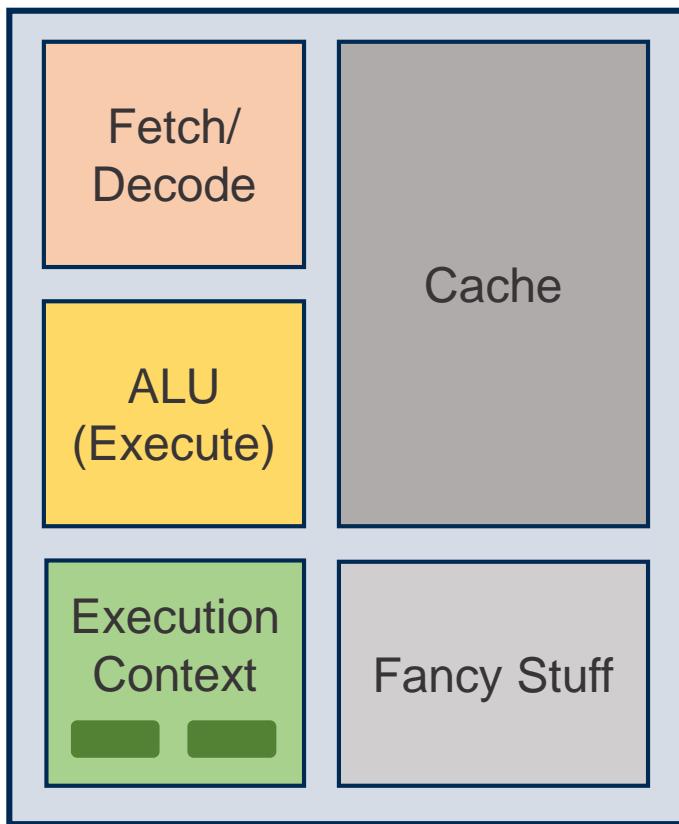
CG-Verfahren

IVNCG-Verfahren

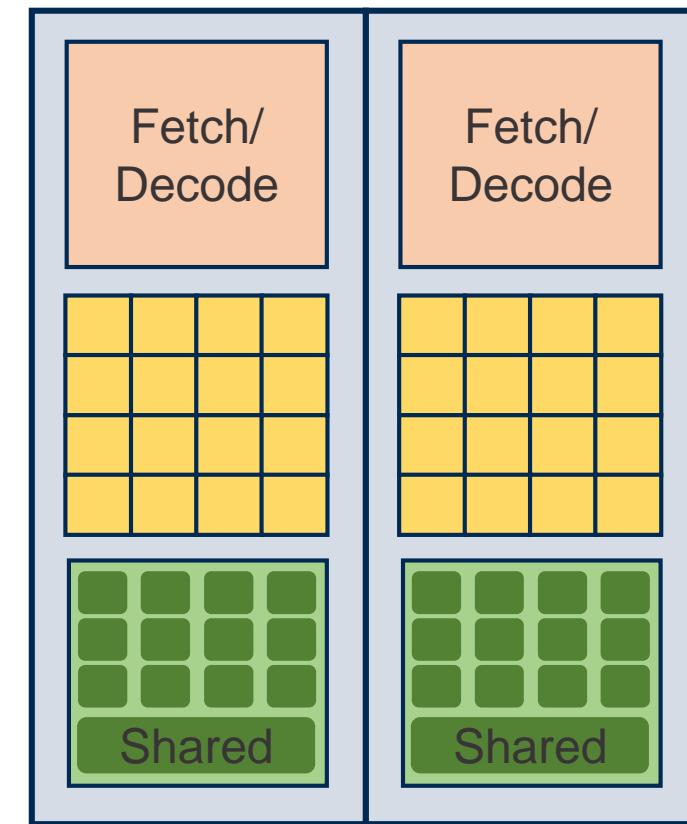
CG-Verfahren mit
gemischt genauer
Inner-Outer-Iteration

CPU vs. GPU

CPU (Single-Core)

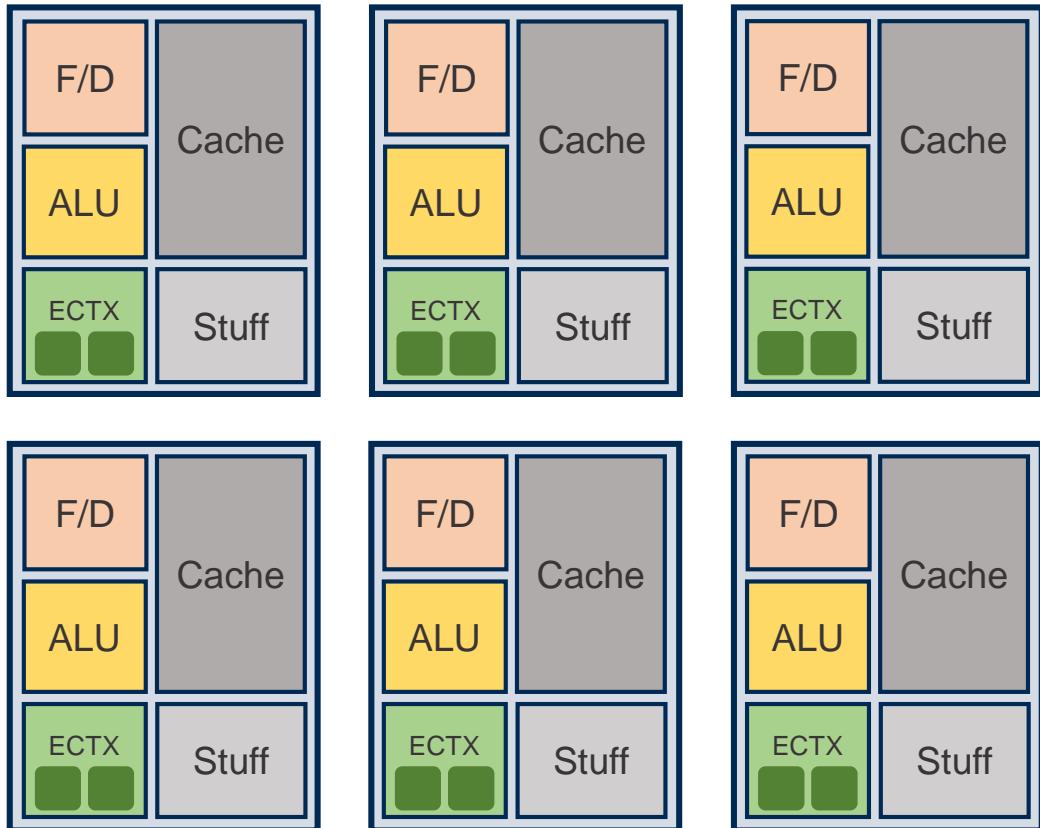


GPU (Single-SM)

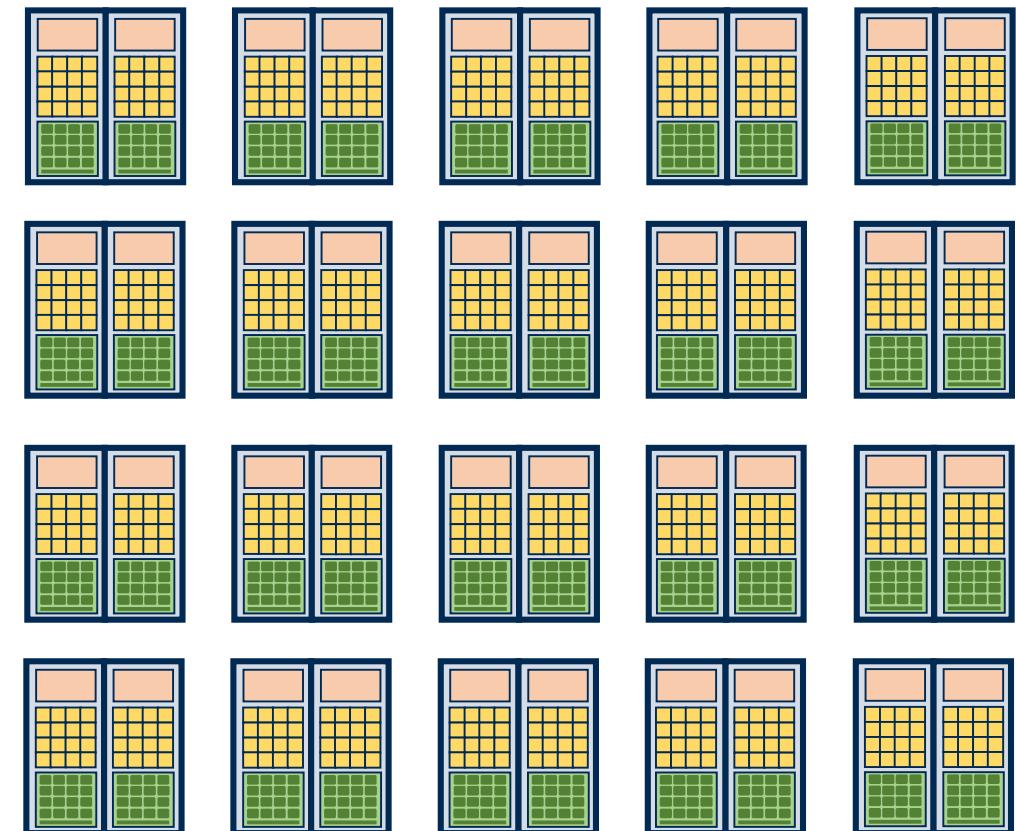


CPU vs. GPU

CPU (Multi-Core)

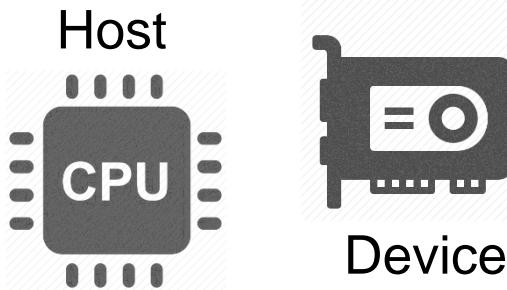


GPU (Multi-SM)

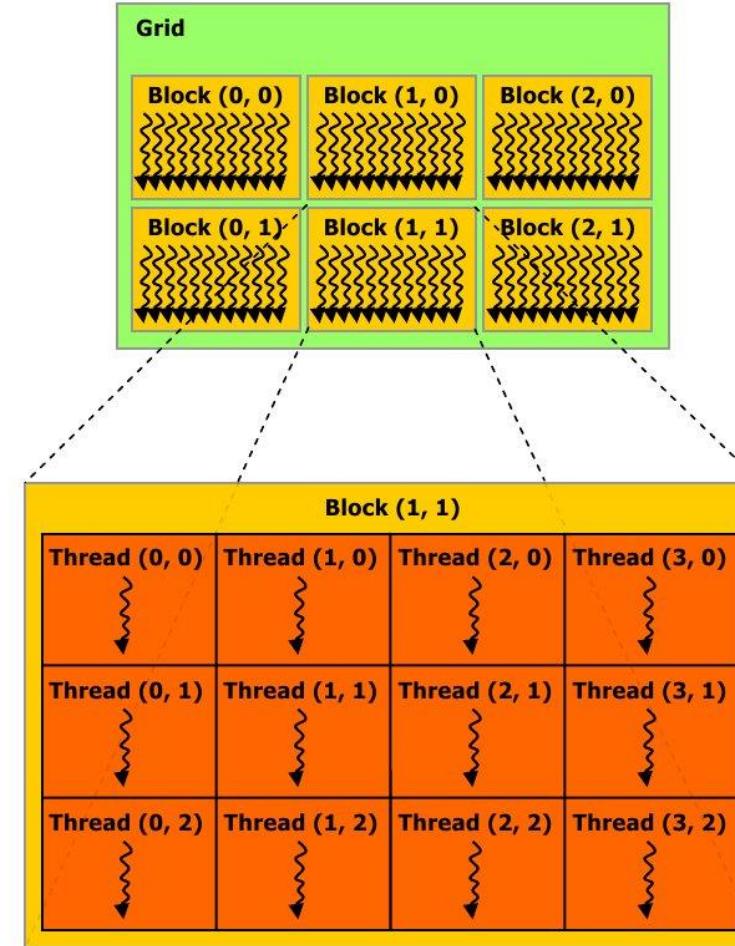


CUDA

- Erweiterung von C++
- Unterscheidung zwischen



- Device-Funktion = Kernel
- Kernel-Code → pro Thread
- Globaler Thread-Index





Parallelisierung



Parallelisierung

CG-Verfahren

Setze $r_0 = p_0 = b$

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

Konvergenzkontrolle

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$



Parallelisierung

CG-Verfahren

Setze $r_0 = p_0 = b$

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

Konvergenzkontrolle

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

- Skalierte Vektor-Vektor-Operation (AXPY)

Parallelisierung

CG-Verfahren

Setze $r_0 = p_0 = b$

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

Konvergenzkontrolle

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

- Skalierte Vektor-Vektor-Operation (AXPY)
- Matrix-Vektor-Multiplikation (MATVEC)

Parallelisierung

CG-Verfahren

Setze $r_0 = p_0 = b$

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

Konvergenzkontrolle

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

- Skalierte Vektor-Vektor-Operation (AXPY)
- Matrix-Vektor-Multiplikation (MATVEC)
- Skalarprodukt (DOT)

Parallelisierung

CG-Verfahren

Setze $r_0 = p_0 = b$

Schleife über $k = 1, 2, \dots$

$$\gamma_{k-1} = \frac{\langle r_{k-1}, r_{k-1} \rangle_2}{\langle p_{k-1}, Ap_{k-1} \rangle_2}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} Ap_{k-1}$$

Konvergenzkontrolle

$$\delta_k = \frac{\langle r_k, r_k \rangle_2}{\langle r_{k-1}, r_{k-1} \rangle_2}$$

$$p_k = r_k + \delta_k p_{k-1}$$

- Skalierte Vektor-Vektor-Operation (AXPY)
- Matrix-Vektor-Multiplikation (MATVEC)
- Skalarprodukt (DOT)
- Skalare Operationen → CPU



Parallelisierung

Iterative Verfeinerung

Schleife über $m = 1, 2, \dots$

1. Berechnung des Residuums

$$r_m = b - Ax_m$$

Konvergenzkontrolle

2. Lösung des Gleichungssystems

$$Ad_m = r_m$$

3. Addition eines Korrekturterms

$$x_{m+1} = x_m + d_m$$

- Skalierte Vektor-Vektor-Operation (AXPY)
- Matrix-Vektor-Multiplikation (MATVEC)
- Skalarprodukt (DOT)
- Skalare Operationen → CPU
- Residuenberechnung (DEFECT)

Parallelisierung

Iterative Verfeinerung

Schleife über $m = 1, 2, \dots$

1. Berechnung des Residuums

$$r_m = b - Ax_m$$

Konvergenzkontrolle

2. Lösung des Gleichungssystems

$$Ad_m = r_m$$

3. Addition eines Korrekturterms

$$x_{m+1} = x_m + d_m$$

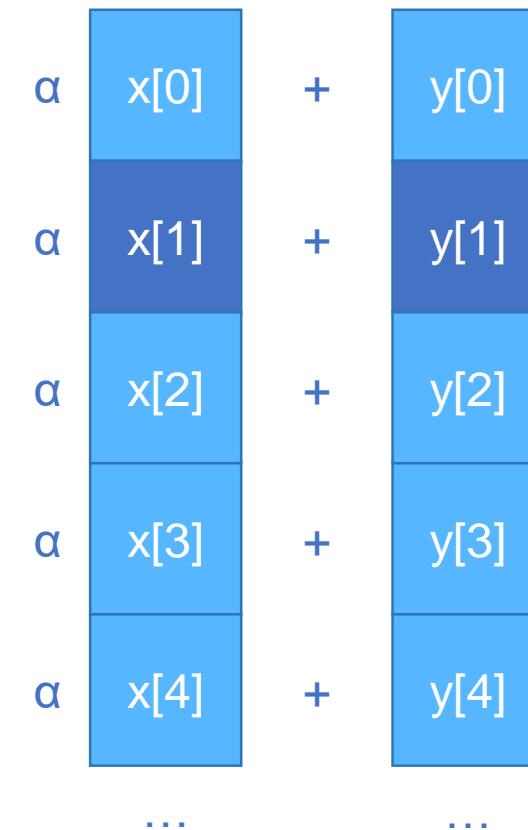


- Skalierte Vektor-Vektor-Operation (AXPY)
- Matrix-Vektor-Multiplikation (MATVEC)
- Skalarprodukt (DOT)
- Skalare Operationen → CPU
- Residuenberechnung (DEFECT)
- Konvertierung implizit in den Kernen

AXPY-Kernel: $\alpha \cdot x + y$

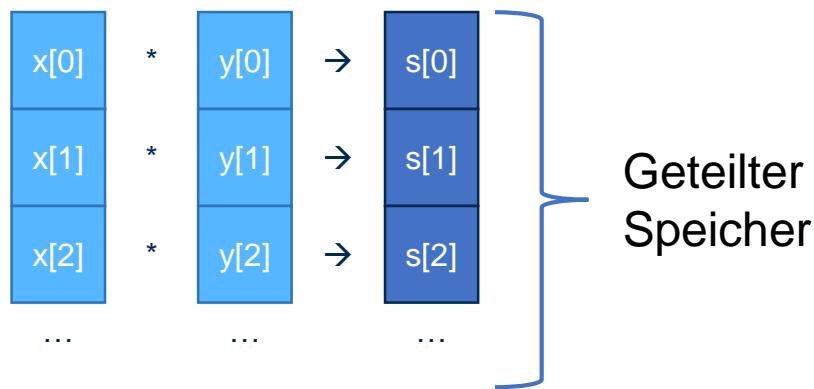
OpenMP Implementierung:

```
--global--  
void axpy(double a, double *x,  
          double *y, int n, double *z)  
{  
    int idx = blockDim.x * blockIdx.x  
            + threadIdx.x;  
    if (idx < n)  
    {  
        z[idx] = a * x[idx] + y[idx];  
    }  
}  
...  
axpy<<<grid, block>>>(a, x, y, n, z);  
...
```

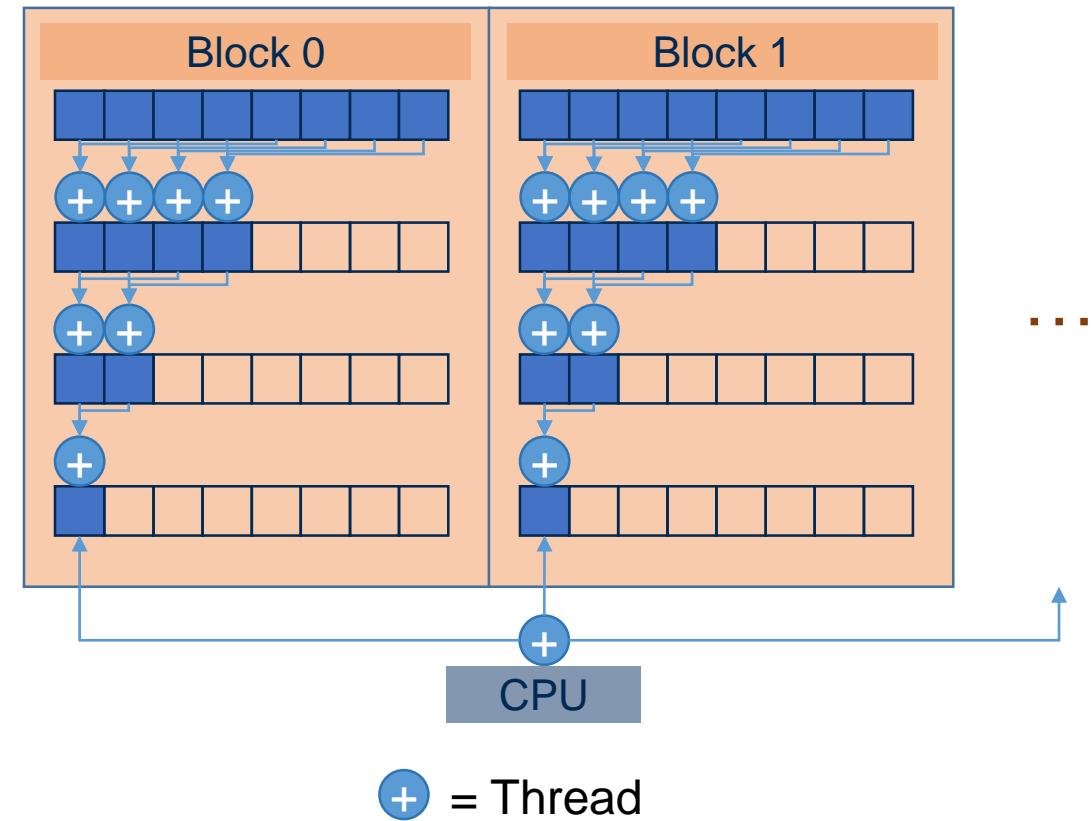


DOT-Kernel: $\sum_{i=1}^n x_i y_i$

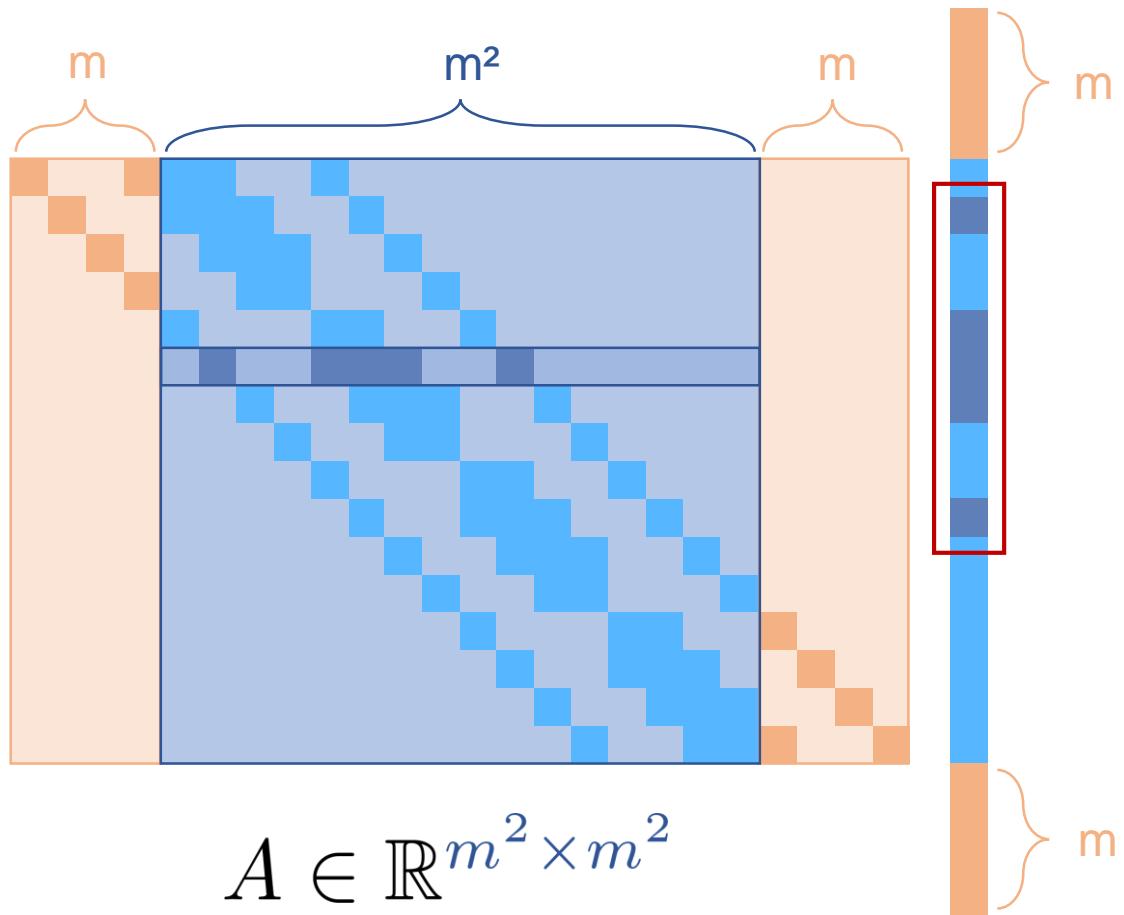
■ 1. Schritt



■ 2. Schritt



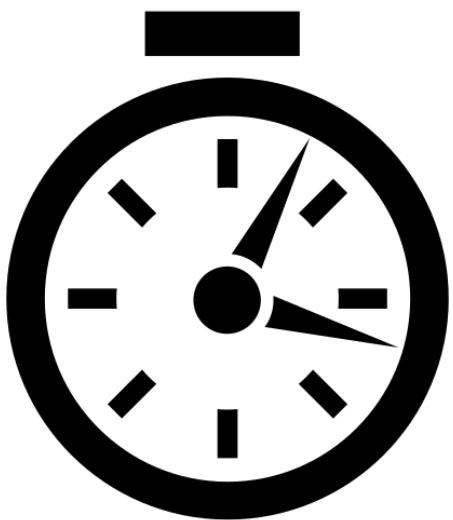
MATVEC-Kernel: Ax



- Koeffizientenvektor nicht linear
→ geteilter Speicher
- Formal $A \in \mathbb{R}^{m^2 \times (m^2 + 2 \cdot m)}$
→ zeilenweise Berechnung

Mess-Metriken

Ausführungszeit



Datendurchsatz

$$\frac{\text{Datenmenge}}{\text{Zeit}}$$

gemessen in

$$\frac{\text{GByte}}{\text{s}}$$

Rechendurchsatz

$$\frac{\# \text{ Operationen}}{\text{Zeit}}$$

gemessen in

$$\frac{\text{GFLOP}}{\text{s}}$$

Testsysteme

Intel Core i7-5820K



- Hexa-Core
- Hyper-Threading
- 3,3 GHz
- 68 GByte/s

NVIDIA GTX 960



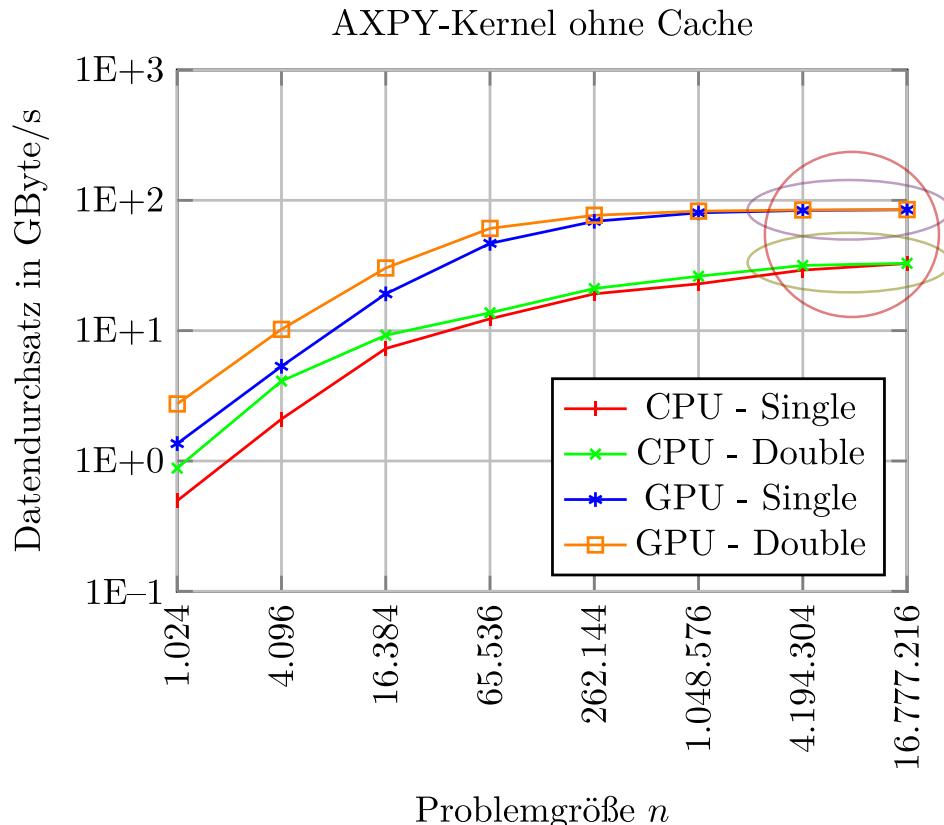
- 8 SM
- 1.024 CUDA-Cores
- 1,1 GHz
- 112 GByte/s
- 2,3 TFLOP/s SP
- DP = 1/32 SP

NVIDIA Tesla P100



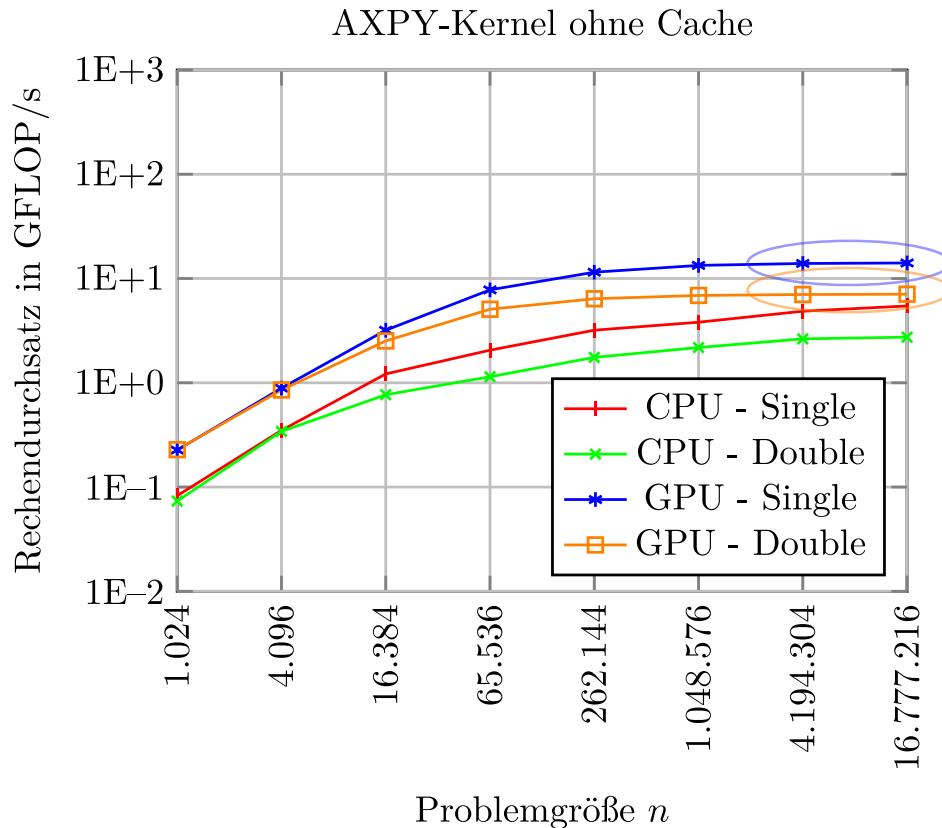
- 56 SM
- 3.584 CUDA-Cores
- 1,3 GHz
- 720 GByte/s
- 10,6 TFLOP/s SP
- DP = 1/2 SP

Datendurchsatz des AXPY-Kernels



- Intel Core i7 ↔ GTX 960
- Kein Cache → glatter Verlauf
- Höherer Datendurchsatz auf GPU
- Double dauerhaft über Single
- n groß → Double = Single
- GPU: 84,86 GByte/s → 76 %
- CPU: 32,64 GByte/s → 48 %

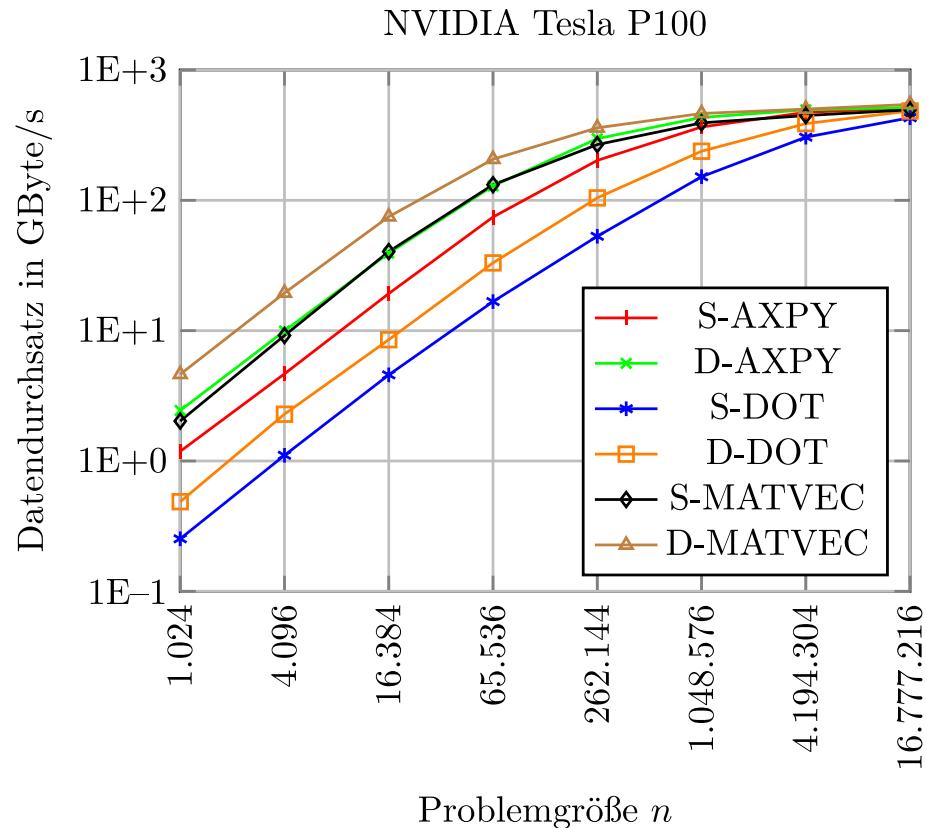
Rechendurchsatz des AXPY-Kernels



- Intel Core i7 ↔ GTX 960
- Höherer Rechendurchsatz auf GPU
- Gespiegelt mit Datendurchsatz
- Single dauerhaft über Double
- n groß → Double = 1/2 Single
- **GPU-Single: 14 GFLOP/s → < 1 %**
- **GPU-Double: 7 GFLOP/s → < 1 %**

Warum nicht Double = 1/32 Single?
→ AXPY-Kernel speicher gebunden

Datendurchsatz auf der Tesla P100



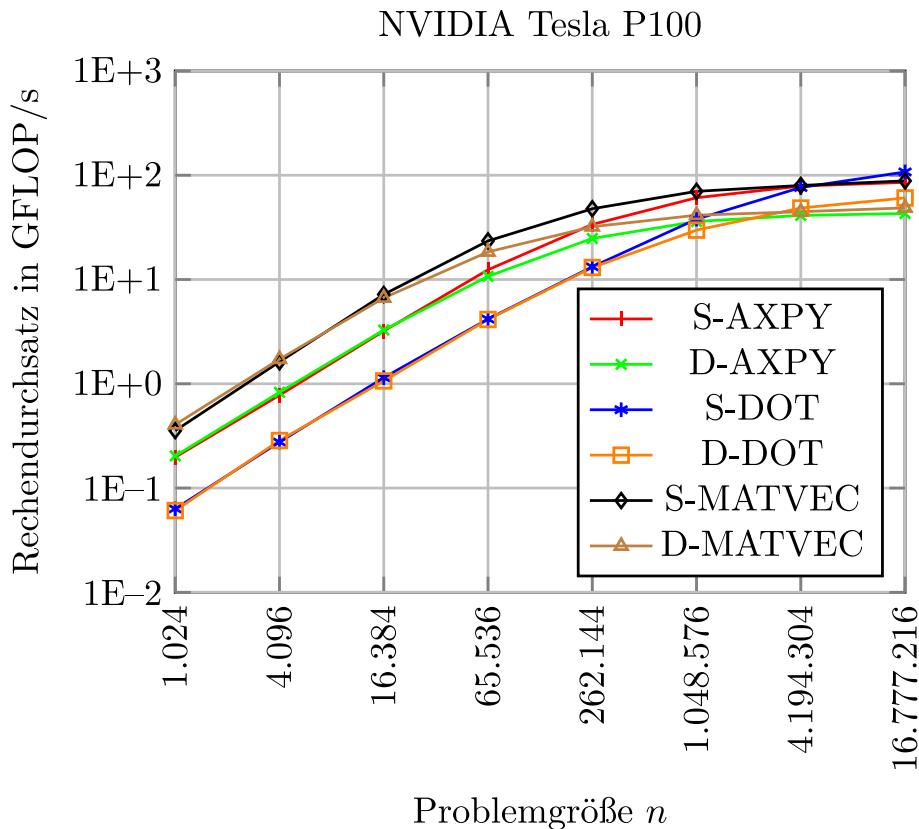
- Kurven fallen später ab

	Tesla P100	GTX 960
SM	56	8
CUDA-Cores	3.584	1.024

→ Aufwand besser verteilbar

- D-AXPY: 515,69 GByte/s → 72 %
- S-DOT: 430,00 GByte/s → 60 %

Rechendurchsatz auf der Tesla P100



- n sehr groß → Double = 1/2 Single
→ P100 mehr als 6x schneller als GTX 960

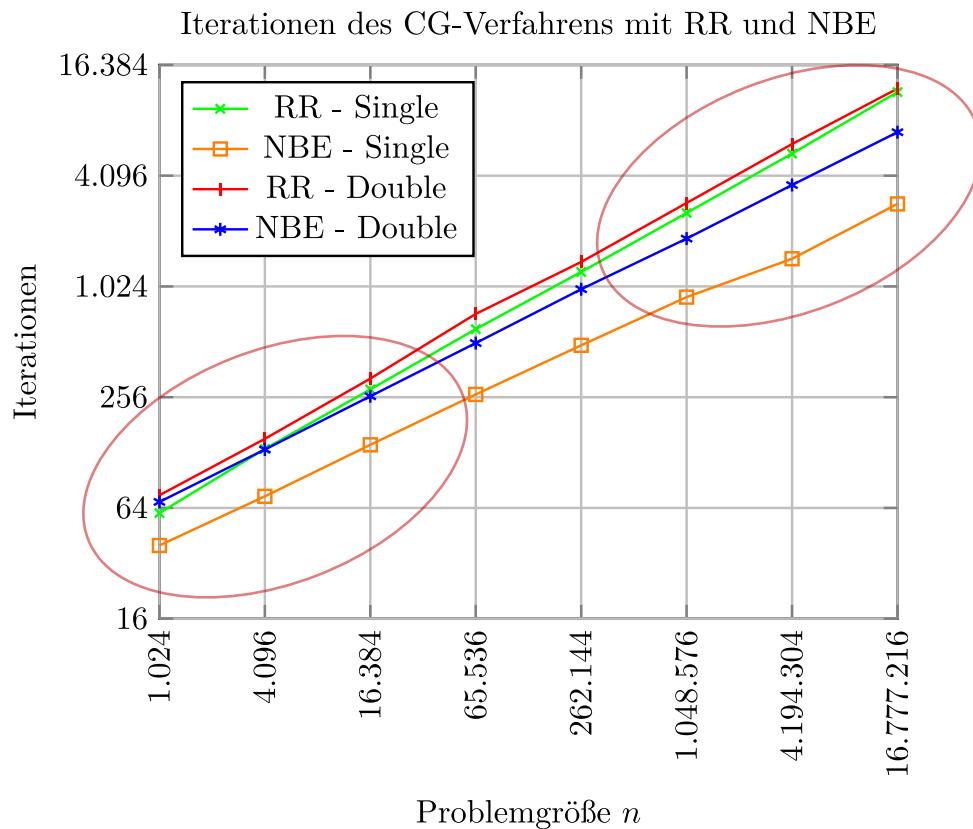
	Tesla P100	GTX 960
Theor. Daten-durchsatz	720 GByte/s	112 GByte/s

- Ausnahme: DOT-Kernel



CG-Verfahren und NBE

RR vs. NBE



$$\text{NBE}(x_k) = \frac{\|r_k\|}{\|A\| \|x_k\| + \|b\|} \leq \frac{\|r_k\|}{\|b\|} = \text{RR}(x_k)$$

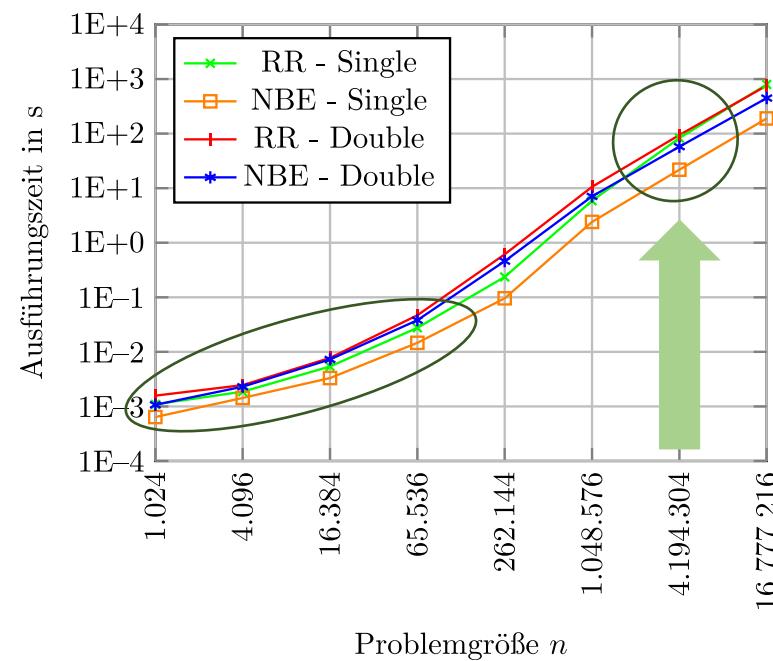
- $\epsilon_{Single} = 1E-08$
- $\epsilon_{Double} = 1E-16$
- $n \text{ klein}$ { ~45 % Gewinn bei Single
~10 % Gewinn bei Double }
- $n \text{ groß}$ { ~75 % Gewinn bei Single
~40 % Gewinn bei Double }

→ Kein zeitlicher Gewinn!

ABER: $n \text{ groß} \rightarrow \text{Berechnung von } \|x_k\| \text{ vernachlässigbar}$

Laufzeitanalyse des CG-Verfahrens

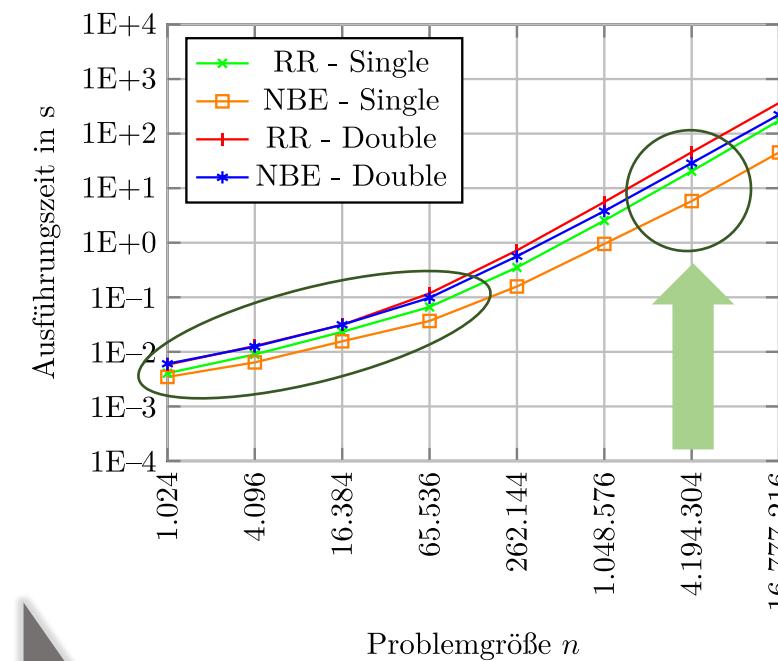
Intel Core i7-5820K



NBE – Single: 22 Sek.
RR – Double: 95 Sek.

2x bis 4x schneller

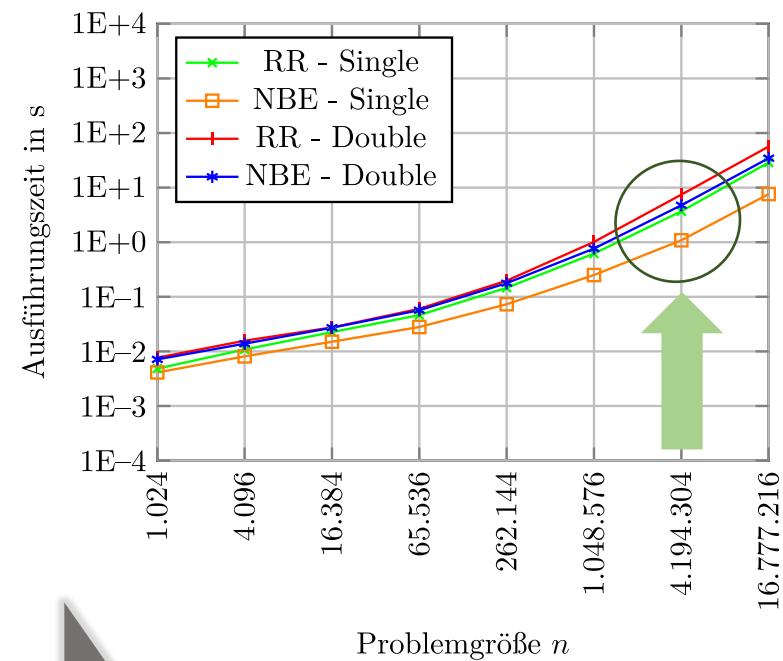
NVIDIA GTX 960



NBE – Single: 6 Sek.
RR – Double: 46 Sek.

5x bis 7x schneller

NVIDIA Tesla P100



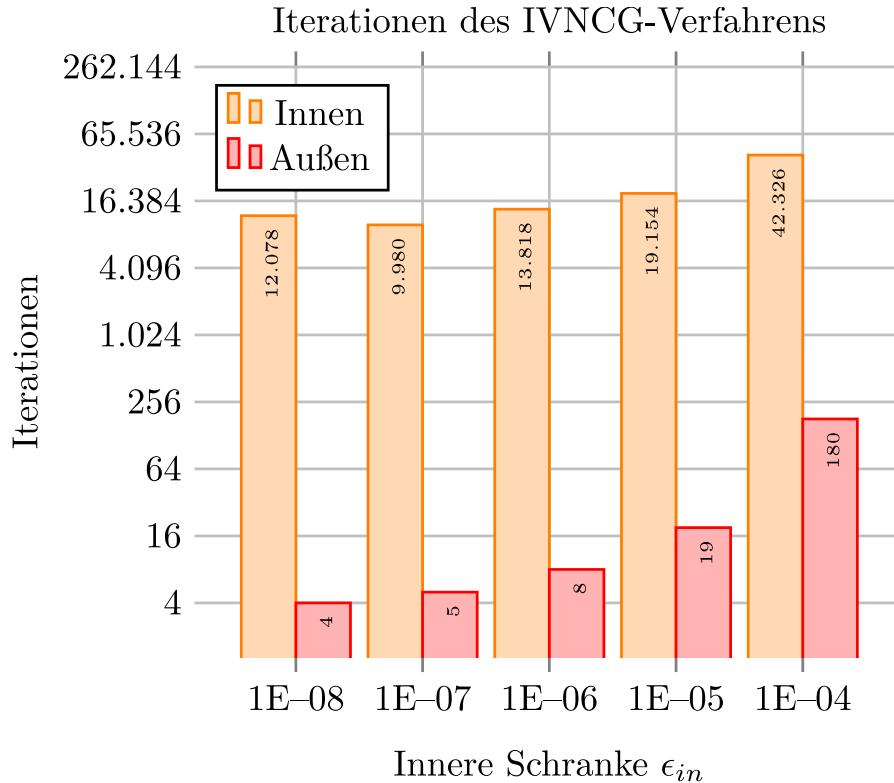
NBE – Single: 1 Sek.
RR – Double: 7 Sek.





IVNCG-Verfahren

Laufzeitanalyse des IVNCG-Verfahrens



- Keine Konvergenz für $\epsilon_{out} < 1E-10$
- $\epsilon_{out} = 1E-10$
- $n = 4.194.304$
- # Äußerer Durchläufe steigt monoton
- # Innerer Durchläufe: Minimum bei $\epsilon_{in} = 1E-07$

	GTX 960	Tesla P100
CG	28 Sek.	5 Sek.
IVNCG	40 Sek.	8 Sek.



CG-Verfahren mit gemsicht genauer Inner-Outer-Iteration



CG-Verfahren mit gemischt genauer Inner-Outer-Iteration

$$Ax = b \quad \Rightarrow \quad K^\top AKy = K^\top b \text{ mit } y = K^{-1}x$$

Wenn $KK^\top = M^{-1}$, dann ist $K^\top AK$ spd und das CG-Verfahren anwendbar.

Für $M^{-1} = A^{-1}$ folgt die Konvergenz in einem Schritt.

Vorkonditioniertes CG-Verfahren

Setze $r_0 = b$

Berechne $p_0 = M^{-1}r_0$

Berechne $\alpha_0 = \langle r_0, p_0 \rangle_2$

Schleife über $m = 0, 1, \dots$

$$v_m = Ap_m$$

$$\lambda_m = \alpha_m / \langle v_m, p_m \rangle_2$$

$$x_{m+1} = x_m + \lambda_m p_m$$

$r_{m+1} = r_m - \lambda_m v_m$ und Konvergenzkontrolle

$$z_{m+1} = M^{-1}r_{m+1}$$

$$\alpha_{m+1} = \langle r_{m+1}, z_{m+1} \rangle_2$$

$$p_{m+1} = z_{m+1} + (\alpha_{m+1} / \alpha_m) p_m$$



CG-Verfahren mit gemischt genauer Inner-Outer-Iteration

$$Ax = b \quad \Rightarrow \quad K^\top AKy = K^\top b \text{ mit } y = K^{-1}x$$

Wenn $KK^\top = M^{-1}$, dann ist $K^\top AK$ spd und das CG-Verfahren anwendbar.

Für $M^{-1} = A^{-1}$ folgt die Konvergenz in einem Schritt.

Wir wählen $M^{-1} = A_{Single}^{-1}$.

CG-Verfahren mit gemischt genauer Inner-Outer-Iteration

Setze $r_0 = b$

Löse $Ap_0 = r_0$ mit CG-Verfahren und NBE $< \epsilon_{in}$

Berechne $\alpha_0 = \langle r_0, p_0 \rangle_2$

Schleife über $m = 0, 1, \dots$

$$v_m = Ap_m$$

$$\lambda_m = \alpha_m / \langle v_m, p_m \rangle_2$$

$$\mathbf{x}_{m+1} = \mathbf{x}_m + \lambda_m \mathbf{p}_m$$

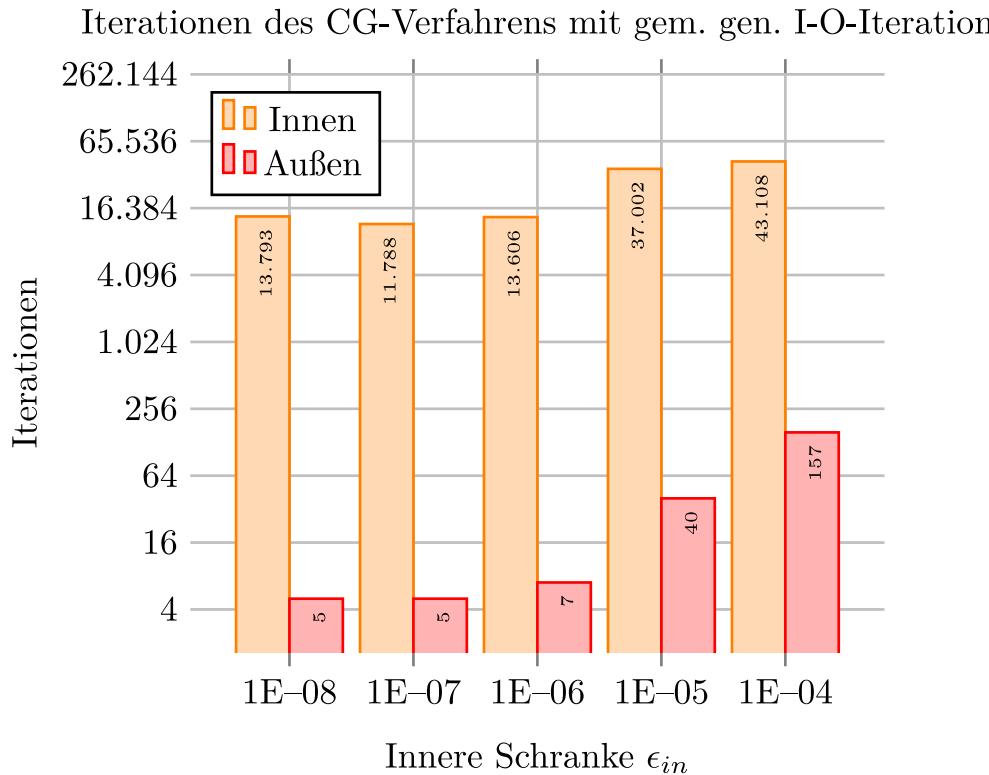
$\mathbf{r}_{m+1} = \mathbf{r}_m - \lambda_m \mathbf{v}_m$ und Konvergenzkontrolle: $\mathbf{R}\mathbf{R} < \epsilon_{out}$

Löse $Az_{m+1} = r_{m+1}$ mit CG-Verfahren und NBE $< \epsilon_{in}$

$$\alpha_{m+1} = \langle r_{m+1}, z_{m+1} \rangle_2$$

$$p_{m+1} = z_{m+1} + (\alpha_{m+1} / \alpha_m) p_m$$

CG-Verfahren mit gemischt genauer Inner-Outer-Iteration



- Abbruchschranke $\epsilon_{out} = 1E-10$
- $n = 4.194.304$
- Verhalten wie beim IVNCG-Verfahren
- Aber: mehr innere Durchläufe

	GTX 960	Tesla P100
CG	28 Sek.	5 Sek.
IVNCG	40 Sek.	8 Sek.
CG-IOT	47 Sek.	9 Sek.

- Allerdings: Konvergenz für $\epsilon_{out} < 1E-10$



Fazit

- Korollar 6 → garantie Genauigkeit für den NBE möglich
- ABER: in hoher Dimension weit weg von Double Genauigkeit

$$\frac{\|b - Ax_\infty\|}{\|A\| \|x_\infty\| + \|b\|} \leq \frac{\beta_{NBE}}{1 - \alpha_{NBE}}$$

- Für Konvergenz muss ϵ_{in} klein genug sein:

$$\frac{1}{1 + 2\kappa(A)} > \epsilon_{in}$$



Fazit

- Mit NBE wird deutlich früher abgebrochen
- Umstieg auf Grafikkarte lohnt sich erst ab mittleren Problemgrößen
- Auf High-End-Grafikkarten lohnt sich gemischte Genauigkeit erst bei großen Problemen
- IVNCG-Verfahren keine Alternative zum CG-Verfahren
- CG-Verfahren mit gem. gen. I-O-Iteration keine Alternative zum CG-Verfahren

ABER: vergleichbare Genauigkeit



Ausblick

- Inkrementelle Normabschätzung macht NBE außen nutzbar
- 1. und 3. Schritt von IVNCG mit **Quadruple**-Datentyp → bessere Genauigkeit
- 2. Schritt mit **Half**-Datentyp → kürzere Laufzeit

- Optimierung der CUDA-Blockgröße
- Verwendung mehrerer CUDA-Streams

Literatur

- Magnus R. Hestens und Eduard Stiefel: *Methods of Conjugate Gradients for Solving Linear Systems*. Journal of Research of the National Bureau of Standards, 49(6):409-436, Dezember 1952.
- Petr Tichy: *On error estimation in the conjugate gradient method: Normwise backward error*. Proceedings of the Conference Algoritmy, Seiten 323–332, 2016.
- J. L. Rigal und J. Gaches: *On the Compatibility of a Given Solution With the Data of a Linear System*. J. ACM, 14(3):543–548, Juli 1967, ISSN 0004-5411.
- Gérard Meurant: *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*, Band 19 der Reihe Software, Environments, and Tools. SIAM, 2006.
- I.S. Duff und C. Vömel: *Incremental Norm Estimation for Dense and Sparse Matrices*. BIT Numerical Mathematics, 42(2):300 – 322, 2002.
- James H. Wilkinson: *Rundungsfehler*. pub-Springer, pub-Springer:adr-B, 1969.



Literatur

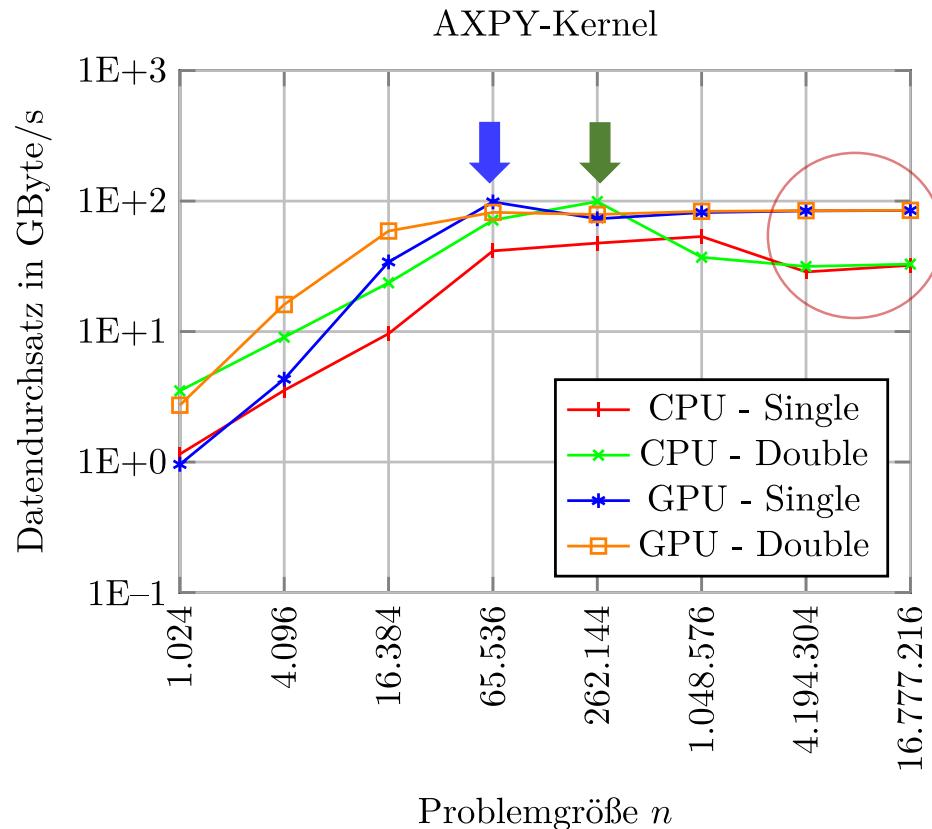
- Dominik Göddeke und Malte Shirwon: *GPU Programming with CUDA*. Doktorandenkurs, September 2016.
- NVIDIA Corporation: *CUDA Toolkit Documentation v9.2.148*. <https://docs.nvidia.com/cuda/index.html>. Abgerufen: Juli 2018.
- Dominik Göddeke: *Wissenschaftliches Rechnen*. Vorlesungsskript, März 2017.
- Dominik Göddeke: *Fast and Accurate Finite-Element Multigrid Solvers for PDE Simulations on GPU Clusters*. Dissertation, Technische Universität Dortmund, Februar 2010.
- NVIDIA Corporation: *GeForce GTX 960 - Technische Daten*. <https://www.nvidia.de/object/geforce-gtx-960-de>. Abgerufen: Juli 2018.
- Purch Group: AnandTech High Tech Online Magazine: *NVIDIA Launches GeForce GTX 960*. <https://www.anandtech.com/show/8923/nvidia-launches-geforce-gtx-960>. Abgerufen: Juli 2018.
- NVIDIA Corporation: *NVIDIA Pascal Architektur-Whitepaper*. <https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf>. Abgerufen: August 2018.



Literatur

- Purch Group: AnandTech High Tech Online Magazine: *NVIDIA Announces Tesla P100 Accelerator.* <https://www.anandtech.com/show/10222/nvidia-announces-tesla-p100-accelerator-pascal-power-for-hpc>. Abgerufen: August 2018.
- Intel Corporation: *Intel Core i7-5820K Prozessor.* https://ark.intel.com/de/products/82932/Intel-Core-i7-5820K-Processor-15M-Cache-up-to-3_60-GHz. Abgerufen: Juli 2018.

Datendurchsatz des AXPY-Kernels



- Intel Core i7 ↔ GTX 960
- Mittel über 100 Durchläufe
- n klein → Double über Single
- CPU-Double: Maximum 99,13 GByte/s
→ $2 \cdot 262.144 \cdot 8 \text{ Byte} \approx 4,19 \text{ MByte} < 15 \text{ MByte}$
- GPU-Single: Maximum 98,53 GByte/s
→ $2 \cdot 65.536 \cdot 4 \text{ Byte} \approx 0,53 \text{ MByte} < 1 \text{ Mbyte}$
- n groß → Double = Single



Normwise Backward Error

$$\frac{\|r_m\|}{\|b\|}$$

„Das relative Residuum ist kein zuverlässiges Maß für die Exaktheit der approximierten Lösung“ [Hestenes, Stiefel 1952]

Gemischt genaue iterative Verfeinerung

Theorem 5 (Relativer Rückwärtsfehler). Sei $\|F_m\| \leq \kappa(A)\epsilon < 1$, $x_1 = 0$ und $\alpha_V + \frac{\beta_V}{1 - \alpha_V} < 1$, so gilt für den relativen Rückwärtsfehler

$$\frac{\|b - Ax_{m+1}\|}{\|A\| \|x_{m+1}\|} \leq \alpha_R \frac{\|b - Ax_m\|}{\|A\| \|x_m\|} + \beta_R$$

mit

$$\alpha_R \in O(\kappa(A)\epsilon)$$

$$\beta_R \in O(\epsilon_d)$$

Gemischt genaue iterative Verfeinerung

Korollar 6 (Limes relativer Rückwärtsfehler). *Unter der Annahme von Theorem 6 und $\alpha_R < 1$, folgt*

$$\frac{\|b - Ax_\infty\|}{\|b\| \|Ax_\infty\|} \leq \frac{\beta_R}{1 - \alpha_R}$$

mit $x_\infty = \lim_{m \rightarrow \infty} x_m$.



Gemischt genaue iterative Verfeinerung

n	α_R	$\frac{\beta_R}{1-\alpha_R}$
16	2,09E-06	1,58E-14
100	9,87E-06	8,70E-14
1.024	8,83E-05	8,71E-13
10.000	8,27E-04	8,49E-12
22.500	1,84E-03	1,91E-11

α_R und Limes des relativen Rückwärtsfehlers $\frac{\beta_R}{1-\alpha_R}$ abhängig von der Problemgröße n . Zur Berechnung der Konstanten wurde $\epsilon = 1\text{E}-07$ gesetzt und die Matrix aus dem Modellproblem verwendet.

RR vs. NBE

n	$\Delta \approx \ A\ $	$\ x\ $
1.024	7,5E+03	6,7E-01
4.096	3,0E+04	2,6E-01
16.384	1,2E+05	1,0E-04
65.536	4,8E+05	4,1E-04
262.144	1,9E+11	1,6E-03
1.048.576	7,7E+11	6,5E-03
4.194.304	3,1E+12	2,6E-02
16.777.216	1,3E+13	1,0E-01

$$\text{NBE}(x_k) = \frac{\|r_k\|}{\|A\| \|x_k\| + \|b\|} \leq \frac{\|r_k\|}{\|b\|} = \text{RR}(x_k)$$

RR vs. NBE

Fehlerentwicklungen für $n = 4.194.304$ 