Iterative Gleichungssystemlöser und parallele Algorithmen
Hamburg, 19.–23. Februar 2001

Vorkonditionierungstechniken für unstrukturierte und strukturierte Probleme

- Einleitung
  - Direkte Methoden
  - Iterative Verfahren, Vorkonditionierung
- Unstruktuierte Probleme
  - Unvollständige Dreieckszerlegungen
  - Approximative Inverse
- Strukturierte Probleme (partielle Dgln.)
  - Einfache Lösungstechniken
  - Mehrgitterverfahren
  - Algebraische Mehrgitterverfahren

Direkte Löser — Beispiel

$$A = \begin{pmatrix} * & * & * & \cdots & * \\ * & * & & & \\ * & & * & & \\ \vdots & & & \ddots & \\ * & & & & * \end{pmatrix}$$

Gauß–Algorithmus (LU–Zerlegung) ohne Vertauschung liefert

$$A = \begin{pmatrix} * & & & & 0 \\ * & * & & & \\ * & * & * & & \\ \vdots & \vdots & \ddots & \ddots & \\ * & * & * & \cdots & * \end{pmatrix} \begin{pmatrix} * & * & * & \cdots & * \\ & * & * & \cdots & * \\ & & * & \ddots & * \\ & & & \ddots & \vdots \\ 0 & & & & * \end{pmatrix}$$

Faktoren sind voll! Riesiger Speicher– und Rechenzeitbedarf!

Direkte Löser — Beispiel

Hier. Problem behebbar durch Umordnen

$$A \longrightarrow \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & \ddots & & \vdots \\ & & & * & * \\ * & * & \cdots & * & * \end{pmatrix}, \quad A = \begin{pmatrix} * & & & & 0 \\ & * & & & \\ & & * & & \\ & & & \ddots & \\ * & * & * & \cdots & * \end{pmatrix} \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & \ddots & \vdots \\ 0 & & & & * \end{pmatrix}$$

Leider: Umordnen hilft nicht immer, z.B.

Matrix LNS3937 aus der Harwell–Boeing–Collection.

Linearisierte kompressible Navier–Stokes–Gleichung

$n = 3937$. Anzahl Einträge: $nnz(A) = 25407$.

LU–Zerlegung mit MATLAB (enthält Vertauschungen, um fill–in gering zu halten)

Algorithmus liefert Faktoren L, U mit insgesamt $nnz(L + U) = 1162464$, das 46–fache!

Iterative Löser — Beispiel

Wozu Vorkonditionierung?

Beispiel. Matrix WEST0067 aus der Harwell–Boeing–Collection.

$n = 67$, Anzahl Einträge $nnz(A) = 294$.

Einige iterative Verfahren. Abbruch, falls $||Ax - b|| \leqslant 10^{-8}$, Maximal 77 Schritte

Iter. Verfahren	Anz. Iterationen
BICG	keine Konvergenz
CGS	keine Konvergenz
BICGSTAB	keine Konvergenz
QMR	keine Konvergenz
GMRES(20)	keine Konvergenz
GMRES	67

Iterative Verfahren funktionieren nur für “gutartige” Matrizen

Also $B \approx A^{-1}$, $Ax = b \longrightarrow (BA)x = Bb$. Verwende $BA \approx I$ statt A.

Unvollständige Dreieckszerlegung

Verwende Gauß–Algorithmus (LU–Zerlegung), lasse dabei einiges weg.

Strategien Einträge wegzulassen.

- Nach Größe der Einträge
- Nach Besetzungsmuster der Matrix
- Vorhandene Strukturen (Symmetrie, Positivdefinitheit, Diagonaldominanz, ...) sollten ausgenutzt und erhalten werden

Wichtig. Effiziente Algorithmen benötigen adäquate Implementierung (Datenstrukturen), sonst langsam (Rechenzeit, Implementation)

Deshalb: Umordnen der Matrix zur Reduktion von fill–in unüblich

Im folgenden: Matrix A zeilenweise gespeichert, nur die Nichtnulleinträge

$$A = \begin{pmatrix} 2 & -1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -7 & 0 & 6 \end{pmatrix}. \text{ Speichere } \begin{array}{ccc|cc|c|cc|l} 1 & 4 & 6 & 7 & 9 & & & & \text{Anfang Zeile 1–5} \\ 1 & 2 & 4 & 1 & 4 & 3 & 2 & 4 & \text{Spaltenindizes} \\ \hline 2 & -1 & 1 & 2 & 1 & -1 & -7 & 6 & \text{numerische Werte} \end{array}$$

(Un)vollständige Dreieckszerlegung

Angepasste LU–Zerlegung bei zeilenweiser Speicherung von A

- Zugriff zeilenweise
- Elimination in einer Zeile von links nach rechts

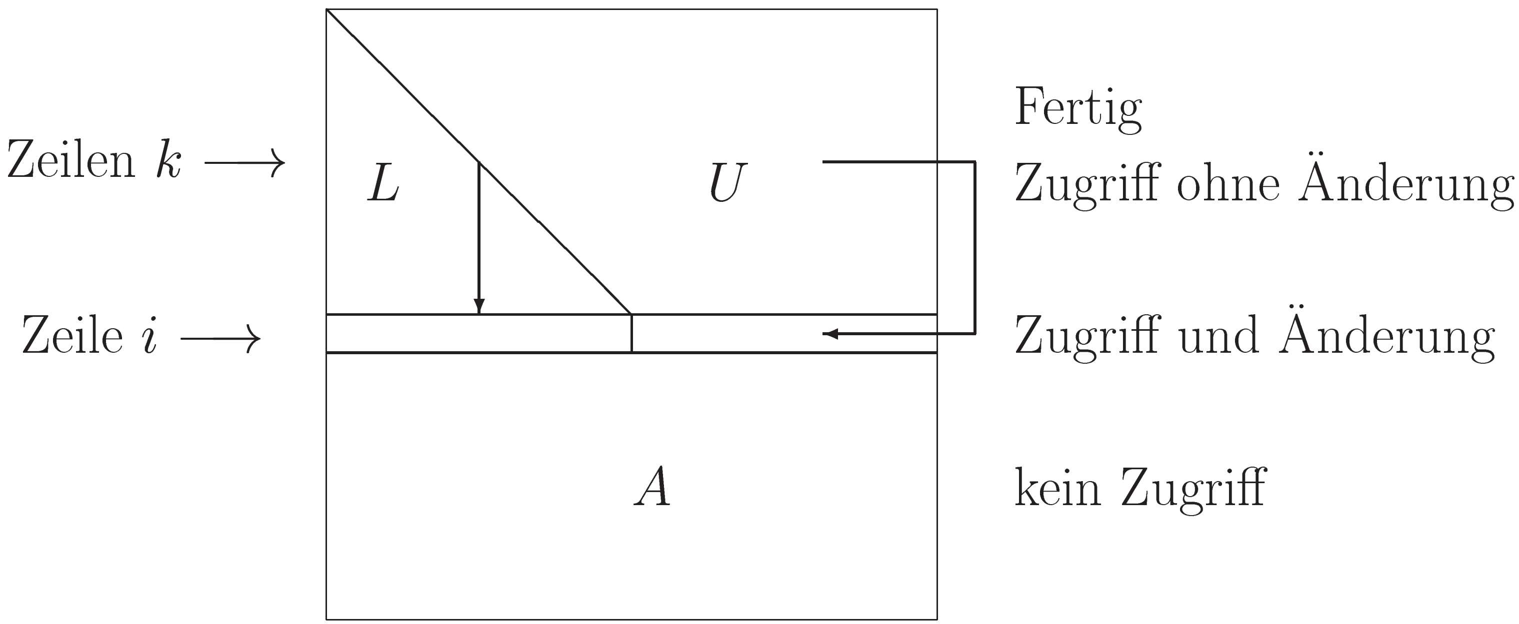
Illustration Eliminationsreihenfolge

$$\left(\begin{array}{cccc} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{array}\right) \to \left(\begin{array}{cccc} * & * & * & * \\ \hline 0 & * & * & * \\ \hline * & * & * & * \\ * & * & * & * \end{array}\right) \to \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \\ \hline 0 & 0 & * & * \\ \hline * & * & * & * \end{array}\right) \to \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ \hline 0 & 0 & 0 & * \\ \hline \end{array}\right)$$

(Un)vollständige Dreieckszerlegung

Die LU–Zerlegung für $A \in \mathbb{R}^{n,n}$ bei zeilenweisem Vorgehen.
Überschreibe A mit L, U (Diagonale von L weglassen)

for $i = 2, \ldots, n$:
  for $k = 1, \ldots, i-1$:
    $a_{ik} := a_{ik}/a_{kk}$
    for $j = k+1, \ldots, n$:
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$



Unvollständige Dreieckszerlegung

Eine einfache unvollst. Dreieckszerlegung: kein fill–in $ILU(0)$

for $i = 2, \ldots, n$:
  for $k = 1, \ldots, i-1$, sofern $a_{ik} \neq 0$
    $a_{ik} := a_{ik}/a_{kk}$
    for $j = k+1, \ldots, n$, sofern $a_{ij} \neq 0$
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Sehr einfach zu implementieren.

$$\begin{pmatrix} 2 & 0 & 0 & 1 \\ 2 & -1 & 0 & 1 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix} \to \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix} \to \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ 0 & 0 & -1 & 6 \end{pmatrix} \to \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

I.a. Stabilitätsprobleme

$$\begin{pmatrix} 2 & -1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix} \to \begin{pmatrix} 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix}$$

Unvollständige Dreieckszerlegung

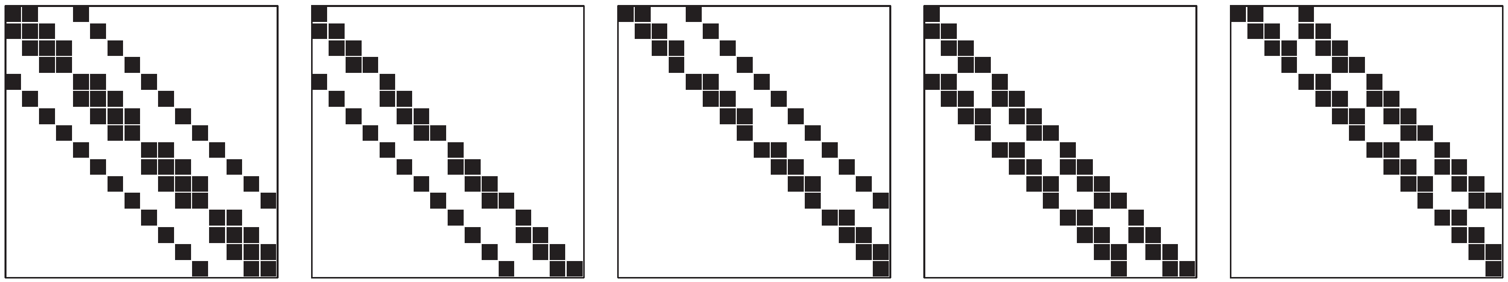
Verallgemeinerungen von $ILU(0)$: $ILU(p)$, $p = 0, 1, 2, \ldots$ fest.

fill–in hängt davon ab, welche Art Eintrag ihn erzeugt. Z.B. $ILU(1)$:

Sei $A \equiv A^{(0)} = \left(a_{ij}^{(0)}\right)_{i,j}$ Ausgangsmatrix. Überschreibe A wieder.

for $i = 2, \ldots, n$:
  for $k = 1, \ldots, i-1$, sofern $a_{ik} \neq 0$
    $a_{ik} := a_{ik}/a_{kk}$
    for $j = k+1, \ldots, n$, sofern $a_{ij} \neq 0$ oder $a_{ik}^{(0)} a_{kj}^{(0)} \neq 0$
      $a_{ij} = a_{ij} - a_{ik} a_{kj}$

Auch einfach zu implementieren. fill–in in seperater Liste abspeichern $\Rightarrow$ Regel von neuer fill–in entsteht



A $L(0)$ $U(0)$ $L(1)$ $U(1)$

Unvollständige Dreieckszerlegung

Andere ILU–Variante. Lasse betraglich kleine Einträge weg: $ILUT$

Parameter $\varepsilon \ll 1$. Neue Faktoren L, U

for $i = 1, \ldots, n$:

  $w = (a_{i1}, \ldots, a_{in})$, $\varepsilon_i = \varepsilon \|w\|$

  for $k = 1, \ldots, i-1$, sofern $w_k \neq 0$

    $w_k := a_{ik}/u_{kk}$. Falls $|w_k| \leqslant \varepsilon_i$: $w_k = 0$

    Falls $w_k \neq 0$

      for $j = k+1, \ldots, n$, sofern $u_{kj} \neq 0$

        $w_j = w_j - w_k u_{kj}$

  for $j = 1, \ldots, i-1$, sofern $w_j \neq 0$

    Falls $|w_j| > \varepsilon_i$: $l_{i,j} = w_j$

  for $j = i, \ldots, n$, sofern $w_j \neq 0$

    Falls $|w_j| > \varepsilon_i$: $u_{i,j} = w_j$

  $w = 0$

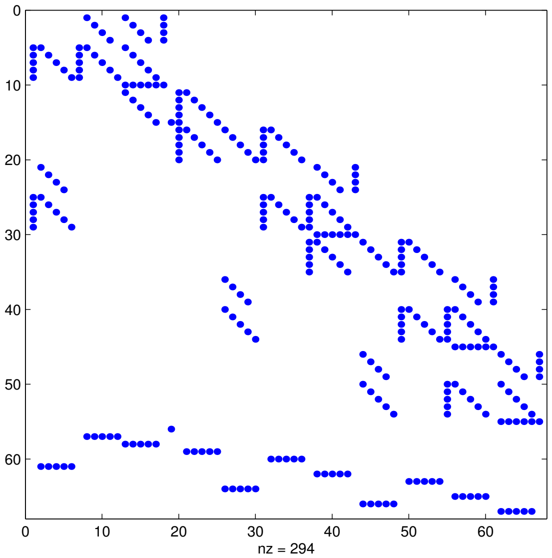
Wesentlicher Trick. Arbeitsvektor w ist nicht voll gespeichert, sondern nur die Nichtnulleinträge analog zu einer Zeile von A

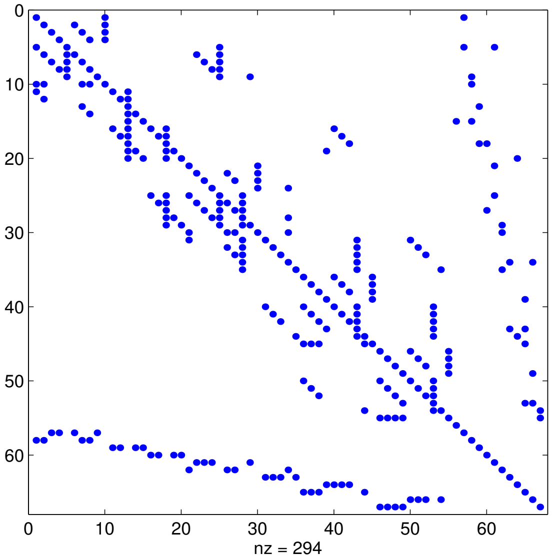
Unvollständige Dreieckszerlegung

- Mehrere Varianten unvollständiger Dreieckszerlegungen
- Prinzipiell kann man alles auch mit Spaltenvertauschungen machen, etwas komplizierter
- Je nach Problem braucht man Spaltenvertauschungen
- Algorithmen immer noch sehr anfällig für viele Probleme
- Ergänzung der Algorithmen durch Vorabpermutation und Skalierung der Matrix möglich (mehr Diagonaldominanz)

Matrix WEST0067

Harwell–Boeing–Coll.





Unvollständige Dreieckszerlegung

Ein Beispiel. Matrix WEST2021 aus der Harwell–Boeing–Collection. $n = 2021, nnz(A) = 7310$

Als iteratives Verfahren nehme $GMRES(30)$

Vorkonditionierer	flops ILU	$\frac{nnz(L+U)}{nnz(A)}$	flops GMRES
ILU(0)	$1.4 \cdot 10^4$	1.0	∞
ILU(0) + Vorabperm.	$1.7 \cdot 10^4$	1.0	∞
ILUTP, $\varepsilon = 10^{-1}$	$2.2 \cdot 10^4$	0.7	∞
ILUTP, $\varepsilon = 10^{-3}$	$4.6 \cdot 10^4$	1.2	∞
ILUTP, $\varepsilon = 10^{-5}$	$1.5 \cdot 10^5$	2.1	$4.3 \cdot 10^5$
ILUTP, $\varepsilon = 10^{-1}$ + Vorabperm.	$2.7 \cdot 10^4$	0.8	∞
ILUTP, $\varepsilon = 10^{-2}$ + Vorabperm.	$3.8 \cdot 10^4$	1.1	$1.1 \cdot 10^6$

Für unstrukturierte Probleme: ILU(0) oft ungeeignet, besser ILU mit Schwelle ε, ggf. Umordnen der Matrixeinträge

Schwachbesetzte Approximative Inverse

Vorkonditionierer für Parallelrechner

ILU in der Regel ungeeignet.

- Zerlegung $A \approx LU$ stark sequentiell
- Iteratives Verfahren mit ILU–Vorkonditionierung sequentiell, Vorwärts–/Rückwärtseinsetzen mit L, U

Ausweg. Approximiere explizit $M \approx A^{-1}$

Vorteil

- Iteratives Verfahren mit M–Vorkonditionierung hochgradig parallel, nur Matrix–Vektormultiplikation mit M
- Aber: Konstruktion von B sollte auch parallelisierbar sein

Hier nur Konzept von SPAI, es gibt auch andere Techniken

Idee: Minimiere $\|AM - I\|_F$, M mit vorgegebenem Besetzungsmuster.

Schwachbesetzte Approximative Inverse — SPAI

Sei $A = (a_{ij})_{i,j} = [a_1, \ldots, a_n]$, $M = (m_{ij})_{i,j} = [m_1, \ldots, m_n]$.

In jedem Fall ist

$$\min \|AM - I\|_F^2 = \min \sum_{j=1}^{n} \|Am_j - e_j\|_F^2 = \sum_{j=1}^{n} \left(\min \|Am_j - e_j\|_F^2\right)$$

Minimierung geht für jede Spalte von M separat!

$\mathcal{I}_j$ beschreibe die Nichtnulleinträge von M in Spalte j.

$\Longrightarrow m_j = \sum_{i \in \mathcal{I}_j} m_{ij} e_j$.

$$\Longrightarrow \min \|Am_j - e_j\|_F^2 = \min \|\sum_{i \in \mathcal{I}_j} a_i m_{ij} - e_j\|_F^2$$

Lösung des Problem erfordert Spalten $(a_i)_{i \in \mathcal{I}_j}$ von A und berechnet m_j.

Hochgradig parallelisierbar!

Schwachbesetzte Approximative Inverse — SPAI

Wie bestimmt ein "gutes" Besetzungsmuster für M?

Fange mit einem Besetzungsmuster an, z.B. von I, A, $A^\top$ oder $|A| + |A|^\top$

In diesem Falle ist $\mathcal{I}_j \; \forall j$ definiert und wir haben ein Problem Form

$$\|A_j x - e_j\|_2^2 = \min$$

wobei A_j diejenigen Spalten a_j von A aus $\mathcal{I}_j$ besitzt und x den Einträgen m_{ij}, $i \in \mathcal{I}_j$ entspricht.

$\rightarrow$ Kleinste–Fehler–Quadrate–Problem. Kann mit QR–Zerlegung oder durch Lösen der Normalengl. gelöst werden.

Adaptiere Muster!

Schwachbesetzte Approximative Inverse — SPAI

Adaptiere Muster:

Sei $r = e_j - A_j x$. Nach Berechnung von x können wir $\|r\|_2^2$ bilden und auch $\|a_k m_{kj} - r\|_2^2$, $k \notin \mathcal{I}_j$.

Minimierung liefert

$$m_{kj} = \frac{r^\top a_k}{\|a_k\|_2^2}, \quad \|A_j x + a_k m_{kj} - e_j\|_2^2 = \|r\|_2^2 - \left(\frac{|r^\top a_k|}{\|a_k\|_2}\right)^2$$

Folgerung: Müssen k mit größtem $\frac{|r^\top a_k|}{\|a_k\|_2}$ bestimmen.

Vorgang wird mehrfach wiederholt (KEINE Lösung des Problems mehr)

Ist A schwachbesetzt, so kommen nur wenige k mit $r^\top a_k \neq$ in Frage
$\Rightarrow$ beschränkte Suche.
$\longrightarrow$ Neue Indexmenge $\hat{\mathcal{I}}_j$. Neues Minimierungsproblem.

Aufhören, falls $\|r\| \leqslant \varepsilon$ ist, ε vorgegeben.

Schwachbesetzte Approximative Inverse — SPAI

Matrix ORSIRR2, Harwell–Beoing–Coll. $n = 886$, $nnz(A) = 5970$.

Fehler des Residuums

ε	0.6	0.5	0.4	0.3	0.2
$\|AM - I\|_F$	14.3	11.3	8.98	7.12	4.82

Anzahl Einträge M

ε	0.6	0.5	0.4	0.3	0.2
$nnz(M)/nnz(A)$	0.32	0.61	0.89	1.53	3.39

Anzahl Iterationen GMRES(20)

ε	0.6	0.5	0.4	0.3	0.2
$\|AM - I\|_F$	458	146	84	52	29

Problem. Als (sequentielles) Programm oft langsam, Berechnung von M sehr teuer.

Zwei Modellprobleme

Eindimensionales elliptisches Randwertproblem

$$\begin{aligned} -u''(x) &= f(x), \; x \in [0,1] \\ u(0) &= g_0, \; u(1) = g_1 \end{aligned}$$

Zweidimensionales elliptisches Randwertproblem

$$\begin{aligned} \Omega &= [0,1] \times [0,1] \\ \overbrace{-u_{xx}(x,y) - u_{yy}(x,y)}^{-\Delta u(x,y)} &= f(x,y), \; (x,y) \in \Omega \\ u(x,y) &= g(x,y), \; (x,y) \in \partial\Omega \end{aligned}$$

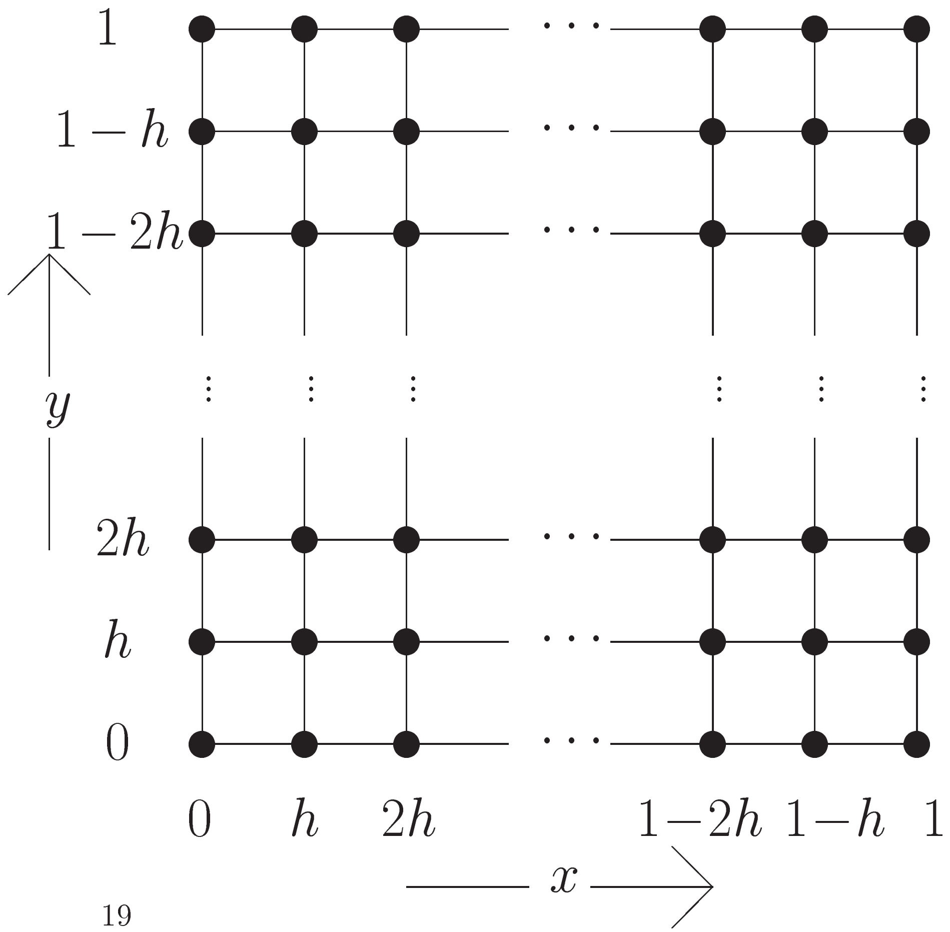
Diskretisierung mittels Finiter Differenzen

1D–Problem: $h = \frac{1}{N+1}$. $\quad [0,1] \to \Omega_h = \{0, h, 2h, \ldots, 1-h, 1\}$

2D–Problem:

$\Omega = [0,1]^2 \to \Omega_h =$

$\{(k,l)h : \; k,l = 0, \ldots N+1\}$



Diskretisierung mittels Finiter Differenzen

1D Randwertproblem

Differentialgleichung

$$-y''(x) = f(x),\ \ y(0) = g_0, y(1) = g_1$$

$$\downarrow$$

Differenzengleichung

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i, u_0 = g_0, u_{N+1} = g_1,\ \text{ wobei } u_i \equiv u(ih)\ \forall i$$

Differenzenstern

$$-1 \;—\; 2 \;—\; -1$$

Lineares Gleichungssystem

$$T_h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \longrightarrow\ T_h u = f.$$

Diskretisierung mittels Finiter Differenzen

2D Randwertproblem

Differentialgleichung

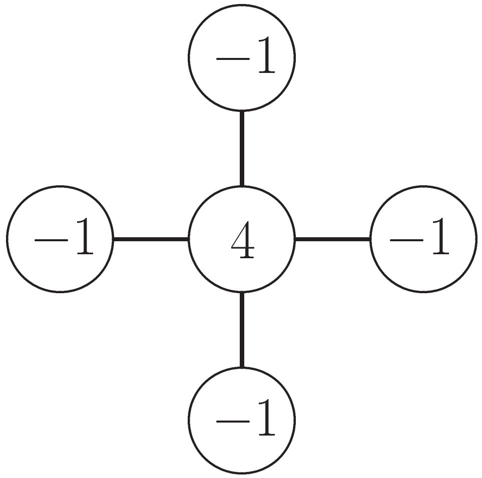
$$-\Delta y(x,y) = f(x,y), \text{ in } \Omega = [0,1]^2, \; y = g \text{ auf } \partial\Omega$$

$$\downarrow$$

Differenzengleichung

$$\frac{-u_{i-1,j} - u_{i,j-1} + 4u_{ij} - u_{i+1,j} - u_{i,j+1}}{h^2} = f_{ij}$$

Differenzenstern



Lineares Gleichungssystem

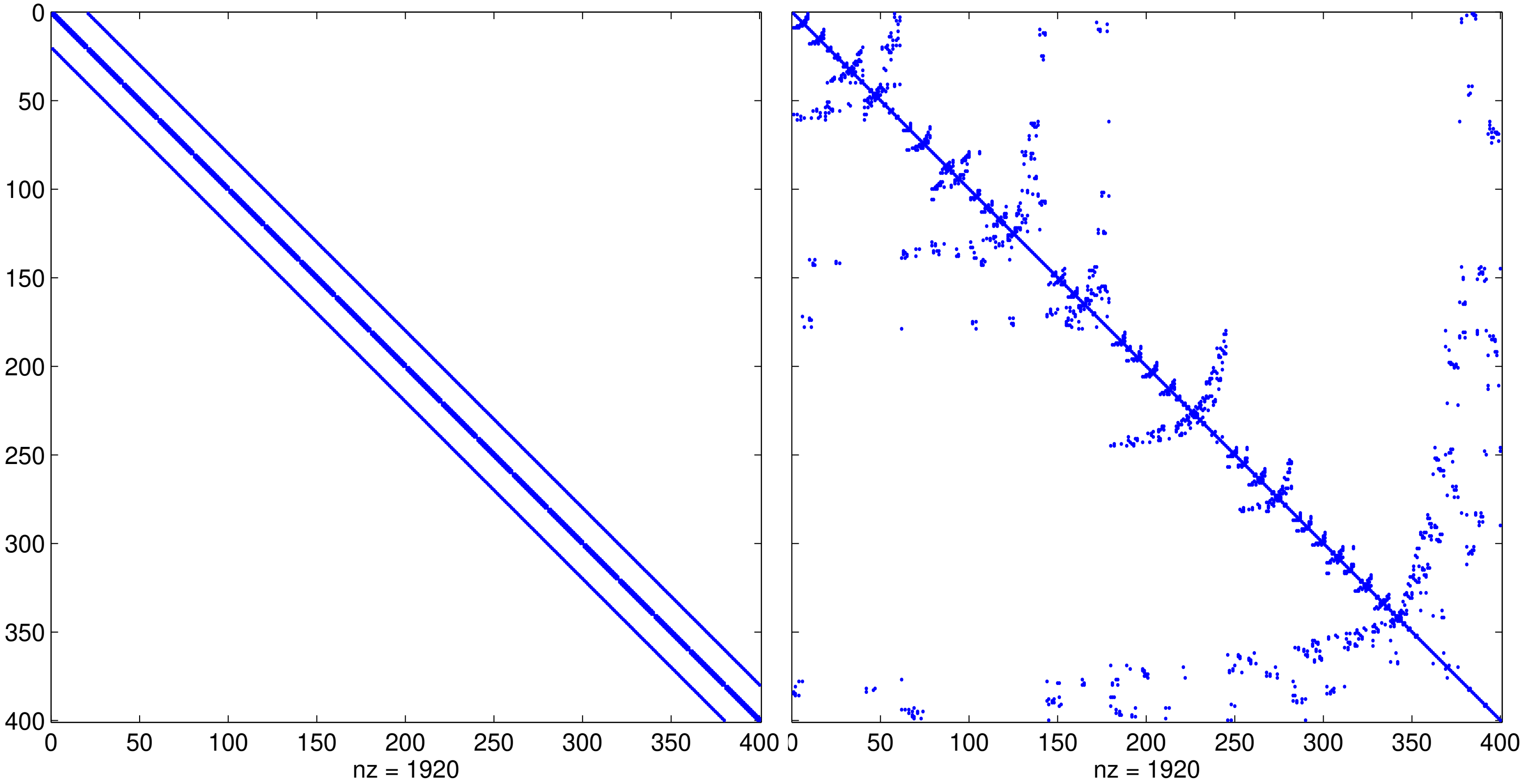
$$A_h u = f$$

Elementare Lösungstechniken — Direkter Löser

1D/2D RWP: Matrix schwach besetzt, symmetrisch positiv definit

Angepasster direkter Löser $A = LL^\top$, Cholesky–Zerlegung

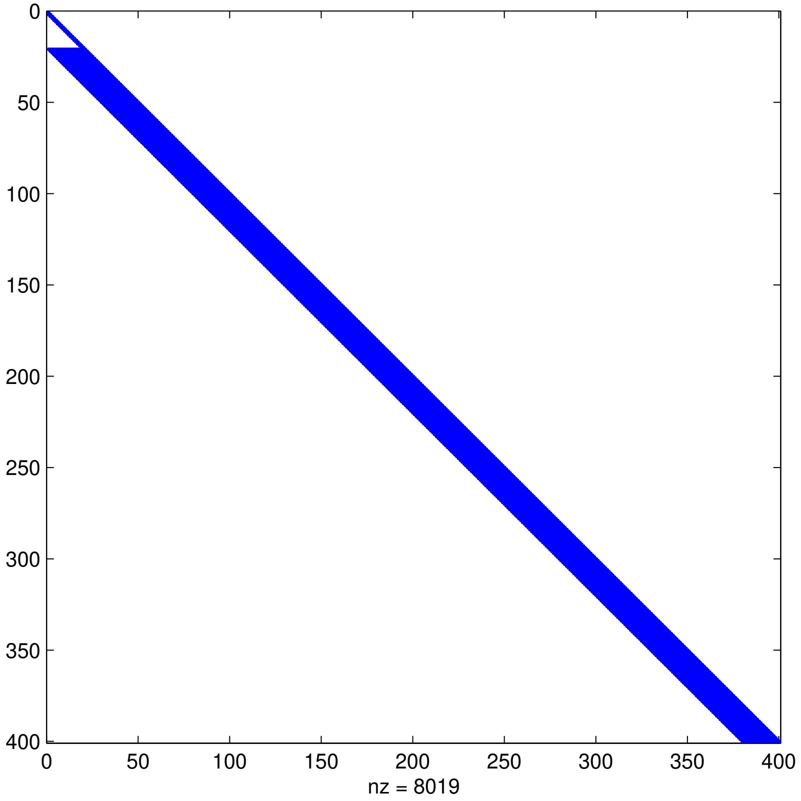
Dabei Matrixeinträge umordnen ("symmetric minimum degree")

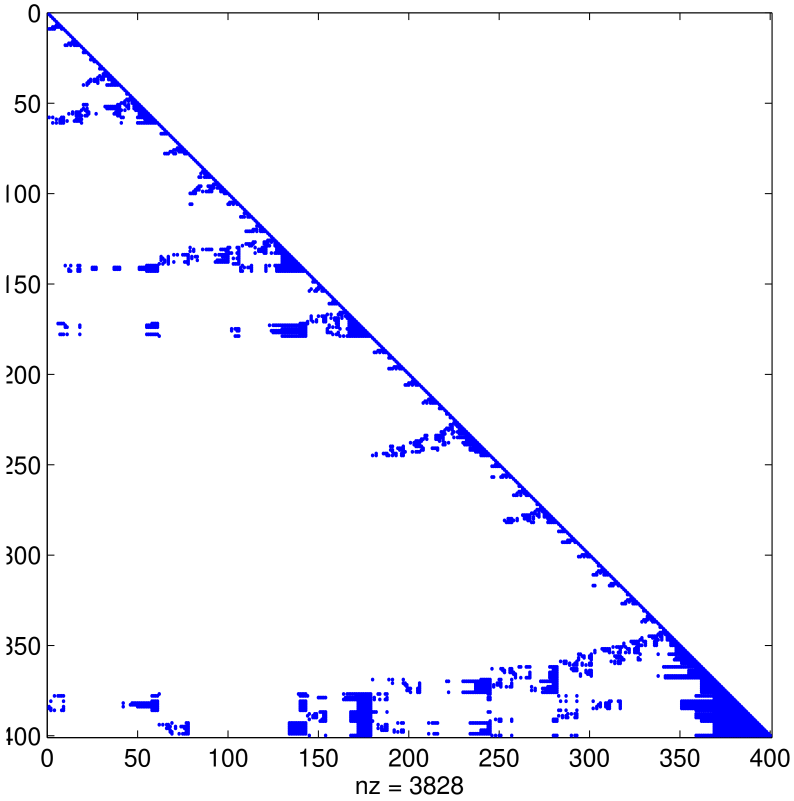


Direkter Löser

2D Randwertproblem

Dreiecksfaktoren L aus Choleskyzerlegung





Direkter Löser

2D Randwertproblem

h	Problemgröße	Rechenzeit (flops)	Speicherbedarf (Nichtnullel.)
$\frac{1}{31}$	961	$2.68 \cdot 10^5$	$1.11 \cdot 10^4$
$\frac{1}{63}$	3969	$2.95 \cdot 10^6$	$6.55 \cdot 10^4$
$\frac{1}{127}$	16129	$2.89 \cdot 10^7$	$3.56 \cdot 10^5$
$\frac{1}{255}$	65025	$2.92 \cdot 10^8$	$1.89 \cdot 10^6$

- Rechenzeit wächst mehr als linear
  (Faktor 10 bei Vervierfachung der Problemgröße)
- Speicherbedarf wächst mehr als linear
  (Faktor 6 bei Vervierfachung der Problemgröße)

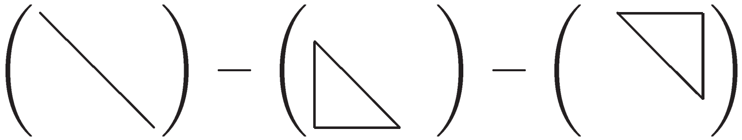
Elementare Lösungstechniken — Iterativer Löser

Grunditerationsverfahren.

$$x^{(k+1)} = x^{(k)} + B^{-1}(b - Ax^{(k)}), \forall k, \quad x^{(0)} = 0.$$

Zerlegung von A

$$A = D - E - E^{\top} =$$



Mögliche Approximationen B an A.

B	D	$D - E$	$(D - E)D^{-1}(D - E^{\top})$
Bezeichnung	Jacobi	Gauß–Seidel	symmetrisches Gauß–Seidel

Iteration stoppen, falls $\|b - Ax^{(k)}\|_{\infty} \leqslant 10^{-8}\|b\|_{\infty}$.

Einfache Iterativer Löser

2D Randwertproblem

		Jacobi		Gauß–Seidel	
h	Problem–größe	Rechenzeit (flops)	Anzahl Iterationen	Rechenzeit (flops)	Anzahl Iterationen
$\frac{1}{31}$	961	$5.43 \cdot 10^7$	3834	$3.43 \cdot 10^7$	1918
$\frac{1}{63}$	3969	$9.06 \cdot 10^8$	15354	$5.73 \cdot 10^8$	7678
$\frac{1}{127}$	16129	$1.48 \cdot 10^{10}$	61432	$9.37 \cdot 10^9$	30717

- Ergebnis niederschmetternd, schlimmer als direkter Löser
- Rechenzeit wächst mehr als linear
  (Faktor 16 bei Vervierfachung der Problemgröße)
- Anzahl Iterationen wächst!
  (Faktor 4 bei Vervierfachung der Problemgröße)

Iterativer Löser — Polynomielle Verfahren

A symmetrisch positiv definit $\Rightarrow$ verwende vorkond. cg–Verfahren.

2D Randwertproblem

		Jacobi Vork.		sym. Gauß–Seidel Vork.	
h	Problem–größe	Rechenzeit (flops)	Anzahl Iterationen	Rechenzeit (flops)	Anzahl Iterationen
$1/31$	961	$1.29{\cdot}10^6$	59	$1.16{\cdot}10^6$	34
$1/63$	3969	$1.09{\cdot}10^7$	120	$9.08{\cdot}10^6$	64
$1/127$	16129	$9.06{\cdot}10^7$	245	$7.00{\cdot}10^7$	121
$1/255$	65025	$7.45{\cdot}10^8$	499	$5.09{\cdot}10^8$	218

- Ergebnis deutlich besser, aber immer noch unbefriedigend
- Rechenzeit wächst weiterhin mehr als linear, Iterationszahl steigt!
- Approximation an die Ausgangsmatrix anscheinend zu grob

Iterativer Löser — Unvollst. Dreieickszerl.

1 Schritt Cholesky:

$$A = \begin{pmatrix} \alpha & v^\top \\ v & B \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ v & I \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} \alpha & v^\top \\ 0 & I \end{pmatrix},$$

wobei $S = B - \frac{vv^\top}{\alpha}$ Schur–Komplement. Wiederhole Vorgang für S.

$$S = \tilde{L}\tilde{D}^{-1}\tilde{L}^\top \Rightarrow A = \begin{pmatrix} \alpha & 0 \\ v & \tilde{L} \end{pmatrix} \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{D} \end{pmatrix}^{-1} \begin{pmatrix} \alpha & v^\top \\ 0 & \tilde{L}^\top \end{pmatrix},$$

Unvollständige Zerlegung. Ersetze S:

$$S \longrightarrow \tilde{S}$$

wobei $\tilde{S}$ identisch mit S, dort wo B von Null verschieden ist.

Liefert approximative Zerlegung von A.

$$A \approx (\hat{D} - \hat{E})\hat{D}^{-1}(\hat{D} - \hat{E})^\top.$$

"ICHOL(0)" (bzw. ILU(0) im unsymmetrischen Fall).

Iterativer Löser — Unvollst. Dreieickszerl.

ICHOL(0) als Vorkond. für das cg–Verfahren

2D Randwertproblem

		ICHOL(0) Vork.	
h	Problem–größe	Rechenzeit (flops)	Anzahl Iterationen
$1/31$	961	$9.40 \cdot 10^5$	29
$1/63$	3969	$7.40 \cdot 10^6$	55
$1/127$	16129	$5.69 \cdot 10^7$	104
$1/255$	65025	$4.51 \cdot 10^8$	204

- Ergebnis etwas besser als sym. Gauß–Seidel, trotzdem unbefriedigend
- Verfahren skaliert immer noch nicht

Elementare Lösungstechniken

Zusammenfassung

- Direkte Löser
  - $\oplus$ Keine Konvergenzprobleme
  - $\ominus$ hohen Speicherbedarf
  - $\ominus$ Rechenzeit
- Einfache Iterationsverfahren (Jacobi, Gauß–Seidel)
  - $\oplus$ geringer Speicherbedarf
  - $\ominus$ große Konvergenzprobleme
  - $\ominus$ Enorme Rechenzeit
- cg mit Vorkond. (ICHOL(0), symm. Gauß–Seidel)
  - $\oplus$ geringer Speicherbedarf
  - $\ominus$ Konvergenzprobleme für große n
  - $\ominus$ Rechenzeit für große n

Glättungsanalyse

Warum sind elementare Verfahren so schlecht?

Der Einfachheit halber: 1D–Problem, Jacobi $B = \frac{2}{h^2} I$, Grunditeration.

Fehler $e^{(k+1)} = x - x^{(k+1)}$ der Gleichung $T_h x = b$.

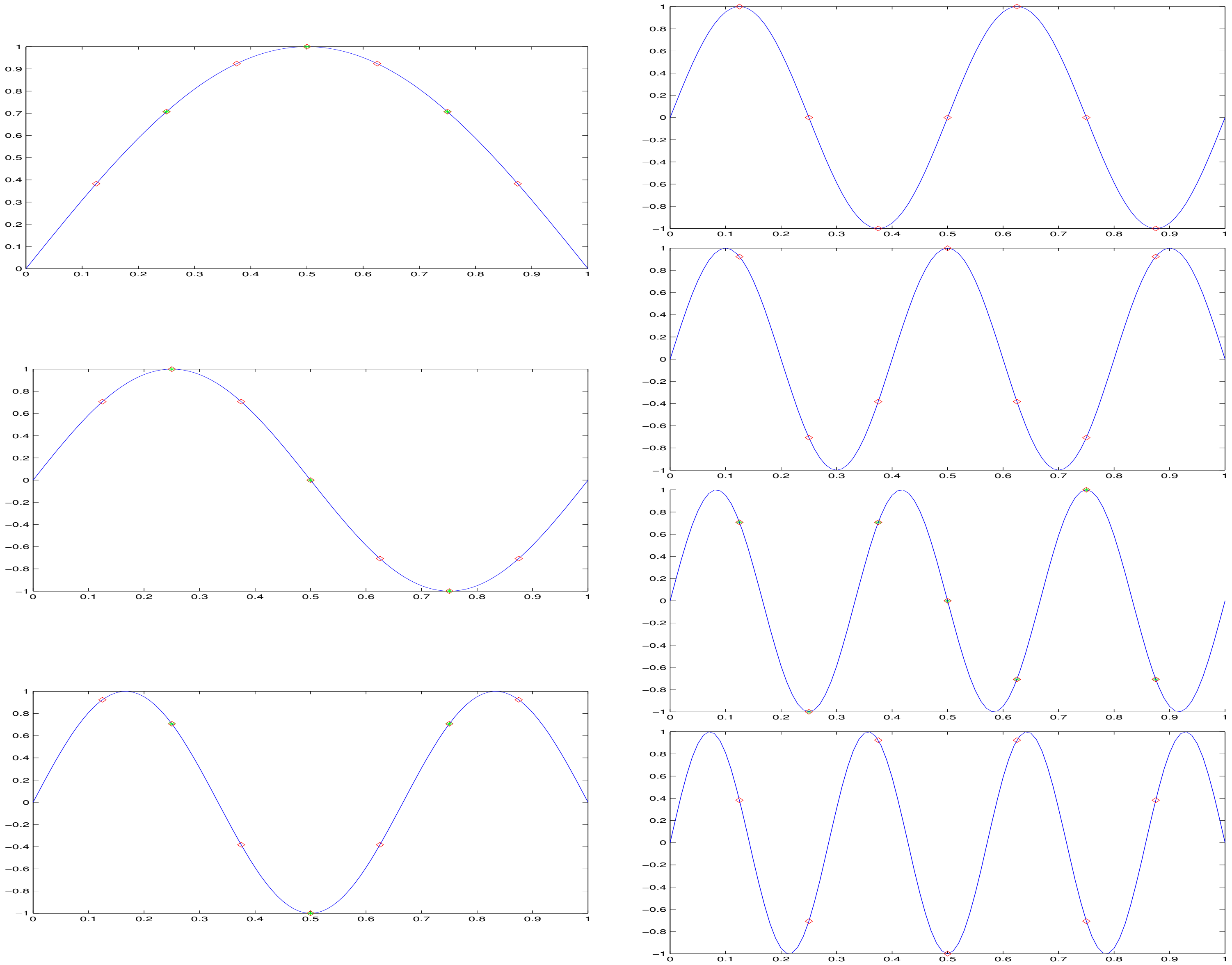
$$e^{(k+1)} = x - x^{(k+1)} = x - \left[x^{(k)} + \frac{h^2}{2}(T_h x - T_h x^{(k)}) \right] = (I - \frac{h^2}{2} T_h) e^{(k)}$$

Zur Analyse verwende für $e^{(k)}$ die Eigenvektoren von T_h.

Eigenvektoren $s^{(k)}$ von T_h: Restriktion von $s^{(k)}(x) = \sin(k\pi x)$ auf Ω_h, $\forall k$.

Zughehörige Eigenwerte: $\lambda_k = \frac{4}{h^2} \sin^2 \frac{k\pi h}{2}, \ \forall k$

Glättungsanalyse — Eigenvektoren

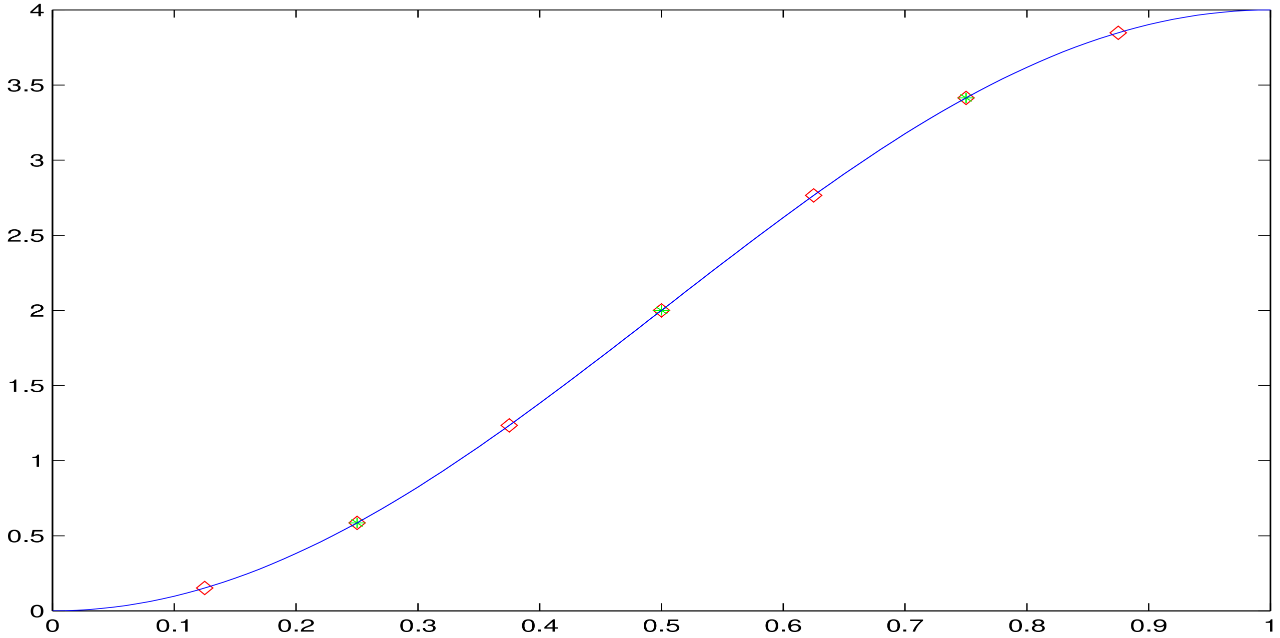


Glättungsanalyse — Eigenvektoren

Beobachtung. Bei höherer Diskretisierung $N \to 2N$:

- niedrige Frequenzen bleiben erhalten
- höhere Frequenzen kommen hinzu

Eigenwerte (bis auf den Faktor $\frac{1}{h^2}$)

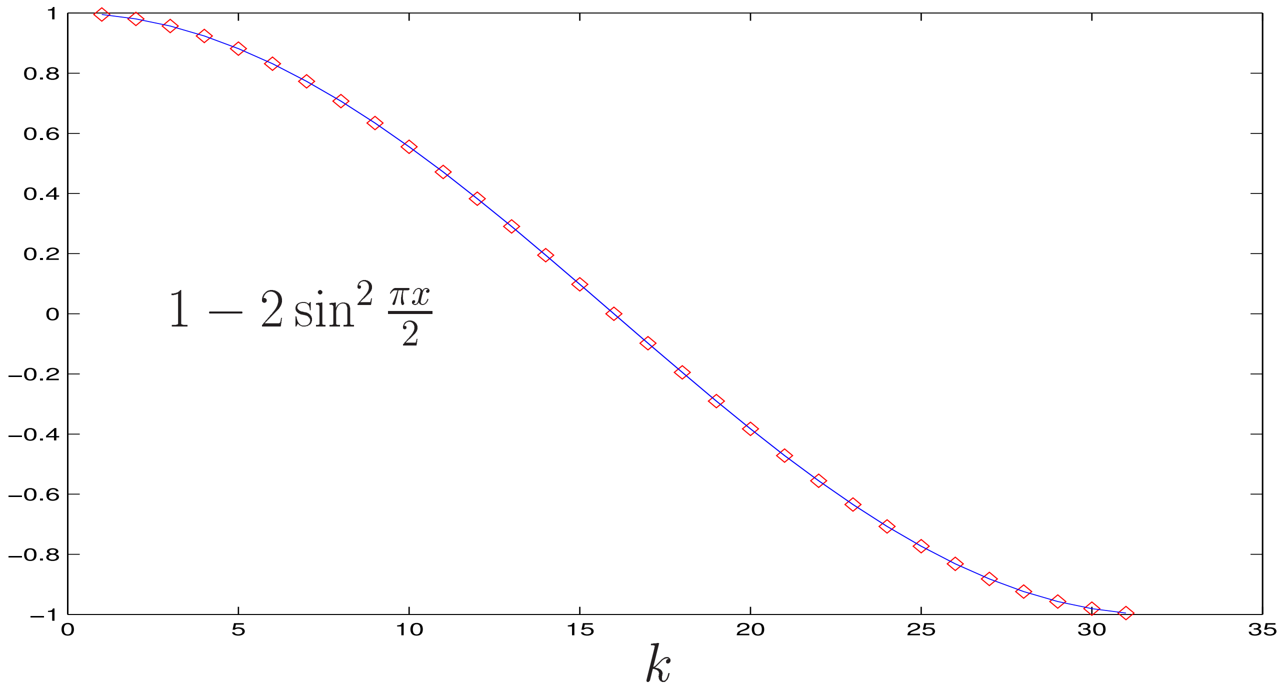


Glättungsanalyse — Jacobi–Verfahren

Schreibe Fehler e als $e = \sum_{k=1}^{N} \alpha_k s^{(k)}$

$$(I - \frac{h^2}{2} T_h)e = \sum_{k=1}^{N} \alpha_k (s^{(k)} - \frac{h^2}{2} T_h s^{(k)}) = \sum_{k=1}^{N} \alpha_k \left(1 - 2\sin^2 \frac{k\pi h}{2}\right) s^{(k)}.$$

Dämpfungsfaktoren $\left|1 - 2\sin^2 \frac{k\pi h}{2}\right|$ für die Eigenvektoren



Glättungsanalyse — Jacobi–Verfahren

Beobachtung.

- Frequenzen im mittleren Bereich (um $\frac{N}{2}$) werden stark gedämpft,
- keine Dämpfung für niedrige Frequenzen oder hohe Frequenzen

Steuere Dämpfungsverhalten durch Dämpfungsparameter ω, $B \to \frac{1}{\omega} B$.

Im Falle des Jacobi–Verfahrens

$$(I - \frac{\omega h^2}{2} T_h) e = \sum_{k=1}^{N} \alpha_k (s^{(k)} - \frac{\omega h^2}{2} T_h s^{(k)}) = \sum_{k=1}^{N} \alpha_k \left(1 - 2\omega \sin^2 \frac{k\pi h}{2}\right) s^{(k)}.$$

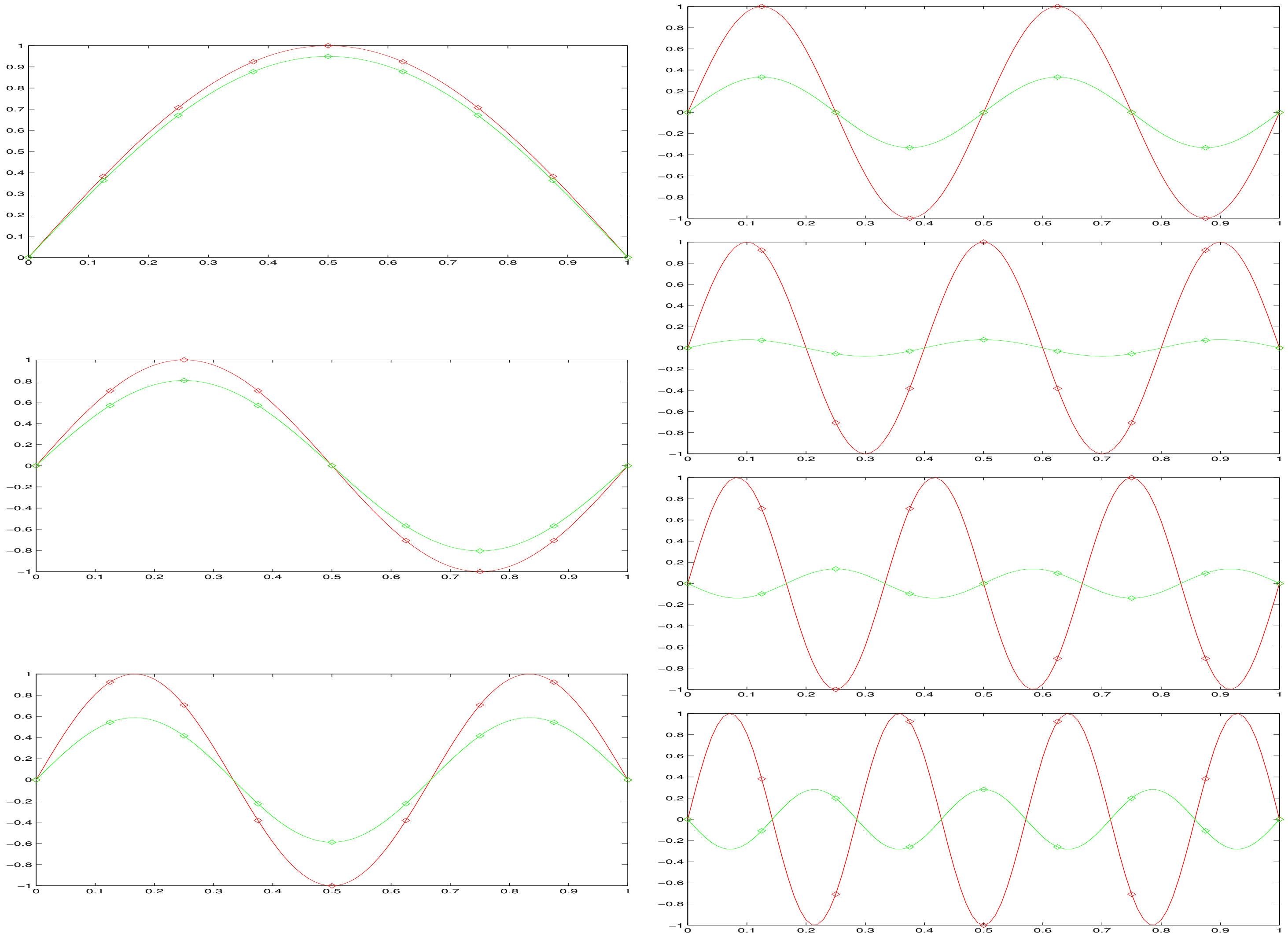
Optimiere ω für hohe Frequenzen ($k \geqslant \frac{N}{2}$)

$$\max_{k > \frac{N}{2}} |1 - 2\omega \sin^2 \frac{k\pi h}{2}| = \min$$

$$\Longrightarrow \omega = \frac{2}{3}, \quad \max_{k > \frac{N}{2}} |1 - \frac{4}{3} \sin^2 \frac{k\pi h}{2}| \leqslant \frac{1}{3}$$

Glättungsanalyse — Jacobi–Verfahren

Geglättete Eigenvektoren vorher und nachher



Glättungsanalyse — Zusammenfassung

Beobachtungen.

- ungedämpftes Jacobi–Verfahren glättet nur Frequenzen im mittleren Bereich
- gedämpftes ($\omega = \frac{2}{3}$) Jacobi–Verfahren dämpft hohe Frequenzen
- Praktisch keine Dämpfung bei niedrigen Frequenzen
- **ABER:** Niedrigen Frequenzen treten bei Problem T_{2h} der Dimension $2h$ wieder auf
- Konsequenz. Brauchen zusätzlichen Korrekturschritt, der die glatten Anteile reduziert.
  Idee: Verwende hierzu gröberes Gitter Ω_{2h} und T_{2h}.

Grobgitterkorrektur

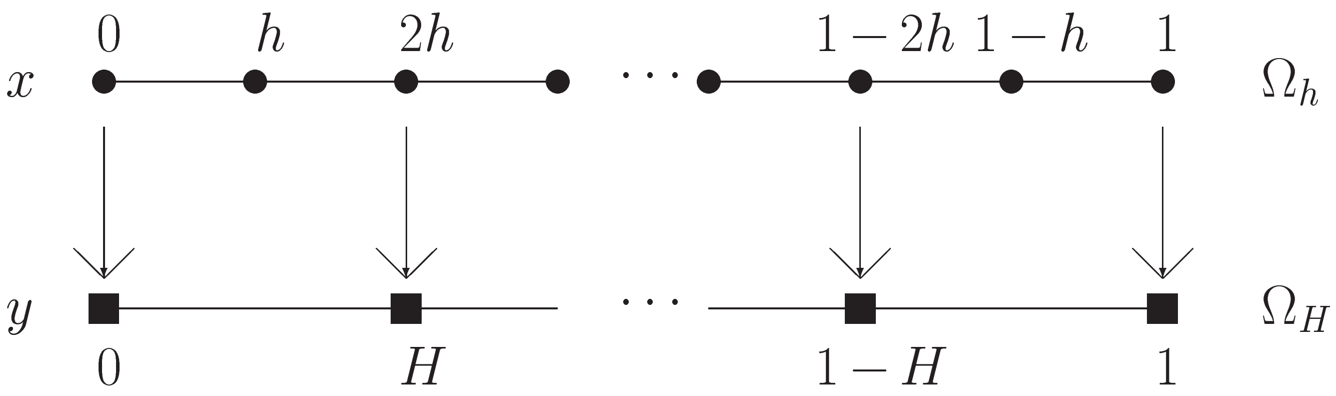
Glättungsschritt benötigt zusätzlichen Grobgitterkorrekturschritt.

$H = 2h$, Übergang von $\begin{array}{lll} \Omega_h \longrightarrow \Omega_H & \text{Restriktion} \\ \Omega_H \longrightarrow \Omega_h & \text{Prolongation} \end{array}$

$x \in \Omega_h$, Werte auf dem Gitter $x(h), \ldots, x((N-1)h)$

$y \in \Omega_H$, Werte auf dem Gitter $y(H), \ldots, y(\frac{N-1}{2}H)$

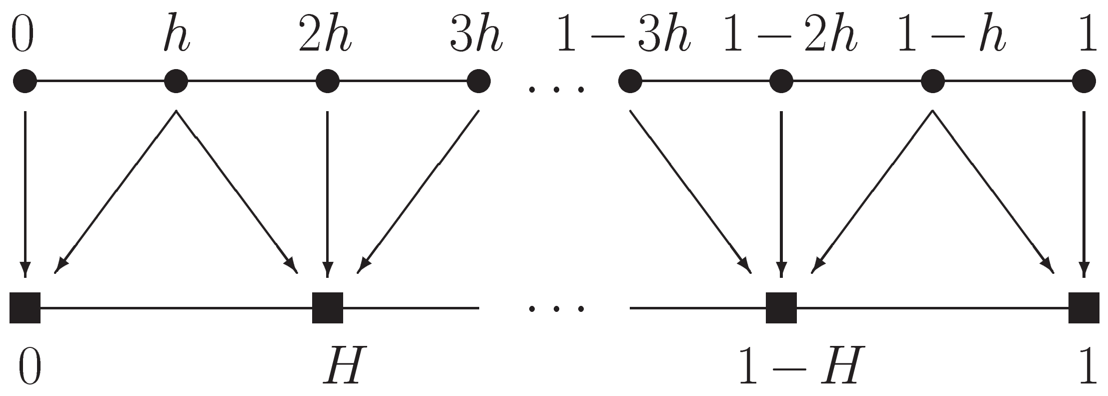
Triviale Restriktion schlecht: $y(iH) = x(iH) = x(2ih)$, $\forall i$

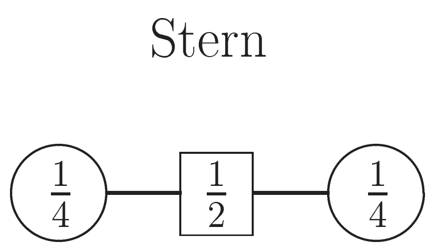


Grobgitterkorrektur — Gitterübergang

Restriktion R. Mittellung über die Nachbarn

$$y(iH) = \frac{1}{4}x(iH-h) + \frac{1}{2}x(iH) + \frac{1}{4}x(iH+h),\ \forall i$$





Prolongation P. Lineare Interpolation.

$$x(ih) = \begin{cases} y(jH) & \text{falls } i = 2j \\ \frac{1}{2}y(jH) + \frac{1}{2}y(jH+H) & \text{falls } i = 2j+1 \end{cases},\ \forall i$$

Prinzip der Restriktion wird umgekehrt. $P = 2R^\top$

Zweigitterverfahren

Kombination Jacobi (gedämpft) / Grobgitterkorrektur $T_h^{-1} \approx P T_H^{-1} R$.

$$
\begin{aligned}
x_h^{(k+\frac{1}{2})} &= x_h^{(k)} + \tfrac{2}{3} \cdot \tfrac{h^2}{2}(b_h - T_h x_h^{(k)}) \\
x_h^{(k+1)} &= x_h^{(k+\frac{1}{2})} + P_h T_H^{-1} R_h (b_h - T_h x_h^{(k+\frac{1}{2})})
\end{aligned}
$$

Bisher: für T_H Gleichungssystem lösen (z.B. mit Cholesky)

Ausweg: wende die Zweigitter–Verfahren wiederum (rekursiv) auf T_H an

Für $b_H = R_h(b_h - T_h x_h^{(k+\frac{1}{2})})$, ersetze $T_H x_H = b_H$ durch

$$
\left.
\begin{aligned}
x_H^{(\frac{1}{2})} &= x_H^{(0)} + \tfrac{2}{3} \cdot \tfrac{H^2}{2}(b_H - T_H x_H^{(0)}) \\
x_H^{(1)} &= x_H^{(\frac{1}{2})} + P_H T_{2H}^{-1} R_H (b_H - T_H x_H^{(\frac{1}{2})})
\end{aligned}
\right\} \Longrightarrow x_h^{(k+1)} \approx x_h^{(k+\frac{1}{2})} + P_h x_H^{(1)}
$$

Problemreduktion $h \longrightarrow 2h = H \longrightarrow 2H = 4h \longrightarrow \cdots$

Mehrgitterverfahren

Hierarchische Gitter $\Omega_{h_1} \subseteq \Omega_{h_2} \subseteq \cdots \subseteq \Omega_{h_l}$, z.B. $h_{s+1} = h_s/2,\ \forall s$

Zu jedem Gitter Ω_{h_s} diskretisierte Gleichung $A_{h_s} x_{h_s} = b_{h_s}$

Zwischen den Gittern Übergange
$$\begin{array}{ll} \Omega_{h_s} \longrightarrow \Omega_{h_{s-1}} & \text{Restriktion } R_s \\ \Omega_{h_{s-1}} \longrightarrow \Omega_{h_s} & \text{Prolongation } P_s \end{array}$$

Auf jedem Gitter Glätter S_{h_s}, z.B. mit Hilfe Zerlegung von A_{h_s}

$$A_{h_s} = D_{h_s} - E_{h_s} - E_{h_s}^\top = \begin{pmatrix} \diagdown \end{pmatrix} - \begin{pmatrix} ◺ \end{pmatrix} - \begin{pmatrix} ◹ \end{pmatrix}$$

$$\begin{array}{rll} \text{Wähle z.B.} & S_{h_s} = \frac{1}{\frac{2}{3}} \cdot D_{h_s} & \text{(gedämpftes Jacobi–Verf.)} \\ \text{oder} & S_{h_s} = D_{h_s} - E_{h_s} & \text{(Gauß–Seidel)} \\ \text{oder} & S_{h_s} = (D_{h_s} - E_{h_s}) D_{h_s}^{-1} (D_{h_s} - E_{h_s}^\top) & \text{(symm. Gauß–Seidel)} \end{array}$$

Mehrgitterverfahren — Algorithmus

Gegeben: h_s für ein $s \in \{1, 2, \ldots, l\}$, rechte Seite b_{h_s}, Näherung x_{h_s}

Mehrgitterschritt berechnet rekursiv neue Näherung y_{h_s}

Falls $s = 1$, löse $A_{h_1} y_{h_1} = b_{h_1}$ direkt.

Sonst $y_{h_s} := x_{h_s}$

Vorglättung
$$\left\{ \begin{array}{l} \textbf{Für } k = 1, 2, \ldots, \nu_1 \\ \quad y_{h_s} := y_{h_s} + S_{h_s}^{-1}(b_{h_s} - A_{h_s} y_{h_s}) \end{array} \right.$$

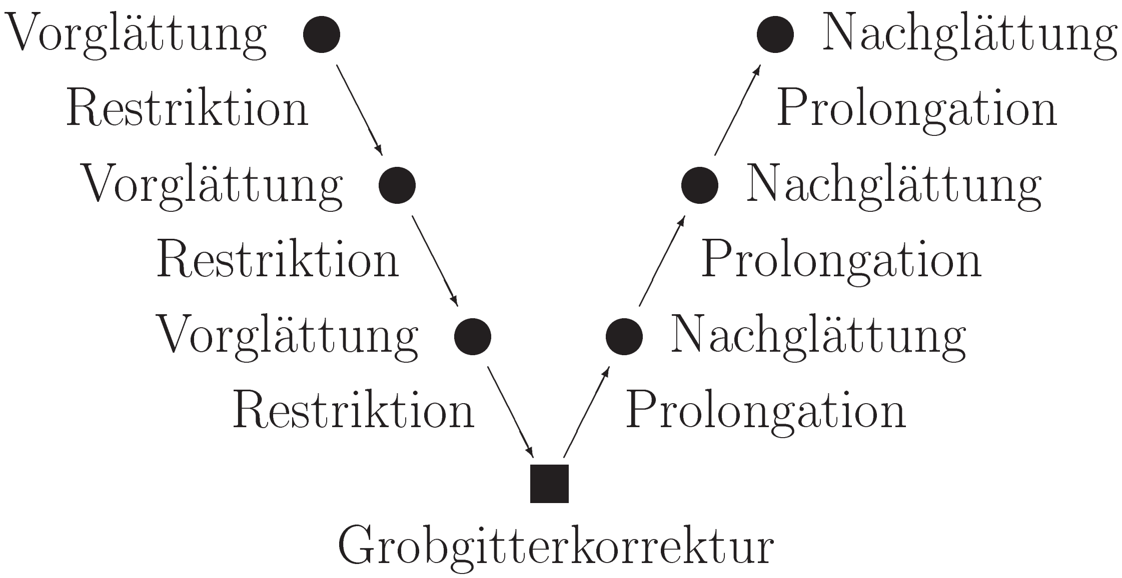
Grobgitterkorrektur
$$\left\{ \begin{array}{l} \textbf{Für } k = 1, 2, \ldots, \mu \\ \quad x_{h_{s-1}} := 0, b_{h_{s-1}} := R_s(b_{h_s} - A_{h_s} y_{h_s}) \\ \quad \text{Rufe } \underline{\textit{Mehrgitterschritt}} \text{ auf mit } h_{s-1}, b_{h_{s-1}}, x_{h_{s-1}} \\ \quad \to \ \text{Ergebnis } y_{h_s}. \\ \quad y_{h_s} := y_{h_s} + P_s y_{h_{s-1}} \end{array} \right.$$

Nachglättung
$$\left\{ \begin{array}{l} \textbf{Für } k = 1, 2, \ldots, \nu_2 \\ \quad y_{h_s} := y_{h_s} + S_{h_s}^{-1}(b_{h_s} - A_{h_s} y_{h_s}) \end{array} \right.$$

Mehrgitterverfahren — Anmerkungen

- Bei mehrmaliger Anwendung nur Summe $\nu = \nu_1 + \nu_2$ benötigt.
- *Mehrgitterschritt* für $\mu = 1$ rekursiven Aufruf für Ω_{h_s} $\rightarrow$ “V–Zyklus”
- *Mehrgitterschritt* für $\mu = 2$ rekursive Aufrufe für Ω_{h_s} $\rightarrow$ “W–Zyklus”

Schema eines Mehrgitterschrittes “V–Zyklus”



Mehrgitterverfahren — 1D Fall

1D Randwertproblem, Mehrgitterverfahren

	V–Zykl.,$\nu = 1$		V–Zykl.,$\nu = 2$		W–Zykl.,$\nu = 1$		W–Zykl.,$\nu = 2$	
h	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.
$\frac{1}{511}$	$5.8{\cdot}10^5$	21	$5.7{\cdot}10^5$	15	$2.4{\cdot}10^6$	17	$1.6{\cdot}10^6$	9
$\frac{1}{1023}$	$1.2{\cdot}10^6$	21	$1.2{\cdot}10^6$	16	$5.5{\cdot}10^6$	17	$3.7{\cdot}10^6$	9
$\frac{1}{2047}$	$2.5{\cdot}10^6$	22	$2.4{\cdot}10^6$	16	$1.2{\cdot}10^7$	16	$9.2{\cdot}10^6$	10
$\frac{1}{4095}$	$4.9{\cdot}10^6$	22	$4.9{\cdot}10^6$	16	$2.6{\cdot}10^7$	16	$1.8{\cdot}10^7$	9

- Verfahren skaliert linear!
- Mehr Glättung $\rightarrow$ weniger Iterationen, aber teurer
- W–Zyklus hier aufwendiger, dafür weniger Iterationen

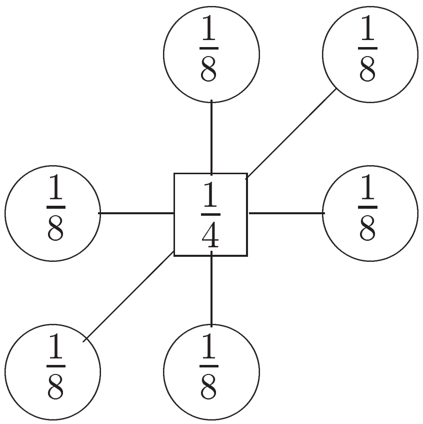
Mehrgitterverfahren im 2D–Fall

$x \in \Omega_h$, $y \in \Omega_H$. Werte auf dem Gitter $x(ih, jh)$, $y(kH, lH)$ $\forall i, j, k, l$.

Übertragung Restriktion: $\forall i, j$

$$\begin{aligned} y(iH, jH) \ &= \frac{1}{4} x(iH, jH) + \frac{1}{8} [x(iH-h, jH) + x(iH+h, jH) \\ &\quad + x(iH, jH-h) + x(iH, jH+h) \\ &\quad + x(iH-h, jH-h) + x(iH+h, jH+h)] \end{aligned}$$

In Form eines Sternes



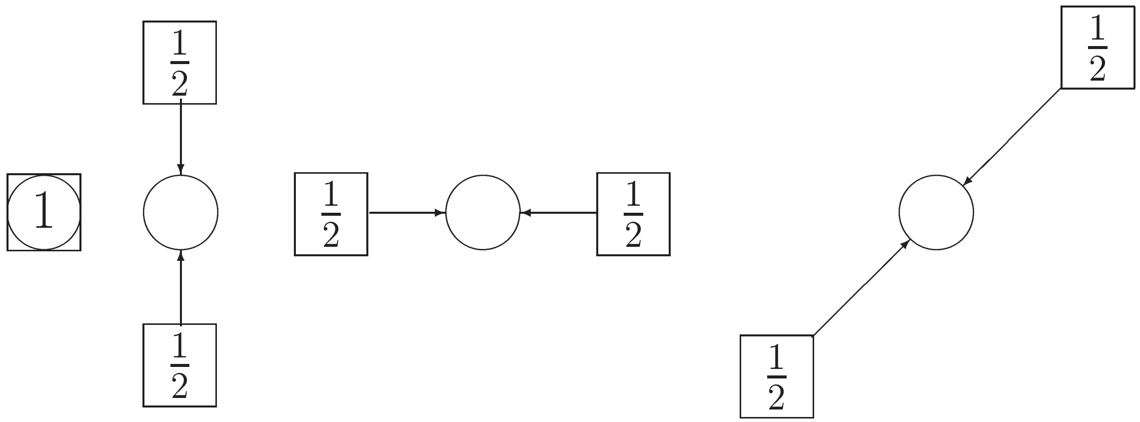
Ein weiteres Beispiel wäre die fehlenden Nachbarn auch mit einzubeziehen.

Mehrgitterverfahren im 2D–Fall

Hierzu passende Prolongation.

$$x(ih, jh) = \begin{cases} y(kH, lH) & i = 2k, j = 2l \\ \frac{1}{2}\left[y(kH, lH) + y(kH + H, lH)\right] & i = 2k+1, j = 2l \\ \frac{1}{2}\left[y(kH, lH) + y(kH, lH + H)\right] & i = 2k, j = 2l+1 \\ \frac{1}{2}\left[y(kH, lH) + y(kH + H, lH + H)\right] & i = 2k+1, j = 2l+1 \end{cases}$$

Zugehörige Sterne



Mehrgitterverfahren im 2D–Fall

2D Randwertproblem, Mehrgitterverfahren

	Cholesky	V–Zykl.,$\nu = 2$		W–Zykl.,$\nu = 2$	
h	Zeit (flops)	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.
$\frac{1}{31}$	$2.7{\cdot}10^5$	$1.7{\cdot}10^6$	26	$2.3{\cdot}10^6$	20
$\frac{1}{63}$	$3.0{\cdot}10^6$	$7.7{\cdot}10^6$	27	$1.0{\cdot}10^7$	20
$\frac{1}{127}$	$2.9{\cdot}10^7$	$3.2{\cdot}10^7$	27	$4.5{\cdot}10^7$	21
$\frac{1}{255}$	$2.9{\cdot}10^8$	$1.3{\cdot}10^8$	28	$1.9{\cdot}10^8$	21

- direktes Verfahren schneidet noch ganz gut ab
- Mehrgitterverfahren hier mit Jacobi. Verbesserung mit Gauß–Seidel–Glätter

Mehrgitterverfahren als Vorkonditionierer

Mehrgitterapproximation als Vorkonditionierer für *cg*. Hierzu nötig:

- $\nu_1 = \nu_2$ (Genauso viel Nachglättung wie Vorglättung)
- bei Nachglättung $S_{h_s}^{\top}$ anstelle von S_{h_s}.

2D Randwertproblem cg mit Mehrgittervork.

h	V–Zykl.,$\nu_1 = \nu_2 = 1$				W–Zykl.,$\nu_1 = \nu_2 = 1$			
	ged. Jacobi		Gauß–Seidel		ged. Jacobi		Gauß–Seidel	
	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.
$\frac{1}{31}$	$6.8{\cdot}10^5$	11	$6.6{\cdot}10^5$	9	$1.1{\cdot}10^6$	10	$9.7{\cdot}10^5$	8
$\frac{1}{63}$	$2.9{\cdot}10^6$	11	$2.8{\cdot}10^6$	9	$4.7{\cdot}10^6$	10	$4.3{\cdot}10^6$	8
$\frac{1}{127}$	$1.3{\cdot}10^7$	12	$1.2{\cdot}10^7$	9	$2.0{\cdot}10^7$	10	$1.8{\cdot}10^7$	8
$\frac{1}{255}$	$5.3{\cdot}10^7$	12	$4.7{\cdot}10^7$	9	$8.2{\cdot}10^7$	10	$7.5{\cdot}10^7$	8

Mehrgitterverfahren — Zusammenfassung

- $\oplus$ überlegene auf das Problem angepasstes Verfahren
- $\oplus$ Verfahren skaliert linear
- $\oplus$ Speicherbedarf moderat
- $\oplus$ Auf ähnliche Probleme übertragbar (Finite Elemente, Sattelpunktprobleme, Parabolische Aufgaben, ...)
- $\ominus$ Benötigt problembedingt eine vorher generierte Gitterhierarchie
- $\ominus$ Nicht mehr einsetzbar falls nur das feinste Gitter bekannt ist.
- $\ominus$ Für viele Problemklassen muss es individuell angepasst werden (robustere Glätter, andere Restriktion/Prolongation)

Versuch eines Ausweges in den letzten drei Fällen:

Algebraische Mehrgitterverfahren

Algebraische Mehrgitterverfahren

Was sind AMGs?

- Idee ist, nur die Matrix als Information zu verwenden
- Künstlich Gitter generieren ohne geometrische Bedeutung
- Restriktion/Prolongation aus der Matrix ableiten
- In Analogie zu Mehrgitterverfahren einsetzen

…und wozu braucht man sie?

- Anwendungen wo Mehrgitterverf. nicht anwendbar sind, etwa fehlende Gitterhierarchie, einige FEM–Pakete, Chip–Design, keine part. Dgl.
- Anwendungen, die erhebliche Anpassungen des Mehrgitterverfahrens erfordern, bzw. keine Mehrgitterverfahren vorhanden
- Bei Verwendung von Geomterieinformation komplexe Datenstrukturen verbunden mit vorhandenen Hierarchien

AMG — Schema

Zur formalen Anwendung des Mehrgitterverfahren benötigt man

- Ausgangsmatrix $A_1 = A$
- Folge R_s, P_s von Restriktionen und Prolongationen, hier: $R_s = P_s^\top$
- zugehörige Grobgittersysteme, hier: $A_{s+1} := P_s^\top A_s P_s$
- Folge von Glättern, hier Gauß–Seidel / symm. Gauß–Seidel

Aufgabe des AMG. A_1 gegeben. AMG erzeugt

$$
\begin{array}{rl}
 & \downarrow \\
P_1, & A_2 = P_1^\top A_1 P_1 \\
 & \downarrow \\
P_2, & A_3 = P_2^\top A_2 P_2 \\
 & \downarrow \\
P_3, & A_4 = P_3^\top A_3 P_3 \\
 & \downarrow \\
 & \vdots
\end{array}
$$

Vorgehen in entgegengesetzter Richtung wie bei geom. Mehrgitterverf.

Das AMG von Ruge/Stüben

Voraussetzung an die Matrix

- $A = (a_{ij})_{i,j}$ mit positiven Diagonaleinträgen, Ausserhalb der Diagonale Null oder negative Einträge.
- Zeilensummen von A nichtnegtiv (schwache Diagonaldominanz)
- Eigenschaften fordern i.w. M–Matrixeigenschaft
- wesentliche Einschränkung für die Anwendbarkeit
- bei wenigen positiven Aussendiagonaleinträgen diese ignorieren

Im folgenden werden Begriffe wie Glättung und Grobgitterkorrektur verwendet (analog zu geometrischen Mehrgitterverfahren), ohne das hierbei geometrische Eigenschaften gemeint sind.

Das AMG von Ruge/Stüben — Die Grundidee

Ziel von Restriktion/Prolongation: Die Fehleranteile e nähern mit

$$(Ae)_i \approx 0, \forall i.$$

Fehleranteile e mit dieser Eigenschaft: "glatt" i.S. des AMG

$$\Rightarrow e_i \approx \sum_{j:\ a_{ij}<0} \frac{|a_{ij}|}{a_{ii}} e_j$$

ausserdem (wg. schwacher Diagonaldominanz)

$$\sum_{j:\ a_{ij}<0} \frac{|a_{ij}|}{a_{ii}} \leqslant 1.$$

Eigentlich nur Zeilen mit Zeilensummen $\sum_j a_{ij} \approx 0$ interessant, sonst Diagonaldominanz, Sonderbehandlung.

$\rightarrow$ glatte Fehler (nahezu) Konvexkombination ihrer Nachbarkomponenten (im Sinne des durch A erzeugten Graphen)

Das AMG von Ruge/Stüben — Die Grundidee

Sei $\omega \in (0,1]$ fest, etwa $\omega = \frac{1}{4}$.

$$\mathcal{S}_i = \{j \neq i : \ |a_{ij}| \geqslant \omega \max_{k \neq i} |a_{ik}|\}$$

$\mathcal{S}_i$ "starke Nachbarn" von i. Diejenigen j für die $\frac{|a_{ij}|}{a_{ii}}$ hinreichend groß ist.
Restliche Nachbarn von i: schwache Nachbarn.

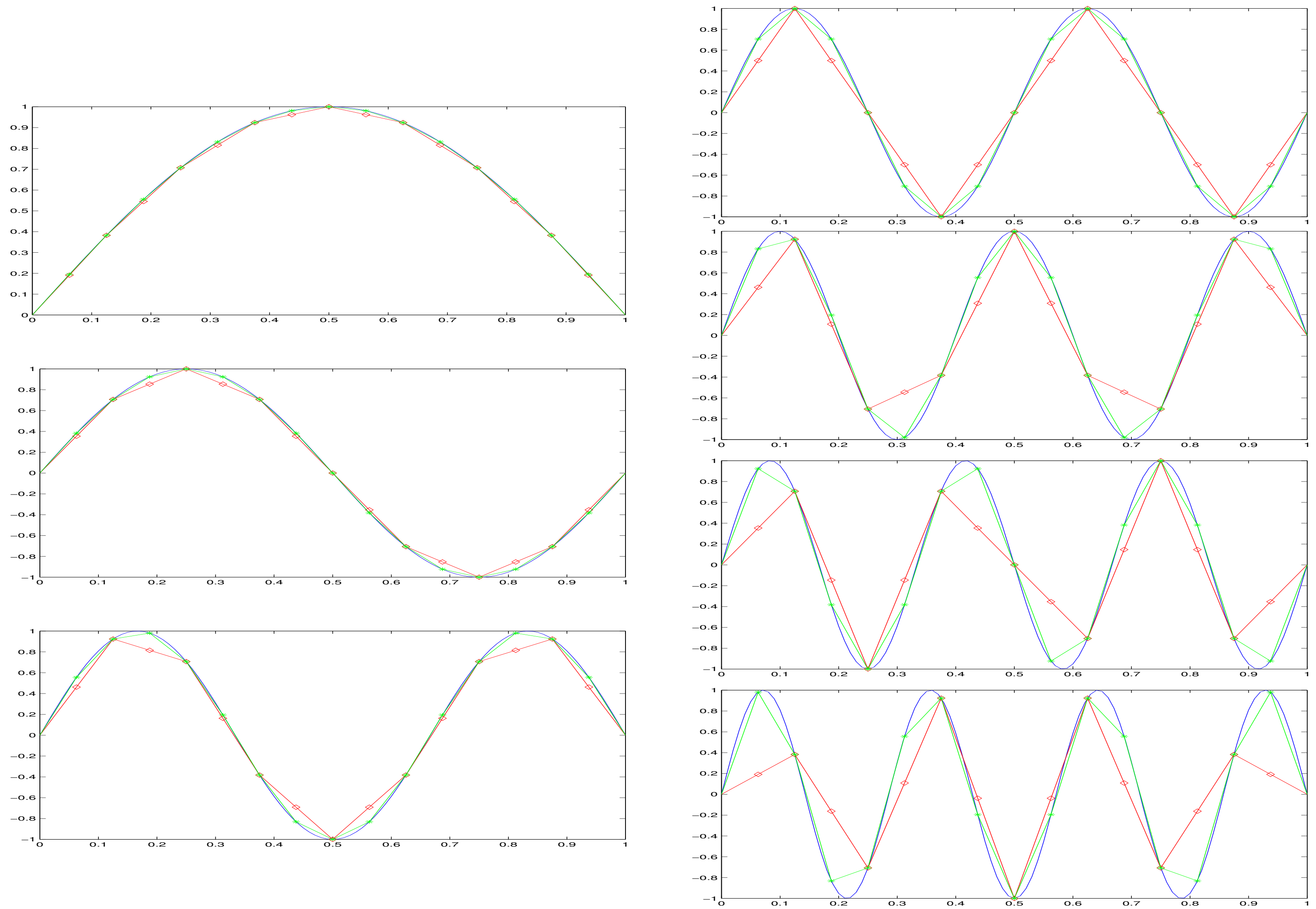
$\rightarrow$ glatte Fehler e_i sollte durch Konvexkombination der "starken Nachbarn" und zugehörigen Gewichte $\frac{|a_{ij}|}{a_{ii}}$ interpolierbar sein.

Beobachtung: "glatte" Fehler ändern sich nur geringfügig ändern in Richtung der starken Nachbarn.

Zur Erinnerung: Bei geometrischen Mehrgitterverfahren sind glatte Frequenzen nach Glättungsschritt durch gröberes Gitter darstellbar.

Das AMG von Ruge/Stüben — Die Grundidee

1D–Randwertproblem. Niedrige Frequenzen durch Grobgitter interpoliert



Das AMG von Ruge/Stüben — Die Grundidee

Strategie: e_i mit $\sum_j a_{ij} \approx 0$ sukzessive durch Interpolation ersetzen.

$\Rightarrow$ "starke Nachbarn" von i ins Grobgitter, i ins "Feingitter"

Strategie feiner. Versuche kleine Zahl an Grobgitterknoten zu erreichen.

$$\mathcal{S}_i^\top = \{j : \; i \in S_j\}.$$

$\mathcal{S}_i$: Solche Knoten i, welche für andere j starke Nachbarn sind.
i wird von vielen j für Interpolation benötigt.
$\to$ schiebe i ins Grobgitter, j ins Feingitter.

Auswahl von Knoten erfolgt mit Kostenfunktion λ, zu Anfang $\lambda_i = \#\mathcal{S}_i^\top$.
Bei Auswahl von i geeignete Anpassung der Nachbarwerte λ_j.

Algorithmus mit zwei Durchläufen

1. Vorläufige Wahl der Grobgitterknoten
2. Endgültige Wahl und Definition der Prolongation

Das AMG von Ruge/Stüben — Algorithmus 1

Vorläufige Wahl der Grobgitterpunkte.

$\mathcal{C}$ Grobgitterpunkte, $\mathcal{F}$ Feingitterpunkte, $\mathcal{U}$ unbestimmte Knoten.

Zu Beginn setze $\lambda_j = \#\mathcal{S}_j^\top,\ \forall j,\ \ \mathcal{C} = \mathcal{F} = \emptyset, \mathcal{U} = \{1, \ldots, n\}$.

Solange $\mathcal{U} \neq \emptyset$.
  Wähle ein i mit maximalem λ_i.

  % Verschiebe i ins Grobgitter
  $\mathcal{C} = \mathcal{C} \cup \{i\}, \mathcal{U} = \mathcal{U} \setminus \{i\}$
  Für alle $j \in \mathcal{S}_i^\top \cap \mathcal{U}$:
    % Nachbarn, für die i starker Nachbar ist, verschiebe ins Feingitter
    $\mathcal{F} = \mathcal{F} \cup \{j\}, \mathcal{U} = \mathcal{U} \setminus \{j\}$
    % Zum Ausgleich erhöhe Gewicht der starken Nachbarn
    Für alle $k \in \mathcal{S}_j \cap \mathcal{U}$: $\lambda_k = \lambda_k + 1$

  % Adaptiere λ_j bei starken Nachbarn, weil i jetzt fehlt
  Für alle $j \in \mathcal{S}_i \cap \mathcal{U}$: $\lambda_j = \lambda_j - 1$

Das AMG von Ruge/Stüben — Beispiel

$$T_h = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \end{pmatrix} \text{ aus 1D–Randwertproblem.}$$

$\mathcal{S}_1 = \{2\}$, $S_i = \{i-1, i+1\}$, $i = 2, \ldots n-1$, $\mathcal{S}_n = \{n-1\}$. $\mathcal{S}_i^\top = \mathcal{S}_i \; \forall i$.

Algorithmus startet mit maximalem λ_i, z.B. $i = 2$. Illustrieren einige Schritte anhand des Graphen von T_h.

$i \in \mathcal{U}$: Wert λ_i angeben $\qquad$ $i \in \mathcal{C}$: $\blacksquare$ $\qquad$ $j \in \mathcal{F}$: $\bullet$

$$\begin{array}{c}
1 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - \quad \cdots \\
\downarrow \\
\bullet - \blacksquare - \bullet - 3 - 2 - 2 - 2 - 2 - \quad \cdots \\
\downarrow \\
\bullet - \blacksquare - \bullet - \blacksquare - \bullet - 3 - 2 - 2 - \quad \cdots \\
\downarrow \\
\bullet - \blacksquare - \bullet - \blacksquare - \bullet - \blacksquare - \bullet - 3 - \quad \cdots
\end{array}$$

Das AMG von Ruge/Stüben — Beispiel

$- \operatorname{div}\,(a \operatorname{grad} u) = f. \quad a$ mit Koeffizientensprung

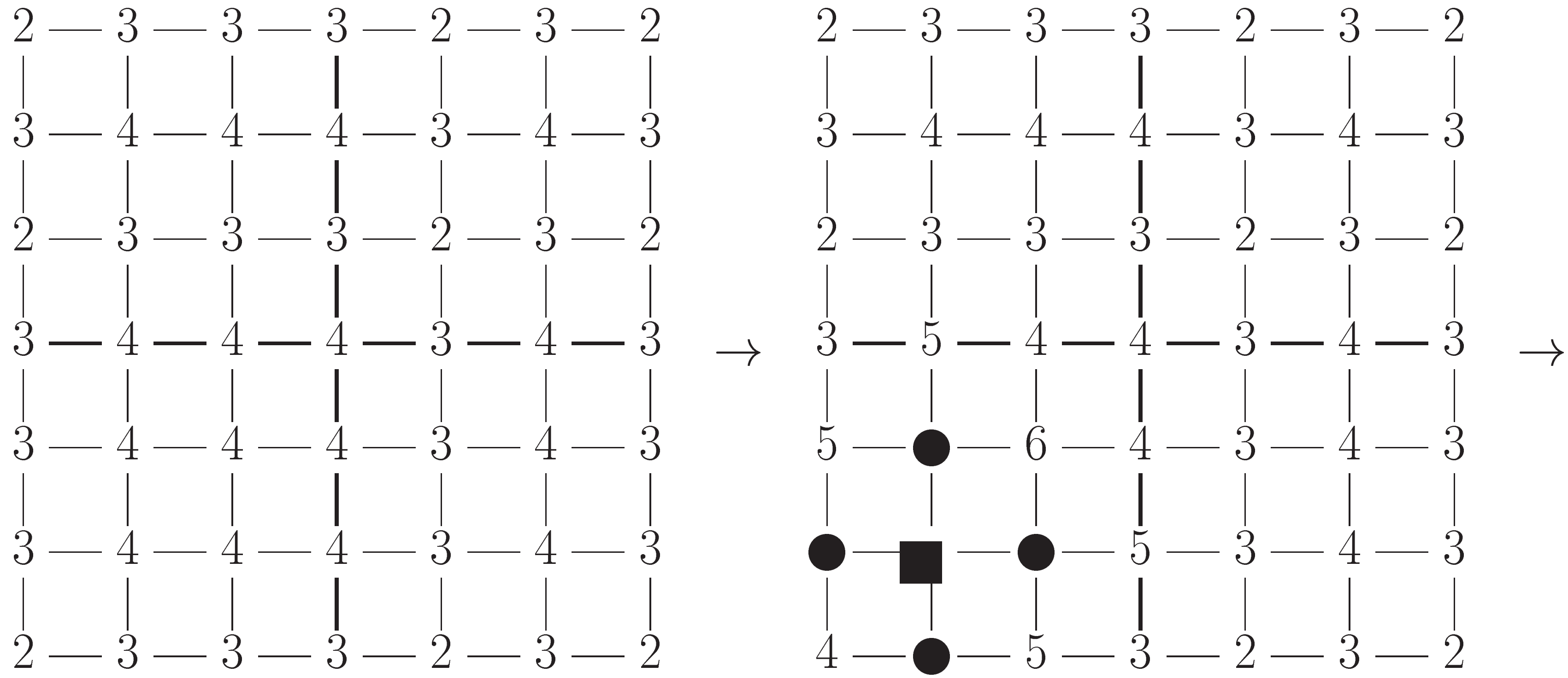
10	1
100	10

Werte anhand eines 7×7 Gitters. Zuerst die Werte für $\#\mathcal{S}$.

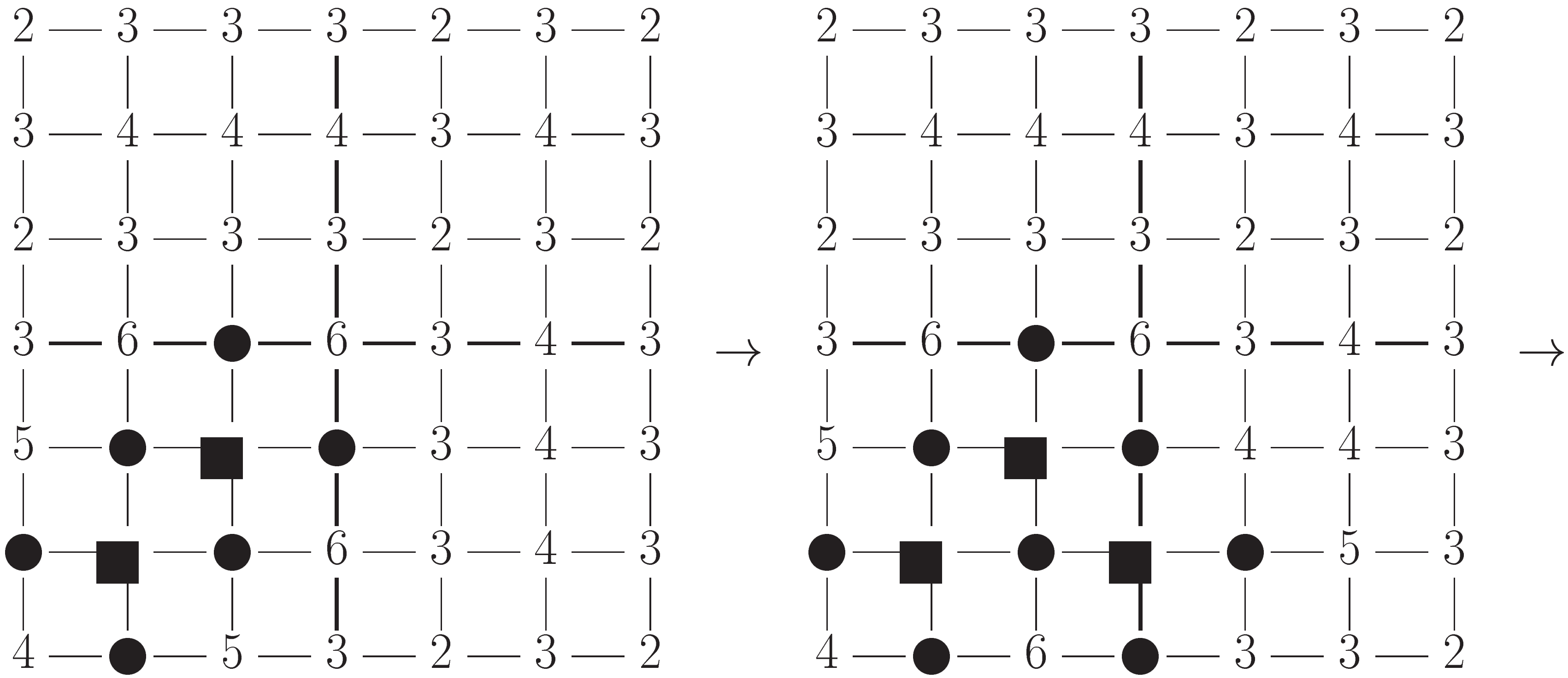
$$
\begin{array}{ccccccccccccc}
2 & — & 3 & — & 3 & — & 2 & — & 3 & — & 3 & — & 2 \\
| & & | & & | & & | & & | & & | & & | \\
3 & — & 4 & — & 4 & — & 3 & — & 4 & — & 4 & — & 3 \\
| & & | & & | & & | & & | & & | & & | \\
3 & — & 4 & — & 4 & — & 3 & — & 4 & — & 4 & — & 3 \\
| & & | & & | & & | & & | & & | & & | \\
2 & — & 3 & — & 3 & — & 2 & — & 3 & — & 3 & — & 2 \\
| & & | & & | & & | & & | & & | & & | \\
3 & — & 4 & — & 4 & — & 3 & — & 4 & — & 4 & — & 3 \\
| & & | & & | & & | & & | & & | & & | \\
3 & — & 4 & — & 4 & — & 3 & — & 4 & — & 4 & — & 3 \\
| & & | & & | & & | & & | & & | & & | \\
2 & — & 3 & — & 3 & — & 2 & — & 3 & — & 3 & — & 2
\end{array}
$$

Das AMG von Ruge/Stüben — Beispiel

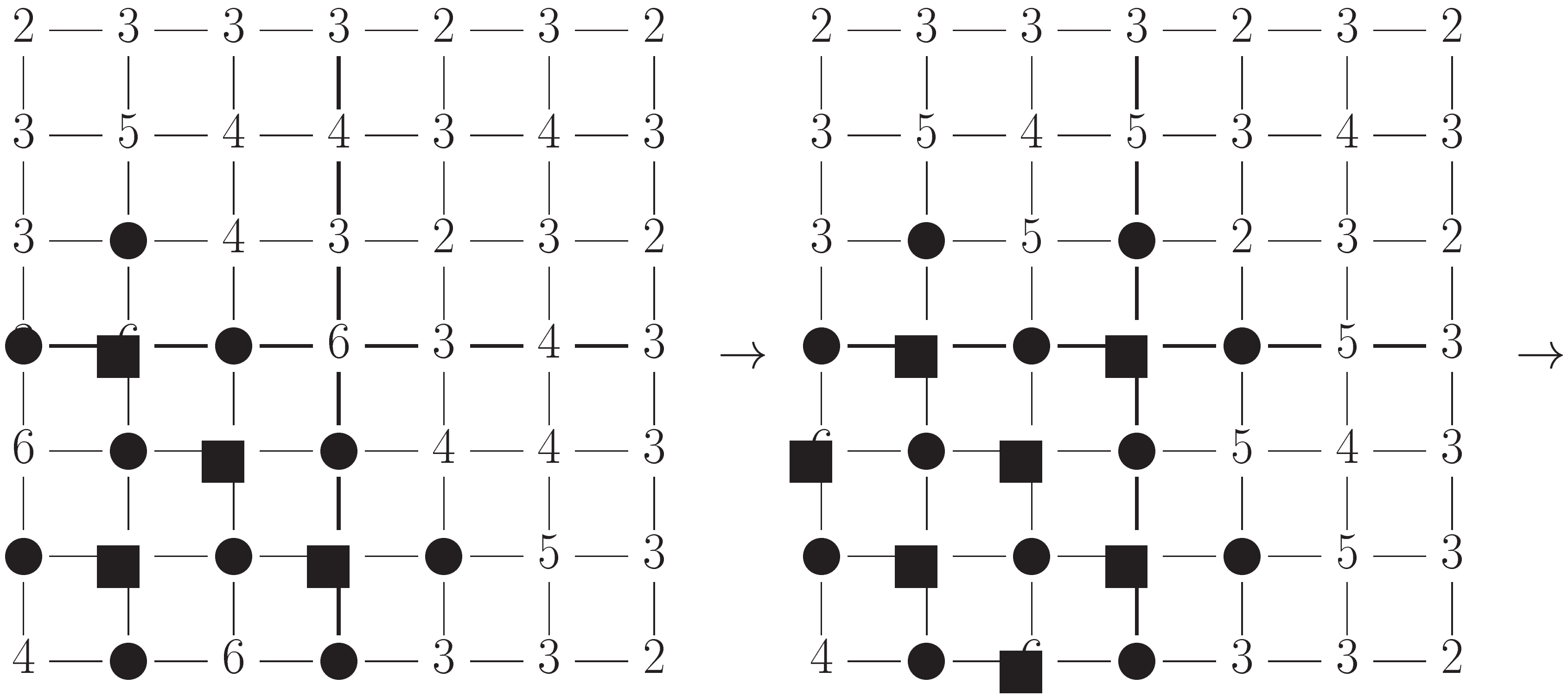
Wir fangen mit Knoten $(2, 2)$ an und zeigen $\lambda_j = \#\mathcal{S}^\top$.



Das AMG von Ruge/Stüben — Beispiel



Das AMG von Ruge/Stüben — Beispiel



Das AMG von Ruge/Stüben

- Algorithmus 1 selektiert Grobgitterknoten nach Priorität.
- Gleichzeitig werden Feingitterpunkte bestimmt
- Algorithmus geht "lokal" vor
- Nachteil. Eventuell zu viele Feingitterpunte.

2. Algorithmus nötig. Ziel des Algorithmus

- Nach Alg. 1 gibt es Punkte $i \in \mathcal{F}$, mit $\mathcal{S}_i \not\subseteq \mathcal{C}$.
  $i \longrightarrow \mathcal{C}$ nur im Notfall ein Ausweg (zu viele Grobgitterpunkte)
- Alternativen.
  - Für $j \in \mathcal{S}_i \setminus \mathcal{C}$ Gewicht $\frac{|a_{ij}|}{a_{ii}}$ verschieben auf $\mathcal{S}_j \subseteq \mathcal{C}$, falls möglich
    Brauchen $\mathcal{S}_j \cap \mathcal{S}_i \cap \mathcal{C} \neq \emptyset$
  - Sonst: Abwegen, ob $i \longrightarrow \mathcal{C}$ oder $j \longrightarrow \mathcal{C}$ besser ist
- Interpolationsgewichte entsprechend anpassen und definieren
- Dabei schwache Nachbarn durch Diagonalkompensation beseitigen

Das AMG von Ruge/Stüben — Algorithmus 2

$\mathcal{C}$, $\mathcal{F}$ (aus Alg. 1), $\mathcal{T}$ getestete Feingitterknoten. Zu Beginn $\mathcal{T} = \emptyset$.

Solange $\mathcal{T} \not\supseteq \mathcal{F}$.

% Teste einen noch nicht getesteten Feingitterknoten

Wähle $i \in \mathcal{F} \setminus \mathcal{T}$, $\mathcal{T} = \mathcal{T} \cup \{i\}$

$\mathcal{C}_i = \mathcal{S}_i \cap \mathcal{C}$, $\mathcal{D}_i = \mathcal{S}_i \setminus \mathcal{C}_i$ % Starke Nachbarn im Grob–/Feingitter

$\tilde{\mathcal{C}}_i = \emptyset$ % Zahl der Elemente, die nach $\mathcal{C}$ verschoben werden sollen

$d_i = a_{ii} + \sum_{j:\ j \notin \mathcal{S}_i} a_{ij}$, für alle $j \in \mathcal{C}_i$: $d_j = |a_{ij}|$ % Diagonalkompensation

Für alle $j \in \mathcal{D}_i$:

% hat j mit i gemeinsamen starken Grobgitternachbarn?

Falls $\mathcal{S}_j \cap \mathcal{C}_i \neq \emptyset$

% Verschiebe Gewicht d_j auf alle $k \in \mathcal{C}_i$

$s = \sum_{k \in \mathcal{C}_i} |a_{jk}|$. Für alle $k \in \mathcal{C}_i$: $d_k = d_k + |a_{ij}| \frac{|a_{jk}|}{s}$

Das AMG von Ruge/Stüben — Algorithmus 2

Sonst: falls $\tilde{C}_i = \emptyset$ % Bisher kein $j \in \mathcal{D}_i$ versuchsweise im Grobgitter

% Versuche erstmalig für i, j dem groben Gitter hinzuzufügen

$\tilde{\mathcal{C}}_i = \{j\}$, $\mathcal{C}_i = \mathcal{C}_i \cup \{j\}$, $\mathcal{D}_i = \mathcal{D}_i \setminus \{j\}$, $d_j = |a_{ij}|$

Sonst: % ein $j \in \mathcal{D}_i$ bereits versuchsweise im Grobgitter

$\tilde{\mathcal{C}}_i = \{i\}$, $\mathcal{D}_i = \mathcal{C}_i = \emptyset$ % Lieber $i \longrightarrow \mathcal{C}$ statt $j \longrightarrow \mathcal{C}$

$\mathcal{C} = \mathcal{C} \cup \tilde{\mathcal{C}}_i$, $\mathcal{F} = \mathcal{F} \setminus \tilde{\mathcal{C}}_i$ % Endgültig: $\tilde{\mathcal{C}}_i \longrightarrow \mathcal{C}$

Für alle $j \in \mathcal{C}_i$: $p_{ij} = \frac{d_j}{d_i}$ % Bilde gemittelte Interpolationsgewichte

Für alle $i \in \mathcal{C}$: $p_{ii} = 1$ % Triviale Einbettung für Grobgitterpunkte

Das AMG von Ruge/Stüben — Algorithmus 1/2

Anmerkungen.

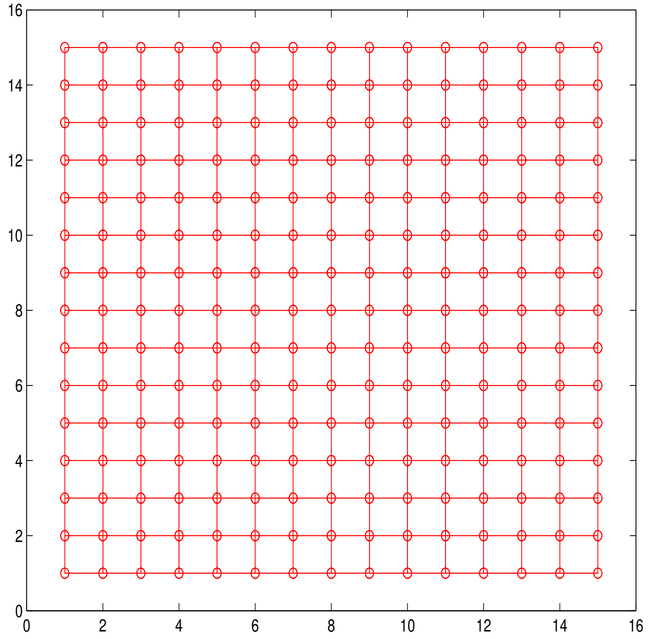
- Algorithmus 2 berechnet Prolongation P vom Grobgitter ins Feingitter (Nullspalten zu Einträgen aus $\mathcal{F}$ streichen)
- Unter relativ allgemeinen Bedingungen kann man zeigen, dass $A_H = P^\top AP$ wieder die Vorzeicheneigenschaft für die Einträge erfüllt. Ausserdem bleiben die Zeilensummen nichtnegativ
- AMG somit nicht nur Zweigitter–Verfahren, sondern wird mittels Rekursion zum Mehrgitter–Verfahren
- Als Glätter üblicher Weise Gauß–Seidel
- Kosten für Generierung der der Gitter inklusive $P_1, A_2,\ P_2, A_3,\ \ldots,$ P_{l-1}, A_l überlagern zum Teil den numerischen Anteil zur Lösung des Gleichungssystems.

Das AMG von Ruge/Stüben — Beispiel

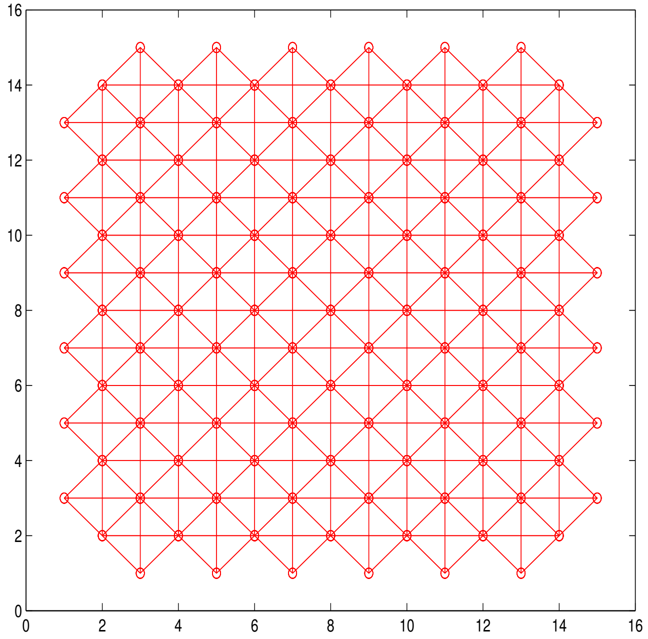
2D Randwertproblem

Algebraisch erzeugte Gitter

Level 1



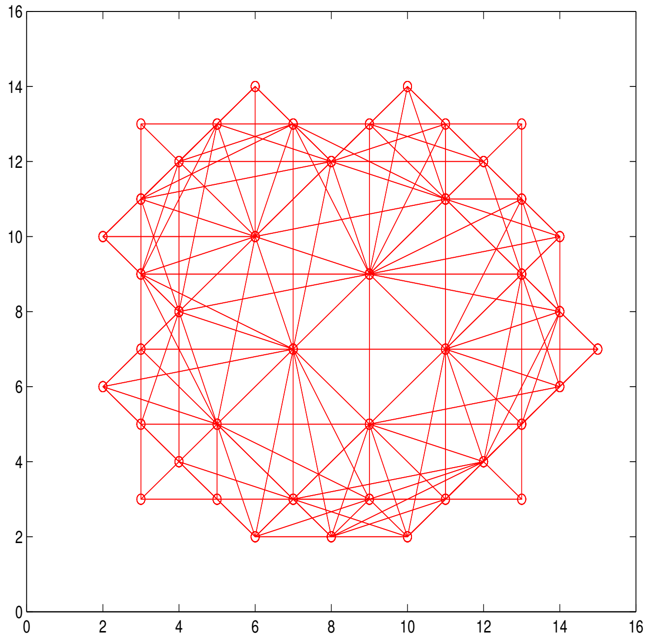
Level 2



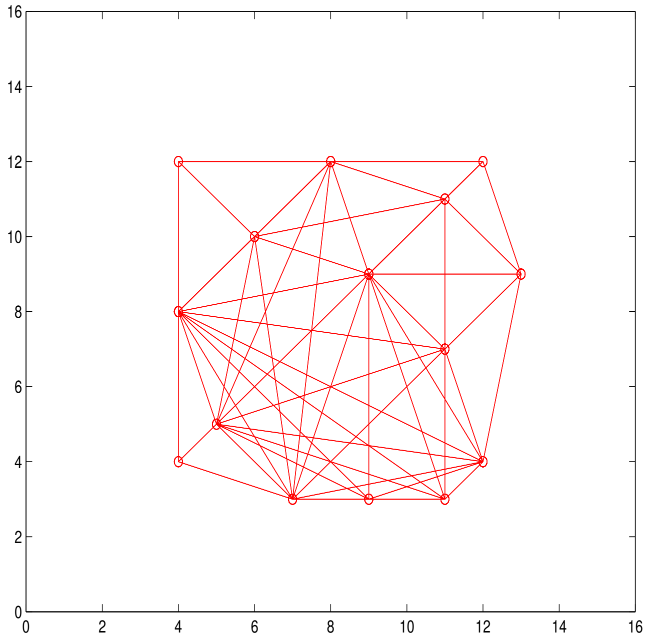
Das AMG von Ruge/Stüben — Beispiel

Algebraisch erzeugte Gitter

Level 3



Level 4



Das AMG von Ruge/Stüben — Beispiel

Vergröberungsprozess

Level	n	# Nichtnullel./n
1	3969	4.9
2	1981	8.6
3	608	10.7
4	208	14.1
5	62	13.5
6	18	10.8

flops für vorkond. cg–Verfahren

n	Gauß–Seidel	
225	$2.7{\cdot}10^5$	11
961	$1.4{\cdot}10^6$	12
3969	$5.8{\cdot}10^6$	12

- Verglichen mit geom. Mehrgitterverfahren schwächere Reduktion der Gittergröße, trotzdem recht gut
- Durchschnittliche Zahl der Nichtnullelemente moderat, aber Anstieg
- Sicherlich langsamer als geom. Mehrgitterverfahren, aber noch recht gut
- Zeit für Genereierung der Gitter muss mit einkalkuliert werden, überlagert in der Regel die Zeit für den Lösungsprozess.

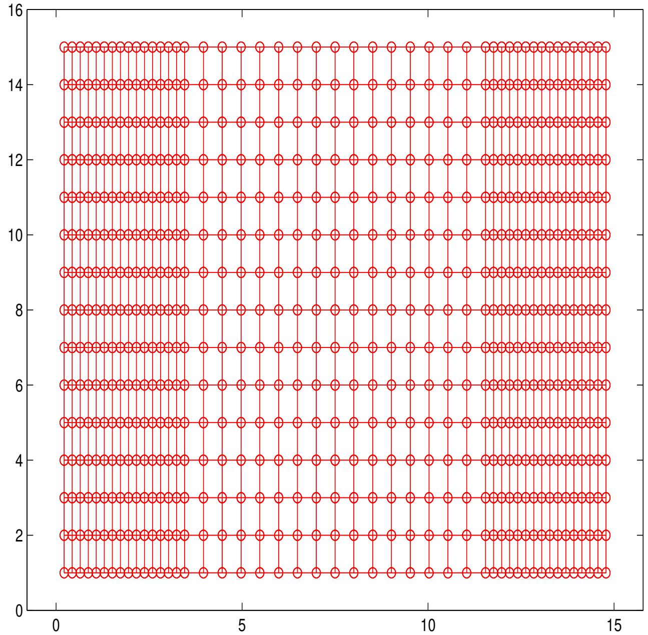
Das AMG von Ruge/Stüben — Beispiel

$$-\varepsilon^2 u_{xx} - u_{yy} = f \text{ in } [0,1]^2,$$

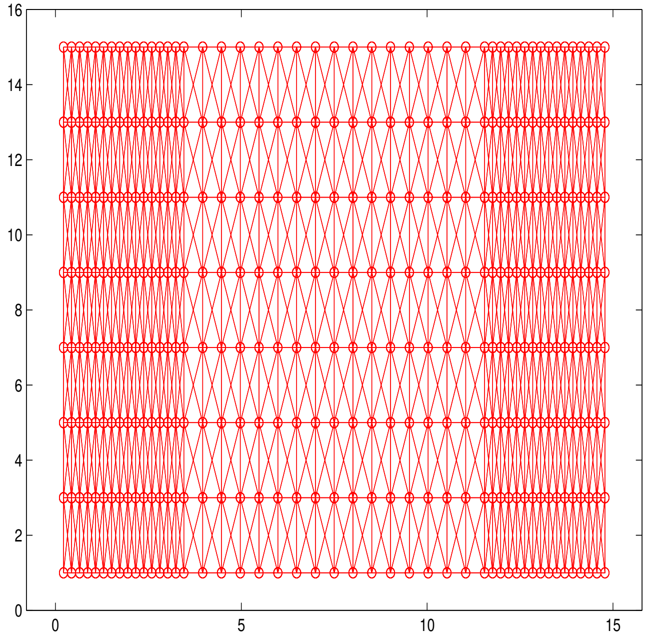
Hier $\varepsilon = .1$. Am linken und rechten Rand Grenzschicht der Breite $\varepsilon |\ln \varepsilon|$ vorsehen

Klare Ausrichtung in Richtung der y–Achse

Level 1

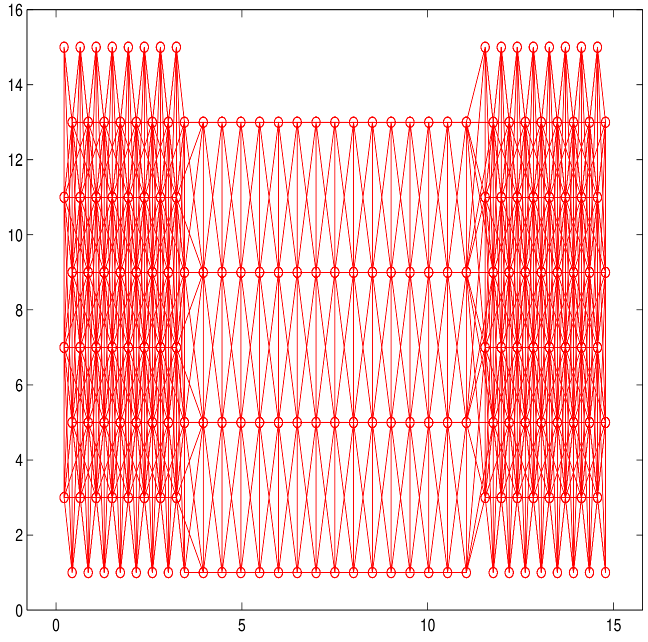


Level 2

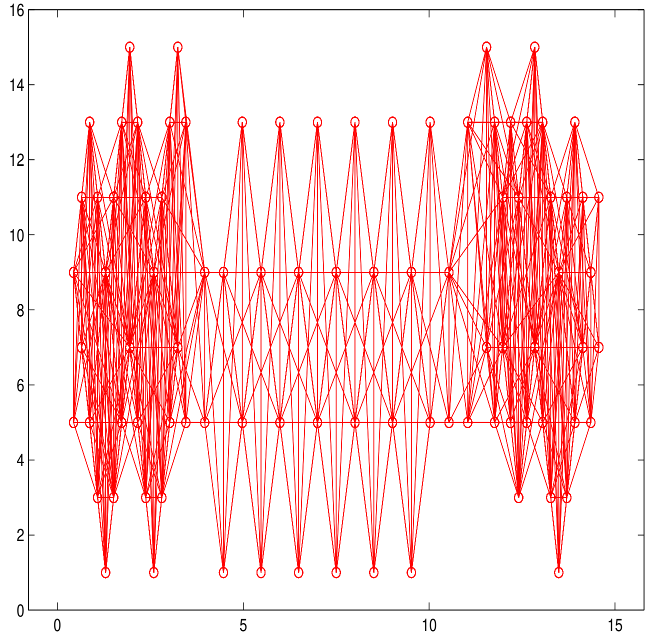


Das AMG von Ruge/Stüben — Beispiel

Level 3



Level 4



Das AMG von Ruge/Stüben — Beispiel

Vergröberungsprozess

Level	n	# Nichtnullel./n
1	12033	5.0
2	6112	8.8
3	3056	15.8
4	1171	18.0
5	428	17.6
6	184	20.9
7	56	11.1

flops für vorkond. GMRES(20)

n	Gauß–Seidel	
705	$9.2 \cdot 10^5$	9
2945	$4.7 \cdot 10^6$	10
12033	$1.9 \cdot 10^7$	10

- Aufgrund der Diskretisierung leichte Unsymmetrie, deshalb GMRES statt cg
- Bei FEM ließe sich das vermeiden
- Auch hier erkennbarer Anstieg beim fill–in
- Verfahren skaliert noch recht gut

Das AMG von Ruge/Stüben — Matrixnotation

Algorithmus 1:

$$A \to \begin{pmatrix} I & P_{\mathcal{C},\mathcal{F}} \\ P_{\mathcal{F},\mathcal{C}} & P_{\mathcal{F},\mathcal{F}} \end{pmatrix} + E,$$

E entspricht i.w. Beiträgen der schwachen Nachbarn (zeilenweise)

Algorithmus 2.

Diagonalkompensation $E \longrightarrow D$, D diagonal, gleiche Zeilensummen

Einträge ins Grobgitter verschieben $\Rightarrow$ partitioniere Matrix neu

Sonst: Verschiebung von Gewichten $\begin{pmatrix} P_{\mathcal{F},\mathcal{C}} & P_{\mathcal{F},\mathcal{F}} \end{pmatrix}$ weg vom Diagonalblock $P_{\mathcal{F},\mathcal{F}}$ hin zum Aussendiagonalblock $P_{\mathcal{F},\mathcal{C}}$, so dass $\begin{pmatrix} P^{(neu)}_{\mathcal{F},\mathcal{C}} & D_{\mathcal{F},\mathcal{F}} \end{pmatrix}$ gleiche Zeilensummen hat und $D_{\mathcal{F},\mathcal{F}}$ schließlich diagonal ist.

$$A \longrightarrow \begin{pmatrix} I + E_{\mathcal{C},\mathcal{C}} & P_{\mathcal{C},\mathcal{F}} \\ P^{(neu)}_{\mathcal{F},\mathcal{C}} & D_{\mathcal{F},\mathcal{F}} \end{pmatrix}, \quad P = \begin{pmatrix} I \\ -D^{-1}_{\mathcal{F},\mathcal{F}} P^{(neu)}_{\mathcal{F},\mathcal{C}} \end{pmatrix}$$

Das AMG von Braess

Voraussetzung hier: A symmetrisch positiv definit. (keine M–Matrix Eig.)

Hauptidee: Knoten $\{1, \ldots, n\}$ geschickt in disjunkten Clustern $\mathcal{I}_1, \ldots, \mathcal{I}_m$ zusammenzufassen und daraus Restriktion R durch Summation bilden.

$$y = Rx, \text{ wobei } y_k = \sum_{i \in \mathcal{I}_k} x_i, \forall k$$

Prolongation $P = R^\top$, Grobgittersystem $A_H = RAR^\top$.

$\Longrightarrow$ Grobgittereintrag a^H_{kl} aus $A = (a_{ij})_{i,j=1,\ldots,n}$ durch Summation:

$$a^H_{kl} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} (a_{ij})_{i \in \mathcal{I}_k, j \in \mathcal{I}_l} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \sum_{i \in \mathcal{I}_k, j \in \mathcal{I}_l} a_{ij}.$$

Aufgabe: geeignete Cluster $\mathcal{I}_1, \ldots, \mathcal{I}_m$ aufspüren.

Das AMG von Braess — Beispiel

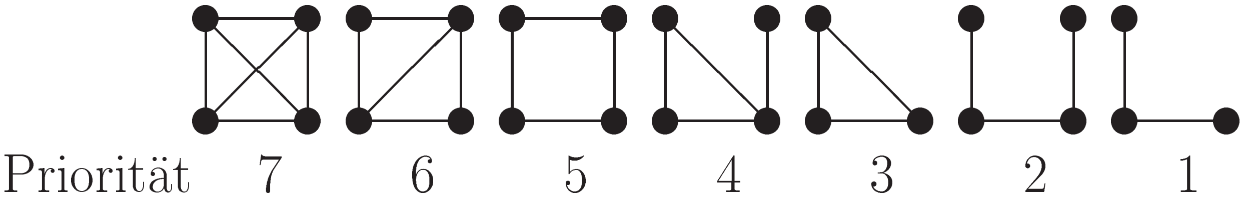
1D–Randwertproblem, T_h, $I_j = \{2j-1, 2j\}$, $\forall j$

$$\begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 2, \quad \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = -1$$

$$\left(\begin{array}{cc|cc|cc|cc}
2 & -1 & & & & & & \\
-1 & 2 & -1 & & & & & \\
\hline
 & -1 & 2 & -1 & & & & \\
 & & -1 & 2 & -1 & & & \\
\hline
 & & & -1 & \ddots & \ddots & & \\
 & & & & \ddots & \ddots & -1 & \\
\hline
 & & & & & -1 & 2 & -1 \\
 & & & & & & -1 & 2
\end{array}\right) \rightarrow \left(\begin{array}{c|c|c|c|c}
2 & -1 & & & \\
\hline
-1 & 2 & -1 & & \\
\hline
 & -1 & \ddots & \ddots & \\
\hline
 & & \ddots & \ddots & -1 \\
\hline
 & & & -1 & 2
\end{array}\right)$$

Das AMG von Braess — Algorithmus 1

Muster für die Bildung von Gruppen nach Priorität



- Gruppen werden jetzt sukzessive gebildet.
- Konstruktion in zwei Schritten. Zunächst Gruppe mit 2 Knoten, derart dass diesse und 2 Knoten einer bereits existierenden Gruppe einen Graphen der Liste enthalten. Anschließend mit größtmöglicher Priorität vervollständigen

Das AMG von Braess — Algorithmus 1

Algorithmus 1

$\mathcal{U} = \{1, \ldots, n\}$. $p = 0$

Solange $\mathcal{U} \neq \emptyset$

  $p = p + 1$

  Wähle Gruppe $\mathcal{G}_p$ von $\leqslant 4$ Knoten mit zulässigem Graphen (vernachlässige Graphen mit $\leqslant 2$ Knoten).

  $\mathcal{U} = \mathcal{U} \setminus \mathcal{G}_p$.

  Solange $\mathcal{G}_1, \ldots, \mathcal{G}_p$ Nachbarn in $\mathcal{U}$ haben

    Sei $i \in \{1, \ldots, p\}$ kleinstmöglich, so dass $\mathcal{G}_i$ Nachbarn in $\mathcal{U}$ hat

    Falls zulässiger Graph mit je 2 Elementen aus $\mathcal{U}$ und $\mathcal{G}_i$ möglich

      Bestimme von diesen den mit höchster Priorität.

      Sind etwa j, k die Knoten aus $\mathcal{U}$, bestimme bestes Muster aus $\mathcal{U}$ das j, k umfasst. Bezeichne Menge als G_{p+1}.

    Sonst:

      Wähle $\leqslant 3$ Nachbarn aus $\mathcal{U}$ die Nachbarn von $\mathcal{G}_i$ sind mit höchster Priorität. Bezeichne Menge als G_{p+1}.

    $p = p + 1, \mathcal{U} = \mathcal{U} \setminus \mathcal{G}_p$

Das AMG von Braess — Algorithmus 1/2

- Algorithmus versucht, Gruppen mit hoher Priorität und hoher Zahl von Verbindungen zu den Nachbarn zu bilden.
- Nachteil: Komplexität, sehr viele Fallunterscheidungen. Numerische Werte von A haben keinerlei Einfluss auf die Konstruktion.

Deshalb zweiter alternativer Algorithmus

- Algorithmus bildet zuerst Paare (i, j) nach größtmöglichem $\frac{a_{ij}^2}{a_{ii}a_{jj}}$.
- Danach Zusammenfassung zu Quadrupeln größtmöglicher Priorität

Das AMG von Braess — Algorithmus 2

Algorithmus 2

1. Schritt: bilde Paare

$\mathcal{U} = \{1, \dots, n\}$. $p = 0$
Solange $\mathcal{U} \neq \emptyset$
    $p = p + 1$. Wähle ein $i \in \mathcal{U}$.
    Falls i mindestens einen Nachbarn in $\mathcal{U}$ hat
        Bestimme Nachbar $j \in \mathcal{U}$ von i mit maximalem $\frac{a_{ij}^2}{a_{ii}a_{jj}}$
        $\mathcal{P}_p = \{i\}$
    Sonst: $\mathcal{P}_p = \{i, j\}$
    $\mathcal{U} = \mathcal{U} \setminus \mathcal{P}_p$

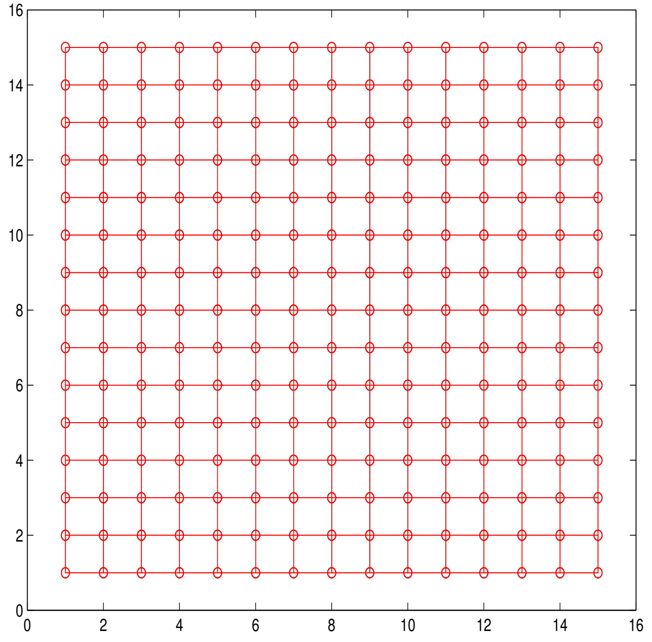
2. Schritt: bilde Gruppen

$\mathcal{U} = \bigcup_{i=1}^{p} \{\mathcal{P}_i\}$. $p = 0$
Solange $\mathcal{U} \neq \emptyset$
    $p = p + 1$. Wähle ein $\mathcal{P}_i \in \mathcal{U}$.
    Sei $m_{ij} = \#\{a_{kl} \neq 0 : k \in \mathcal{P}_i, l \in \mathcal{P}_j\}$
    Falls $m_{ij} > 0$ für ein $\mathcal{P}_j \in \mathcal{U}$
        Bestimme j so, dass m_{ij} maximal für alle $\mathcal{P}_j \in \mathcal{U}$
        $G_p = \mathcal{P}_i \cup \mathcal{P}_j, \mathcal{U} = \mathcal{U} \setminus \{\mathcal{P}_i, \mathcal{P}_j\}$
    Sonst: $\mathcal{G}_p = \mathcal{P}_i, \mathcal{U} = \mathcal{U} \setminus \{\mathcal{P}_i\}$

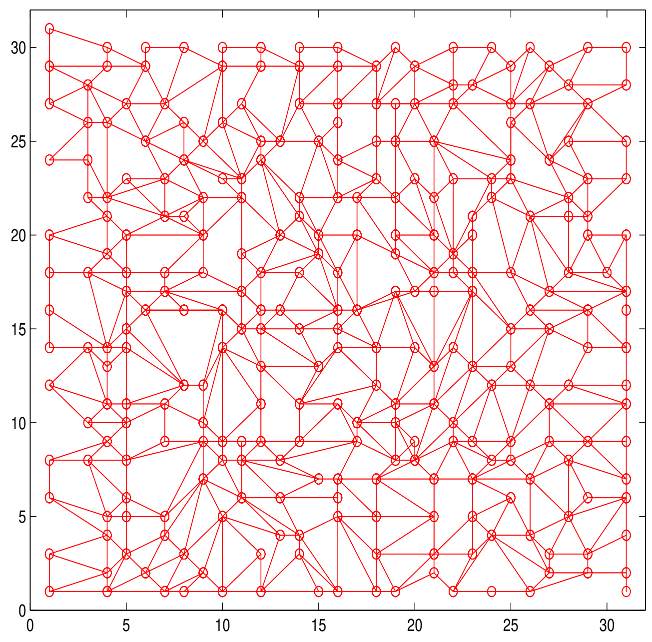
Das AMG von Braess — Beispiel

2D Randwertproblem

Level 1

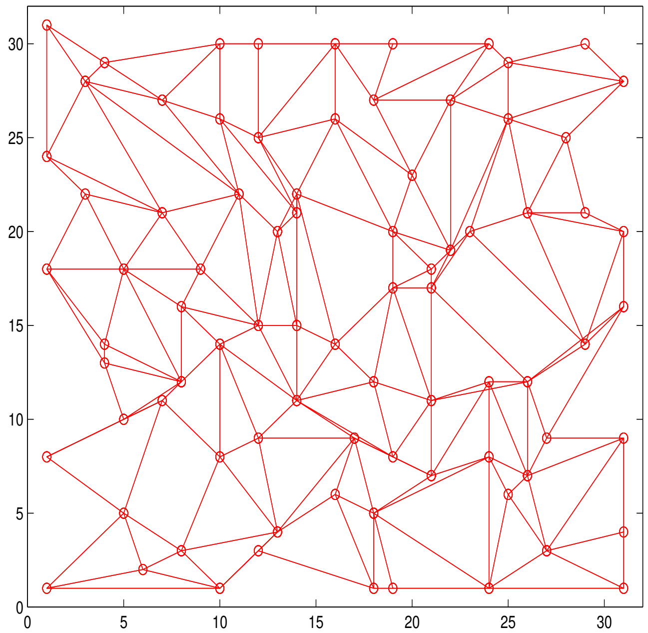


Level 2

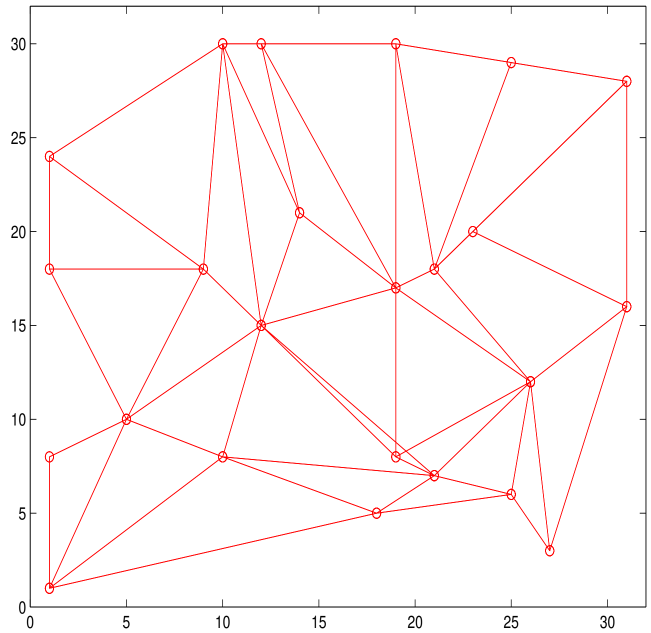


Das AMG von Braess — Beispiel

Level 3



Level 4



Algorithmus vergröbert vorliegendes System recht gut, ähnliche Gittergrößen und Besetztheit wie beim geom. Mehrgitterverfahren

Das AMG von Braess — Beispiel

Vergröberungsprozess

Level	n	# Einträge$(A)/n$
1	3969	4.9
2	1150	6.2
3	340	6.4
4	104	6.1
5	32	5.4
6	11	4.6
7	4	3.0

Vork. cg–Verfahren

	Gauß–Seidel	
n	flops	# Iter.
225	$2.2{\cdot}10^5$	13
961	$1.5{\cdot}10^6$	19
3969	$9.2{\cdot}10^6$	28

- Hier: *cg*–Verfahren mit Mehrgittervorkonditionierung (V11), Gauß–Seidel Glätter
- Verfahren skaliert nicht so gut wie geom. Mehrgitterverfahren.

Das AMG von Braess — Beispiel

Verbesserung

- Verwende als Vor– und Nachglätter symm. Gauß–Seidel
- bei mehr als 2 Leveln Zahl der Glättungsschritte auf den gröberen Gittern zu erhöhen.
- Grobgitterkorrektur wird zusätzlich mit Relaxationsparameter 1.8 versehen.

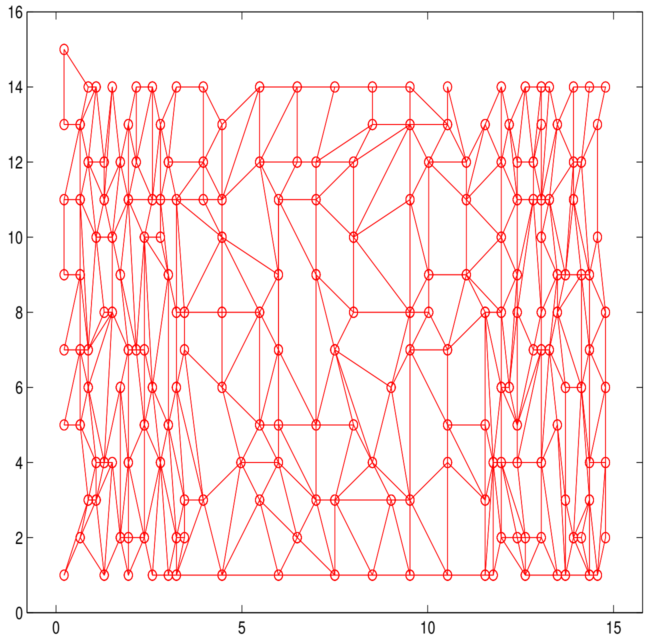
	Symm. Gauß–Seidel	
n	flops	# Iter.
225	$2.0 \cdot 10^5$	9
961	$1.2 \cdot 10^6$	12
3969	$6.0 \cdot 10^6$	14
16129	$3.0 \cdot 10^7$	17

Das AMG von Braess — Beispiel

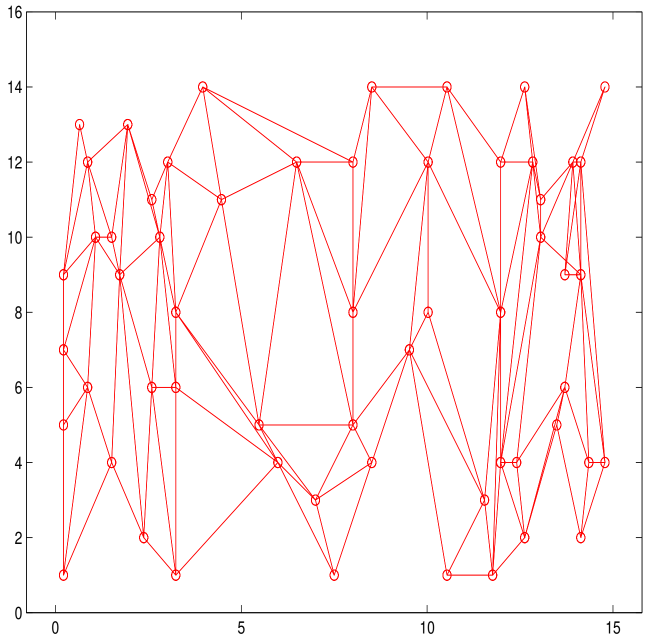
$$-\varepsilon^2 u_{xx} - u_{yy} = f$$

Hier $\varepsilon = .1$. Randschichten der Größe $\varepsilon|\ln \varepsilon|$ werden berücksichtigt.
Konstruktion des AMG bevorzugt y–Richtung,

Level 2

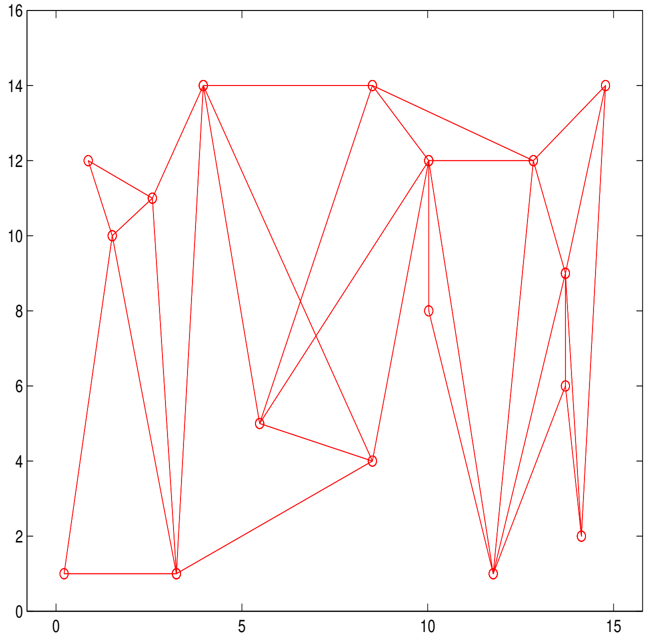


Level 3

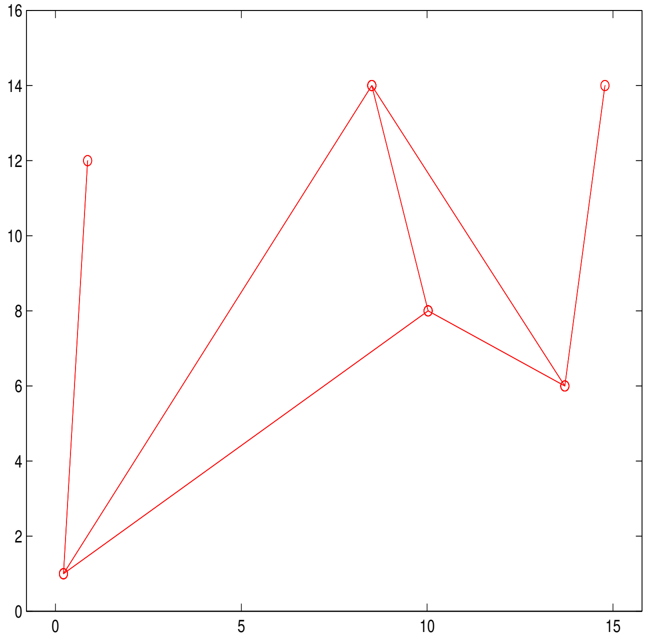


Das AMG von Braess — Beispiel

Level 4



Level 5



Vergröberung verhält sich wiederum ähnlich einem geom. Mehrgitterverfahren.

Das AMG von Braess — Beispiel

Vergröberungsprozess

Level	n	# Nichtnullel.$(A)/n$
1	12033	5.0
2	3464	6.3
3	1005	6.5
4	296	6.3
5	88	5.9
6	26	5.0
7	8	3.5

vorkond. GMRES(20)

	Sym. Gauß–Seidel	
n	flops	# Iter.
705	$1.6 \cdot 10^6$	17
2945	$9.9 \cdot 10^6$	23
12033	$5.4 \cdot 10^7$	31

- zufriedenstellendes Verhalten des AMG
- skaliert nicht so gut wie im ersten Beispiel
- Zeit für die Konstruktion des AMG eher gering (ungefähr drei Schritte des vorkonditionierten *cg*–Verfahrens)

AMGs — Zusammenfassung

- Beide vorgestellte Verfahren liefern brauchbare Werte sofern die Koeffizientenmatrix die passenden Voraussetzungen hat (Ruge/Stüben M–Matrix, Braess: symm. pos. def.)
- Verfahren sind dort einsetzbar, wo zugrunde liegendes Problem keinerlei natürliche Gitterhierarchie hat
- Es gibt viele weitere AMGs, die auf unvollst. Dreieckszerlegungen beruhen. Können hier aus Zeitgründen leider nicht gemacht werden. Dort einige Verallgemeinerungen für unsymmetrische Probleme

Referenzen zu unvollst. Dreieckszerl.: [18, 17, 15, 19, 20, 21, 22].

Referenzen zur Umordnung von Matrizen. [13, 3]

Referenzen zu SPAI: [14, 1]. Symm. Pos. def. Fall: [16]. Weitere Literatur: [12].

Referenzen zu weiteren schwach besetzten approximativen Inversen. “AINV”: [5, 6, 2, 4, 7, 8]. “APINV”: [9, 10, 11].

Referenzen zu Mehrgitter und AMG siehe Kursmaterial.

Literatur

[1] S. T. Barnard, L. M. Bernardo, and H. D. Simon. An mpi implementation of the SPAI preconditioner on the t3e. Preprint LBNL–40794 UC–405, Lawrence Berkeley National Laboratory, Computer Science Directorate, 1997.

[2] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. Technical report LA–UR–99–2899, Los Alamos National Laboratory, Scientific Computing Group, 1999.

[3] M. Benzi, J. C. Haws, and M. Tůma. Preconditioning highly indefinite and nonsymmetric matrices. Technical Report LA–UR–99–4857, Los Alamos National Laboratory, Scientific Computing Group (CIC–19), 1999.

[4] M. Benzi, J. Marin, and M. Tůma. A two–level parallel preconditioner based on sparse approximate inverses. In D. R. Kincaid and A. C. Elster, editors, *Iterative Methods in Scientific Computation IV*, volume 5 of *IMACS Series in Computational and Applied Mathematics*, pages 167–178, 1999.

[5] M. Benzi and C. D. Meyer. A direct projection method for sparse linear systems. *SIAM J. Sci. Comput.*, 16(5):1159–1176, 1995.

[6] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17:1135–1149, 1996.

[7] M. Bollhöfer and Y. Saad. *ILU*s and factorized approximate inverses are strongly related. Part I: Overview of results. Technical Report umsi–2000-39, Minnesota Supercomputer Institue, University of Minnesota, 2000. Submitted to SIAM Matrix. Anal. App.

[8] M. Bollhöfer and Y. Saad. *ILU*s and factorized approximate inverses are strongly related. Part II: Applications to stabilization. Technical Report umsi–2000-70, University of Minnesota at Minneapolis, Dep. of Computer Science and Engineering, 2000. Submitted to SIAM Matrix. Anal. App.

[9] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. Research Report UMSI 94/101, University of Minnesota, Super Computing Institute, Minneapolis, Minnesota, 1994.

[10] E. Chow and Y. Saad. Approximate inverse techniques for block–partitioned matrices. *SIAM J. Sci. Comput.*, 18(6):1657–1675, 1997.

[11] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse–sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023, 1998.

[12] J. Cosgrove, J. Diaz, and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *Intl. J. Comp. Math.*, 44:91–110, 1992.

[13] I. S. Duff and J. Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. *SIAM J. Matrix Anal. Appl.*, 20:889–901, 1999.

[14] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18(3), 1997.

[15] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[16] Y. Kolotilina and Y. Yeremin. Factorized sparse approximate inverse preconditionings I. theory. *SIAM J. Matrix Anal. Appl.*, 14:45–58, 1993.

[17] T. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34:473–490, 1980.

[18] J. Meijerink and H. A. V. der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m–matrix. *Math. Comp.*, 31:148–162, 1977.

[19] N. Munksgaard. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradient method. *ACM Trans. Math. Software*, 6:206–219, 1980.

[20] Y. Saad. ILUT: a dual treshold incomplete ILU factorization. *Numer. Lin. Alg. w. Appl.*, 1:387–402, 1994.

[21] Y. Saad. SPARSKIT and sparse examples. NA Digest, 1994.

[22] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.