Geometrische Mehrgitterverfahren
und
Algebraische Mehrgitterverfahren

—

Eine Einführung

Kompaktkurs

Iterative Gleichungssystemlöser
und parallele Algorithmen

19.–23. Februar 2001

Matthias Bollhöfer

Sekretariat MA 4–5
Fachbereich Mathematik
Technische Universität Berlin
Straße des 17. Juni 136
D–10623 Berlin

bolle@math.tu-berlin.de
http://www.math.tu-berlin.de/~bolle

Inhaltsverzeichnis

1 Die Grundidee der Mehrgitterverfahren **5**

1.1 Modellprobleme 5

1.2 Direkte Verfahren 8

1.3 Warum keine einfachen Iterationsverfahren verwenden? 9

1.4 Glättungsanalyse 12

1.5 Die Grobgitterkorrektur 15

1.6 Mehrgitterverfahren 19

1.6.1 Mehrgitterverfahren im eindimensionalen Fall 19

1.6.2 Mehrgitterverfahren im 2D–Fall 22

1.7 Eine elementare Konvergenzanalyse 26

1.7.1 Konvergenzanalyse im 1D–Fall 27

1.7.2 Übertragung auf den 2D–Fall 31

1.8 Mehrgitterartige Verfahren für Variationsprobleme 32

1.8.1 Variationsformulierung 32

1.8.2 Finite Elemente 34

1.8.3 Mehrgitterverfahren für FEM–Diskretisierung 35

1.8.4 Weitere mehrgitterartige Verfahren 38

2 Algebraische Mehrgitterverfahren **41**

2.1 Was sind und wozu braucht man algebraische Techniken 41

2.2 Konzepte für algebraische Mehrgitterverfahren 42

2.2.1 Mehrgitterverfahren mit algebraischer Restriktion/Prolongation 42

2.2.2 AMG basierend auf Blockeliminationstechniken 43

2.3 Das AMG von Ruge und Stüben 45

2.3.1 Die Grundidee des Ansatzes 45

2.3.2 Der Algorithmus an sich, 2 Durchläufe 46

2.3.3 Die Konstruktion der Interpolation in Matrixschreibweise 55

2.3.4 Ergänzungen bei Diagonaldominanz 57

2.3.5 Erhaltung der Vorzeicheneigenschaft 57

2.3.6 Weitere Ruge–Stüben–artige AMGs 61

2.4 AMGs basierend auf Aggregation 61

2.5 Unvollst. Block–Eliminationstechniken als AMG 68

Kapitel 1

Die Grundidee der Mehrgitterverfahren

In diesem Kapitel untersuchen wir zunächst die gewöhnlichen Mehrgitterverfahren für zwei sehr einfache Modellprobleme. Die Techniken und Erkenntnisse hier lassen sich natürlich in allgemeinerem Rahmen durchführen. Insbesondere kann man die Techniken auch als Motivation für die Konstruktion algebraischer Mehrgitterverfahren verstehen. Als Literatur kann [15, 16, 18] sowie die dort enthaltenen Referenzen empfohlen werden.

1.1 Modellprobleme

Wir betrachten zwei einfache Modellprobleme von Differentialgleichungen. Zunächst einmal ein einfaches eindimensionales Problem.

Gesucht ist eine Funktion $u : [0,1] \to \mathbb{R}$ die der folgenden Gleichung genügt.

$$-u''(x) = f(x), \text{ für alle } x \in [0,1] \tag{1.1}$$

sowie

$$u(0) = g_0,\ u(1) = g_1. \tag{1.2}$$

Dabei ist f eine gegebene Funktion auf $[0,1]$. Ebenfalls gegeben sind $g_0, g_1 \in \mathbb{R}$.

Das analoge Problem in zwei Raumdimensionen lautet wie folgt. Wir bezeichnen mit Ω das Gebiet $\Omega = [0,1] \times [0,1]$.

Gesucht ist eine Funktion $u : \Omega \to \mathbb{R}$ die der folgenden Gleichung genügt.

$$-\Delta u(x,y) \equiv -u_{xx}(x,y) - u_{yy}(x,y) = f(x,y), \text{ für alle } (x,y) \in \Omega \tag{1.3}$$

sowie

$$u(x,y) = g(x,y), \text{ für alle } (x,y) \in \partial\Omega \tag{1.4}$$

Dabei ist f eine gegebene Funktion auf Ω und g eine gegebene Funktion auf $\partial\Omega$.

Im Prinzip lässt sich vieles was hier gemacht wird, problemlos übertragen auf allgemeinere Probleme in Gebieten Ω des $\mathbb{R}^n$ der Form

$$-\ \operatorname{div}\ (A \text{ grad } u) = f \text{ in } \Omega, \tag{1.5}$$

$$u = g \text{ auf } \Gamma_1,\ \nu^\top A \text{ grad } u = h,\ \text{ auf } \Gamma_2. \tag{1.6}$$

Dabei ist $A : \Omega \to \mathbb{R}^{n\times n}$ eine für alle Werte aus Ω symmetrisch positiv definite Matrix derart, dass der kleinste Eigenwert von A nach unten durch eine positive Konstante beschränkt ist. $\Gamma_1 \cup \Gamma_2 = \partial\Omega$ und f, g, h sind gegebene Funktionen. ν bezeichnet die äussere Normale.

Wir wollen uns aber im folgenden auf einfache Probleme der Form (1.1) und (1.2) beschränken, um die Darstellung einfach zu halten und die grundlegenden Ideen besser darstellen zu können.

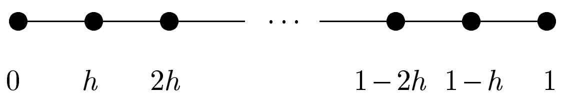
Zur numerischen Behandlung dieser beiden Problemfälle kannn man z.B. das Gebiet (also $[0,1]$ bzw. $[0,1]^2$) durch ein diskretes Gitter ersetzen und die Näherung nur an diesen Gitterpunkten verwenden.

Im Falle des 1D–Problems (1.1) ersetzen wir etwa

$$[0,1] \to \Omega_h = \{\frac{k}{N+1} :\ k = 0,\ldots,N+1\}. \tag{1.7}$$

Dabei ist N eine natürliche Zahl, ein sogenannter Diskretisierungsparameter und $h = \frac{1}{N+1}$ der Abstand zwischen zwei Gitterpunkten.

Geometrisch kann man diese Einteilung so interpretieren:



Mit den gleichen Bezeichnungen ersetzt man im Falle des 2D–Problems (1.3)

$$\Omega = [0,1]^2 \to \Omega_h = \{(kh, lh) :\ k, l = 0,\ldots,N+1\}. \tag{1.8}$$

Mit Hilfe dieser diskreten Menge Ω_h ersetzen wir jetzt die Differentialgleichung durch eine Differenzengleichung. Mit Hilfe der Taylorentwicklung bekommen wir, dass

$$-\ y''(x) = \frac{1}{h^2}\left(-y(x-h) + 2y(x) - y(x+h)\right) + \mathcal{O}(h^2). \tag{1.9}$$

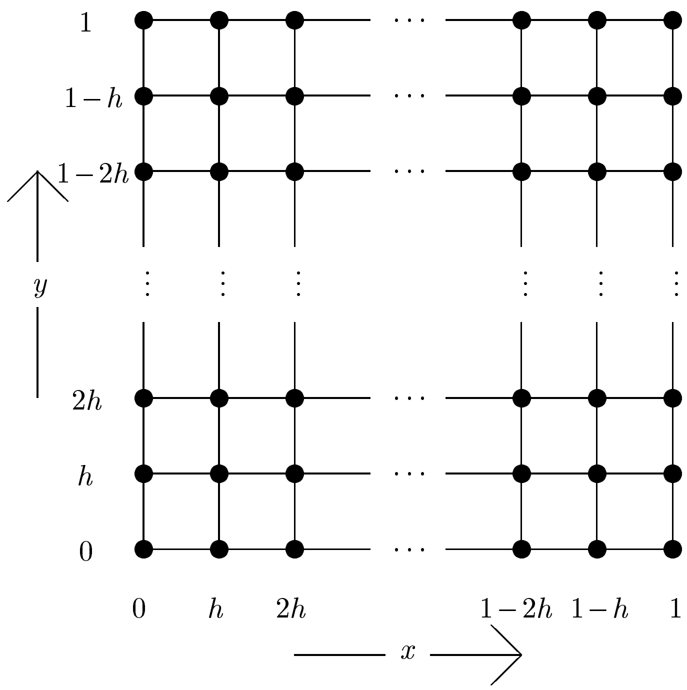
Man stellt das üblicher Weise in Form eines sogenannten Differenzensterns dar. Hier in der Form

$$-1 \;\text{———}\; 2 \;\text{———}\; -1$$

Den Faktor $\frac{1}{h^2}$ lässt man hierbei üblicher Weise weg. Man kann ihn später beim linearen Gleichungssystem auch in der rechten Seite unterbringen. Der Stern lebt dann genau auf dem diskreten Gitter, d.h. man kann immer in drei benachbarten Punkten auf dem Gitter diese Relation verwenden. Am Rande kann man stattdessen die vorgegebenen Randwerte einsetzen. Zusammen hat das diskrete System dann die Form

$$-\ u((i-1)h) + 2u(ih) - u((i+1)h) = h^2 f(ih), i = 1,\ldots,N, \tag{1.10}$$

Abbildung 1.1: **Diskretisierung des Gebietes Ω**



wobei $u(0) = g_0, u(1) = g_1$ vorgegeben sind. Setzen wir

$$u_i = u(ih),\ i = 1, \ldots, N,\ f_i = f(ih),\ i = 2, \ldots, N-1,\ f_1 = f(h) + \frac{g_0}{h^2},\ f_N = f(1-h) + \frac{g_1}{h^2}$$

und

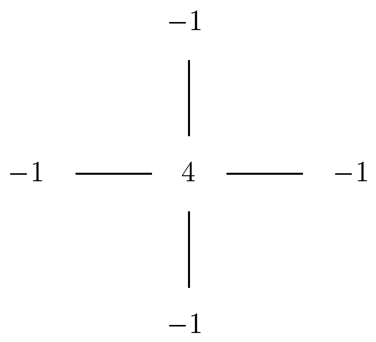
$$T_h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}. \tag{1.11}$$

Dann haben wir ein Gleichungssystem der Form

$$T_h u = f. \tag{1.12}$$

Dabei sind u, f die Vektoren, die u_i, f_i als Komponenten haben.

Die gleiche Argumentation im zweidimensionalen Fall ergibt den sogenannten "Fünf–Punkte–Stern", der der Diskretisierung des Laplace–Operators $-\Delta u(x, y)$ entspricht.



Hier bekommen wir in völliger Analogie zu dem eindimensionalen Fall eine diskrete Gleichung die auf dem zweidimensionalen Gitter Ω_h lebt d.h. setzen wir $u_{ij} = u(ih, jh)$ dann haben wir ein diskretes Problem der Form

$$-u_{i-1,j} - u_{i,j-1} + 4u_{ij} - u_{i+1,j} - u_{i,j+1} = h^2 f(ih, jh) \tag{1.13}$$

für $i, j = 1, \ldots, N$ mit entsprechenden Randbedingungen.

Setzt man wieder die Vorgaben auf dem Rand ein, so bekommt man ein System der Form

$$A_h u = f \tag{1.14}$$

wobei A_h eine $N^2 \times N^2$ Matrix ist. x entspricht der der diskreten Funktion auf den inneren Gitterpunkten ohne Rand, und f der Funktion f unter Berücksichtigung der Randpunkte.

Man kann im Falle von A_h zeigen, dass

$$A_h = \frac{1}{h^2} \begin{pmatrix} h^2 T_h + 2I_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & \ddots & -I_N \\ & & -I_N & h^2 T_h + 2I_N \end{pmatrix}. \tag{1.15}$$

Vorausgesetzt ist, dass man die Gitterpunkte z.B. von unten nach oben zeilenweise einen nach dem anderem nimmt.

Selbstverständlich ist dieser Ansatz hier stark vereinfacht. Aber natürlich lässt sich das alles viel allgemeiner formulieren. Nun stellt sich die Frage, wie man dieses Gleichungssystem lösen soll. Bei dem 1D–Problem stellt sich eigentlich nicht so sehr die Frage, denn die Matrix ist tridiagonal, symmetrisch und positiv definit und natürlich wäre die Cholesky–Zerlegung eine gute und schnelle Möglichkeit (Kosten ungefähr $9N$ Operationen). Im 2D–Fall klappt dieser Trick aber nicht mehr so. Sehen wir uns einmal für zunehmendes N an, wie lange eine Cholesky–Zerlegung, z.B. in MATLAB dauern würde.

Bei realistischen Problemen kann ohne weiteres $N^2 \approx 10^6$ und größer vorkommen.

Diese Möglichkeit müssen wir bei numerischen Tests im Auge behalten.

1.2 Direkte Verfahren

Die hier zugrunde liegende Matrix ist symmetrisch positiv definit. Der hierfür angepasste direkte Löser wird als Cholesky–Zerlegung bezeichnet. Grundlage hierfür ist die folgende Zerlegung der Ausgangsmatrix A.

$$A = \begin{pmatrix} \alpha & v^\top \\ v & B \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ v & I \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} \alpha & v^\top \\ 0 & I \end{pmatrix}, \tag{1.16}$$

wobei $S = B - \frac{vv^\top}{\alpha}$ das Schur–Komplement (oder auch die Restmatrix) bezeichnet. Von diesem weiss man, dass es wieder positiv definit ist. Wendet man sukzessive (1.16) auf S anstelle von A an so erhält man schließlich eine Zerlegung der Form

$$A = (D - E)D^{-1}(D - E)^\top. \tag{1.17}$$

Dabei entstehen D, E während der Zerlegung. Man beachte, dass im allgemeinen das Schur–Komplement sich während der Elimination immer weiter auffüllt.

Natürlich kann man im eindimensionalen Fall des Modellproblems einfach die Cholesky–Zerlegung verwenden und das Problem dadurch in sehr kurzer Zeit lösen. Der eindimensionale Fall dient hier auch mehr der Illustration statt der ernsthaften Behandlung.
Im zweidimensionalen Fall ist der Aufwand schon erheblich höher und selbst bei Verwendung vielfältiger Techniken zur Minimierung des Rechenaufwandes lässt sich eigentlich im günstigsten Fall nicht viel mehr als $\mathcal{O}(N^3)$ erreichen. D.h. man hat man hat einen Rechenaufwand der deutlich mehr als linear mit der Zahl der Unbekannten steigt. Hinzu kommt der enorme Speicheraufwand der in der gleichen Größenordnung liegt. Im dreidimensional Fall verschlechtert sich diese Beziehung noch mehr. Wir betrachten das mal für den zweidimensionalen Fall (1.3) mit Hilfe von MATLAB. Dort verwenden wir zunächst einen (heuristischen aber recht effizienten) Algorithmus um die Unbekannten geschickt durchzunummerieren. Der Algorithmus wird mit "symmetric minimum degree" bezeichnet. Anschließend verwenden wir eine Cholesky–Zerlegung basierend auf dieser Anordnung der Unbekannten. Wir betrachten Rechenaufwand und Speicherbedarf in Abhängigkeit von N. Die Ergebnisse sind in Tabelle 1.1 zusammengefasst.

Tabelle 1.1: **Direkte Verfahren für die 2–dim. Laplace–Gl.**

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

Wir sehen das prognostizierte Verhalten, wenn auch erkennbar ist, dass sich der Algorithmus noch recht gut verhält.

1.3 Warum keine einfachen Iterationsverfahren verwenden?

Wir haben bereits im vorherigen Abschnitt gesehen, dass direkte Verfahren aus Gründen der Rechenzeit aber auch vor allem wegen ihres Speicherbedarfs ab einer gewissen Problemgröße nicht mehr in Frage kommen.
Wir untersuchen im folgenden einige einfache Standarditerationsverfahren zur Lösung der Probleme (1.1), (1.3) bzw. ihrer diskretisierten Varianten (1.10), (1.13). Wir werden dabei sehen, dass diese erhebliche Zeit, d.h. sehr viele Iterationsschritte benötigen und daher so nicht verwendet werden können.

Wir beginnen mit sehr einfachen Iterationsverfahren der Form

$$x^{(k+1)} = x^{(k)} + B^{-1}(b - Ax^{(k)}), k = 0, 1, 2, \dots \tag{1.18}$$

mit Startvektor $x^{(0)}$. Hier verwenden wir als Startwert die 0. B soll hier eine nichtsinguläre Approximation an A darstellen. Die Theorie solcher Verfahren (siehe z.B. [17, 10]) besagt, dass die Folge der $x^{(k)}$ gegen die Lösung von $Ax = b$ konvergiert, sofern der Spektralradius $\rho(T) = \max\{|\lambda| : \lambda \text{ Eigenwert von } T\}$ kleiner als 1 ist für $T = I - B^{-1}A$.

Als Approximationen B an die Ausgangsmatrix betrachten wir mehrere Varianten. Wir nehmen an, es sei $A = D - E - E^\top$ eine Zerlegung der Ausgangsmatrix in den Diagonalanteil D, den negativen strikten unteren Dreiecksanteil E und entsprechend $E^\top$ den negativen oberen Dreiecksanteil aus Symmetriegründen. Als erstes nehmen wir den Diagonalanteil $B = D$ von A. Das zugehörige Verfahren ist auch als Gesamtschrittverfahren oder Jacobi–Verfahren bekannt. Als zweites betrachten wir den unteren Dreiecksanteil von $B = D - E$ von A. Das zugehörige Iterationsverfahren wird als Einzelschrittverfahren oder Gauß–Seidel–Verfahren bezeichnet.

Es soll in diesem Zusammenhang so verstanden werden, dass man B^{-1} natürlich nicht explizit bildet sondern stattdessen immer ein Gleichungssystem mit B löst. Wir bekommen in Abhängigkeit von N folgende Ergebnisse, die in Tabelle 1.2 zusammengefasst sind. Dabei haben wir $x^{(0)} = 0$ gewählt und als Abbruchkriterium $\|b - Ax^{(k)}\|_\infty \leqslant 10^{-8}\|b\|_\infty$ gewählt.

Tabelle 1.2: **Standarditerationsverfahren für die 2–dim. Laplace–Gl.**

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

Das Ergebnis ist ähnlich niederschmetternd wie zuvor das Experiment mit dem direkten Löser. Zwar benötigen wir erheblich weniger Speicher, die Zahl der Iterationen ist aber derart hoch, dass diese Techniken nicht weiter in Frage kommen.

Nun können wir natürlich anstelle von diesen einfachen Iterationsfverfahren auch polynomielle Beschleunigungstechniken verwenden, speziell hier ist natürlich das cg–Verfahren die naheliegende Variante, da die Ausgangsmatrix symmetrisch positiv definit ist. Als Vorkonditionierung B an A benötigen wir hier aber eine symmetrisch positiv definite Matrix. Wir können einmal wieder $B = D$ als Diagonalanteil verwenden und zum anderen ersetzen wir das Gauß–Seidel Verfahren durch seine symmetrisierte Variante $B = (D - E^\top)D^{-1}(D - E)$. Tabelle 1.3 zeigt, dass das Ergebnis zwar schon besser ist, aber es ist weit entfernt von effizient.

Tabelle 1.3: **cg–Verfahren für die 2–dim. Laplace–Gl.**

		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

Allem Anschein nach sind diese Approximationen an die Ausgangsmatrix zu grob.

Natürlich können wir versuchen andere allgemeine Techniken zu verwenden. Die symmetrisierte Variante des Gauß–Seidel Verfahrens $(D - E)D^{-1}(D - E)^\top$ kann man als angenäherte Cholesky–Zerlegung von A interpretieren, wenn auch der Fehler sicherlich sehr groß ist. Ein Beispiel für eine etwas aufwendigere Approximation wäre die unvollständige Cholesky–Zerlegung, bei der man nur dort Einträge speichert, wo bereits in der Ausgangsmatrix Nichtnullelemente vorhanden sind. Wir erinnern uns daran, dass folgende Gleichung Grundlage für die Cholesky–Zerlegung ist.

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

wobei $S = B - \frac{vv^\top}{\alpha}$. Wir ersetzen einfach S durch eine Näherung $\tilde{S}$, die dadurch entsteht, dass nur die Einträge b_{ij} von B durch $b_{ij} - \frac{v_i v_j}{\alpha}$ ersetzt werden, die vorher schon von Null verschieden waren. Wie beim symmetrischen Gauß–Seidel–Verfahren bekommen wir eine approximative Zerlegung der Matrix A in der Form

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

Die so entstehende approximative Cholesky–Zerlegung wird auch als ICHOL(0) bezeichnet (ILU(0) im unsymmetrischen Fall). Die Kosten dieser Zerlegung liegen in derselben Größenordnung wie die Anzahl der Nichtnullelemente der Matrix. Ein vertretbarer und geringer Rechenaufwand also. Es bleibt die Frage wie sich das cg–Verfahren mit der unvollständigen Cholesky–Zerlegung als Vorkonditionerer in der Praxis verhält. Dieses zeigt uns Tabelle 1.4

Tabelle 1.4: **ICHOL(0) cg–Verfahren für die 2–dim. Laplace–Gl.**

		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

Die numerischen Ergebnisse zeigen schon eine erhebliche Steigerung gegenüber dem cg–Verfahren mit Jacobi–Vorkonditionierung und auch gegenüber dem symmetrisierten Gauß–Seidel als Vorkonditionerer. Nichtsdestotrotz auch hier knickt das Verfahren für große N nach und nach ein.

Feststellung:

- Direkte Löser kommen wegen des hohen Speicherbedarfs aber auch wegen der Rechenzeit nicht in Betracht.
- Einfache Iterationsverfahren (Jacobi, Gauß–Seidel) bieten zwar eine Alternative bzgl. Speicherplatz, sind aber zu langsam.
- Aufwendigere Techniken wie die unvollständige Cholesky–Zerlegung funktionieren erheblich besser und sind auch nicht sonderlich speicherplatzintensiv, aber für große N auch nicht mehr befriedigend.

1.4 Glättungsanalyse

Wir nehmen die obigen Beobachtungen zum Anlass, zu untersuchen warum sich die obigen Verfahren so schlecht verhalten. Um die Problematik einfach zu halten, betrachten wir zunächst den eindimensionalen Fall, bei dem Jacobi–Verfahren und Gauß–Seidel–Verfahren sogar noch schlechter abschneiden. Der Fehler $e^{(k)} = x - x^{(k)}$ der Gleichung $Ax = b$ lässt sich schreiben als

$$e^{(k+1)} = x - x^{(k+1)} = x - x^{(k)} - B^{-1}(Ax - Ax^{(k)}) = (I - B^{-1}A)e^{(k)}.$$

D.h. die Fehlerkomponenten werden mit der Matrix $(I - B^{-1}A)$ gedämpft. Im Falle der Modellprobleme sowie der Jacobi–Glättung ist diese Matrix recht einfach zu beschreiben. Denn dann ist B nur ein Vielfaches der Identität und die Eigenvektoren sind die von A.

Im Falle des eindimensionalen Problems (1.1) haben wir für ein beliebiges $k \in \mathbb{N} \setminus \{0\}$

$$-(\sin(k\pi x))'' = k^2\pi^2 \cdot \sin(k\pi x), \text{ für alle } x \in [0,1]$$

und $\sin(k\pi \cdot 0) = \sin(k\pi \cdot 1) = 0$. D.h. die Funktionen $\sin(k\pi x)$ sind Eigenfunktionen der homogenen Differentialgleichung

$$-u''(x) = f(x) \text{ in } [0,1], u(0) = u(1) = 0.$$

Es liegt nahe im Falle der diskretisierten Gleichung

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i$$

die gleichen Eigenfunktionen, natuürlich auf das Gitter beschränkt zu verwenden. Wir setzen $s_i^{(k)} = \sin k\pi ih$ Dann erhalten wir mit Hilfe des Additionstheorems $\sin(\phi \pm \psi) = \sin\phi\cos\psi \pm \sin\psi\cos\phi$ die Gleichung

$$-s_{i-1}^{(k)} + 2s_i^{(k)} - s_{i+1}^{(k)} \quad = \quad 2(1 - \cos(k\pi h))s_i^{(k)}$$

Für $i = 1, N$ treffen die Nullstellen von $s^{(k)}$ zusammen mit den Randwerten (hier homogen). Also sind die Vektoren

$$s^{(k)} = \left(s_i^{(k)}\right)_{i=1,\ldots,N} = \left(\sin(k\pi ih)\right)_{i=1,\ldots,N} \tag{1.19}$$

die wir als diskrete Punkte der Eigenfunktionen bekommen haben genau die Eigenvektoren der diskreten Matrix T_h aus (1.11). Die zughehörigen Eigenwerte sind

$$h^2\lambda_k = 2 - 2\cos(k\pi h) = 2(1 - \cos^2\frac{k\pi h}{2} + \sin^2\frac{k\pi h}{2}) = 4\sin^2\frac{k\pi h}{2}, \tag{1.20}$$

und zwar können wir das für $k = 1, \ldots, N$ machen. Ab $k = N+1, N+2, \ldots$ bekommen wir wieder dieselben Werte wie bei $k = 0, 1, \ldots,$ weil wir ja nur diskrete Punkte betrachten. Allerdings kehren sich Reihenfolge und Vorzeichen um.

Die Eigenvektoren sind für $N = 3$ (grüne Sterne) und $N = 7$ (rote Kästchen) in Abbildung 1.2 abgetragen.

Bei den Eigenvektoren fällt insbesondere auf, dass mit zunehmenden N höhere und höhere Frequenzen mit hineinkommen. Bei $N = 3, 7$ haben wir in Abbildung 1.2 einmal auf der linken

Abbildung 1.2: **Eigenvektoren für** $N = 3, 7$

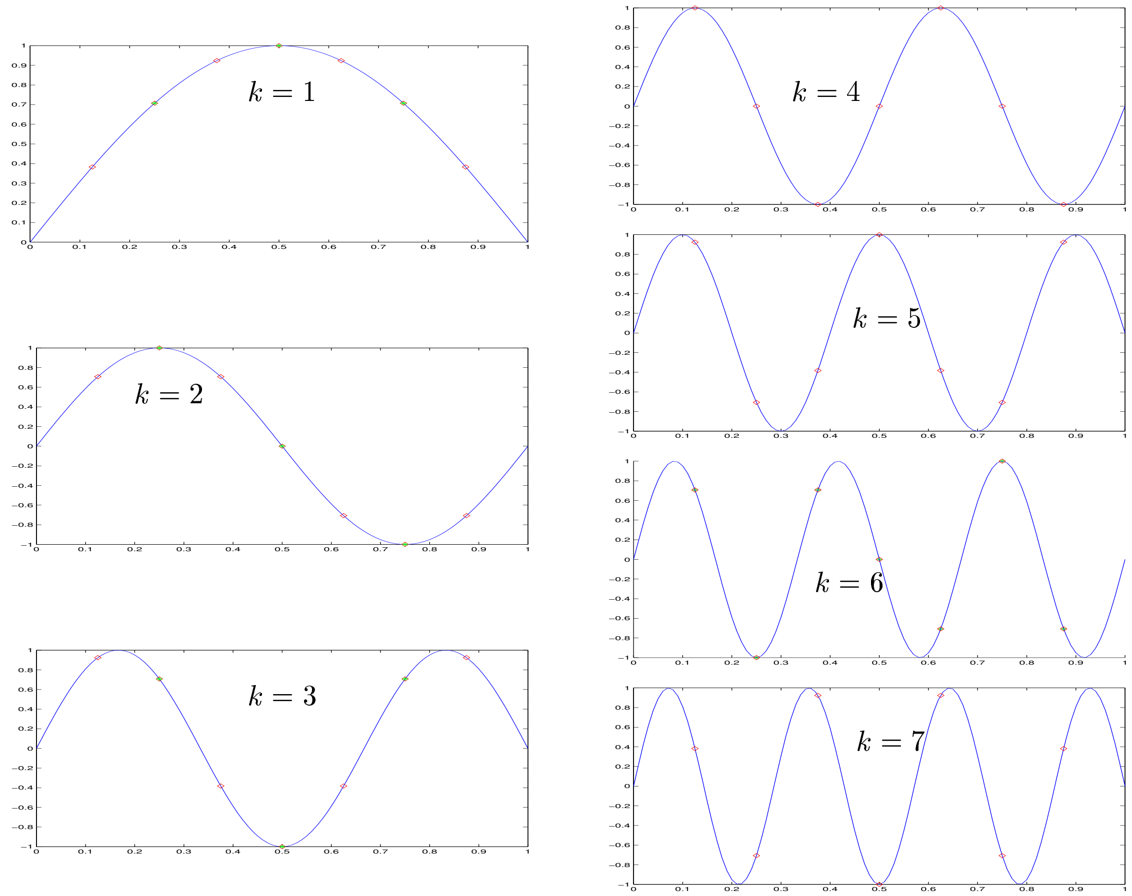
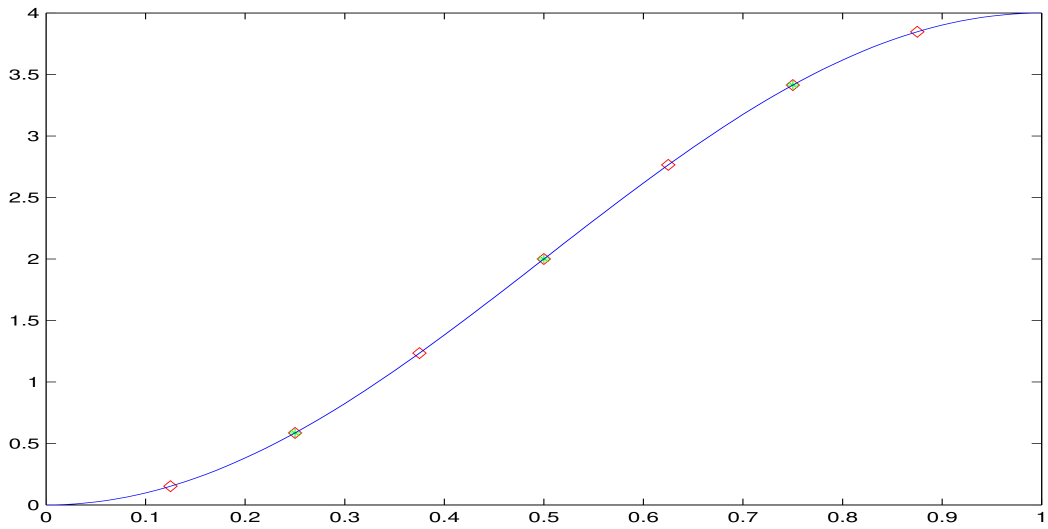


Abbildung 1.3: **Eigenwerte für** $N = 3, 7$



Seite die niedrigen Frequenzen mit Eigenvektoren für beide N, auf der rechten Seite die für $N = 7$ zusätzlichen Frequenzen.

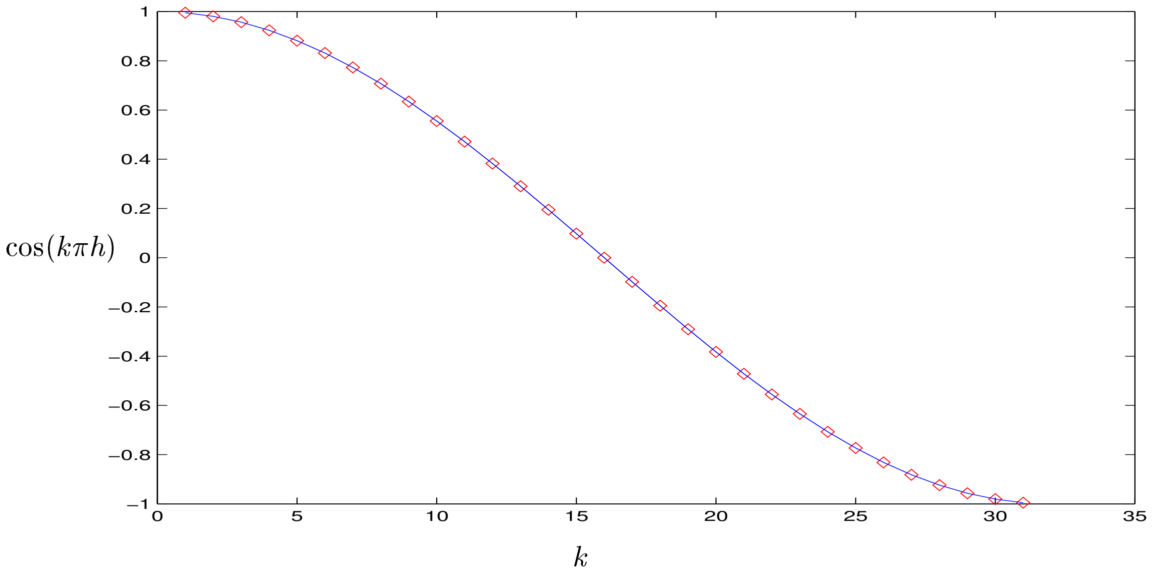
Die Eigenwerte sind bis auf den Faktor $\frac{1}{h^2}$ in Abbildung 1.3 abgetragen. Dabei ist zu beachten, dass $\frac{1}{h^2}$ unterschiedlich für $N = 3$ und $N = 7$ wäre.

Nachdem wir Eigenvektoren und Eigenwerte betrachtet haben, sehen wir uns an, wie die Fehler in Richtung der Eigenvektoren durch das Jacobi–Verfahren gedämpft werden. Ist also ein gegebener Fehler $e = \sum_{k=1}^{N} \alpha_k s^{(k)}$ bezüglich der Basis $s^{(1)}, \ldots, s^{(N)}$ dargestellt, so ist

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

Die Terme $\left|1 - 2\sin^2 \frac{k\pi h}{2}\right|$, $k = 1, \ldots, N$ treten gerade als Dämpfungsfaktoren auf. Nur für k um $N/2$ herum sind diese Werte richtig klein. Wir sehen das in Abbildung 1.4.

Abbildung 1.4: **Gewichte** $1 - 2\sin^2 \frac{k\pi h}{2}$ **für** $N = 31$**, d.h.** $h = \frac{1}{31}$



Also werden Anteile der Lösung in Richtung gewisser Frequenzen $s^{(k)}$ nur unzureichend gedämpft. Wir können versuchen dieses Dämpfungsverhalten durch Einbringen eines Dämpfungsparameters ω zu beeinflussen.

D.h. wir setzen

$$x^{(k+1)} = x^{(k)} + \omega B^{-1}(b - Ax^{(k)})$$

was im Falle des Jacobi–Verfahrens folgende Fehlerdämpfung liefern würde.

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

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

Wir können nun versuchen ω zu optimieren, d.h. ω so zu wählen, dass

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

ist. Es stellt sich allerdings heraus, dass dann bereits $\omega = 1$ ist. Wir können mit ω also das globale Verhalten nicht verbessern. Was wir aber machen können, ist ω so zu wählen, dass für möglichst viele Frequenzen $s^{(k)}$ der Wert $|1 - 2\omega \sin^2 \frac{k\pi h}{2}|$ gleichmäßig von 1 weg beschränkt ist. Das erreichen wir dadurch, dass wir $\omega = \frac{2}{3}$ wählen. In dem Fall ist nämlich für $k > N/2$

$$\frac{4}{3} \geqslant \frac{4}{3} \sin^2 \frac{k\pi h}{2} \geqslant \frac{4}{3} \cdot \frac{1}{2} = \frac{2}{3}.$$

Und somit ist

$$\max_{k>N/2} |1 - 2\omega \sin^2 \frac{k\pi h}{2}| \leqslant \frac{1}{3}. \tag{1.21}$$

Wir erhalten somit ein Verfahren, dass für die hohen Eigenfrequenzen $s^{(k)}$, $k > N/2$ ein sehr gutes Dämpfungsverhalten hat, aber für die verbleibenden niedrigen Frequenzen keine Verbesserung bringt.

Wir illustrieren das Dämpfungsverhalten in der folgenden Abbildung 1.5 für einige ausgesuchte Frequenzen.

Wir sehen in Abbildung 1.5 insbesondere, dass nur hohe Frequenzen gedämpft werden, während die niedrigen Frequenzen nahezu unverändert bleiben.

Wir haben gesehen, dass wir durchaus in der Lage sind ein Verfahren zu konstruieren, welches auf einem großen Unterraum sehr gut arbeitet. Um aber ein Verfahren zu bekommen, welches das Problem vernünftig approximiert, müssen wir noch etwas tun.

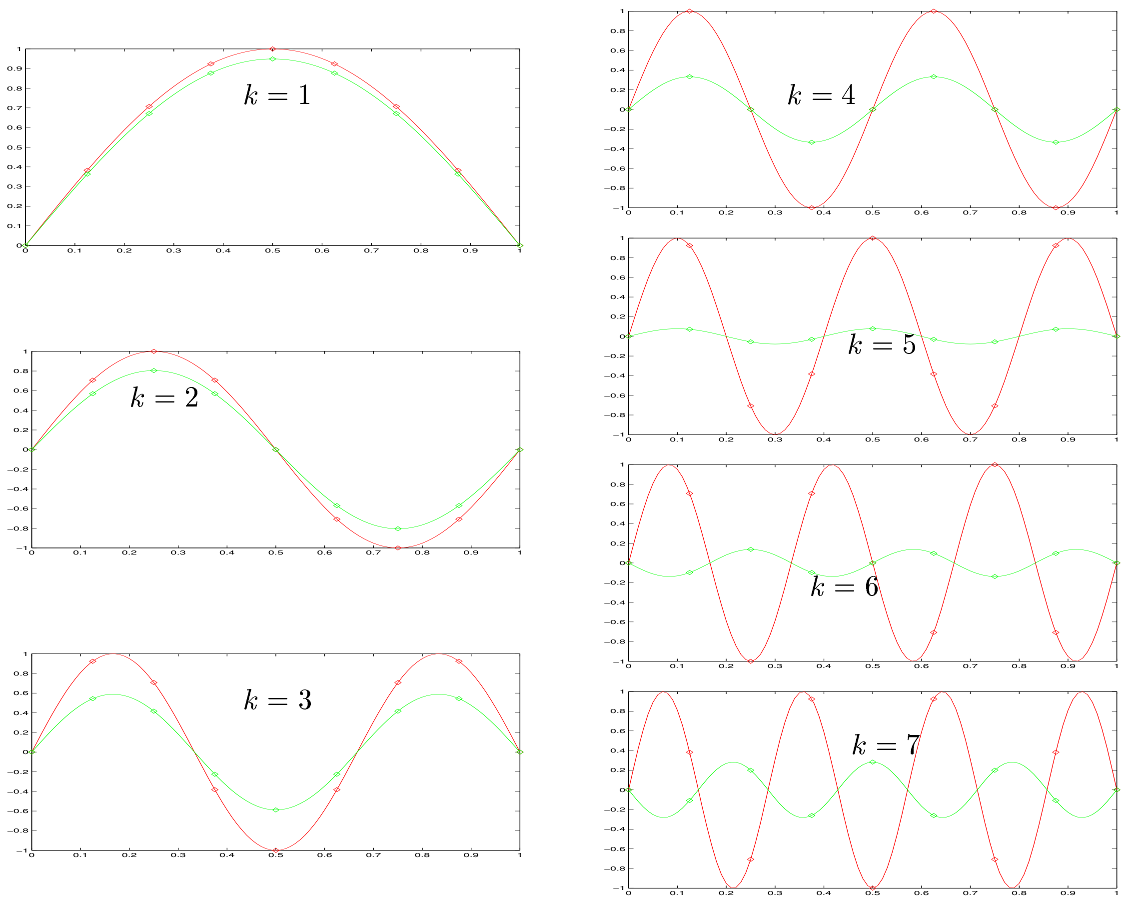
1.5 Die Grobgitterkorrektur

Wir motivieren im folgenden die sogenannte Grobgitterkorrektur.

Aus Abschnitt 1.4 haben wir entnommen, dass wir Verfahren, die auf einem sehr großem Spektrum zufriedenstellend arbeiten relativ leicht konstruieren können. Wir sprechen von einer sogenannten Glättung, in Anlehnung an den Effekt, dass hohe Frequenzen stark reduziert werden, während niedrige Frequenzen nahezu unverändert bleiben. Nun wissen wir (siehe z.B. Abbildung 1.2, dass mit zunehmendem N immer mehr hohe Frequenzen in unser Problem hineinkommen. Anders herum gesagt, bei kleinerem N haben wir vornehmlich niedrige Frequenzen. Wir können also erwarten, dass der verbleibende Fehler nach der Glättung auf einem gröberen Gitter approximiert werden kann. Im folgenden müssen wir lediglich die Übergänge zwischen feinerem Gitter un gröberen Gitter beschreiben.

Wir beginnen mit dem Schritt vom feineren Gitter zum gröberen Gitter. In diesem Falle sprechen wir von einer sogenannten Restriktion. Haben wir etwa zwei Gitter Ω_h und Ω_H mit $H = 2h$ gegeben, so könnten wir natürlich Ω_H wegen der gröberen Maschenweite als Teilmenge von Ω_h interpretieren und schlichtweg die triviale Restriktion auf diese Punkte verwenden. Abbildung 1.6 illustriert diese Art der Restriktion. Identifizieren wir jedes $x \in$

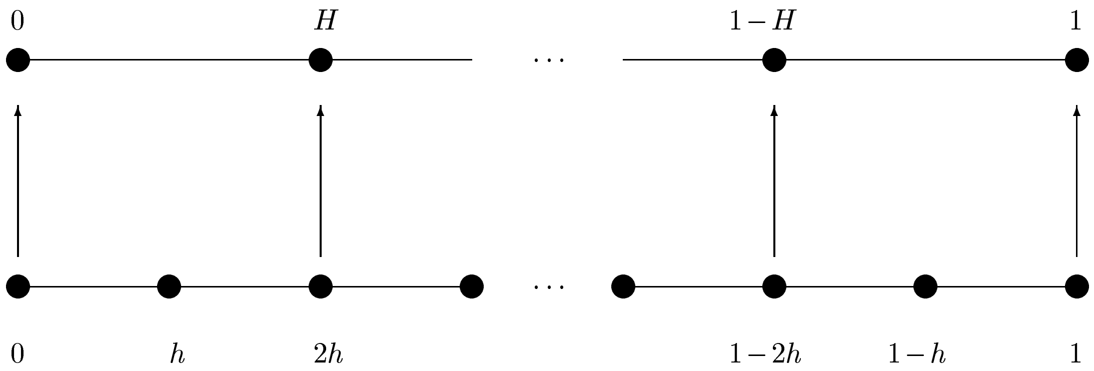
Abbildung 1.5: **Geglättete Eigenvektoren ($N = 7$), vorher (rot), nachher (grün)**



$\mathbb{R}^N, y \in \mathbb{R}^{\frac{N-1}{2}}$ mit der Stelle im geometrischen Gitter, so können wir statt von $x_1, \ldots, x_N$ nun von $x(h), \ldots, x((N-1)h)$ im Falle des feineren Gitters sprechen, bzw. von $y(H), \ldots, y(\frac{N-1}{2}H)$ anstelle von $y_1, \ldots, y_{\frac{N-1}{2}}$. Diese Notation veranschaulicht die Lage der Unbekannten und erlaubt eine einfache Beschreibung der Restriktion. D.h. $y \in \Omega_H$ ist definiert durch

$$y(iH) = x(iH) = x(2ih), \; i = 1, \ldots, \frac{N-1}{2}, x \in \Omega_h. \tag{1.22}$$

Abbildung 1.6: **Triviale Restriktion** $\Omega_h \to \Omega_H$



Dass die triviale Restriktion aus Abbildung 1.6 nicht sehr viel Sinn macht zeigt schon folgende Plausibilitätsbetrachtung. Ignorieren wir jeden zweiten Knoten, so gehen die durch diese Knoten verursachten Fehler gar nicht mit ein. Wir können diese Fehler nicht korrigieren und somit auch keine berauschende Konvergenz erwarten, wenn überhaupt.

Viel mehr Sinn macht bei der Restriktion die Mittellung über die Nachbarn, d.h. $y \in \Omega_H$ wird definiert durch

$$y(iH) = \frac{1}{4}x(iH-h) + \frac{1}{2}x(iH) + \frac{1}{4}x(iH+h), \; i = 1, \ldots, \frac{N-1}{2}, x \in \Omega_h. \tag{1.23}$$

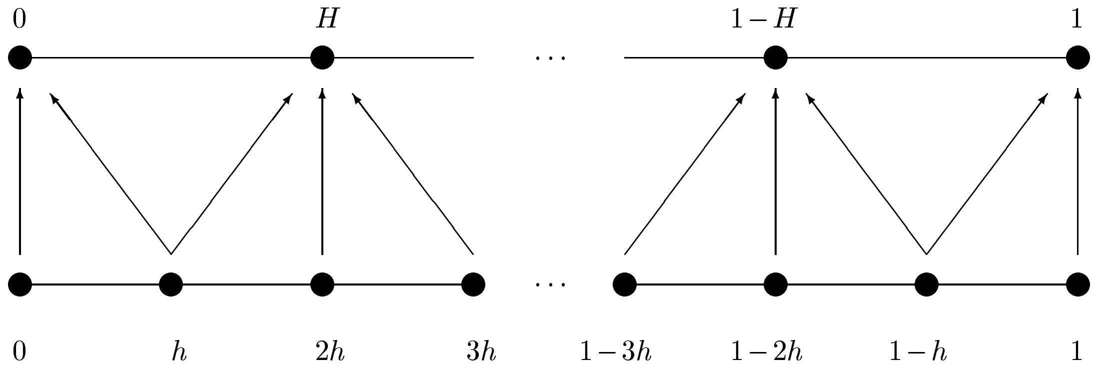
Dieses kann man wieder mit Hilfe eines Sterns verdeutlichen

$$\tfrac{1}{4} \quad \text{———} \quad \tfrac{1}{2} \quad \text{———} \quad \tfrac{1}{4}$$

Geometrisch illustriert Abbildung 1.7 diese Art der Restriktion.

Neben der Restriktion benötigen wir als nächstes die umgekehrte Richtung, nämlich die Einbettung des gröberen Gitters Ω_H in das feinere Gitter Ω_h. Diese Einbettung nennt man üblicher Weise Prolongation. Es ist naheliegend, die Punkte des groben Gitters natürlich in das feinere Gitter einzubetten. Die fehlenden Punkte im feineren Gitter kann man mittels linearer Interpolation aus den Nachbarn des groben Gitters bekommen. D.h. wir bekommen folgende Interpolationsvorschrift für einen gegebenen Wert $y \in \Omega_H$ um daraus $x \in \Omega_h$ zu konstruieren. Wir nutzen wieder die Schreibweise $x(ih), y(iH)$ als Gitterfunktionen um Beziehungen zwischen den Komponenten von x, y und den zugehörigen Gittern zu erleichtern.

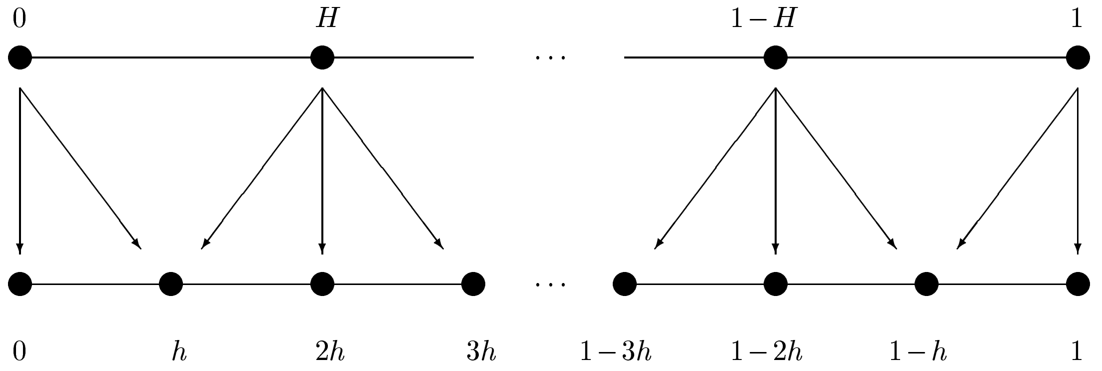
Abbildung 1.7: **Gemittelte Restriktion** $\Omega_h \to \Omega_H$



$$(1.24) \qquad x(ih) = \begin{cases} y(jH) & \text{falls } i = 2j \\ \frac{1}{2}y(jH) + \frac{1}{2}y(jH+H) & \text{falls } i = 2j+1 \end{cases}, \; i = 1, \ldots, N, \; y \in \Omega_H.$$

Anhand von Abbildung 1.8 erkennt man bereits, dass hier das Prinzip der Restriktion umgekehrt wird.

Abbildung 1.8: **Interpolation (Prolongation)** $\Omega_H \to \Omega_h$



In Matrixschreibweise lässt sich die Restriktion aus (1.23) darstellen mit Hilfe der Matrix

$$(1.25) \qquad R = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 2 & 1 \end{pmatrix}$$

Die Prologation P ist bis auf einen Faktor 2 die transponierte Matrix.

$$(1.26) \qquad P = 2R^\top.$$

Nach den Betrachtungen die wir in Abschnitt 1.4 angestellt haben wissen wir, dass das gedämpfte Jacobi–Verfahren eine glättende Wirkung hat. Speziell die hohen Frequenzen werden gleichmässig gedämpft. Was fehlt, ist eine Korrektur der niedrigen Frequenzen.

Deshalb kombinieren wir das gedämpfte Jacobi–Verfahren mit einer Korrektur auf dem gröberen Gitter. Im nächsten Korrekturschritt wählen wir als Approximation anstelle des Diagonalanteils von T_h die exakte Korrektur, allerdings auf dem gröberen Gitter Ω_H.

$$
\begin{aligned}
x^{(k+\frac{1}{2})} &= x^{(k)} + \tfrac{2}{3} \cdot \tfrac{h^2}{2}(b - T_h x^{(k)}) \\
x^{(k+1)} &= x^{(k+\frac{1}{2})} + P T_H^{-1} R (b - T_h x^{(k+\frac{1}{2})})
\end{aligned}
\tag{1.27}
$$

Das so entstandene Verfahren bezeichnet man als (exaktes) Zweigitter–Verfahren. Im Gegensatz zum späteren Mehrgitterverfahren betrachten wir zunächst nur diesen Fall. Die Verallgemeinerung auf mehrere Gitter lässt sich dann rekursiv weiterführen. Natürlich ist das Zweigitter–Verfahren noch keine echte Alternative, weil man das Problem der Dimension N auf eins der Größe $\frac{N-1}{2}$ reduziert hat. Für das Problem der Größe $\frac{N-1}{2}$ haben wir aber erst mal die exakte Lösung z.B. mittels Cholesky–Verfahren unterstellt. Nichtsdestotrotz können wir natürlich dieses einfache Zweigitter–Verfahren bezüglich Iterationsschritten, Glättungsverhalten mit dem gewöhnlichen Jacobi– oder Gauß–Seidel–Verfahren vergleichen. Tabelle 1.5 zeigt die gravierende Unterschiede. Der Volländigkeit halber müssen wir erähnen, dass einmalig der Aufwand für die Cholesky-Zerlegung mit hinzukommt. Dieser Aufwand ist vernachlässigbar, da das ganze Problem tridiagonal ist und sowieso nur zur Illustration dient. Im zweidimensionalen Fall wäre die Cholesky–Zerlegung bereits erheblich teurer.

Tabelle 1.5: **Zweigitterverf. für die 1–dim. Laplace–Gl.**

	Jacobi		Gauß–Seidel		Zweigitterverf.		
h	Zeit (flops)	Anzahl Iter.	Zeit (flops)	Anzahl Iter.	Zeit (flops)	Anzahl Iter.	Cholesky (flops)
$\frac{1}{31}$	$1.3{\cdot}10^6$	3784	$7.6{\cdot}10^5$	1893	$1.9{\cdot}10^4$	17	$5.7{\cdot}10^1$
$\frac{1}{63}$	$1.0{\cdot}10^7$	15154	$6.2{\cdot}10^6$	7578	$3.6{\cdot}10^4$	16	$1.2{\cdot}10^2$
$\frac{1}{127}$	$8.5{\cdot}10^7$	60632	$5.0{\cdot}10^7$	30317	$7.3{\cdot}10^4$	16	$2.5{\cdot}10^2$
$\frac{1}{255}$	$6.8{\cdot}10^8$	242555	$4.0{\cdot}10^8$	121272	$1.5{\cdot}10^5$	16	$5.1{\cdot}10^2$
$\frac{1}{511}$	—		—		$2.9{\cdot}10^5$	16	$1.0{\cdot}10^3$
$\frac{1}{1023}$	—		—		$5.9{\cdot}10^5$	16	$2.0{\cdot}10^3$
$\frac{1}{2047}$	—		—		$1.2{\cdot}10^6$	16	$4.1{\cdot}10^3$

1.6 Mehrgitterverfahren

Wir werden jetzt im folgenden das bisher motivierte Zweigitterverfahren zum Mehrgitterverfahren verallgemeinern. Dabei werden wir auch auf den zweidimensionalen Fall eingehen sowie den Einfluss der Zahl der Glättungs– bzw. Korrekturschritte auf die Konvergenz untersuchen. Außerdem untersuchen wir weitere Gätter wie z.B. das Gaß–Seidel–Verfahren.

1.6.1 Mehrgitterverfahren im eindimensionalen Fall

Bisher haben wir nur zwei Gitter betrachtet. Dies ist zwar vom Prinzip her ausreichend, weil man natürlich die Idee rekursiv weiterentwickeln kann. Jedoch muss dies auch in der Praxis durch einen Algorithmus umgesetzt werden. Wir motivieren dies anhand des Zweigitterverfahrens. Nach (1.27) bekommen wir durch Kombinations von Glättung und Grobgitterkorrektur eine neue Näherung $x^{(k+1)}$ durch folgende Berechnungsvorschrift.

$$\begin{aligned} x^{(k+\frac{1}{2})} &= x^{(k)} + \tfrac{2}{3} \cdot \tfrac{h^2}{2}(b - T_h x^{(k)}) \\ x^{(k+1)} &= x^{(k+\frac{1}{2})} + P T_H^{-1} R(b - T_h x^{(k+\frac{1}{2})}) \end{aligned}$$

Wir betrachten den Fall mehrerer hierarchischer Gitter $\Omega_{h_1} \subseteq \Omega_{h_2} \subseteq \cdots \subseteq \Omega_{h_l}$. Der Einfachheit halber seien sie durch Halbieren der Schrittweite gegeben, d.h. $h_{s+1} = h_s/2$, $s = 1, 2, \dots, l-1$, wobei $h_1 = \frac{1}{N_1+1}$. Zu jedem dieser Gitter haben wir die diskretisierte Gleichung in Form der Matrix T_{h_s}, $s = 1, 2, \dots, l$. Die Übergangsmatrizen (Restriktion und Prolongation) seien jetzt mit

$$R_s : \Omega_{h_s} \to \Omega_{h_{s-1}},\ P_s : \Omega_{h_{s-1}} \to \Omega_{h_s},\ s = 2, 3, \dots, l \tag{1.28}$$

bezeichnet. Eigentlich wollen wir nur das System

$$T_{h_l} x_{h_l} = b_{h_l}$$

auf dem feinsten Gitter lösen. Was wir jetzt machen ist im zweiten Schritt der kombinierten Iteration T_H^{-1} durch weitere rekursive Aufrufe zu ersetzen. D.h. statt ein Gleichungssystem der Form $T_H y = b_H$ mit $b_H = R(b - T_h x^{(k+\frac{1}{2})})$ exakt zu lösen, verwenden wir wieder einen Schritt des Zweigitteralgorithmus.

Algorithmus 1 (Mehrgitterschritt)

Gegeben: h_s *für ein* $s \in \{1, 2, \dots, l\}$, *eine zugehörige rechte Seite* b_{h_s}, *eine Näherungslösung* x_{h_s}.

Der folgende Algorithmus berechnet rekursiv eine neue Näherung y_{h_s}.

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

Sonst $y_{h_s} := x_{h_s} + \frac{2}{3} \cdot \frac{h_s^2}{2}(b_{h_s} - T_{h_s} x_{h_s})$

$x_{h_{s-1}} := 0$, $b_{h_{s-1}} := R_s(b_{h_s} - T_{h_s} y_{h_s})$

Rufe Algorithmus 1 auf mit h_{s-1}, $b_{h_{s-1}}$, $x_{h_{s-1}}$. *Verwende das Ergebnis* $y_{h_{s-1}}$ *aus dem rekursiven Aufruf um* y_{h_s} *zu definieren.*

$y_{h_s} := y_{h_s} + P_s y_{h_{s-1}}$.

In der Praxis rufen wir Algorithmus 1 in jedem Iterationsschritt k nur einmal auf dem feinsten Gitter auf. D.h. wir verwenden $h_s = h_l$, b_{h_l}, $x_{h_l} = x_{h_l}^{(k)}$, der Rest erfolgt rekursiv. Die neue Näherung ist dann $x_{h_l}^{(k+1)} = y_{h_s}$.

Wir erwähnen noch der Vollständigkeit halber, dass sich das Mehrgitterverfahren durch mehrere Schritte Vorglättung und Nachglättung sowie mehrere rekursive Aufrufe verallgemeinern lässt. Hierzu wählen wir zwei Parameter $\nu_1, \nu_2 \in \mathbb{N}$ die die Anzahl Gättungsschritte beschreiben soll sowie einen weiteren Parameter $\mu \in \mathbb{N}$, der die Anzahl rekursiver Aufrufe steuert. Mit Hilfe der Parameter ν_1, ν_2, μ können wir nun das allgemeine Mehrgitterverfahren definieren. Zum Zwecke der Allgeinheit ersetzen wir unsere konkrete Matrix T_{h_s} durch eine allgemeine Matrix A_{h_s}, die z.B. im Moment T_{h_s} selbst sein kann oder später die analoge Matrix aus dem zweidimensionalen Problem (1.15). Den Glätter $\frac{2}{3} \cdot \frac{h_s^2}{2} I$ ersetzen wir durch einen abstrakten Glätter $S_{h_s}^{-1}$.

Algorithmus 2 (Mehrgitterschritt)

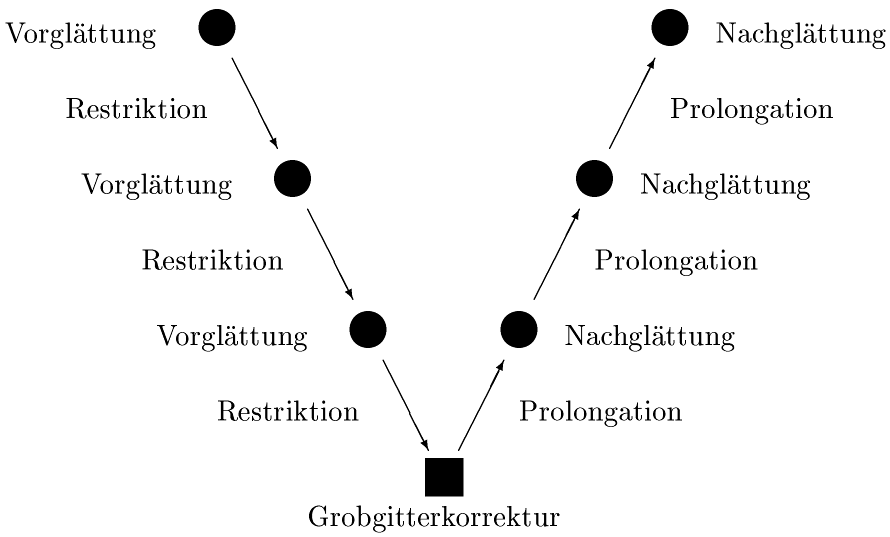
Gegeben: h_s *für ein* $s \in \{1, 2, \ldots, l\}$, *eine zugehörige rechte Seite* b_{h_s}, *eine Näherungslösung* x_{h_s}.

Der folgende Algorithmus berechnet rekursiv eine 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
Für $k = 1, 2, \ldots, \nu_1$
$y_{h_s} := y_{h_s} + S_{h_s}^{-1}(b_{h_s} - A_{h_s} y_{h_s})$
Grobgitterkorrektur
Für $k = 1, 2, \ldots, \mu$
$x_{h_{s-1}} := 0$, $b_{h_{s-1}} := R_s(b_{h_s} - A_{h_s} y_{h_s})$
Rufe Algorithmus 2 auf mit h_{s-1}, $b_{h_{s-1}}$, $x_{h_{s-1}}$. *Verwende das Ergebnis* $y_{h_{s-1}}$ *aus dem rekursiven Aufruf um* y_{h_s} *zu definieren.*
$y_{h_s} := y_{h_s} + P_s y_{h_{s-1}}$.
Nachglättung
Für $k = 1, 2, \ldots, \nu_2$
$y_{h_s} := y_{h_s} + S_{h_s}^{-1}(b_{h_s} - A_{h_s} y_{h_s})$

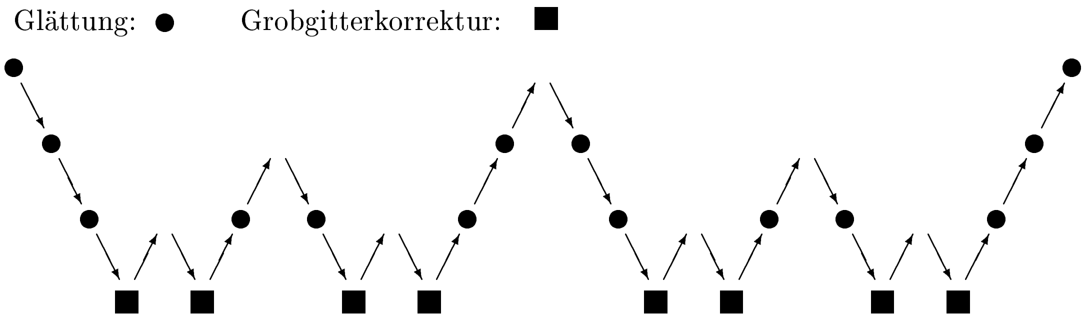
Beim genaueren Hinsehen sieht man, dass bei mehrmaliger Anwendung von Algorithmus 2 eigentlich nur die Summe $\nu = \nu_1 + \nu_2$ von nu_1 und ν_2 benötigt wird. Wenden wir Algorithmus 2 mit $\mu = 1$ and, so können wir in Abbildung 1.9 eine Skizze des rekursiven Aufrufs sehen ($l = 4$). Man bezeichnet dies als "V–Zyklus".

Abbildung 1.9: **Schema eines Mehrgitterschrittes "V–Zyklus"**



Verwenden wir anstelle von $\mu = 1$ jetzt $\mu = 2$, dann werden in jedem Schritt zwei rekursive Aufrufe statt einem durchgeführt. Da dies aber auf jedem Gitter h_s geschieht, steigt die Zahl der Aufrufe deutlich an. Da dies graphisch wie ein immer wiederholtes "W" aussieht, bezeichnet man dies als "W–Zyklus". Abbildung 1.10 zeigt die Aufrufe.

Abbildung 1.10: **Schema eines Mehrgitterschrittes “W–Zyklus”**



Wir ergänzen diese Abbildungen jetzt durch einige numerische Experimente. Dabei vergleichen wir einmal das Mehrgitterverfahren mit größtmöglicher Zahl Gitter mit dem Zweigitterverfahren. Zum anderen testen wir das Verhalten bei Zunahme von Glättungsschritten. Als Mehrgittervarianten wählen wir den “V–Zyklus” sowie den “W–Zyklus”. Tabelle 1.6 zeigt die Ergebnisse.

Tabelle 1.6: **Mehrgitterverf. für die 1–dim. Laplace–Gl.**

	Zweigitterv.		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.	Zeit (flops)	Anz. Iter.
$\frac{1}{511}$	$2.9{\cdot}10^5$	16	$5.8{\cdot}10^5$	21	$5.7{\cdot}10^5$	15	$2.4{\cdot}10^6$	17	$1.6{\cdot}10^6$	9
$\frac{1}{1023}$	$5.9{\cdot}10^5$	16	$1.2{\cdot}10^6$	21	$1.2{\cdot}10^6$	16	$5.5{\cdot}10^6$	17	$3.7{\cdot}10^6$	9
$\frac{1}{2047}$	$1.2{\cdot}10^6$	16	$2.5{\cdot}10^6$	22	$2.4{\cdot}10^6$	16	$1.2{\cdot}10^7$	16	$9.2{\cdot}10^6$	10
$\frac{1}{4095}$	$2.4{\cdot}10^6$	16	$4.9{\cdot}10^6$	22	$4.9{\cdot}10^6$	16	$2.6{\cdot}10^7$	16	$1.8{\cdot}10^7$	9

Tabelle 1.6 zeigt, dass durch Zunahme eines Glättungsschrittes die Zahl der Iterationen natürlich verringert wird, aber auf Kosten zusätzlicher Glättungen pro Schritt auf jedem Gitter. Der W–Zyklus ist natürlich aufwendiger, hat dafür weniger Iterationen. Da dieses Beispiel nur eindimaensional ist, sind die numerischen Beispiele sehr akademisch. Insbesondere wäre natürlich die Cholesky–Zerlegung für tridiagonale Matrizen hier das schnellste Verfahren. Das ändert sich im zweidimensionalen Fall.

1.6.2 Mehrgitterverfahren im 2D–Fall

Wir übertragen nun das Mehrgitterverfahren auf den zweidimensionalen Fall. Das einzige, was wir dafür tun müssen ist Restriktion und Prolongation zu übertragen.

Es ist klar, dass die triviale Restriktion auch hier nicht allzuviel Sinn macht. Als erstes Beispiel wählen wir jeden Knoten aus dem groben Gitter als Mittel über sein Analogon im feinen Gitter sowie sechs seiner Nachbarn. Zu diesem Zweck sei $x \in \Omega_h$ und $y \in \Omega_H$. Wir schreiben x, y wieder als Gitterfunktionen in der Form $x(ih, jh), y(ih, jH)$.

$$
\begin{aligned}
y(iH,jH) \quad &= \quad \frac{1}{4}x(iH,jH) + \frac{1}{8}\left[x(iH-h,jH) + x(iH+h,jH)\right. \\
&\qquad\qquad + x(iH,jH-h) + x(iH,jH+h) \\
&\qquad\qquad \left. + x(iH-h,jH-h) + x(iH+h,jH+h)\right], \\
&\quad i,j = 1,\dots,\frac{N-1}{2}, x \in \Omega_h.
\end{aligned}
\tag{1.29}
$$

In Form eines Stern sieht das so aus.

$$
\begin{array}{ccccc}
 & & \frac{1}{8} & & \frac{1}{8} \\
 & & | & / & \\
\frac{1}{8} & — & \frac{1}{4} & — & \frac{1}{8} \\
 & / & | & & \\
\frac{1}{8} & & \frac{1}{8} & &
\end{array}
\tag{1.30}
$$

Ein weiteres Besipiel wäre die fehlenden Nachbarn auch mit einzubeziehen. Zum Beispiel so.

$$
\begin{array}{ccccc}
\frac{1}{16} & & \frac{1}{8} & & \frac{1}{16} \\
 & \backslash & | & / & \\
\frac{1}{8} & — & \frac{1}{4} & — & \frac{1}{8} \\
 & / & | & \backslash & \\
\frac{1}{16} & & \frac{1}{8} & & \frac{1}{16}
\end{array}
\tag{1.31}
$$

Die hierzu passende Prolongationen sehen dann so aus. Sei dazu $y \in \Omega_H$. Dann wird $x \in \Omega_h$ wie folgt definiert. Als erstes die Analogie zu (1.30):

$$
x(ih,jh) = \begin{cases}
y(kH,lH) & i=2k, j=2l \\
\frac{1}{2}\left[y(kH,lH) + y(kH+H,lH)\right] & i=2k+1, j=2l \\
\frac{1}{2}\left[y(kH,lH) + y(kH,lH+H)\right] & i=2k, j=2l+1 \\
\frac{1}{2}\left[y(kH,lH) + y(kH+H,lH+H)\right] & i=2k+1, j=2l+1
\end{cases}
\tag{1.32}
$$

$i,j = 1,\dots,N,\ y \in \Omega_H$ im Falle von Restriktion (1.29),(1.30).

Als zweites für den Fall von Restriktion (1.31) wäre die Analogie bei der Prolongation diese.

$$x(ih,jh) = \begin{cases} y(kH,lH) & i=2k, j=2l \\ \frac{1}{2}\left[y(kH,lH)+y(kH+H,lH)\right] & i=2k+1, j=2l \\ \frac{1}{2}\left[y(kH,lH)+y(kH,lH+H)\right] & i=2k, j=2l+1 \\ \frac{1}{4}\left[y(kH,lH)+y(kH,lH+H)\right. & i=2k+1, \\ \left.+y(kH+H,lH+H)+y(kH+H,lH)\right] & j=2l+1 \end{cases} \tag{1.33}$$

$i,j=1,\ldots,N,\ y\in\Omega_H$.

Bezeichnen wir die daraus resultierenden Matrizen wieder mit $R:\Omega_h\to\Omega_H$ und $P:\Omega_H\to\Omega_h$ so gilt in beiden Fällen $P=4R^\top$ und wir erhalten damit das Mehrgitterverfahren für das zweidimensionale Problem (1.3). Tabelle 1.7 wendet wieder Algorithmus 2 an. Diesmal mit den Restriktionen, Prolongationen aus (1.29), (1.32). Als Dämpfungsparameter für das Jacobi–Verfahren verwenden wir wieder $\frac{2}{3}$ (man kann zeigen, dass die analoge Aussage aus dem eindimensionalen Fall im zweidimensionalen Fall auch gültig ist). Als Vergleichswerte nehmen wir das in den Experimenten bisher schnellste Verfahren, die Cholesky–Zerlegung.

Tabelle 1.7: **Mehrgitterverf. für die 2–dim. Laplace–Gl.**

	Cholesky	V–Zykl.,$\nu=1$		V–Zykl.,$\nu=2$		W–Zykl.,$\nu=1$		W–Zykl.,$\nu=2$	
h	Zeit (flops)	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.	Zeit (flops)	Anz. Iter.
$\frac{1}{31}$	$2.7\cdot10^5$	$2.2\cdot10^6$	44	$1.7\cdot10^6$	26	$3.7\cdot10^6$	40	$2.3\cdot10^6$	20
$\frac{1}{63}$	$3.0\cdot10^6$	$9.5\cdot10^6$	45	$7.7\cdot10^6$	27	$1.6\cdot10^7$	40	$1.0\cdot10^7$	20
$\frac{1}{127}$	$2.9\cdot10^7$	$4.0\cdot10^7$	46	$3.2\cdot10^7$	27	$7.0\cdot10^7$	40	$4.5\cdot10^7$	21
$\frac{1}{255}$	$2.9\cdot10^8$	$1.7\cdot10^8$	48	$1.3\cdot10^8$	28	$2.9\cdot10^8$	40	$1.9\cdot10^8$	21

Das direkte Verfahren schneidet immer noch ganz gut ab. Wir können das Mehrgitterverfahren noch etwas verbessern. Anstelle des gedämpften Jacobi–Glätters kann man das ganz normale Gauß–Seidel–Verfahren einsetzen. Ausserdem können wir einen Schritt des Mehrgitterverfahrens als Vorkonditionierer für ein cg–Verfahren einsetzen.

Wir erklären kurz, wie man das macht. Bei dem Mehrgitterverfahren handelt es sich um ein zusammengesetztes Iterationsverfahren (Vorglättung, Grobgitterkorrektur, Nachglättung). Im Prinzip kann jedes zusammengesetze iterative Verfahren gelesen werden als ein Schritt eines anderen Verfahren. Betrachten wir zum Beispiel das kombinierte Verfahren

$$\begin{aligned} x^{(k+1/2)} &= x^{(k)} + B_1(b-Ax^{(k)}) \\ x^{(k+1)} &= x^{(k+1/2)} + B_2(b-Ax^{(k+1/2)}) \end{aligned}$$

mit zwei irgendwie gewählten Approximationen B_1, B_2 an A^{-1}. Der erste Schritt besitzt die Iterationsmatrix $I-B_1A$, der zweite Schritt $I-B_2A$. Für das zusammengesetzte Verfahren bekommt man

$$I-BA \equiv (I-B_2A)(I-B_1A) = I - \underbrace{[B_2+B_1-B_2AB_1]}_{B} A$$

Wir können also (zumindest formal) die beiden Schritte in einem Schritt zusammenfassen.

$$\Rightarrow x^{(k+1)} = x^{(k)} + [B_1+B_2-B_2AB_1]\,(b-Ax^{(k)})$$

Dieses Prinzip ist natürlich auf mehr als 2 Schritte übertragbar. Ein einfaches Beispiel ist das symmetrische Gauß–Seidel Verfahren, das aus dem Gauß–Seidel–Verfahren mit Vorwärtseinsetzen gefolgt von dem Gauß–Seidel–Verfahren mit Rückwärtseinsetzen zusammengefasst werden kann. Sei dazu wieder A zerlegt in $A = D - E - E^\top$ mit Diagonalanteil D, negativem strikten unteren Dreiecksanteil E und entsprechend $E^\top$ dem negativen oberen Dreiecksanteil (aus Symmetriegründen). Dann ist $B_1 = (D-E)^{-1}$, $B_2 = (D-E^\top)^{-1}$ und wir erhalten

$$\begin{aligned} B &= B_1 + B_2 - B_2AB_1 \\ &= (D-E)^{-1} + (D-E^\top)^{-1} - (D-E^\top)^{-1}A(D-E)^{-1} \\ &= (D-E^\top)^{-1}\Big((D-E^\top) + (D-E) - A\Big)(D-E)^{-1} \\ &= (D-E^\top)^{-1}D(D-E)^{-1} \end{aligned}$$

Letzteres ist gerade die Inverse des symmetrischen Gauß–Seidel–Verfahrens. Vom Prinzip her können wir beim Mehrgitterverfahren genauso vorgehen. Ein Schritt des Zweigitterverfahrens (der Einfachheit halber) besitzt die zusammengesetzte Iterationsmatrix

$$I - M_2A \equiv (I - S_{h_2}A_{h_2})^{\nu_2}(I - P_2A_{h_1}^{-1}R_2A_{h_2})^{\mu}(I - S_{h_2}A_{h_2})^{\nu_1} \tag{1.34}$$

Hier haben wir gleich insgesamt $k = \nu_1 + \mu + \nu_2$ einzelne Iterationsschritte zusammenzufassen. Dies ist formal etwas aufwendiger. Wir beschreiben daher kurz die Herangehensweise.

Ist etwa ein aus k Schritten zusammengesetztes iteratives Verfahren gegeben durch die Iterationsmatrix

$$(I - A_kA)(I - A_{k-1}A)\cdots(I - A_1A) \equiv I - BA$$

so erhalten wir durch Auflösen nach B die Formel

$$B = [I - (I - A_kA)(I - A_{k-1}A)\cdots(I - A_1A)]\,A^{-1}.$$

Tatsächlich taucht A^{-1} nur formal auf. Durch Ausmultiplizieren könnten wir verfizieren, dass sich A^{-1} immer gegen A herauskürzt. Trotzdem ist diese Formel nicht praktikabel, wenn wir sie etwa beim Mehrgitterverfahren einsetzen wollen. Uns wäre uns lieber, wenn wir ein Schema hätten, das wir so auch ohne zusätzlichen Aufwand umsetzen könnten. Sei dazu etwa

$$B_{l,k} \equiv [I - (I - A_kA)(I - A_{k-1}A)\cdots(I - A_lA)]\,A^{-1}.$$

Dann ist $B = B_{1,k}$ und wir erhalten die Rekursionsformel

$$\begin{aligned} B &\equiv B_{1,k} \\ &= [I - (I - A_kA)(I - A_{k-1}A)\cdots(I - A_1A)]\,A^{-1} \\ &= [I - (I - A_kA)(I - A_{k-1}A)\cdots(I - A_2A)]\,A^{-1}(I - AA_1) + A_1 \\ &= B_{2,k}(I - AA_1) + A_1 \end{aligned} \tag{1.35}$$

Allgemein haben wir damit eine rekursive Darstellung von $B = B_{1,k}$ für $s = 1, 2, \ldots, k-1$.

$$\begin{aligned} s < k: \quad B_{s,k} &= B_{s+1,k}(I - AA_s) + A_s \\ s = k: \quad B_{k,k} &= A_k \end{aligned} \tag{1.36}$$

Die Anwendung von B auf einen Vektor x würde jetzt so aussehen.

Berechne $y = Bb$
$y = 0$
for $s = 1, 2, \ldots, k$
  if $s > 1$: $b = b - Ad$
  Bestimme $d = A_s b$ (ggf. durch Lösen eines Gleichungssystems)
  $y = y + d$

Dieses Schema lässt sich natürlich auch auf einen Schritt des Zweigitter– oder des Mehrgitterverfahrens anwenden. Damit bekommen wir auf dem feinsten Gitter einen Vorkonditionierer des cg–Verfahrens in Form eines Mehrgitterschrittes. Um sicherzustellen, dass der aus einem Schritt des Mehrgitterverfahrens entstandene Vorkonditionierer symmetrisch ist, muss lediglich $\nu_1 = \nu_2$ gewählt werden und bei der Nachglättung $S_{h_s}^{\top}$ anstelle von S_{h_s} verwendet werden.

Die Darstellung (1.34) bezieht sich nur auf das Zweigitterverfahren. Man kann sich überlegen, das im Falle des Mehrgitterverfahrens die zu (1.34) analoge Iterationsmatrix gegeben ist durch M_l, welche selbst rekursiv für $s = l, l-1, \ldots, 2$ definiert ist.

$$\begin{aligned} s > 1: \quad I - M_s A &= (I - S_{h_s} A_{h_s})^{\nu_2} (I - P_s M_{s-1} R_s A_{h_s})^{\mu} (I - S_{h_s} A_{h_s})^{\nu_1}, \\ s = 1: \quad M_1 &= A_1^{-1} \end{aligned} \tag{1.37}$$

Tabelle 1.8 zeigt die deutliche Verbesserung bei der Kombination des cg–Verfahrens mit einer Mehrgittervorkonditionierung.

Tabelle 1.8: **cg mit Mehrgittervork. für die 2–dim. Laplace–Gl.**

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

Wir haben in diesem Abschnitt ein– und zweidimensionale Mehrgitterverfahren gesehen in verschiedenen Raumdimensionen und mit verschiedenen Glättern. Die Überlegenheit dieser Verfahren ist anhand numerischer Beispiele demonstriert worden.

1.7 Eine elementare Konvergenzanalyse

In diesem Abschnitt geben wir eine genaue Konvergenzanalyse des Zweigitterverfahrens an. Wir beschränken uns dabei auf den eindimensionalen Fall. Auf den zweidimensionalen Fall werden wir aber auch kurz eingehen und die Übertragungen andiskutieren.

Wir unterscheiden in diesem Abschnitt zwischen feinem Gitter Ω_h und groben Gitter Ω_H. Zur besseren Unterscheidung tragen die entsprechenden Eigenvektoren deshalb einen zusätzlichen Index.

1.7.1 Konvergenzanalyse im 1D–Fall

Wir gehen bei der Konvergenzanalyes in drei Schritten vor. Schritt eins ist die Glättung, der zweite Schritt besteht aus der Restriktion und der dritte schließlich behandelt die Prolongation. Alle drei zusammen liefern dann die Fehlerabschätzung für einen Schritt des Mehrgitterverfahrens.

Wie wir bereits im Abschnitt 1.4 gesehen haben ist bei gegebenen Eingangsfehler $e = \sum_{k=1}^{N} \alpha_k \cos(k\pi h) s_h^{(k)}$ der Fehler nach einem Schritt Glättung gegeben durch

$$(I - \omega \frac{h^2}{2} T_h)e = \sum_{k=1}^{N} \alpha_k (1 - 2\omega \sin^2 \frac{k\pi h}{2}) s_h^{(k)}. \tag{1.38}$$

Dabei wählen wir $\omega = \frac{2}{3}$, um die Gättungseigenschaft zu sichern. $s_h^{(k)} = \Big(\sin(ik\pi h)\Big)_{i=1,\dots,N}$, $k = 1, \dots, N$ sind die Eigenvektoren von T_h aus (1.19). T_h besitzt nach (1.20) die Eigenwerte $\lambda_h^{(k)} = \frac{4}{h^2} \sin^2 \frac{k\pi h}{2}$. Wir betrachten den Fall, dass das exakte Zweigitterverfahren $\nu = \nu_1$ Schritte Vorglättung sowie $\mu = 1$ Schritt Grobgitterkorrektur verwendet. Man kann übrigens zeigen, dass beim exakten Zweigitterverfahren mehr als eine Grobgitterkorrektur keine Verbesserung bringen, weil ein Schritt Grobgitterkorrektur eine exakte orthogonale Projektion im Sinne des durch T_h erzeugten inneren Produktes darstellt.

Schritt 1: die Glättung

Wenden wir insgesamt ν Glättungsschritte an, so erhalten wir in Analogie zu (1.38) nun

$$(I - \frac{2}{3} \frac{h^2}{2} T_h)^\nu e = \sum_{k=1}^{N} \alpha_k (1 - \frac{4}{3} \sin^2 \frac{k\pi h}{2})^\nu s_h^{(k)}. \tag{1.39}$$

Gleichung (1.39) beschreibt die Fehlergewichte nach ν Schritten Glättung. Aus (1.21) wissen wir, dass für $k > N/2$, $\left|1 - \frac{4}{3} \sin^2 \frac{k\pi h}{2}\right|^\nu \leqslant \frac{1}{3^\nu}$ ist.

Wir kommen nun zum anschließenden Korrekturschritt. Der Fehler ist dann

$$f = (I - PT_H^{-1} RT_h)(I - \frac{2}{3} \frac{h^2}{2} T_h)^\nu e. \tag{1.40}$$

Mit Hilfe von (1.39) bekommen wir

$$\begin{aligned} f &= \sum_{k=1}^{N} \alpha_k (1 - \frac{4}{3} \sin^2 \frac{k\pi h}{2})^\nu (I - PT_H^{-1} RT_h) s_h^{(k)} \\ &= \sum_{k=1}^{N} \alpha_k (1 - \frac{4}{3} \sin^2 \frac{k\pi h}{2})^\nu (s_h^{(k)} - \frac{4}{h^2} \sin^2 \frac{k\pi h}{2} PT_H^{-1} R s_h^{(k)}) \end{aligned} \tag{1.41}$$

Schritt 2: die Restriktion

Als nächstes betrachten wir die Restriktion $y = Rs^{(k)}$. Wegen

$$\begin{aligned}
\sin((i-1)k\pi h) + 2\sin(ik\pi h) + \sin((i+1)k\pi h) &= \sin(ik\pi h)\cos(k\pi h) - \sin(k\pi h)\cos(ik\pi h) \\
&\quad +2\sin(ik\pi h) \\
&\quad + \sin(ik\pi h)\cos(k\pi h) + \sin(k\pi h)\cos(ik\pi h) \\
&= 2\,(1+\cos(k\pi h))\sin(ik\pi h) \qquad (1.42)
\end{aligned}$$

bekommen wir für die Restriktionsvorschrift (1.23)

$$y(iH) = \frac{1}{4}x(iH-h) + \frac{1}{2}x(iH) + \frac{1}{4}x(iH+h),\ i = 1,\ldots,\frac{N-1}{2}, x \in \Omega_h.$$

folgende Beziehung (angewendet auf jedes einzelne $s_h^{(k)}$):

$$\begin{aligned}
y(jH) &= \frac{1}{4}s_h^{(k)}(jH-h) + \frac{1}{2}s_h^{(k)}(jH) + \frac{1}{4}s_h^{(k)}(jH+h) \\
&= \frac{1+\cos(k\pi h)}{2}\sin(jk\pi H) \\
&= \cos^2\frac{k\pi h}{2}\ \sin(jk\pi H),\ j = 1,\ldots,\frac{N-1}{2}. \qquad (1.43)
\end{aligned}$$

Dabei haben wir nur gerade Werte $i = 2j$ verwendet und $2ih = jH$ ausgenutzt. Folgerung:

$$Rs_h^{(k)} = \cos^2\frac{k\pi h}{2}s_H^{(k)},\ k = 1,\ldots,N. \qquad (1.44)$$

Aus Symmetriegründen gilt bereits

$$Rs_h^{(k)} = -\cos^2\frac{k\pi h}{2}s_H^{(N+1-k)},\ k = \frac{N+1}{2},\ldots,N.$$

und $Rs_h^{(\frac{N+1}{2})}$ wird auf $s_H^{(\frac{N+1}{2})} = 0$ abgebildet. Die Analyse benötigt diese Details aber nicht.

Sei $f = (I - PT_H^{-1}RT_h)(I - \frac{2}{3}\frac{h^2}{2}T_h)^\nu e$. Damit erhalten wir für unser Zweigitterverfahren

$$\begin{aligned}
f &= \sum_{k=1}^N \alpha_k(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - \frac{4}{h^2}\sin^2\frac{k\pi h}{2}PT_H^{-1}Rs_h^{(k)}) \\
&= \sum_{k=1}^N \alpha_k(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - \frac{4}{h^2}\cos^2\frac{k\pi h}{2}\sin^2\frac{k\pi h}{2}PT_H^{-1}s_H^{(k)}) \\
&= \sum_{k=1}^N \alpha_k(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - \frac{4}{h^2}\cos^2\frac{k\pi h}{2}\sin^2\frac{k\pi h}{2}\frac{H^2}{4\sin^2\frac{k\pi H}{2}}Ps_H^{(k)}) \\
&= \sum_{k=1}^N \alpha_k(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - Ps_H^{(k)}). \qquad (1.45)
\end{aligned}$$

Dabei haben wir ausgenutzt, dass nach (1.20), $s_H^{(k)}$ Eigenvektor von T_H^{-1} zum Eigenwert

$$\frac{H^2}{4\sin^2\frac{k\pi H}{2}} = \frac{h^2}{4\cos^2\frac{k\pi h}{2}\ \sin^2\frac{k\pi h}{2}}$$

ist. (1.45) zeigt eindrucksvoll, dass nach einem Schritt Grobgitterkorrektur der Fehler genau durch den Fehler $s_h^{(k)} - Ps_H^{(k)}$ bestimmt wird. D.h. das ist der Fehler der zwischen den interpolierten Eigenvektoren $Ps_h^{(k)}$ des gröberen Gitters (die zu den niedrigen Frequenzen gehören) und den Eigenvektoren $s_h^{(k)}$ des feineren Gitters entsteht. Wir erwarten, dass dieser Fehler klein ist, sofern $k < N/2$ ist, also in dem Fall wo $s_h^{(k)}$ selbst zu den niedrigen Frequenzen zählt. Aus Abschnitt 1.4 werden die hochfrequenten Anteile für $k > N/2$ durch die Glättung gedämpft werden. Dieser Anteil entspricht gerade dem Faktor $(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu$.

Schritt 3: die Prolongation

Als letzten Schritt ist jetzt die Prolongation $x = Ps_H^{(k)}$ zu untersuchen. Für gerades $i = 2j$ ist nach (1.24)

$$x(ih) = \sin(jk\pi H) = \sin(ik\pi h) = -\sin(i(N+1-k)\pi h). \tag{1.46}$$

Ansonsten ist für ungerades $i = 2j+1$ nach (1.24)

$$\begin{aligned} x(ih) &= \frac{1}{2}\sin(jk\pi H) + \frac{1}{2}\sin((j+1)k\pi H) \\ &= \frac{1}{2}\sin((i-1)k\pi h) + \frac{1}{2}\sin((i+1)k\pi h) \\ &= \cos(k\pi h)\sin(ik\pi h) = \cos(k\pi h)\sin(i(N+1-k)\pi h) \end{aligned} \tag{1.47}$$

Wir können die Gleichungen (1.46), (1.47) für gerades und ungerades i vereinheitlichen.

Es ist für gerades $i = 2j$

$$x(ih) = \frac{1+\cos(k\pi h)}{2}\ \sin(ik\pi h) - \frac{1-\cos(k\pi h)}{2}\ \sin(i(N+1-k)\pi h)$$

und für ungerades $i = 2j+1$

$$x(ih) = \frac{\cos(k\pi h)+1}{2\cos(k\pi h)}\ \cos(k\pi h)\sin(ik\pi h) + \frac{\cos(k\pi h)-1}{2\cos(k\pi h)}\ \cos(k\pi h)\sin(i(N+1-k)\pi h).$$

D.h. wir haben unter Ausnutzung der Additionstheoreme für alle $i = 1, \dots, N$

$$x(ih) = \cos^2\frac{k\pi h}{2}\ \sin(ik\pi h) - \sin^2\frac{k\pi h}{2}\ \sin(i(N+1-k)\pi h). \tag{1.48}$$

Hieraus folgt sofort

$$Ps_H^{(k)} = \left[s_h^{(k)}, s_h^{(N+1-k)}\right]\begin{bmatrix} \cos^2\frac{k\pi h}{2} \\ -\sin^2\frac{k\pi h}{2} \end{bmatrix}. \tag{1.49}$$

Für den Fehler f des Zweigitterverfahrens aus (1.45) folgt hieraus

$$\begin{aligned} f &= \sum_{k=1}^{N}\alpha_k(1-\frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - Ps_H^{(k)}) \\ &= \sum_{k=1}^{N}\alpha_k(1-\frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu(s_h^{(k)} - \left[s_h^{(k)}, s_h^{(N+1-k)}\right]\begin{bmatrix} \cos^2\frac{k\pi h}{2} \\ -\sin^2\frac{k\pi h}{2} \end{bmatrix}) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=1}^{N} \alpha_k (1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu \sin^2\frac{k\pi h}{2}\left[s_h^{(k)} + s_h^{(N+1-k)}\right] \\
&= \sum_{k=1}^{N} (\alpha_k + \alpha_{N+1-k})\,(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu \sin^2\frac{k\pi h}{2} s_h^{(k)}
\end{aligned}
$$

Als Fehlergewichte treten also die Faktoren $(1 - \frac{4}{3}\sin^2\frac{k\pi h}{2})^\nu \sin^2\frac{k\pi h}{2}$ auf (neben der Summe $\alpha_k + \alpha_{N+1-k}$ anstelle von α_k).

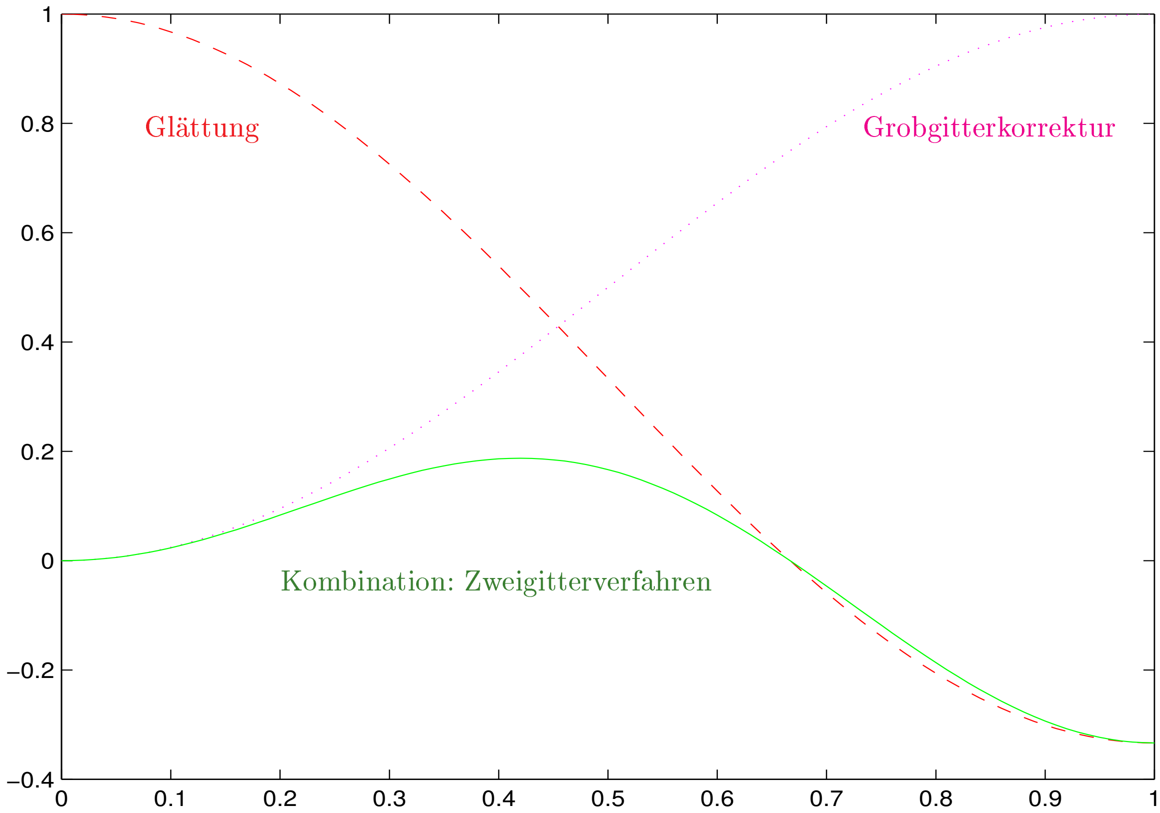
Die Funktion

$$f_\nu(t) = (1 - \frac{4}{3}\sin^2\frac{t\pi}{2})^\nu \sin^2\frac{t\pi}{2},\ t \in [0,1]$$

beschreibt genau das Zusammenspiel zwischen Glättung und Grobgitterkorrektur. Der Anteil $(1 - \frac{4}{3}\sin^2\frac{t\pi}{2})^\nu$ stammt aus der Glättung und ist für $t \geqslant 1/2$ (entspricht $k > N/2$) betraglich klein. Der Anteil $\sin^2\frac{t\pi}{2}$ kommt durch die Grobgitterkorrektur hinzu und ist für $t \leqslant 1/2$ (entspricht $k \leqslant N/2$) klein. Also muss das Produkt immer betraglich klein sein.

Wir betrachten einmal für $\nu = 1$ die Werte der Funktion f_ν und ihrer Glättungs– und Grobgitterkorrekturanteile. siehe Abbildung (1.11).

Abbildung 1.11: **Glättung, Grobgitter und Kombination beider ($\nu = 1$)**



Wir stellen insbesondere fest, dass in Abhängigkeit von ν folgende betragliche Maxima für f_ν auftreten (Tabelle (1.9)).

Diese Maxima sind alle gleichmäßig beschränkt für Werte die deutlich kleiner als 1 sind.

Tabelle 1.9: **Konvergenzraten für das exakte Zweigitterverfahren**

ν	1	2	3	$\nu \geqslant 2$
$\rho = \max_{t\in[0,1]} \lvert f_\nu(t)\rvert$	$\frac{1}{3}$	$\frac{1}{9}$	$\frac{3^4}{4^5}$	$\frac{3}{4}\frac{\nu^\nu}{(\nu+1)^{\nu+1}}$

Zusammengefasst erhalten wir folgenden Satz.

Satz 3 *Das exakte Zweigitterverfahren (k Schritte) mit gedämpftem Jacobi–Gätter ($\omega = \frac{2}{3}$) und ν Glättungsschritten liefert folgende Fehlerabschätzung angewendet auf das eindimensionale Modellproblem (1.1) (unter Verwendung der Diskretisierung (1.10) und Restriktion, Prolongation aus (1.23), (1.24)):*

$$\| \left[(I - PT_H^{-1}RT_h)(I - \frac{2}{3}\frac{h^2}{2}T_h)^\nu \right]^k e\|_2 \leqslant 2\rho^k \|e\|_2, \text{ für alle } e \in \mathbb{R}^n,$$

wobei für ρ die Werte aus Tabelle (1.9) gelten.

Beweis. Dies folgt sofort aus den obigen Betrachtungen, weil die $s_h^{(k)}, k = 1, \ldots, N$ orthogonal zueinander sind. Der Faktor 2 ergibt sich aus der Abschätzung $\left[\sum_{k=1}^N (\alpha_k + \alpha_{N+1-k})^2\right]^{1/2} \leqslant 2\left[\sum_{k=1}^N \alpha_k^2\right]^{1/2}$. Dieser Faktor tritt bei 2 und mehr Schritten nicht mehr zusätzlich auf, da nach einem Schritt des Zweigitterverfahrens dieser Anteil symmetrisiert ist. □

Als Abbruchkriterium für das Mehrgitterverfahren ist bisher immer $\sqrt{\text{eps}} \approx 1.5 \cdot 10^{-8}$ gewählt worden (verglichen mit der Ausgangsnorm). Nach 16 bzw. 17 Schritten ist im Falle von $\nu = 1$ gerade $2\rho^16 \approx 4.6 \cdot 10^{-8}$ sowie $2\rho^17 \approx 1.5 \cdot 10^{-8}$. Somit reichen 16–17 Schritte für diese Genauigkeit aus. Tabelle (1.5) zeigt genau dieses Verhalten!

1.7.2 Übertragung auf den 2D–Fall

Wir betrachten jetzt grob die Verallgemeinerung auf den zweidimensionalen Fall. Als erstes bekommen wir die Eigenfunktionen des zweidimensionalen Problems

$$-\Delta u = f \text{ in } \Omega,\ u = 0 \text{ on } \partial\Omega$$

durch Tensorproduktbildung der eindimensionalen Eigenfunktionen bekommen.

$$-\Delta\left[\sin(k\pi x)\sin(l\pi y)\right] = \left(k^2 + l^2\right)\pi^2 \cdot \sin(k\pi x)\sin(l\pi y).$$

Diese Analogie überträgt sich auf die Eigenvektoren bei Verwendung des 5–Punkte–Sterns (1.13), d.h. die Eigenvektoren der Matrix A_h aus (1.15) sind

$$e^{(k,l)}(ih, jh) = \sin(ik\pi h)\sin(jl\pi h).$$

Als Eigenwerte erhält man

$$\lambda_{k,l} = \frac{4}{h^2}\left(\sin^2\frac{k\pi h}{2} + 4\sin^2\frac{l\pi h}{2}\right).$$

Die Glättungsanalyse aus Abschnitt 1.4 liefert hier ebenfalls $\omega = \frac{2}{3}$. Das liegt daran, dass durch das Tensorieren der Eigenfunktion/Eigenvektoren, für $k > N/2$ oder $l > N/2$ der zugehörige Eigenvektor bereits hochfrequent ist.

Wichtigstes algebraisches Hilfsmittel im zweidimensionalen Fall ist die Tatsache, dass

$$A_h = T_h \otimes I + I \otimes T_h$$

ist, wobei $\otimes$ das Kroneckerprodukt zweier Matrizen bezeichnet. Diese Darstellung erlaubt es i.w. die eindimensionalen Anteile in x–Richtung und y–Richtung weitestgehend separat zu behandeln und auf den eindimensionalen Fall zurückzuführen. Satz 3 kann fast wortwörtlich übertragen werden. Einziger Unterschied ist, dass der Trick mit dem Faktor 2 nicht mehr klappt.

Zusammengefasst erhalten wir folgenden Satz.

Satz 4 *Das exakte Zweigitterverfahren (k Schritte) mit gedämpftem Jacobi–Gätter ($\omega = \frac{2}{3}$) und ν Glättungsschritten liefert folgende Fehlerabschätzung angewendet auf das zweidimensionale Modellproblem (1.3) (unter Verwendung der Diskretisierung (1.13) und Restriktion, Prolongation aus (1.29), (1.32)).*

$$\| \left[(I - PT_H^{-1}RT_h)(I - \frac{2}{3}\,\frac{h^2}{2}T_h)^{\nu} \right]^k e\|_2 \leqslant (2\rho)^k \|e\|_2, \text{ für alle } e \in \mathbb{R}^n,$$

wobei für ρ die Werte aus Tabelle (1.9) gelten.

Die Ergebnisse dieses Abschnittes gelten nur für das exakte Zweigitterverfahren. Eine Verallgemeinerung auf den Fall des Mehrgitterverfahrens fehlt bisher und soll auch aus Gründen der einfacheren Darstellung weggelassen werden.

1.8 Mehrgitterartige Verfahren für Variationsprobleme

Wir haben uns bisher auf sehr spezielle Fälle des Mehrgitterverfahrens beschränkt. Dies war lediglich ein Modellproblem auf einem Rechteckgitter (in 2D) mit spezieller Diskretisierung und dafür zurechtgeschnittenen Restriktionen und Prolongationen. Der für diese Problemklasse adäquate Zugang besteht üblicher Weise in Finite–Elemente–Methoden(FEM). Als Literatur sei z.B. [7, 32] verwiesen. Es gibt aber auch viele weitere Bücher.

1.8.1 Variationsformulierung

Dazu wird das Problem (1.5), (1.6) aus Abschnitt 1.1

(1.50)
$$-\ \mathrm{div}\ (A \ \mathrm{grad}\ u) = f, \ \text{in } \Omega$$

(1.51)
$$u = g \text{ auf } \Gamma_1, \ \nu^\top A \ \mathrm{grad}\ u = h, \ \text{auf } \Gamma_2.$$

durch Integration in die sogenannte schwache Formulierung überführt.

Wir multiplizerien dazu Gleichung (1.50) mit einer Funtkion v und integrieren.

$$\begin{aligned} -\int_\Omega v \text{ div } (A \text{ grad } u)\ dx &= \int_\Omega \Big[(\text{ grad } v)^\top A \text{ grad } u - \text{ div } (vA \text{ grad } u)\Big]\ dx \\ &= \int_\Omega (\text{ grad } v)^\top A \text{ grad } u\ dx - \int_{\partial\Omega} v\nu^\top A \text{ grad } u\ ds \end{aligned} \tag{1.52}$$

Dabei haben wir den Gaußschen Integralsatz

$$\int_\Omega \text{ div } w\ dx = \int_{\partial\Omega} \nu^\top w\ ds$$

benutzt. ν bezeichnet das äussere Einheitsnormalenfeld auf $\partial\Omega$. Wir nehmen jetzt an, dass wir unsere gesuchte Funktion u schreiben können als $u_0 + w$, wobei u_0 eine bekannte Funktion auf Ω ist, welche die Randbedingungen $u_0 = g$ auf Γ_1, $\nu^\top A$ grad $u_0 = h$, auf Γ_2 bereits erfüllt.

Es reicht in diesem Falle Funktionen v, w mit $v = w = 0$ auf Γ_1, $\nu^\top A$ grad $v = \nu^\top A$ grad $w = 0$, auf Γ_2 zu betrachten. Wir setzen

$$(v, f) = \int_\Omega vf\ dx,\ a(v, w) = \int_\Omega (\text{ grad } v)^\top A \text{ grad } w\ dx$$

und erhalten damit

$$\begin{aligned} (v, f) &= \int_\Omega vf\ dx \\ &= \int_\Omega (\text{ grad } v)^\top A \text{ grad } (u_0 + w)\ dx - \int_{\partial\Omega} v\nu^\top A \text{ grad } (u_0 + w)\ ds \\ &= a(v, u) + a(v, u_0) - \int_{\Gamma_2} v\nu^\top A \text{ grad } u_0\ ds \end{aligned}$$

Die schwache Formulierung lautet nun wie folgt. Sei

$$H = \{u : \int_\Omega (u^2 + (\text{ grad } u)^\top A \text{ grad } u)\ dx \text{ existiert, } u = 0 \text{ auf } \Gamma_1,\ \nu^\top A \text{ grad } u = 0 \text{ auf } \Gamma_2\}.$$

Wir bezeichnen mit $H_0^1(\Omega)$ den Raum

$$H_0^1(\Omega) = \overline{H},$$

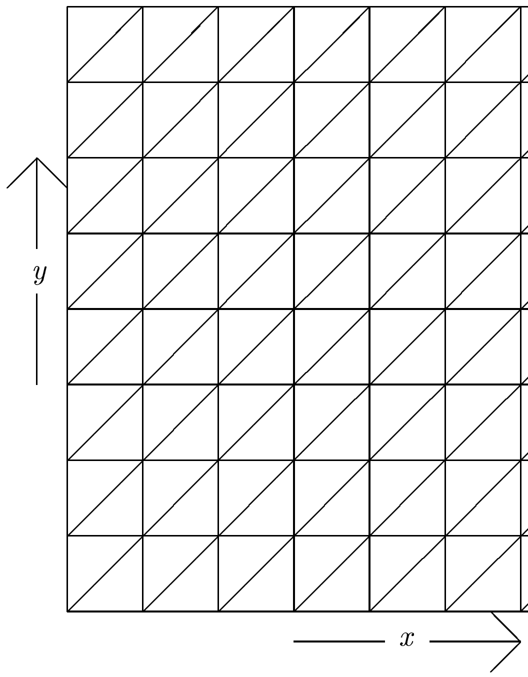
d.h. den Abschluss von H bzgl. der Norm $\|u\|^2 = \int_\Omega (u^2 + (\text{ grad } u)^\top A \text{ grad } u)\ dx$. Eine präzisere Definition von H_0^1 (Sobolev–Räume) sparen wir uns hier.

Gesucht ist eine Funktion $u \in H_0^1(\Omega)$ derart, dass

$$a(v, w) = (v, f) - a(v, u_0) + \int_{\Gamma_2} v\nu^\top A \text{ grad } u_0\ ds, \tag{1.53}$$

für alle $v \in H_0^1(\Omega)$.

Abbildung 1.12: **Triangulierung des Gebietes Ω**



1.8.2 Finite Elemente

Bei der finiten–Elemente–Methode(FEM) wird jetzt der Raum H_0^1 numerisch durch einen kleineren (endlichdimensionalen) Raum ersetzt. Eine Möglichkeit ist, das Gebiet Ω mit einem Dreiecksnetz zu überziehen. Im Falle von $[0,1]^2$ sieht das z.B. so aus (Abbildung 1.12).

Als Ansatzraum einer solchen Triangulierung Ω_h nimmt man üblicher Weise stetige Funktionen, die stückweise Polynome auf den Dreiecken sind. Im einfachsten Fall könnte man stückweise lineare Funktionen nehmen. Wir bezeichnen den Ansatzraum mit $S_h(\Omega_h)$. Da S_h endlichdimensional ist, besitzt er eine Basis. Im Falle linearer Elemente kann man als Ansatzfunktionen stückweise lineare Funktionen ψ_l mit lokalem Täger wählen. Diese Funktionen wären durch Vorgabe der Funktionswerte in den Eckpunkten eindeutig bestimmt (Abbildung 1.13). Bei stückweise quadratischen Funktionen und Funktionen höheren Grades kann man ähnlich vorgehen.

Wir erhalten damit einen Satz von Basisfunktion $\left(\psi_l\right)_{l=1,\ldots,n}$. n sei dabei die Dimension des Raumes S_h, die sogenannte nodale Basis. Wir bezeichnen mit Ψ die (formale) Matrix

$$\Psi = [\psi_1, \ldots, \psi_n] \tag{1.54}$$

Wählen wir nun $w, v \in S_h$ so bekommen wir eine für u, v eine Darstellung

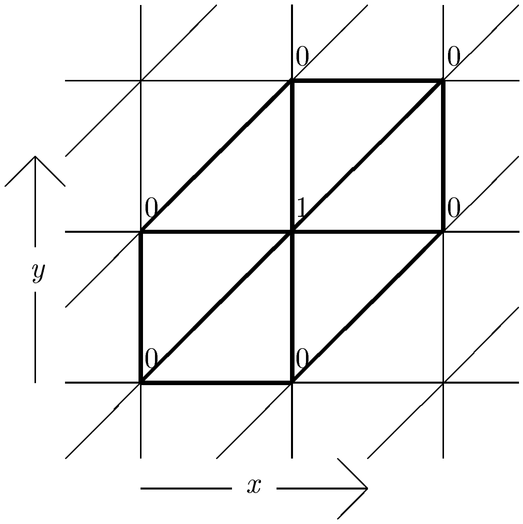
$$w = \Psi \underline{\mathbf{w}}, v = \Psi \underline{\mathbf{v}}. \tag{1.55}$$

Dabei bezeichnen $\underline{\mathbf{u}}, \underline{\mathbf{v}} \in \mathbb{R}^n$ die Koordinaten von u, v bezüglich der Basis $\left(\psi_l\right)_{l=1,\ldots,n}$.

Das Variationsproblem (1.53) ist dann äquivalent zu

$$\sum_{i,j=1}^{n} \underline{\mathbf{v}}_i a(\psi_i, \psi_j) \underline{\mathbf{w}}_j = \sum_{i=1}^{n} \underline{\mathbf{v}}_i \left[(\psi_i, f) - a(\psi_i, u_0) + \int_{\Gamma_2} \psi_i \nu^\top A \text{ grad } u_0 \, ds, \right]$$

Abbildung 1.13: **stückw. lin. Basisfunktion mit lokalem Träger**



Bezeichnen wir mit $A_h = (a_{ij,h})_{i,j=1,\dots,n}$,$b_h = (b_{i,h})_{i=1,\dots,n}$ die Größen

$$a_{ij,h} = a(\psi_i, \psi_j),\ b_{i,h} = (\psi_i, f) - a(\psi_i, u_0) + \int_{\Gamma_2} \psi_i \nu^\top A \text{ grad } u_0 \, ds, \tag{1.56}$$

so ist auf dem Raum S_h das Variationsproblem (1.53) äquivalent zu

$$\underline{\mathbf{v}}^\top A_h \underline{\mathbf{u}} = \underline{\mathbf{v}}^\top b_h, \text{ für alle } \underline{\mathbf{v}} \in \mathbb{R}^n$$

oder einfach

$$A_h \underline{\mathbf{u}} = b_h. \tag{1.57}$$

Die einzelnen Integrale in A_h, b_h lassen sich durch Kubaturformeln numerisch berechnen. Empfehlenswert ist dabei die gleiche Approximationsordnung zu verwenden wie bei den stückweise Polynomen im Ansatzraum S_h.

1.8.3 Mehrgitterverfahren für FEM–Diskretisierung

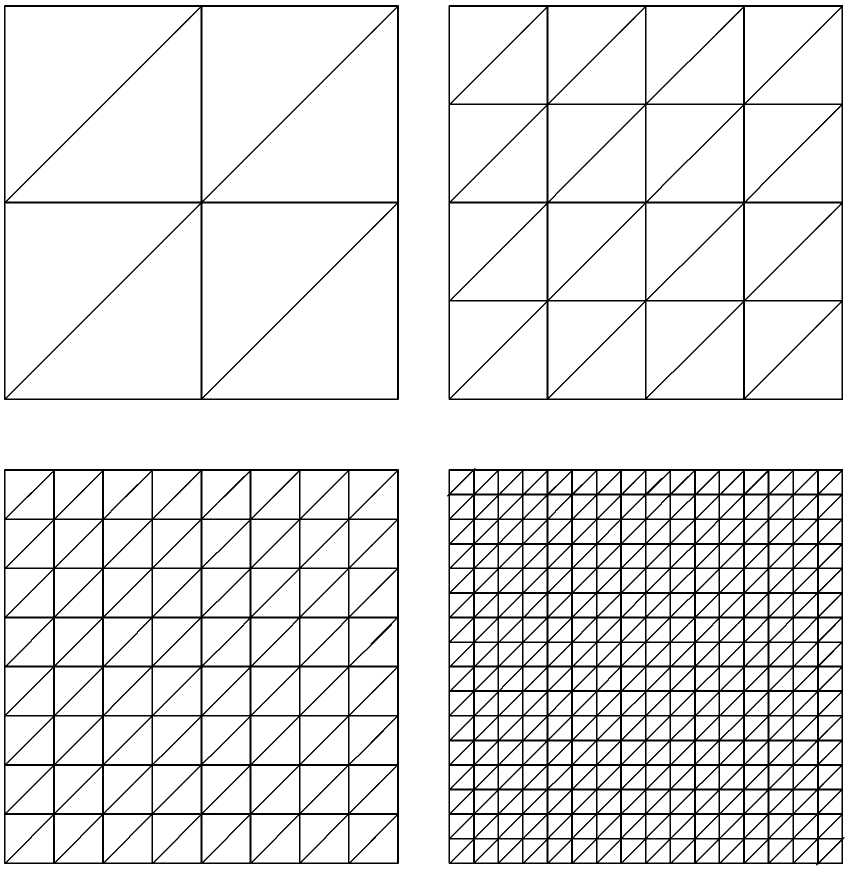
Wie bereits bei der Verwendung finiter Differenzen so wird auch bei der Verwendung finiter Elemente eine Hierarchie verwendet. Die Hierarchie der Gitter bei der Differenzenmethode wird ersetzt durch hierarchisch geschachtelte Dreiecksnetze $\Omega_{h_1} \subseteq \Omega_{h_2} \subseteq \cdots \subseteq \Omega_{h_l}$ und daraus resultiert eine entsprechende natürliche Einbettung der zugehörigen Ansatzräume

$$S_{h_1} \subseteq S_{h_2} \subseteq \cdots \subseteq S_{h_l}.$$

Die Einteilung des Problems (1.53) in eine Hierarchie von Ansatzräumen ist Grundlage für viele mehrgitterartige Verfahren. Dafür braucht man lediglich Interpolationsoperatoren vom gröberen Gitter in das feinere Gitter. Diese sind kanonisch, weil die Räume ineinander geschachtelt sind. Betrachten wir etwa ein Beispiel wo die Hierarchie der Triangulierung durch sukzessive Viertelung der bestehenden Dreiecke entsteht (Abbildung 1.14).

Verwenden wir etwa stückweise lineare Basisfunktionen $\psi_h^{(k)}$ zur Triangulierung der Seitenlänge h eines Dreiecks sowie eine beliebige stückweise lineare Basisfunktion ψ_H zur Triangulierung der Seitenlänge $H = 2h$.

Abbildung 1.14: **Verfeinerte Triangulierung**



Eine Basisfunktion ψ_H kann durch Wahl der Funktionswerten in den Eckpunkte der Dreiecke beschrieben werden. Im Falle von ψ_H können wir annehmen, dass es einen festen Punkt $(x_0, y_0) = (iH, jH)$ gibt an dem ψ_H gerade den Wert 1 hat und sonst überall 0 ist. D.h. wir haben

$$\psi_H(x,y) = \begin{cases} 1 & \text{falls } (x,y) = (x_0, y_0) \\ 0 & \text{falls } |x - x_0| = lH \text{ oder } |y - y_0| = mH, \ l, m = 1, 2, 3, \ldots \end{cases}$$

in den Eckpunkten und ψ_H linear auf jedem Dreieck. Die benachbarten Basisfunktionen auf dem feineren Gitter erfüllen analoge Bedingungen:

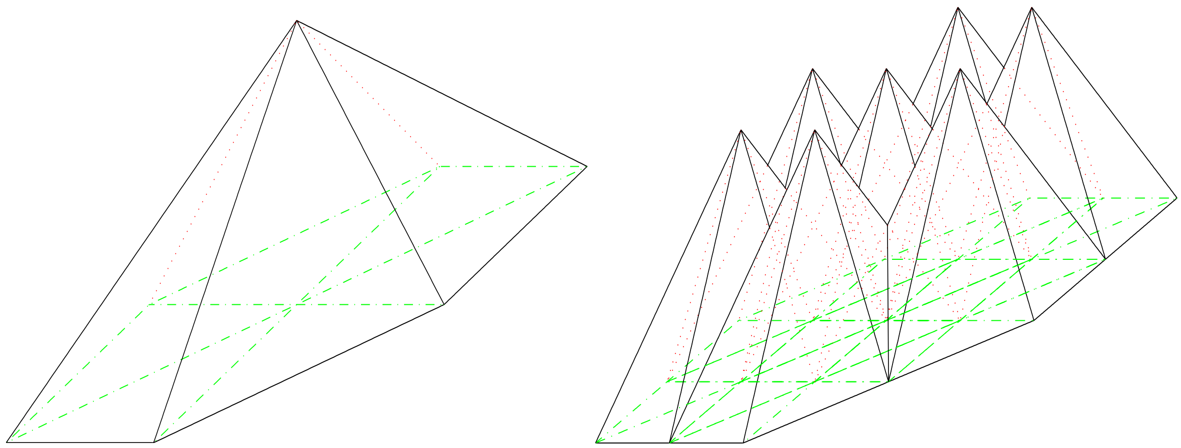
$$\psi_h^{(k)}(x,y) = \begin{cases} 1 & \text{falls } (x,y) = (x_0^{(k)}, y_0^{(k)}) \\ 0 & \text{falls } |x - x_0| = lh \text{ oder } |y - y_0| = mh, \ l, m = 1, 2, 3, \ldots \end{cases}$$

Dabei sind die Punkte $(x_0^{(k)}, y_0^{(k)})$ gerade (x_0, y_0) und dessen Nachbarpunkte auf dem feinen Gitter

$$(x_0^{(k)}, y_0^{(k)}) \in \{(x_0, y_0), (x_0 \pm h, y_0), (x_0, y_0 \pm h), (x_0 + h, y_0 + h), (x_0 - h, y_0 - h)\}.$$

In Abbildung 1.15 sind ψ_H und $\psi_h^{(k)}$ geometrisch dargestellt.

Abbildung 1.15: **Basisfunktionen grobes Gitter, feines Gitter**

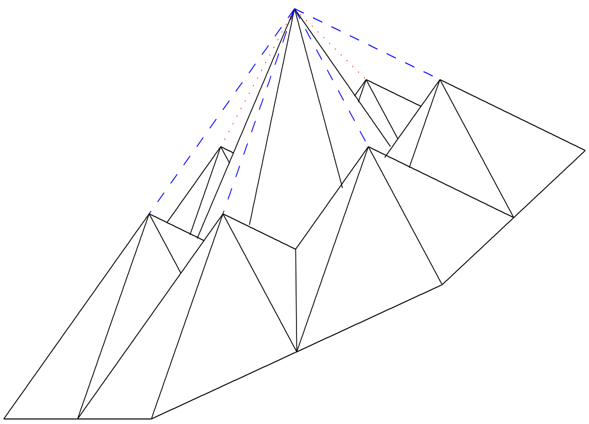


Ist $\psi_h^{(0)}$ die Basisfunktion mit gleichem Mittelpunkt (x_0, y_0) wie ψ_H, so können wir sehr leicht ψ_H als Linearkombination der $\psi_h^{(k)}$ darstellen. Es gilt:

$$\psi_H = \psi_h^{(0)} + \frac{1}{2} \sum_{k \neq 0} \psi_h^{(k)}. \tag{1.58}$$

Dabei sind die verbleibenden (sechs) Basisfunktionen $\psi_h^{(k)}$, $k \neq 0$ jeweils nur mit $\frac{1}{2}$ gewichtet. Beziehung (1.58) ist sehr anschaulich in Abbildung 1.16 dargestellt.

Abbildung 1.16: **Basisfkt. grobes Gitter bzgl. der Basisfktn. des feinen Gitters**



Mit Hilfe von (1.58) erhalten wir auf natürliche Weise den Prolongationsoperator P_s und als Transponierte die Restriktion.

$$P_s : \; S_{h_{s-1}} \to S_{h_s}, \; R_s = P_s^\top, s = 2, \ldots, l. \tag{1.59}$$

Dass die Restriktion genau die Transponierte der Interpolation ist, kann man am besten dadurch verstehen, dass per Konstruktion der finiten Element–Räume und ihrer Basen gilt:

$$A_{h_{s-1}} = P_s^\top A_{h_s} P_s, \; s = 2, \ldots, l. \tag{1.60}$$

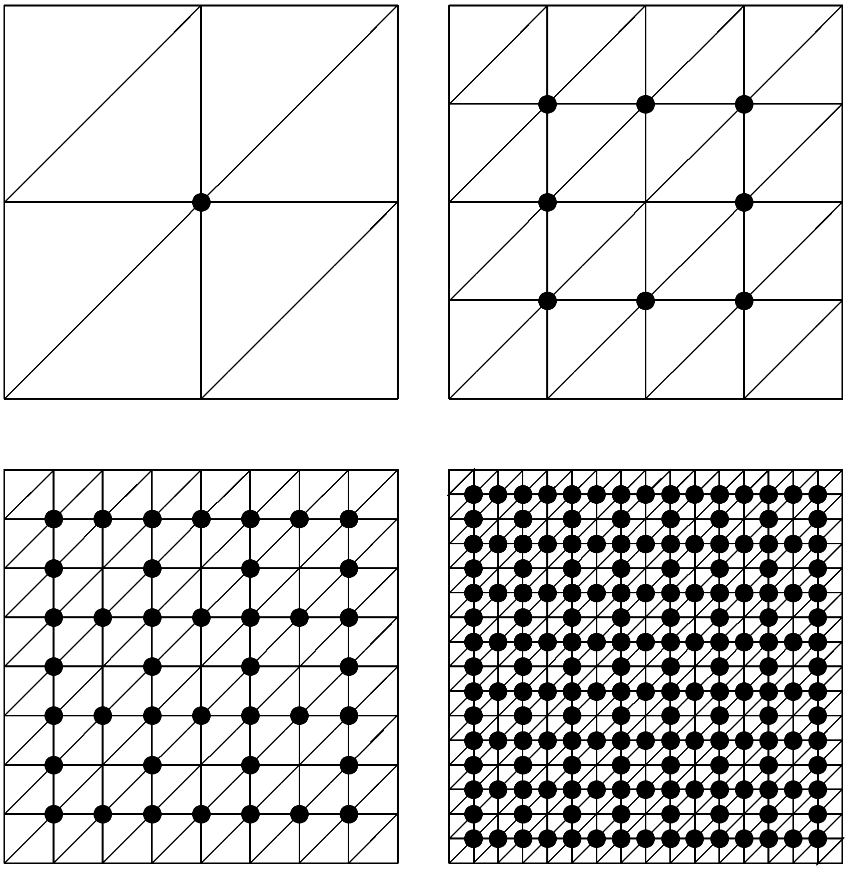
Damit ist die exakte Grobgitterkorrektur $I - P_s A_{h_{s-1}} P_s^\top A_{h_s}$ ein orthogonaler Projektor im Sinne des durch A_{h_s} definierten inneren Produktes.

Nachdem wir gesehen haben, dass bei den Finite–Elemente–Methoden unter Verwendung der sogenannten nodalen Basis (lokaler Träger) auf natürliche Weise Restriktions– und Prolongationsoperatoren entstehen, lässt sich das Mehrgitterverfahren natürlich sofort übertragen. Wir verwenden einfach Algorithmus 2 mit den für die FEM natürlichen P_s, R_s aus (1.59) und den entsprechenden Matrizen A_{h_s}, die aus der Diskretisierung hervorgehen. Insbesondere sind wir nicht mehr an Rechteckgitter gebunden, sondern die Interpolation passt sich der natürlichen gebietsabhängigen Einbettung des gröberen Raumes S_H in den feineren Raum S_h an.

1.8.4 Weitere mehrgitterartige Verfahren

Vor einigen Jahren sind in der numerischen Behandlung der FEM Methoden vorgestellt worden, die ebenfalls die Hierarchie der Räume S_H, S_h beutzen, jedoch etwas anders aufgebaut sind. Der erste Verfahren dieser Art ist die sogenannte Hierarchische Basis [37]. Dabei wird vereinfacht gesagt ein Basiswechsel von der nodalen Basis auf dem feinsten Gitter hin zu einer Hierarchischen Basis durchgeführt, bei der zunächst die Basis des gröbsten Gitters verwendet wird. Diese wird dann ergänzt durch Basisfunktionen des nächst feineren Gitters, um so eine Basis des zweitgröbsten Raumes zu bekommen. Dieses Verfahren wird von Raum zu Raum fortgesetzt (siehe Abbildung 1.17). Man kann zeigen, dass zumindest in zwei Raumdimensionen die so transformierte Matrix lediglich mit einer Diagonalskalierung versehen werden muss, damit man bei Anwendung des cg–Verfahrens eine von h fast unabhängige (logarithmisch) Zahl von Iterationen zu bekommen. Die Matrix wird in dieser Basisdarstellung üblicher Weise nicht gebildet, weil die Martix deutlich mehr Nichtnulleinträge hat als bei der nodalen Basis.

Abbildung 1.17: **Hierarchische Basis**



In drei Raumdimensionen erweist sich dieses Verfahren jedoch als unbefriedigend. Der Ersatz hierfür besteht in der Verwendung eines Erzeugendensystems, wo man die Basis nicht einfach nur ergänzt, sondern durch erheblich mehr Funktionen der jeweils feineren Gitter ergänzt. Auf die Details dieses Verfahrens soll hier verzichtet werden und auf die einschlägige Literatur verwiesen werden, die üblicher Weise unter dem Stichwort BPX (nach den Anfangsbuchstaben der Autoren) [9] zu finden ist.

Weitere Techniken die zur Klasse der geometrischen Mehrgitterverfahren zählen, sind die in [2, 3] vorgestellte Klasse der AMLI–Verfahren. Die Bezeichnung algebraisch bezieht sich dabei nicht auf eine von der Geometrie losgelöste Herangehensweise, sondern auf die verwendeten Techniken wie etwa approximative Schur–Komplementbildung und die damit verbundenen Beweistechniken.

Ebenfalls Techniken der unvollständigen Dreieckszerlegung aber wiederum bei bekannter Geometrie und Hierarchie verwenden Arbeiten in [26, 25, 27]. Diese sind speziell als robuste Mehrgitterverfahren für die Konvexions–Diffusionsproblematik konstruiert.

Kapitel 2

Algebraische Mehrgitterverfahren

In diesem Kapitel werden wir uns mit algebraischen Mehrgitterverfahren (AMG) beschäftigen. Hier gibt es eine Reihe von Methoden die sehr viel Verwandtschaft zu den in Kapitel 1 vorgestellten geometrischen Mehrgitterverfahren haben. Wir werden dazu als erstes motivieren warum und wo man algebraische anstelle von geometrischen Techniken verwendet. Dann werden wir das klassische AMG von Ruge und Stüben betrachten. Daran anschließend werden wir einige andere algebraische und semialgebraische Techniken vorstellen.

2.1 Was sind und wozu braucht man algebraische Techniken

Es gibt eine Reihe von Anwendungen, bei denen es nicht möglich ist, geometrische Mehrgitterverfahren anzuwenden, obwohl die zugrunde liegenden Probleme aus verwandten Bereichen stammen. Dies kann z.B. sein, wenn das Problem, das man lösen will mit einem FEM–Paket auf dem Computer generiert worden ist, dabei aber nur ein Gitter vorhanden ist. Dies ist in vielen Ingenieuranwendungen der Fall. Andere Problem, wie etwa Chip–Design [24], also das Plazieren von gewissen Gattern oder Addieren usw. auf einem Chip beruhen gar nicht auf einer partiellen Differentialgleichung, sondern stellen ein Optimierungsproblem dar. Trotzdem hat man natürlich so etwas wie ein Gebiet (nämlich den Chip) und kann Gleichungen aufstellen, die denen aus den partiellen Differentialgleichungen sehr ähnlich sind. Desweiteren gibt es Probleme mit den geometrischen Mehrgitterverfahren bei einigen Problemen, die nur entfernt mit den in Kapitel 1 behandelten Problemen verwandt sind. Dort ist erheblich mehr Anpassung des geometrischen Mehrgitterverfahrens erforderlich um es ähnlich effizient zu machen. Dies erfordert eine tiefgreifende Analyse und manchmal stehen angepasste effiziente geometrische Mehrgitterverfahren nicht zur Verfügung. Ein weiterer Aspekt für die Nutzung algebraischer Techniken ist natürlich die erheblich einfacheren Datenstrukturen. Die Netze der jeweiligen Diskretisierung mit all ihren Feinheiten und ihrer Administration werden nicht unbedingt benötigt. Bei großen Problemen kann dies entscheidend sein wenn der Hauptspeicher das Problem nicht mehr fasst oder der Zugriff auf die Datenstrukturen zu teuer wird.

Die Idee bei algebraischen Techniken ist, lediglich die Matrix des zugrunde liegenden Gleichungssystems als Information zu verwenden und daraus ein mehrgitterähnliches Verfahren zu stricken, dass das geometrische Mehrgitterverfahren nachahmt, ohne dabei aber den deterministischen Hintergrund zu haben. So wird man bei algebraischen Mehrgitterverfahren wieder den Apparat von Techniken wie Glättung und Grobgitterkorrektur verwenden. Hier jedoch wird der Glätter fixiert sein und Restriktion und Prolongation aus den Koeffizienten der Matrix gewonnen. Damit wird dann ein Grobgittersystem induziert auf welchem man die analoge Technik wieder anwendet. Alles in allem werden damit bei einem gegebenen feinsten virtuellen Gitter künstlich gröbere Gitter erzeugt statt anders herum wie bei den geometrischen Mehrgitterverfahren.

2.2 Konzepte für algebraische Mehrgitterverfahren

In diesem Abschnitt werden wir grob zwei gängige Konzepte vorstellen, die als Grundlage für die hier vorgestellten Mehrgitterverfahren dienen. Das erste Konzept ist direkt an die bereits vorgestellten Konzepte für geometrische Mehrgitterverfahren angelehnt. Das zweite Konzept beruht auf Blockeliminationstechniken mit deren Hilfe man ebenfalls so etwas wie ein Mehrgitterverfahren definieren kann.

2.2.1 Mehrgitterverfahren mit algebraischer Restriktion/Prolongation

Um Algorithmus 2 formal anwenden zu können, brauchen wir lediglich eine Folge R_S, P_s von Restriktions– und Prolongationsoperatoren sowie die zugehörigen Grobgittersysteme A_{h_s} und und zugehörige Glätter. Im einfachsten Falle kann man als Glätter natürlich ein Verfahren wie das Gauß–Seidel–Verfahren oder das Jacobi–Verfahren verwenden. Die Grobgittersysteme kann man direkt mit Hilfe von Prolongation und Restriktion definieren. Ist etwa $A_1 = A$ ein gegebenes System, und R_1, P_1 (irgendwie definierte) Restriktions– und Prolongationsoperatoren, so ist die sogenannte Galerkin–Matrix

$$A_2 = R_1 A_1 P_1 \tag{2.1}$$

eine natürliches Grobgittersystem. Übrigens gilt (2.1) bei den in Kapitel 1 behandelten Mehrgitterverfahren und zwar nicht nur bei der FEM. Durch sukzessive Anwendung bekommen wir eine (künstliche) Hierarchie von Systemen $A_1, A_2, \ldots, A_l$, die im Gegensatz zu Kapitel 1 üblicher Weise anders herum nummeriert sind, weil man vorab noch nicht wissen kann, wieviele Vergröberungsschritte denn durchgeführt werden. Grob schematisch läuft das AMG wie folgt ab.

Algorithmus 5 (Schema algebraisches Mehrgitterverfahren)
Gegeben: $A = A_1$ *Ausgangssystem, eine zugehörige rechte Seite* b*, eine Näherungslösung* x.

Teil 1: Generierung der Operatoren

Bilde für $k = 1, 2, \ldots$ *sukzessive*
einen Glätter S_k *für* A_k
Restriktions–/Prolongationsoperatoren R_k, P_k *zu* A_k
$A_{k+1} = R_k A_k P_k$

Teil 2: Anwendung als Mehrgitterverfahren

Sind etwa l *Systeme* $A_1, \ldots, A_l$ *mit Glättern* $S_1, \ldots, S_{l-1}$ *Restriktionen/Prolongationen* $R_1, \ldots, R_{l-1}$, $P_1, \ldots, P_{l-1}$ *definiert, dann wende Algorithmus 2 für diese Konfiguration an (mit den Systemen, die in umgekehrter Reihenfolge nummeriert sind).*

Als Gätter kann man z.B. fest den Gauß–Seidel–Glätter verwenden. Dann bleibt als einzige Aufgabe die Definition von Restriktion und Prolongation. Hier in diesem Zusammenhang werden wir $R_s = P_s^\top$ wählen. Das klassische Verfahren hierzu stammt von Ruge und Stüben [30]. Mit diesem werden wir uns im Abschnitt 2.3 eingehend beschäftigen.

2.2.2 AMG basierend auf Blockeliminationstechniken

Wir betrachten in diesem Abschnitt ein Konzept zu algebraischen Merhgitterverfahren, welches auf Blockeliminationstechniken beim Gaußschen Algorithmus beruhen. Sei dazu die Ausgangsmatrix A wie folgt partitioniert:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \tag{2.2}$$

mit quadratischen Diagonalblöcken A_{11}, A_{22}. Wir können durch einen Schritt Blockelimination A faktorisieren als

$$A = \begin{pmatrix} I & O \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & O \\ O & S_{22} \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}A_{12} \\ O & I \end{pmatrix},$$

wobei $S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ das Schur–Komplement ist. Eine solche Partitionierung kann z.B. die Einteilung des Gitters in Grobgitterpunkte und deren Rest (Feingitterpunkte) sein. Natürlich beabsichtigt man jetzt nicht wirklich diese ganze Inversen zu bilden. Ist allerdings A_{11}^{-1} leicht zu bekommen bzw. ein Gleichungssystem mit A_{11} leicht zu lösen, ist eine solche Zerlegung nicht zu teuer. Es kann höchstens passieren, dass S_{22} voll besetzt ist, was man natürlich auch nicht will.

Betrachten wir mal was passiert, wenn wir mit Hilfe dieser Zerlegung ein Gleichungssystem lösen wollen (alles unter der Annahme, dass wir zumindest mit A_{11} leicht ein Gleichungssystem lösen können.

Bilde $L = -A_{21}A_{11}^{-1}, U = -A_{11}^{-1}A_{12}$.

Setze

$$P = \begin{pmatrix} U \\ I \end{pmatrix}, \; R = \begin{pmatrix} L & I \end{pmatrix}.$$

Dann gilt

$$S_{22} = RAP \tag{2.3}$$

und

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} & O \\ O & O \end{pmatrix} + PS_{22}^{-1}R. \tag{2.4}$$

Diese Darstellung erlaubt uns, nachdem P, R, S_{22} gebildet sind, das Lösen des Gleichungssystems $Ax = b$ zurückzuführen auf das Lösen eines Gleichungssystemes mit S_{22}. Für S_{22} kann man natürlich dieselbe Technik wiederverwenden.

Sei also $x = \binom{x_1}{x_2}, \binom{b_1}{b_2}$. Es ist dann $Ax = b$ äquivalent zu folgendem groben Algorithmus.
Löse $A_{11}u = b_1$
bilde $v = Rb$
löse $S_{22}w = v$
bilde $y = Pw$.
Dann ist $x = \binom{u}{0} + y$.

Um hieraus ein iteratives Verfahren zu machen, bildet man natürlich L, U nicht explizit, sondern man approximiert diese. Ausserdem muss man beachten, dass die exakten L, U voll sein könnten. Auch das vermeidet man natürlich in der Praxis. Wir erhalten also

$$P = \begin{pmatrix} \tilde{U} \\ I \end{pmatrix}, \; R = \begin{pmatrix} \tilde{L} & I \end{pmatrix}, \text{ wobei } \tilde{L} \approx -A_{21}A_{11}^{-1}, \tilde{U} \approx -A_{11}^{-1}A_{12}. \tag{2.5}$$

Die dabei auftretenden Fehler kann man natürlich dadurch versuchen abzufangen, dass man das Verfahren in iteratives Verfahren einbettet.

Beispielsweise könnte man (2.4) in ein iteratives Verfahren einkleiden:

$x^{(k+1)} = x^{(k)} + \tilde{A}^{-1}(b - Ax^{(k)})$. Dabei ist $\tilde{A}^{-1}$ mit A^{-1} in (2.4) identisch bis auf die Nutzung von $\tilde{L}, \tilde{U}$ anstelle von L, U.

Algorithmus 6 (AMG beruhend auf Blockelimination)
Gegeben: $A = A_1$ *Ausgangssystem, eine zugehörige rechte Seite* b*, eine Näherungslösung* x.

Teil 1: Generierung der Operatoren

Bilde für $k = 1, 2, \ldots$ *sukzessive*
Partitioniere A_k *entsprechend (2.2) analog.*
Berechne P, R *aus (2.5) mit* A *ersetzt durch* A_k
$A_{k+1} = R_k A_k P_k$ *(analog zu (2.3)).*

Teil 2: Anwendung als Mehrgitterverfahren

Seien etwa l *Systeme* $A_1, \ldots, A_l$ *mit Restriktionen/Prolongationen* $R_1, \ldots, R_{l-1}$, $P_1, \ldots, P_{l-1}$ *konstruiert.*
Weiterhin: eine zugehörige rechte Seite $b = b_1$*, eine Näherungslösung* $x = x_1$*,* μ *sei die Anzahl Grobgitterkorrekturschritte.*

Der folgende Algorithmus berechnet rekursiv eine neue Näherung y_s*, angefangen mit* $s = 1$.

Falls $s = l$, *löse* $A_l y_l = b_l$ *direkt.*
Sonst $y_s := b_s - A_s x_s$
Grobgitterkorrektur
$x_{s+1} := 0$, $b_{s+1} := R_s y_s$
Für $k = 1, 2, \ldots, \mu$
Rufe Algorithmus 6, Teil 2 auf mit b_{s+1}*,* x_{s+1}*. Verwende das Ergebnis* y_{s+1} *und setze* $x_{s+1} = y_{s+1}$.
Nachglättung
Verwende das Ergebnis y_{s+1} *um schließlich* y_s *zu definieren.*
$y_s := x_s + \begin{pmatrix} A_{11,s}^{-1} & O \\ O & O \end{pmatrix} y_s + P_s y_{s+1}$.

Man kann natürlich auch alternativ die in Teil 1 gewonnenen Systeme samt Restriktionen und Prolongationen schlicht in ein Mehrgitterverfahren einsetzen und den Glätter jeweils durch $\begin{pmatrix} A_{11,s}^{-1} & O \\ O & O \end{pmatrix}$ ersetzen. Letzteres zeigt natürlich am deutlichsten die Verwandtschaft zu Mehrgitterverfahren. Diese zweite Klasse von Mehrgitterverfahren hat einen Vorteil. Die Restriktion/Prolongation ergibt sich kanonisch, sobald man die Ausgangsmatrix partitioniert hat. Genauso ist der Glätter aus dem Ansatz her festgelegt. Es bleibt natürlich das Problem geeignete Partionierungen zu finden.

2.3 Das AMG von Ruge und Stüben

Im folgenden werden wir uns mit dem klassischen AMG [30] beschäftigen. Die Grundidee hierbei ist, die Operationen wie Restriktion und Prolongation aus den Matrixkoeffizienten zu gewinnen. Zu diesem Zweck nehmen wir in diesem Abschnitt an, dass die Matrix ein festes Vorzeichenmuster habe. Sei also $A = (a_{ij})_{i,j=1,\dots,n}$ die Matrix für die wir das Gleichungssystem $Ax = b$ lösen möchten. Nehme an, dass $a_{ii} > 0$, für alle $i = 1,\dots,n$ ist und $a_{ij} \leqslant 0$ für alle $i \neq j$. Sollte die Matrix A einige wenige Einträge ausserhalb der Diagonalen haben die positiv sind so werden diese zur Konstruktion des AMG einfach ignoriert, d.h. wie 0 behandelt. Für Probleme die diese Bedingungen nicht erfüllen, ist das nachfolgende AMG nicht anwendbar. Desweiteren nehmen wir an, dass ein uns bekannter elementweise positiver Vektor $w \in \mathbb{R}^n$ existiere, so dass Aw elementweise nichtnegativ ist. Die Voraussetzungen hier sind (nahezu) äquivalent zu der Eigenschaft, dass A eine M–Matrix sei (Siehe z.B. [6]). Der Einfachheit sei w der Vektor mit lauter Einsen. Ansonsten verwenden wir die Matrix $\hat{A} = A \operatorname{diag}(w)$ anstelle von A. Diese Bedingungen sind sicherlich recht einschränkend, treten aber in der Praxis bei sehr vielen Problemen auf. Beispielsweise elliptische partielle Differentialgleichungen mit FEM und linearen Ansatzfunktionen, sofern die Innenwinkel der Dreiecke nicht mehr als rechte Winkel ($\pi/2$ in Bogenmaß) sind. Dies trifft z.B. auf die in Kapitel 1 betrachteten Probleme zu. Davon abgesehen gibt es einige weitere Probleme, die solche Gleichungssysteme liefern. Wie bei geometrischen Mehrgitterverfahren so benutzen wir auch hier Glättung und Grobgitterkorrektur, wobei diese Begriffe hier jetzt nur umgangssprachlich zu verstehen sind und nichts mehr mit gedämpften hochfrequenten Schwingungen zu tun haben müssen.

2.3.1 Die Grundidee des Ansatzes

Die Zugang zum AMG erfolgt über den folgenden Ansatz. Man versucht man diejenigen Fehler e mit Hilfe von Restriktion und Prolongation zu nähern, für die gilt:

$$(Ae)_i \approx 0, i = 1,\dots,n. \tag{2.6}$$

Fehler e mit Eigenschaft (2.6) werden wir im folgenden als “glatt” bezeichnen. Unter der Voraussetzung (2.6) erhalten wir, dass

$$e_i \approx \sum_{j:\ a_{ij}<0} \frac{|a_{ij}|}{a_{ii}} e_j, \tag{2.7}$$

wobei wegen $Aw \geqslant 0$, $w = (1,\dots,1)^\top$ ausserdem gilt

$$\sum_{j:\ a_{ij}<0} \frac{|a_{ij}|}{a_{ii}} \leqslant 1. \tag{2.8}$$

Genauer noch sind diejenigen Zeilen von A eigentlich uninteressant, für die $Aw \gg 0$ gilt. Solche Zeilen könnten durch Sonderbehandlung eigentlich vernachlässigt werden, weil in diesen Zeilen die Matrix strikt diagonal dominant ist. D.h. der Anteil der Matrix ist leicht invertierbar. Wir kommen darauf später noch mal zurück. Wenn wir uns also konzentrieren auf diejenigen Zeilen, für die $Aw \approx 0$ ist, dann bedeutet (2.8), dass sich glatte Fehler (nahezu) als Konvexkombination ihrer Nachbarkomponenten (im Sinne des durch A erzeugten Graphen) darstellen lassen. Klar sollte auch sein, dass eigentlich nur Gewichte $\frac{|a_{ij}|}{a_{ii}}$ von Interesse sind, die nicht vernachlässigbar klein sind. Anders gesagt, wir können e_i mit Hilfe seiner sogenannten “starken Nachbarn” und der zugehörigen (großen) Gewichte $\frac{|a_{ij}|}{a_{ii}}$ interpolieren. Wir definieren deshalb jetzt starke und schwache Nachbarn. Sei zu diesem Zweck $\omega \in (0,1]$ eine festgewählte Konstante (z.B. $\omega = \frac{1}{4}$, was bei [30] in der Praxis gewählt worden ist). Die Menge

$$\mathcal{S}_i = \{j \neq i : \ |a_{ij}| \geqslant \omega \max_{k \neq i} |a_{ik}|\} \tag{2.9}$$

heisst Menge der starken Nachbarn von i und beschreibe diejenigen j für die $\frac{|a_{ij}|}{a_{ii}}$ hinreichend groß ist. Die restlichen j für die $a_{ij} < 0$ ist, heissen schwache Nachbarn. Die Bezeichnung starker Nachbar ist übrigens nicht reflexiv. D.h. $j \in \mathcal{S}_i \not\Rightarrow i \in \mathcal{S}_j$. Mit anderen Worten j kann ein starker Nachbar von i sein, ohne dass das andersherum auch richtig sein muss. Man unterscheidet eigentlich aus recht praktischen Gründen zwischen starken und schwachen Nachbarn. Bei einer Reihe von Anwendungen kann es durchaus vorkommen, dass in einigen Zeilen recht viele Nichtnullelemente auftreten. Das will man natürlich aus Effizienzgründen vermeiden. (2.7) zeigt, dass sich “glatte” Fehler nur geringfügig ändern in Richtung der starken Nachbarn. Dies ist im gewissen Sinne die algebraische Analogie zu glatten Fehlern beim geometrischen Mehrgitterverfahren.

Zur Erinnerung: Bei geometrischen Mehrgitterverfahren haben wir zunächst oszillierende Anteile (dort wirklich hochfrequent im geometrischen Sinne) mit Hilfe der Glättung gedämpft. Was geblieben ist, sind die glatten Frequenzen. Bei Restriktion und Interpolation zwischen den Gittern sind das aber genau diejenigen Frequenzen die sich mit Hilfe des groben Gitters ohne großen Qualitätsverlust leicht durch das gröbere Gitter interpolieren lassen. Wir betrachten zur Illustration nochmal das Problem (1.1) und als Interpolation die in (1.24) betrachtete Interpolation. Abbildung 2.1 zeigt, dass der Fehler zwischen den interpolierten Frequenzen $s_H^{(k)}$ des groben Gitters aus (1.19) und den niedrige Frequenzen $s_h^{(k)}$ für $k < N/2$ gering ist während der Fehler recht beachtlich ist für hochfrequente Anteile. Dies ist ein wesentlicher Punkt bei der Konvergenzanalyse In Kapitel 1.7 gewesen. In Abbildung 2.1 werden Grobgitterpunkte rot interpoliert, Feingitterpunkte grün. Als Referenz sind die kontinuierlichen Eigenfunktionen in blau abgetragen.

Nehmen wir uns der Reihe nach diejenigen Zeilen, bei denen $Aw \approx 0$ ist vor, dann können wir eigentlich jedesmal wenn (2.7) erfüllt ist, diesen Entrag i weglassen, da er sich ja leicht durch Konvexkombination seiner Nachbarpunkte ersetzen lässt. Ein solcher Knoten i wird für ein späteres grobes Gitter nicht unbedingt benötigt. Im Gegensatz dazu müssen wir natürlich dann dafür sorgen, dass die starken Nachbarpunkte nicht weggelassen werden. Auf diese Weise werden eine Reihe von Punkten herausgenommen. Ziel ist es natürlich, möglichst viele Punkte herauszunehmen. Solche Punkte heissen dann Feingitterpunkte und die restlichen würden automatisch zu Grobgitterpunkten werden.

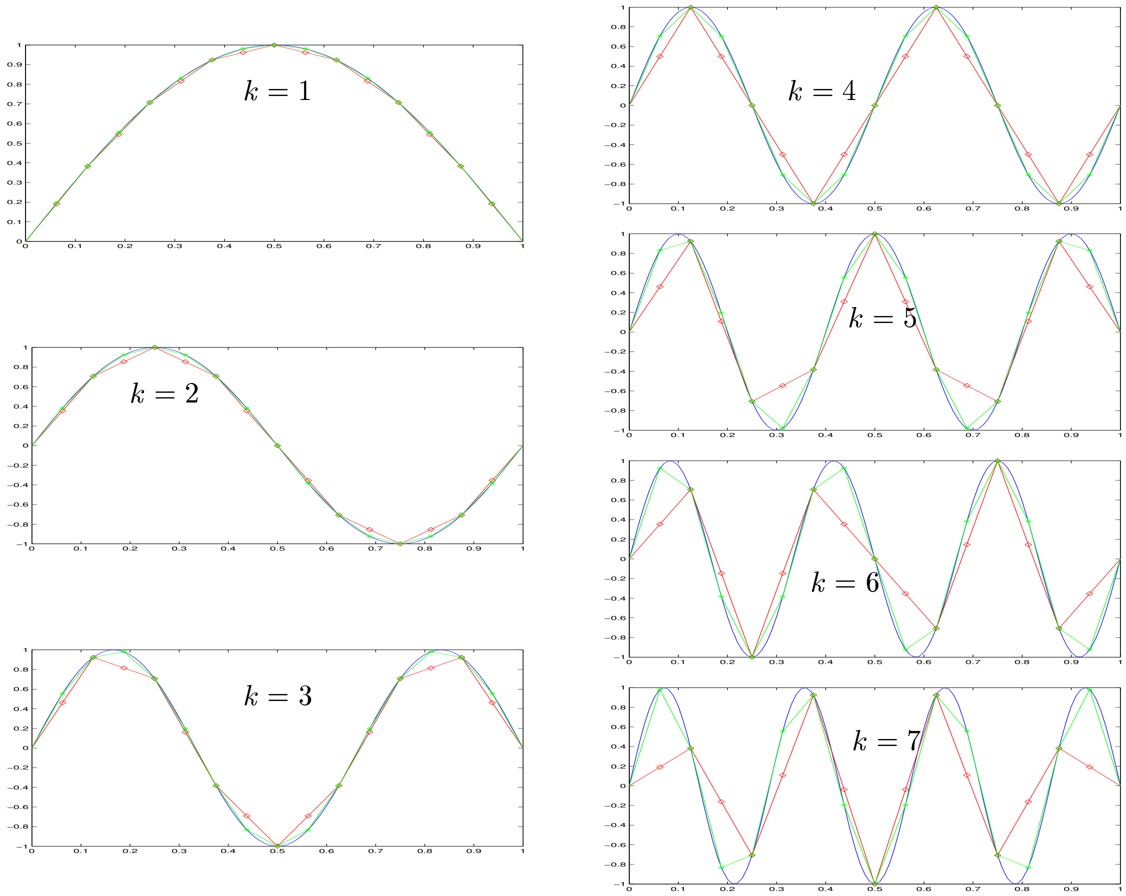
2.3.2 Der Algorithmus an sich, 2 Durchläufe

Einfach naiv eine Zeile zu nehmen, diese herauszunehmen und deren starken Nachbarn dafür (im Grobgitter) zu behalten, könnte sehr leicht dazu führen, dass die Ausbeute der Knoten die man weglässt ziemlich gering ist. Aus diesem Grunde führt man die Menge $\mathcal{S}_i^\top$ ein.

$$\mathcal{S}_i^\top = \{j : \ i \in S_j\}. \tag{2.10}$$

Diese zu $\mathcal{S}_i$ adjungierte Menge $\mathcal{S}_i^\top$ beschreibt somit nicht die starken Nachbarn von i, sondern die Punkte j für die i selbst ein starker Nachbar ist. Wie bereits erwähnt ist die Eigenschaft starker Nachbar zu sein nicht reflexiv. Ein i mit $\#\mathcal{S}_i^\top$ bedeutet, dass i bei sehr vielen j zur

Abbildung 2.1: **Niedrige Frequenzen feines Gitter, interpoliert durch Grobgitterfrequenzen**



Interpolation benötigt wird. Ein i mit sehr großem $\#\mathcal{S}_i^\top$ sollte also möglichst bleiben und damit in das sogenannte Grobgitter wandern. Wählen wir also einen solchen Knoten i für das grobe Gitter aus, so müssen wir natürlich anschließend für seine starken Nachbarn $j \in \mathcal{S}_i$ die Mengen $\mathcal{S}_j^\top$ korrigieren, weil i natürlich nicht länger interpoliert werden muss. Wenn wir jetzt als nächstes wieder einen Knoten mit maximalem $\#\mathcal{S}_j^\top$ nehmen würden (solange bis für alle restlichen j, S_j nur aus Grobgitterpunkten besteht), dann bestünde die Gefahr wieder zu viele Grobgitterpunkte auszuwählen. Darum nimmt man jetzt im Gegenzug die verbleibenden Elemente aus $j \in \mathcal{S}_i^\top$ heraus und packt sie ins feine Gitter. Dies ist ein riskanter Schritt, weil man ja nicht weiss, ob diese Punkte gut interpolierbar sind nur weil i ein starker Nachbar von j ist. Also nimmt man jetzt die Punkte $k \in \mathcal{S}_j$ und fügt j der Menge $\mathcal{S}_k^\top$ hinzu. Dieser so entstandene Algorithmus ist erst der erste Teil des AMG, dem anschließend ein zweiter Teil folgt. Wir fassen ihn in einem Algorithmus zusammen.

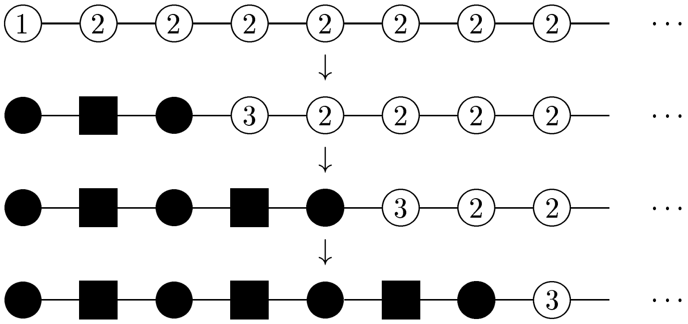
Algorithmus 7 (Vorläufige Wahl der Grobgitterpunkte) *$\mathcal{C}$ bezeichne die Menge der Grobgitterpunkte, $\mathcal{F}$ die verbleibenden Feingitterpunkte sowie $\mathcal{U}$ die Menge der unbestimmten Knoten. Ausserdem sei $\lambda_j = \#\mathcal{S}_j^\top$, $j = 1, \dots, n$.*

Zu Beginn setze $\mathcal{C} = \mathcal{F} = \emptyset$, $\mathcal{U} = \{1, \dots, n\}$.

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 $j \in \mathcal{S}_i^\top \cap \mathcal{U}$:
        % Die Nachbarn, für die i ein 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 Gewicht bei starken Nachbarn, weil i jetzt fehlt ($j \in \mathcal{S}_i \Leftrightarrow i \in \mathcal{S}_j^\top$)
    Für alle $j \in \mathcal{S}_i \cap \mathcal{U}$: $\lambda_j = \lambda_j - 1$

Wir demonstrieren die Wirkungsweise von Algorithmus 7 an einigen Beispielen. Als Parameter ω für die Definition der starken Nachbarn nehmen wir $\omega = \frac{1}{4}$. Im folgenden sind für unbestimmte i die Werte λ_i angezeigt. Grobgitterpunkte bekommen ausgefüllte Rechtecke, Feingitterpunkte ausgefüllte Kreise.

Beispiel 8 *Wir verwenden Matrix T_h aus (1.11) in Kapitel 1. Wir erhalten als Mengen $\mathcal{S}_1 = \{2\}$, $S_i = \{i-1, i+1\}$, $i = 2, \dots, n-1$, $\mathcal{S}_n = \{n-1\}$. Umgekehrt sind $\mathcal{S}_i^\top = \mathcal{S}_i$ für alle $i = 1, \dots, n$. Der Algorithmus startet mit einem i mit maximalem λ_i, z.B. $i = 2$. Wir illustrieren einige Schritte des Algorithmus graphisch anhand des ungerichteten Graphen von T_h.*



Als zweites Beispiel verwenden wir im Prinzip eine ähnliche Gleichung allerdings jetzt mit einem Koeffizientensprung.

Beispiel 9

$$-(au')' = f \ \ in\ [0,1].$$

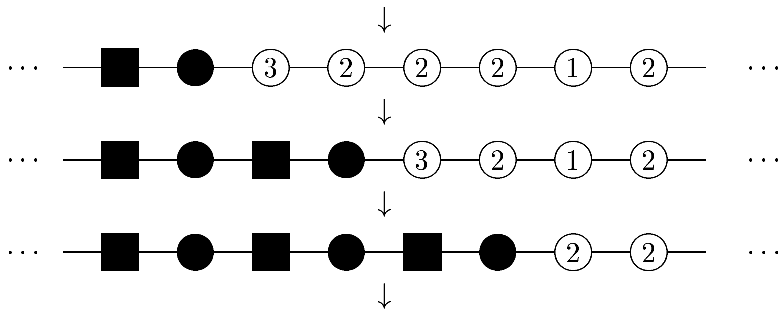
Dabei sei

$$a(x) = \begin{cases} 10 & x \leqslant 0.5 \\ 1 & x > 0.5 \end{cases}$$

Die entsprechende Matrix der Diskretisierung sieht fast genauso aus wie T_h. Sei $M = \lceil N/2 \rceil$. In Zeile M steht die Beziehung

$$-10u_{M-1} + 11u_M - u_M = f_M$$

und bis Zeile $M-1$ werden alle Zeilen von T_h mit 10 multipliziert. Die Menge $\mathcal{S}_M$, $\mathcal{S}_M^\top$ sind fast identisch mit denen aus Beispiel 8. Lediglich für $i = M, M+1$, ändert sich etwas. So ist $\mathcal{S}_{M-1} = \{M-2, M\}$, $\mathcal{S}_M = \{M-1\}$, $\mathcal{S}_{M+1} = \{M, M+2\}$. Folglich sind $\mathcal{S}_{M-1}^\top = \{M-2, M\}$, $\mathcal{S}_M^\top = \{M-1, M+1\}$, $\mathcal{S}_{M+1}^\top = \{M+2\}$. Fangen wir mit dem Knoten $i = 3$ an, so bietet sich zunächst das gleiche Bild wie in Beispiel 8. Beim Erreichen der Sprungstelle jedoch verhält sich der Algorithmus anders.



Wir sehen, dass Knoten $M+1$ keine höhere Priorität mehr hat als die anderen. Je nachdem wie der Algorithmus den nächsten Knoten auswählt, könnte der Vergröberungsprozess irgendwo weitergehen.

Als nächstes betrachten wir ein zweidimensionales Problem vom Typ (1.5) mit springenden Koeffizienten.

Beispiel 10

$$-\operatorname{div}\,(a \operatorname{grad}\, u) = f \;\; in \; [0,1]^2.$$

Dabei sei

$$a(x,y) = \begin{cases} 100 & x,y \leqslant 0.5 \\ 10 & x > 0.5, y \leqslant 0.5 \textit{ oder } x \leqslant 0.5, y > 0.5 \\ 1 & x,y > 0.5 \end{cases}$$

10	1
100	10

Die entsprechende Matrix bei Verwendung einer zu (1.13) analogen Diskretisierung sieht fast genauso aus wie A_h aus (1.15). Sei $M = \lceil N/2 \rceil$. Die zu den Gitterpunkten (ih, jh) mit $i,j < M$ gehörigen Zeilen werden mit 100, die zu (ih,jh) mit $i < M$, $j > M$ oder umgekehrt gehörenden Zeilen werden mit 10 multipliziert. Die verbleibenden interessanten Fälle sind die für $i = M$ oder $j = M$.

$$\begin{array}{ccccc} & & -10 & & \\ & & | & & \\ -100 & \text{---} & 310 & \text{---} & -100 \\ & & | & & \\ & & -100 & & \end{array}
\qquad
\begin{array}{ccccc} & & -100 & & \\ & & | & & \\ -100 & \text{---} & 310 & \text{---} & -10 \\ & & | & & \\ & & -100 & & \end{array}$$

$i = M, j < M$ $\qquad$ $i < M, j = M$

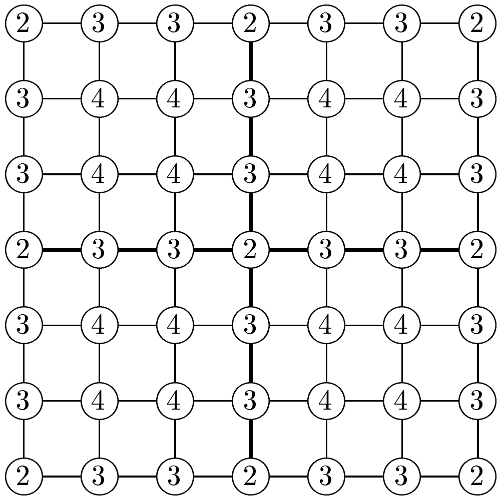
$$\begin{array}{ccccc} & & -1 & & \\ & & | & & \\ -10 & \text{---} & 31 & \text{---} & -10 \\ & & | & & \\ & & -10 & & \end{array}
\qquad
\begin{array}{ccccc} & & -10 & & \\ & & | & & \\ -10 & \text{---} & 31 & \text{---} & -1 \\ & & | & & \\ & & -10 & & \end{array}$$

$i = M, j > M$ $\qquad$ $i > M, j = M$

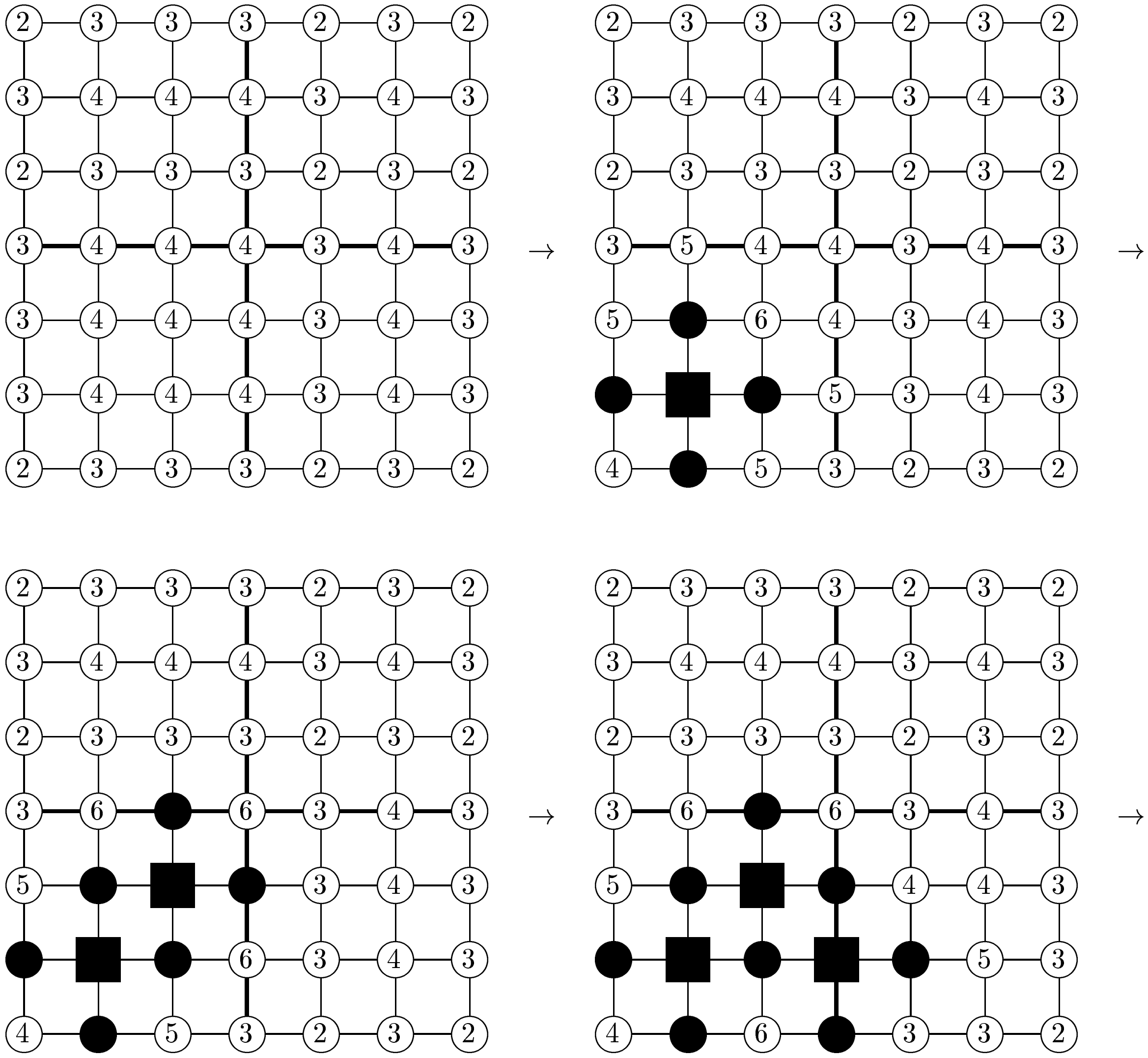
$$\begin{array}{ccccc} & & -10 & & \\ & & | & & \\ -100 & \text{---} & 220 & \text{---} & -10 \\ & & | & & \\ & & -100 & & \end{array}$$

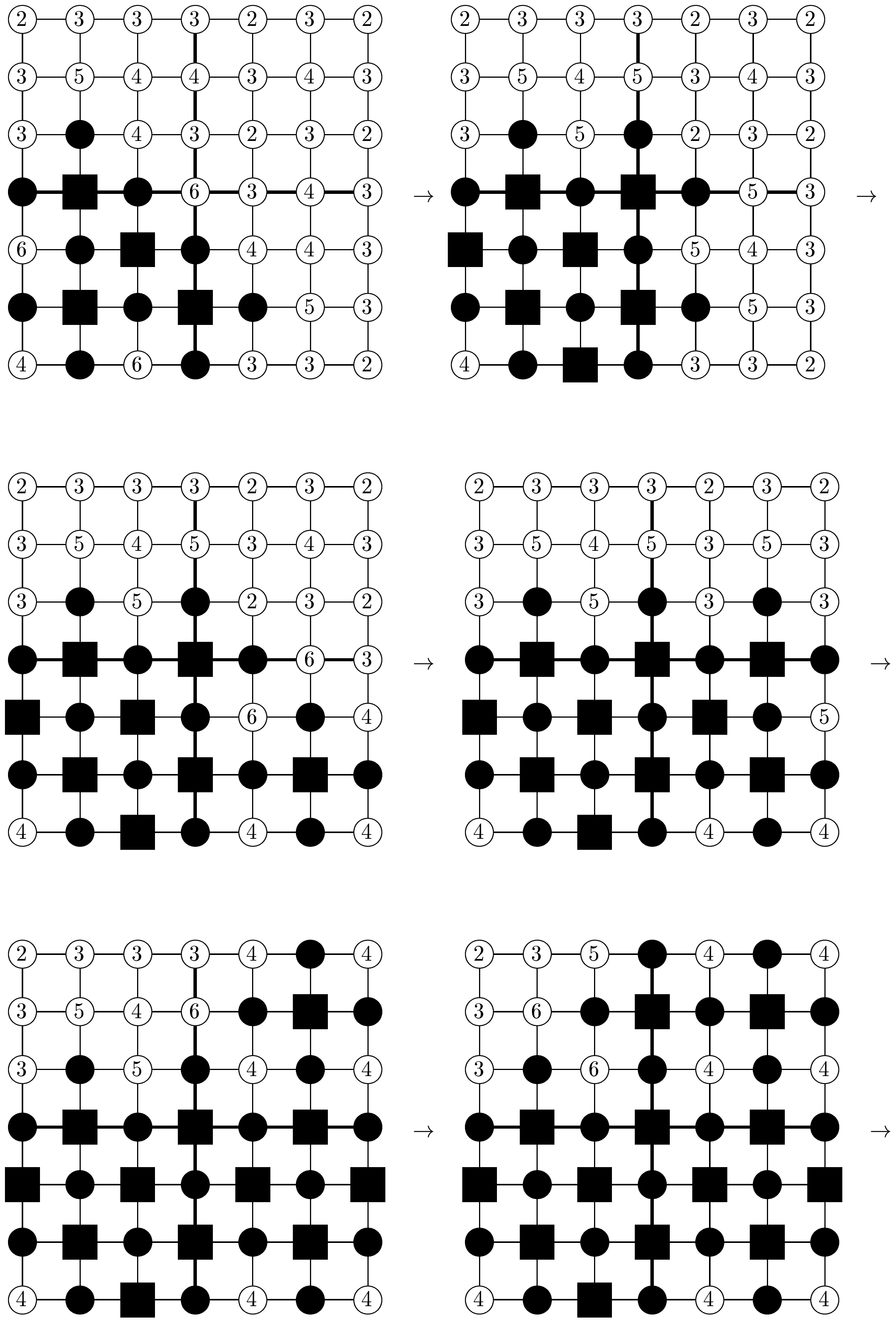
$i = j = M$

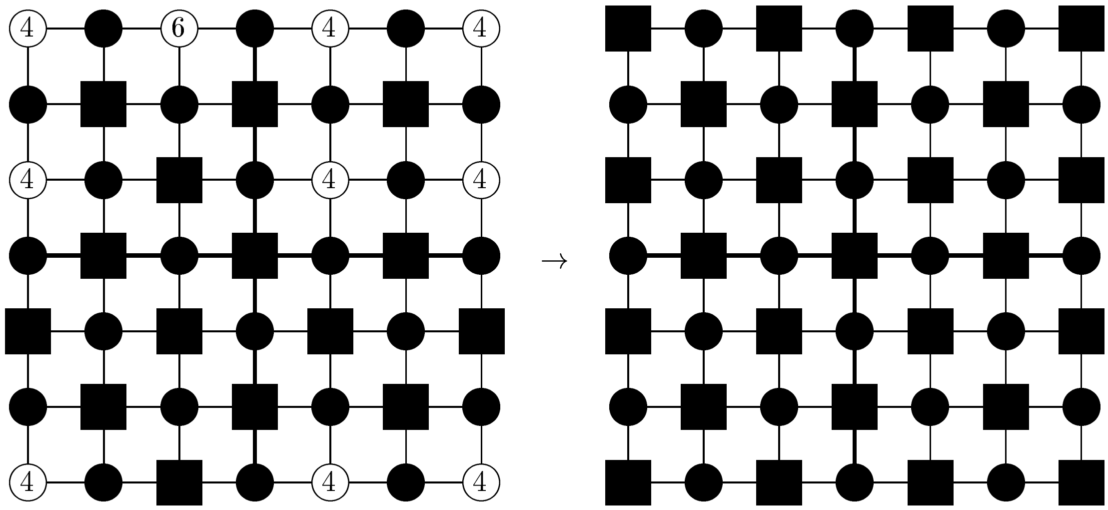
Wir sparen uns hier eine detaillierte Beschreibung der Mengen $\mathcal{S}, \mathcal{S}^\top$. Stattdessen geben wir die Werte anhand eines 7×7 Gitters an. Zuerst die Werte für $\#\mathcal{S}$.



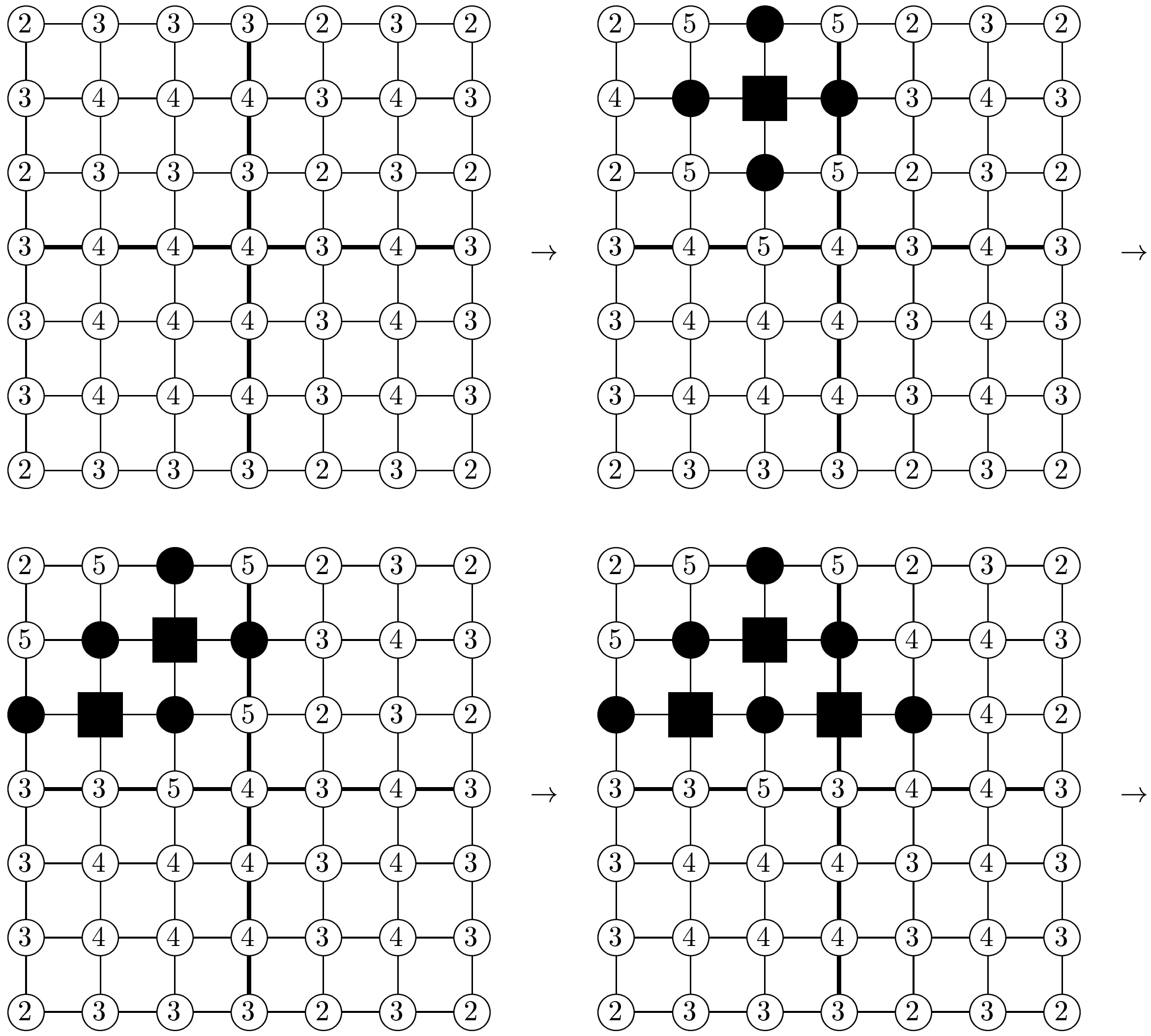
Wir fangen mit Knoten $(2,2)$ an und zeigen zunächst die Werte für $\lambda_j = \#\mathcal{S}^\top$ an.







Im Großen und Ganzen sehen wir den Trend, dass der Algorithmus versucht, in den Teilgebieten mit großem $a(x, y)$ zu bleiben oder zurückzukehren. Dass wie hier das grobe Gitter derart regelmäßig aussieht, muss im allgemeinen nicht sein. Dies sieht man z.B. wenn wir als erstes mit dem Knoten $(3, 6)$ anfangen.



Hier werden nicht automatisch alle Nachbarn (egal ob stark oder nicht) ins Feingitter geschoben.

Algorithmus 7 hat den Nachteil, dass er nicht nur Punkte für das grobe Gitter aussucht, sondern gleichzeitig auch Feingitterpunkte bestimmt. Hierbei kann es passieren, dass Punkte die zu Feingitterpunkten erklärt worden sind, nicht ausreichend durch Grobgitterpunkte interpoliert werden. D.h. es gibt Punkte $i \in \mathcal{F}$, mit $\mathcal{S}_i \not\subseteq \mathcal{C}$. Für diese Punkte könnte man natürlich $\mathcal{S}_i$ der Menge $\mathcal{C}$ hinzuzufügen. Wenn wir dies im großen Stile machen, bekommen wir vermutlich zu viele Grobgitterpunkte. Dies sollte als Option nur der letzte Ausweg sein. Mehr Sinn macht es, in einem solchen Fall die Interpolation mit einem Knoten $j \in \mathcal{S}_i$ zu ersetzen durch andere Punkte. Natürlich müssen diese Punkte aus $\mathcal{C}$ sein, mit Hilfe derer $j \in \mathcal{S}_i$ problemlos interpoliert werden kann. Ist also für $j \in \mathcal{S}_i$ bereits $\mathcal{S}_j \subseteq \mathcal{C}$, so ersetzen wir die Interpolation mittels j durch die Interpolation mit Hilfe von $\mathcal{S}_j$. Ansonsten überprüfen wir, ob es günstiger ist j der Menge $\mathcal{C}$ hinzuzufügen oder alternativ i selbst.
Dieser Prozess wird im wesentlichen durch folgenden Algorithmus 11 realisiert. Dabei werden gleichzeitig die Interpolationsgewichte konstruiert und die Beiträge schwacher Nachbarn kompensiert.

Algorithmus 11 (Endgültige Wahl der Grobgitterpunkte, Interpolation)
$\mathcal{C}$ bezeichne die Menge der Grobgitterpunkte, $\mathcal{F}$ die verbleibenden Feingitterpunkte sowie $\mathcal{T}$ die Menge der getesteten Feingitterknoten.

Zu Beginn sind $\mathcal{C}$, $\mathcal{F} = \{1, \dots, n\} \setminus \mathcal{C}$ aus Algorithmus 7 gewählt.$\mathcal{T} = \emptyset$.

Solange $\mathcal{T} \not\supseteq \mathcal{F}$.
  % Teste einen noch nicht getesteten Feingitterknoten
  Wähle $i \in \mathcal{F} \setminus \mathcal{T}$, $\mathcal{T} = \mathcal{T} \cup \{i\}$
  $\mathcal{C}_i = \mathcal{S}_i \cap \mathcal{C}$ % Starke Nachbarn im Grobgitter
  $\mathcal{D}_i = \mathcal{S}_i \setminus \mathcal{C}_i$ % Starke Nachbarn im Feingitter
  $\tilde{\mathcal{C}}_i = \emptyset$ % Zahl der Elemente, die nach $\mathcal{C}$ verschoben werden sollen
  % Diagonalkompensation für schwache Nachbarn
  $d_i = a_{ii} + \sum_{j:\ j \notin \mathcal{S}_i} a_{ij}$, für alle $j \in \mathcal{C}_i$: $d_j = |a_{ij}|$
  Für alle $j \in \mathcal{D}_i$:
    % hat starker Feingitternachbar j von i einen 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 $k \in \mathcal{C}_i$: $d_k = d_k + |a_{ij}| \frac{|a_{jk}|}{s}$
    Sonst: falls $\tilde{\mathcal{C}}_i = \emptyset$ % Bisher ist 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: % es wurde bereits ein $j \in \mathcal{D}_i$ versuchsweise ins Grobgitter geschoben
      % Schiebe lieber i ins grobe Gitter statt eines weiteren j
      $\tilde{\mathcal{C}}_i = \{i\}$, $\mathcal{D}_i = \mathcal{C}_i = \emptyset$
  % Schiebe Element aus $\tilde{\mathcal{C}}_i$ endgültig ins Grobgitter
  $\mathcal{C} = \mathcal{C} \cup \tilde{\mathcal{C}}_i$, $\mathcal{F} = \mathcal{F} \setminus \tilde{\mathcal{C}}_i$
  % Bilde gemittelte Interpolationsgewichte
  Für alle $j \in \mathcal{C}_i$: $p_{ij} = \frac{d_j}{d_i}$

% Triviale Einbettung für Grobgitterpunkte

Für alle $i \in \mathcal{C}$: $p_{ii} = 1$

Der Algorithmus berechnet eine Prolongationsmatrix P, welche die Interpolation vom Grobgitter ins Feingitter beschreibt (wenn man die Nullspalten zu Enträgen aus $\mathcal{F}$ streicht oder als nicht vorhanden interpretiert).

Wir betrachten ein paar Beispiele. Die Beispiele (8)–(10) sind ungeeignet für das Verschieben von Interpolationsgewichten oder für das Verschieben von Knoten ins grobe Gitter. Das liegt daran, dass diese Beispiele immer alle Nachbarn der Feingitterpunkte bereits im Grobgitter liegen. Sie sind allerdings leicht nachzuvollziehen, wenn es um die Interpolation geht.

Beispiel 12 *Die Matrix P stimmt mit der Interpolationsvorschrift (1.24) überein. Denn für jedes gerade i wird gerade x_i interpoliert mit Hilfe von*

$$x_i = \frac{|T_{i,i-1}|}{T_{ii}} y_{i-1} + \frac{|T_{i,i+1}|}{T_{ii}} y_{i+1} = \frac{1}{2} y_{i-1} + \frac{1}{2} y_{i+1}.$$

Beispiel 13 *Die Matrix P stimmt ebenfalls mit der geometrischen Interpolationsvorschrift (1.24) überein. Das liegt in diesem Beispiel daran, dass Knoten $M = \lceil N/2 \rceil$ zum Grobgitter gehört. Ansonsten wäre für diesen Knoten die Interpolationsvorschrift anders.*

Beispiel 14 *Das Grobgitter ist völlig anders als beim geometrischen Mehrgitterverfahren. Hier haben wir ungefähr doppelt so viele Grobgitterknoten. Für einen Knoten (i,j) aus dem feinen Gitter ist die Interpolation an den Stellen wo keine Koeffizientensprünge sind gegeben durch*

$$x_{i,j} = \frac{1}{4} y_{i,j-1} + \frac{1}{4} y_{i,j+1} + \frac{1}{4} y_{i-1,j} + \frac{1}{4} y_{i+1,j}.$$

Fehlt eine starke Verbindung (etwa $(i,j+1)$) wegen eines Koeffizientensprunges, so bekommen wir

$$x_{i,j} = \frac{1}{3} y_{i,j-1} + \frac{1}{3} y_{i,j+1} + \frac{1}{3} y_{i-1,j}.$$

Das liegt daran, dass Diagonalkompensation gemacht wird. Bei zwei starken Nachbarknoten lauten die Gewichte entsprechend je $\frac{1}{2}$. Ausnahme von dieser Regel bilden die Knoten, die sich in Randnähe befinden. Hier sind die Zeilensummen größer als Null und dementsprechend werden selbst einfache Feingitterknoten (i,j) z.B. durch $x_{i,j} = \frac{1}{4} y_{i,j-1} + \frac{1}{4} y_{i,j+1} + \frac{1}{4} y_{i-1,j}$ interpoliert.

2.3.3 Die Konstruktion der Interpolation in Matrixschreibweise

Die Konstruktion der Interpolation mit Hilfe von A kann man sich auch in Matrixschreibweise vorstellen. Als erstes nehmen wir die Matrix A und ordnen die Zeilen/Spalten so an, dass die z.B. die Grobgitterpunkte als erstes kommen. Schwache Nachbarn in einer Zeile werden sofort auf die Diagonale in jeder Zeile verschoben.

Ist etwa zu Beginn

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

so dass E die Einträge von A beschreibt, die zeilenweise in jeder Zeile i schwache Nachbarn von i sind, dann wird A ersetzt durch

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

wobei D die Diagonalmatrix mit

$$D \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = E \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

ist. Da die Einträge von E Aussendiagonaleinträge sind, sind die Diagonaleinträge von D kleiner oder gleich Null. Per Konstruktion ändern sich die Zeilensummen nicht.

Die Einträge aus $\mathcal{D}_i$ entsprechen den verbleibenden Aussendiagonaleinträgen in Zeile i von Block $P_{\mathcal{F},\mathcal{F}}$. Braucht beim Testen eines Feingitterpunktes i keiner seiner starken Nachbarn zusätzlich ins Grobgitter, so werden lediglich Aussendiagonaleinträge von $P_{\mathcal{F},\mathcal{F}}$ in Zeile i auf den Block $P_{\mathcal{F},\mathcal{C}}$ verteilt. Jeder einzelne Aussendiagonaleintrag p_{ij} in $P_{\mathcal{F},\mathcal{F}}$ wird so auf die Einträge p_{ik} in $P_{\mathcal{F},\mathcal{C}}$ verteilt wie es den Größenverhältnissen der Koeffizienten p_{jk} in Zeile j derselben Matrix $P_{\mathcal{F},\mathcal{C}}$ entspricht.
Setzen wir etwa $s = \sum_{k\in\mathcal{C}} |p_{jk}|$ und für alle $k \in \mathcal{C}$, $w_{jk} = \frac{|p_{jk}|}{s}$, so sind für alle $k \in \mathcal{C}$, $w_{jk} \geqslant 0$ und es ist $\sum_{k\in\mathcal{C}} w_{jk} = 1$. Für jedes j wird

$$e_i^\top \left(\, P_{\mathcal{F},\mathcal{C}} \,\middle|\, P_{\mathcal{F},\mathcal{F}} \, \right)$$

ersetzt durch

$$e_i^\top \left(\, P_{\mathcal{F},\mathcal{C}} \,\middle|\, P_{\mathcal{F},\mathcal{F}} \, \right) + \left(\, \Big(p_{ij} \cdot w_{jk}\Big)_{k\in\mathcal{C}} \,\middle|\, -p_{ij} e_j^\top \, \right).$$

Dabei ändern sich auch hier die Zeilensummen wegen $\sum_{k\in\mathcal{C}} p_{ij} w_{jk} - p_{ij} = p_{ij} \left(\sum_{k\in\mathcal{C}} w_{jk} - 1\right) = 0$ nicht und die Vorzeichen bleiben auch erhalten.

Entscheiden wir uns hingegen in einem Schritt dazu einen starken Nachbarn j von i ins grobe Gitter zu verschieben, so wird $\begin{pmatrix} I & P_{\mathcal{C},\mathcal{F}} \\ P_{\mathcal{F},\mathcal{C}} & P_{\mathcal{F},\mathcal{F}} \end{pmatrix} + D$ zeilenweise und spaltenweise neu umgeordnet, damit Zeile/Spalte j vom $\mathcal{F}$–Block in den $\mathcal{C}$–Block wandert. Gleiches passiert auch, wenn wir stattdessen i vom $\mathcal{F}$–Block in den $\mathcal{C}$–Block verschieben.

Am Ende erhalten wir eine Matrix der Form

$$\begin{pmatrix} I + E_{\mathcal{C},\mathcal{C}} & P_{\mathcal{C},\mathcal{F}} \\ P_{\mathcal{F},\mathcal{C}} & D_{\mathcal{F},\mathcal{F}} \end{pmatrix},$$

wobei $D_{\mathcal{F},\mathcal{F}}$ diagonal ist und $E_{\mathcal{C},\mathcal{C}}$ später ignoriert wird. Es gilt in jedem Falle

$$A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} I + E_{\mathcal{C},\mathcal{C}} & P_{\mathcal{C},\mathcal{F}} \\ P_{\mathcal{F},\mathcal{C}} & D_{\mathcal{F},\mathcal{F}} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Betrachten wir nun die Matrix

$$P = \begin{pmatrix} I \\ -D_{\mathcal{F},\mathcal{F}}^{-1} P_{\mathcal{F},\mathcal{C}} \end{pmatrix},$$

so ist dies genau die Matrix der Interpolation. Für diejenigen Zeilen i von A, für die $e_i^\top A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 0$ ist, gilt

$$0 = e_i^\top A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \sum_{k \in \mathcal{C}} p_{ik} + d_{ii}.$$

Hieraus folgt dann

$$\frac{\sum_{k \in \mathcal{C}} |p_{ik}|}{d_{ii}} = 1,$$

weil die einzelnen $p_{ik} \leqslant 0$ sind. Mit anderen Worten: Für alle Zeilen die zu Grobgitterpunkten gehören sowie zu allen Feingitterpunkten i, für die die Zeilensumme von Zeile i in A Null ist, ist die Zeilensumme von P gleich 1. Ansonsten kann ist die Zeilensumme $1 - \frac{1}{d_{ii}} e_i^\top A \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ zwischen 0 und 1. Mit anderen Worten, wir haben immer

$$P \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \left(I - \begin{pmatrix} \mathrm{O} & \mathrm{O} \\ \mathrm{O} & D_{\mathcal{F},\mathcal{F}}^{-1} \end{pmatrix} A\right) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

2.3.4 Ergänzungen bei Diagonaldominanz

Wir erwähnen kurz die Erweiterung der Algorithmen bei Diagonaldominanz. Im Prinzip können Zeilen i für die $a_{ii} \gg \sum_{j \neq i} |a_{ij}|$, gehören eigentlich sofort ins Feingitter und brauchen auch nicht interpoliert zu werden. Die Änderungen sind schnell beschrieben. Sei etwa $\delta \in (0, 1)$ gegeben, etwa $\delta = \frac{1}{4}$. Dann setzen wir

$$\mathcal{F}_{dd} = \{i : \ (1 - \delta)|a_{ii}| \geqslant \sum_{j:\ j \neq i} |a_{ij}|\}. \tag{2.11}$$

$\mathcal{F}_{dd}$ ist die Menge der strikt diagonal dominanten Zeilen. Wir definieren abweichend

$$\mathcal{S}_i = \emptyset, \quad \text{für alle } i \in \mathcal{F}_{dd}. \tag{2.12}$$

Ausserdem startet Algorithmus 7 mit $\mathcal{C} = \emptyset$, $\mathcal{F} = \mathcal{F}_{dd}$ und $\mathcal{U} = \{1, \ldots, n\} \setminus \mathcal{F}_{dd}$. Algorithmus 11 darf mit $\mathcal{T} = \mathcal{F}_{dd}$ starten, weil das Testen von Punkten $i \in \mathcal{F}_{dd}$ wegen $\mathcal{S}_i = \emptyset$ sowieso keinerlei Änderung bringt. Bleibt ein Punkt $i \in \mathcal{F}_{dd}$ in $\mathcal{F}$, so sind die Interpolationsgewichte Null. Es kann allerdings passieren, dass Punkte aus $\mathcal{F}_{dd}$ ins Grobgitter verschoben werden. Das passiert in dem Augenblick, wo ein zu testender Knoten $i \in \mathcal{F} \setminus \mathcal{T}$ einen Knoten $j \in \mathcal{F}_{dd}$ als starken Nachbarn hat. Da j selbst aber keinen starken Nachbarn (wegen $\mathcal{S}_j = \emptyset$) mehr hat, kann nur i selbst oder j ins Grobgitter verschoben werden. Im letzten Fall wird ein Punkt $j \in \mathcal{F}_{dd}$ zum Grobgitterpunkt.

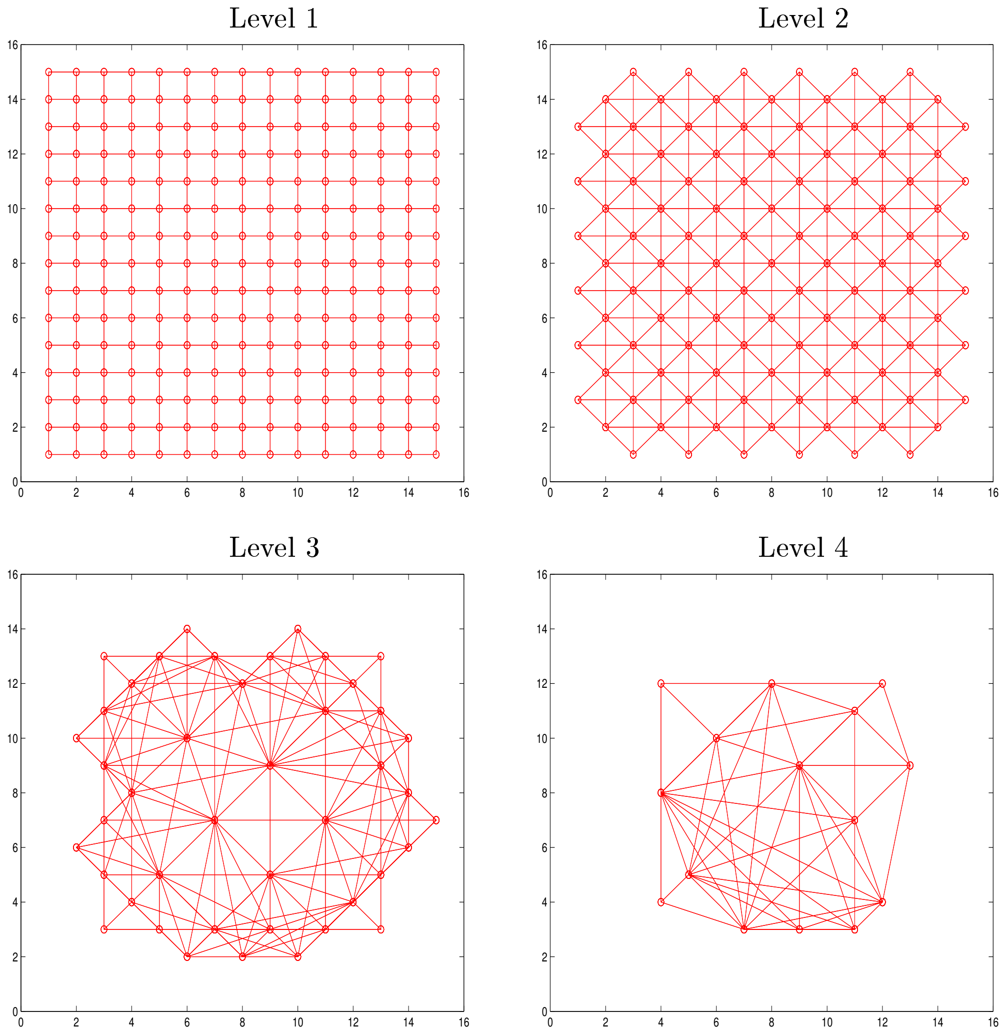
2.3.5 Erhaltung der Vorzeicheneigenschaft

Wir haben bisher nur Techniken gesehen, die die Interpolation P konstruieren. Bei einem Mehrgitterverfahren erzeugen wir jedoch ein neues Grobgittersystem $A_H = P^\top AP$. Dass dieses System die Vorzeicheneigenschaft erfüllt ist nicht sofort klar. Auch ist nicht klar, ob die Zeilensummen vergleichbar mit denen von A sind. Dies lässt sich aber unter recht allgemeinen Voraussetzungen zeigen. Für Details siehe [30].

Damit ist das AMG nicht nur ein Zweigitter–Verfahren, sondern erlaubt auch den rekursiven Einsatz als Mehrgitter–Verfahren. Als Glätter wird üblicher Weise das Gauß–Seidel–Verfahren eingesetzt. Wir betrachten einige Beispiele.

Beispiel 15 *Als erstes verwenden wir das einfache 2D Modellproblem (1.3) mit Fünf–Punkt–Disktretisierung (1.13). Im Gegensatz zum geometrischen Mehrgitter sind die hier entstehende Gitter auf den gröberen Levels erheblich weniger strukturiert und geometrisch z.T. nicht sonderlich sinnvoll (Abbildung 2.2).*

Abbildung 2.2: **Algebraisch erzeugte Gitter**



Im Gegensatz zum geometrischen Mehrgitterverfahren reduziert sich die Zahl der Unbekannten pro Level hier weniger. Eine Halbierung ist für ein rein algebraisches Verfahren aber schon recht gut. Auch die durchschnittliche Zahl der Nichtelemente pro Zeile steigt i.a. an. Details zeigt die folgende Tabelle.

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*
7	*1*	*1.0*

Die Zahl der Iterationen ist sicherlich höher als beim geometrischen Mehrgitterverfahren. Sie hält sich aber doch erkennbar in Grenzen. Als Beispiel betrachten wir das cg–Verfahren mit Gauß–Seidel und gedämpften Jacobi als Glätter. Das Jacobi–Verfahren ist so gedämpft worden, dass die vorkonditionierte Matrix alle Eigenwerte kleiner oder gleich 1 hat.

Zeit in* flops *für das vorkonditionerte cg–Verfahren

n	*Jacobi*		*Gauß–Seidel*	
225	$3.6{\cdot}10^5$	18	$2.7{\cdot}10^5$	11
961	$1.8{\cdot}10^6$	20	$1.4{\cdot}10^6$	12
3969	$8.1{\cdot}10^6$	21	$5.8{\cdot}10^6$	12

Verglichen mit den Werten für das geometrische Mehrgitterverfahren (siehe Tabelle 1.8 auf Seite 26 schlägt sich das AMG an dieser Stelle ziemlich gut. Es soll jedoch nicht verheimlicht werden, dass die Generierung der Mehrgitter–Hierarchie eine erhebliche Zeit in Anspruch nimmt. Diese überlagert in der Regel noch die Zeit für den Lösungsprozess.

Als ein weiteres Beispiel betrachten wir ein einfaches Problem allerdings mit einer Anisotropie und Grenzschichten.

Beispiel 16 *Betrachten wir das Modellproblem*

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

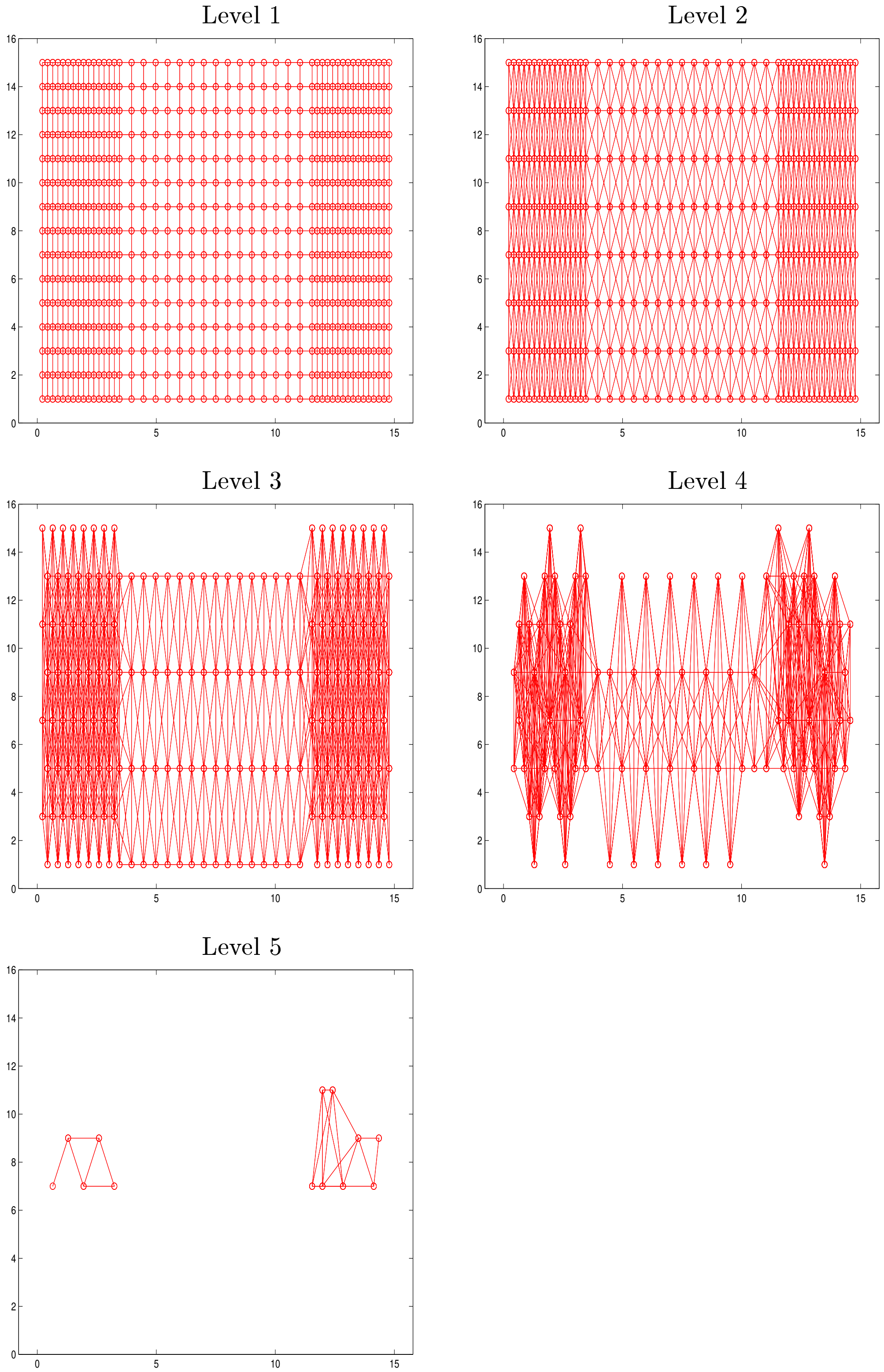
so empfielt es sich in jedem Falle am linken und rechten Rand des Gebietes eine Grenzschicht der Breite $\varepsilon|\ln\varepsilon|$ vorzusehen [1]. Wir verwenden eine angepasste 5–Punkt–Diskretisierung. Dies führt zu einer Unsymmetrie der Matrix (dies liesse sich bei der FEM vermeiden).

Die entstehenden Gitter zeigen eine klare Ausrichtung in Richtung der y–Achse (Abbildung 2.3).

Im Gegensatz zum geometrischen Mehrgitterverfahren reduziert sich die Zahl der Unbekannten pro Level hier weniger. Eine Halbierung ist für ein rein algebraisches Verfahren aber schon recht gut. Auch die durchschnittliche Zahl der Nichtelemente pro Zeile steigt i.a. an. Details zeigt die folgende Tabelle.

Vergröberungsprozess

Abbildung 2.3: **Algebraisch erzeugte Gitter**



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*

Da die Matrix nicht mehr symmetrisch ist, verwenden wir anstelle des cg–Verfahrens jetzt das GMRES(20)–Verfahren [31]. Wie in Beispiel 15 verwenden wir Gauß–Seidel und gedämpften Jacobi als Glätter ($\nu = 2$).

Zeit in flops ***für das vorkonditionerte cg–Verfahren***

n	*Jacobi*		*Gauß–Seidel*	
705	$1.3{\cdot}10^6$	14	$9.2{\cdot}10^5$	9
2945	$6.6{\cdot}10^6$	16	$4.7{\cdot}10^6$	10
12033	$2.9{\cdot}10^7$	17	$1.9{\cdot}10^7$	10

Auch hier überzeugt die reine Rechenzeit für das Lösen des Systems. Wiederum ist die Zeit für die Generierung nicht aufgeführt. Diese in flops *zu messen macht nicht viel Sinn, überlagert aber den Lösungsprozess.*

2.3.6 Weitere Ruge–Stüben–artige AMGs

Neben dem bereits klassischen Mehrgitter–Ansatz gibt es darauf basierend viele weitere Verfahren, die diese Methode in der ein oder anderen Art nutzen. Z.B. wird ein Verfahren dieses Typs in [24] als Vorkonditierer eingesetzt (z.B. zur Optimierung von Chip–Layout). Andere Anwendungen beschäftigen sich z.B. mit der Parallelisierung des Verfahrens (siehe z.B. [13]). Für Probleme die aus Finite–Element–Methoden stammen gibt es z.B. eine neuere Variante, die die FEM–Struktur mit in das AMG einbaut [20, 19]. Für einige Probleme, wo die zugrunde liegende Matrix nicht die gewünschte Vorzeichenstruktur hat, weil etwa bei der Diskretisierung Ansatzfunktionen gewählt werden, die die Vorzeicheneigenschaft der Matrix nicht erhalten, werden äquivalente M–Matrizen konstruiert und darauf dann das Ruge–Stüben–AMG angewandt [14].

2.4 AMGs basierend auf Aggregation

In diesem Abschnitt beschäftigen wir uns mit einem algebraischen Mehrgitter–Verfahren, dass von Braess zur Behandlung von elliptischen Problemen zweiter Ordnung wie z.B. (1.5) vorgeschlagen worden ist [8]. Weitere Arbeiten hierzu findet man z.B. in [34]. Das Verfahren unterscheidet sich von dem AMG von Ruge und Stüben dadurch, dass die M–Matrix–Eigenschaft nicht mehr benötigt wird. Dafür sollte die Matrix aber symmetrisch und positiv definit sein. Es wird ausserdem angenommen, dass eine Diskretisierung mittels FEM in Dreiecks– und/oder Viereckselemente zugrunde liegt.

Die Idee besteht darin die Knoten $\{1,\dots,n\}$ geschickt in disjunkten Clustern $\mathcal{I}_1,\dots,\mathcal{I}_m$ zusammenzufassen und daraus die Restriktion zu bilden. D.h. $R:\ \mathbb{R}^n \to \mathbb{R}^m$ ist dadurch definiert, dass $y = Rx$, wobei

$$y_k = \sum_{i\in\mathcal{I}_k} x_i, \text{ für alle } k = 1,\dots,m. \tag{2.13}$$

Als Prolongation verwendet man wieder $P = R^\top$ und als Grobgittersystem $A_H = RAR^\top$. Dies bedeutet, dass man einen Grobgittereintrag a^H_{kl} wie folgt aus $A = (a_{ij})_{i,j=1,\dots,n}$ bekommt:

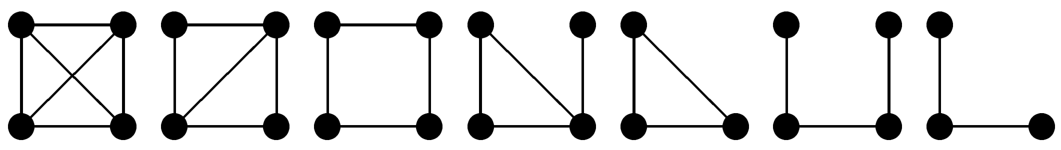
$$a^H_{kl} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix} (a_{ij})_{i\in\mathcal{I}_k, j\in\mathcal{I}_l} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \sum_{i\in\mathcal{I}_k, j\in\mathcal{I}_l} a_{ij}. \tag{2.14}$$

Beispiel 17 *Nehmen wir etwa als Beispiel T_h aus (1.11). Der Einfachheit soll die Dimension gerade sein und $I_j = \{2j-1, 2j\}$, $j = 1,\dots,n/2$ sein. D.h. zwei benachbarte Zeilen und Spalten werden zu einer Zeile/Spalte zusammengeschmolzen.*

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

Die eigentliche Kunst ist es natürlich die Mengen $\mathcal{I}_1,\dots,\mathcal{I}_m$ geschickt zu bestimmen. In [8] wird vorgeschlagen, für elliptische 2D–Probleme, bei denen mittels FEM das Gebiet irgendwie in Dreiecke und Vierecke zerlegt wird, Knoten wie folgt zu clustern.
Die Mengen $\mathcal{I}$ sollen möglichst aus drei bis vier Elementen bestehen. Für deren Auswahl wird zunächst eine Prioritätenliste aufgestellt, siehe Abbildung 2.4.

Abbildung 2.4: **Muster für die Bildung von Gruppen nach Priorität**



Priorität 7 6 5 4 3 2 1

Die Gruppen werden jetzt sukzessive gebildet. Neue Gruppen werden in zwei Schritten konstruiert. Zunächst werden zwei Knoten in eine neue Gruppe zusammengefasst, derart dass sie und zwei Knoten einer bereits existierenden Gruppe einen der Graphen in Abbildung 2.4 enthalten. Anschließ werden sie so vervollständigt, so dass ihr Graph eine möglichst hohe Priorität im Sinne von Abbildung 2.4 hat.

Dies kann durch einen Algorithmus beschrieben werden.

Algorithmus 18 (Gruppenbildung, 1. Algorithmus)
$\mathcal{U} = \{1, \dots, n\}$. $p = 0$
Solange $\mathcal{U} \neq \emptyset$
  $p = p + 1$
  Wähle eine Gruppe $\mathcal{G}_p$ *von maximal 4 Knoten mit zulässigem Graphen entsprechend Abbildung 2.4 (triviale Graphen mit 1 bis zwei Knoten sind weggelassen).*
  $\mathcal{U} = \mathcal{U} \setminus \mathcal{G}_p$.
  Solange $\mathcal{G}_1, \dots, \mathcal{G}_p$ *Nachbarn in* $\mathcal{U}$ *haben*
    Sei $i \in \{1, \dots, p\}$ *kleinstmöglich, so dass* $\mathcal{G}_i$ *einen Nachbarn in* $\mathcal{U}$ *hat*
    Falls ein zulässiger Graph mit je zwei Elementen aus $\mathcal{U}$ *und* $\mathcal{G}_i$ *möglich ist*
      Bestimme unter diesen den besten Graphen i.S. von Abbildung 2.4.
      Sind etwa j, k *die beiden Knoten aus* $\mathcal{U}$*, so bestimme das beste Muster aus* $\mathcal{U}$ *welches* j, k *umfasst bezeichne die Menge als* G_{p+1}.
    Sonst:
      wähle maximal drei Nachbarn aus $\mathcal{U}$ *die Nachbarn von* $\mathcal{G}_i$ *sind, bestmöglich i.S. von Abbildung 2.4, und bezeichne diese Menge als* G_{p+1}.
    $p = p + 1$
    $\mathcal{U} = \mathcal{U} \setminus \mathcal{G}_p$

Dieser Algorithmus versucht soweit es geht, Gruppen mit hoher Priorität als auch einer hohen Zahl von Verbindungen zu den Nachbarn zu bilden. Der Nachteil ist natürlich die Vielzahl von Fallunterscheidungen und die daraus resultierende Komplexität. Ausserdem haben auch die numerischen Werte keinerlei Einfluss auf die Konstruktion. Aus diesem Grund wird in [8] ein weiterer Algorithmus vorgestellt, der diese Aspekte besser berücksichtigt.

Der zweite Algorithmus bildet zuerst Paare (i, j) nach größtmöglichem $\frac{a_{ij}^2}{a_{ii}a_{jj}}$. In einem zweiten Schritt werden diese zu Quadrupeln zusammengefasst, so dass die Priorität in Abbildung 2.4 größtmöglich ist.

Algorithmus 19 (2. Algorithmus, AMG von Braess)
<u>1. Schritt:</u> Paarbildung
$\mathcal{U} = \{1, \dots, n\}$. $p = 0$
Solange $\mathcal{U} \neq \emptyset$
  $p = p + 1$
  Wähle ein $i \in \mathcal{U}$.
  Falls i *mindestens einen Nachbarn in* $\mathcal{U}$ *besitzt*
    Bestimme Nachbar $j \in \mathcal{U}$ *von* i *so, dass* $\frac{a_{ij}^2}{a_{ii}a_{jj}}$ *maximal ist.*
    $\mathcal{P}_p = \{i\}$
  Sonst:
    $\mathcal{P}_p = \{i, j\}$
  $\mathcal{U} = \mathcal{U} \setminus \mathcal{P}_p$
<u>2. Schritt:</u> Gruppenbildung
$\mathcal{U} = \bigcup_{i=1}^{p} \{\mathcal{P}_i\}$. $p = 0$

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:
        $\mathcal{G}_p = \mathcal{P}_i$
        $\mathcal{U} = \mathcal{U} \setminus \{\mathcal{P}_i\}$

Natürlich kann man die Freiheiten in diesem Algorithmus noch etwas nützen. Z.B. bei der Paarbildung könnte man anstelle eines beliebigen unbestimmten Knotens einen Knoten nehmen, der selbst Nachbar eines Paares ist. Unter den Knoten j die m_{ij} maximieren, könnte man diejenigen bevorzugen, die Nachbarn von bereits existierenden Paaren sind, womöglich Nachbarn desselben Paares wie i.

Wir betrachten einige Beispiele.

Beispiel 20 *Als erstes verwenden wir wieder Modellproblem (1.3) mit Diskretisierung (1.13). Wir betrachten zunächst einige Bilder von den erzeugten Grobgittersystemen. Hierzu siehe Abbildung 2.5. Level 1 ist dabei weggelassen worden, da es sich lediglich umd das bekannte Rechteckgitter handelt mit dem man startet.*

Der Algorithmus selbst vergröbert das vorliegende System recht gut, ähnliche Gittergrößen und Besetztheit wie beim geometrischen Mehrgitter werden erreicht.

Vergröberungsprozess

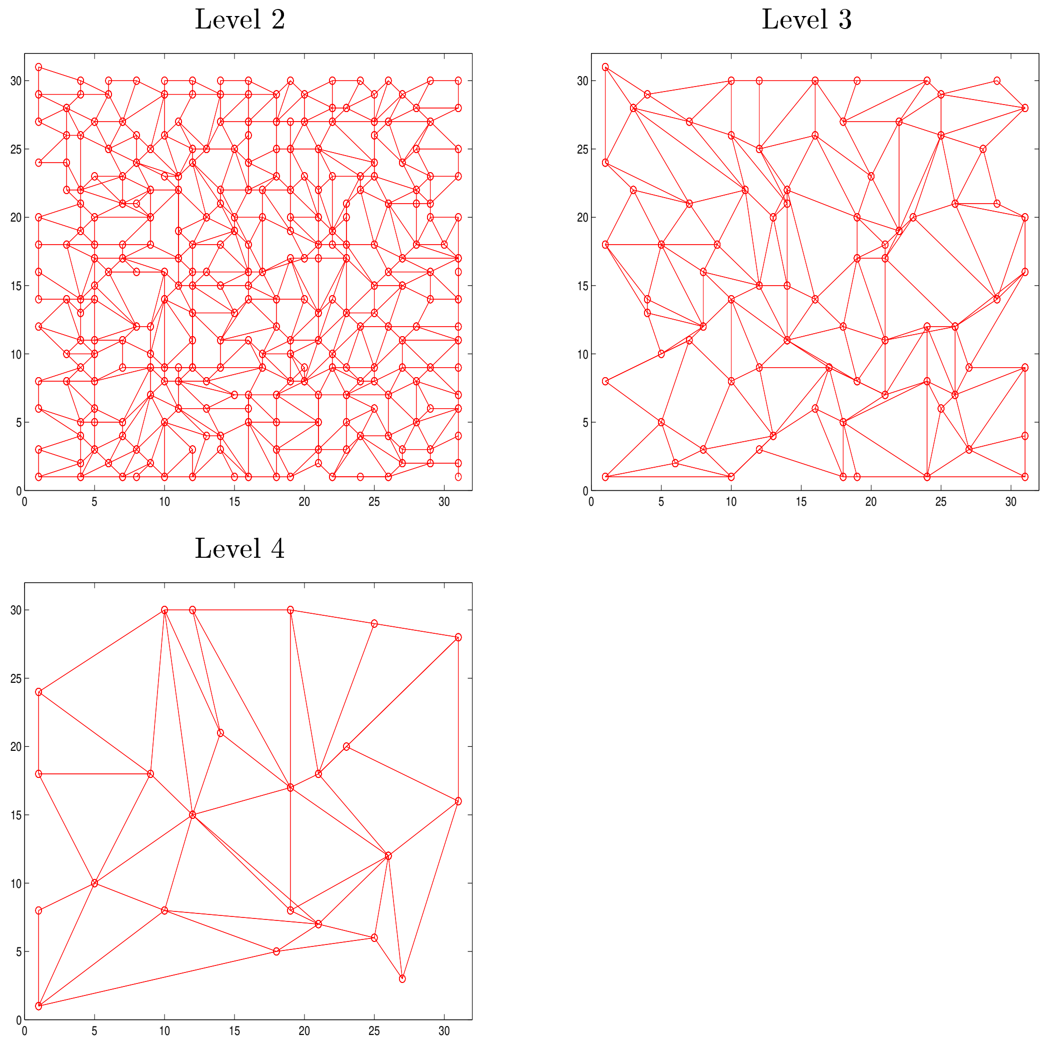
Level	n	*# Nichtnullel.(A)/n*
1	3969	4.9
2	1150	6.2
3	340	6.4
4	104	6.1
5	32	5.4
6	11	4.6
7	4	3.0

Betrachten wir als Mehrgitterverfahren ein cg–Verfahren mit Mehrgittervorkonditionierung (V11), so sehen wir, dass das AMG nicht so gut skaliert wie etwa ein geometrisches Mehrgitterverfahren.

	Jacobi		*Gauß–Seidel*	
n	flops	*# Iter.*	flops	*# Iter.*
225	$3.0 \cdot 10^5$	21	$2.2 \cdot 10^5$	13
961	$1.9 \cdot 10^6$	30	$1.5 \cdot 10^6$	19
3969	$1.3 \cdot 10^7$	48	$9.2 \cdot 10^6$	28

Man kann dies noch deutlich verbessern. In [8] wird als Vor– und Nachglätter das symmetrische Gauß–Seidel–Verfahren verwendet. Es wird ausserdem vorgeschlagen, bei mehr als 2

Abbildung 2.5: **Gitterhierarchie**



Leveln die Zahl der Glättungsschritte auf den gröberen Gittern zu erhöhen. Die Grobgitterkorrektur wird zusätzlich mit einem Relaxationsparameter 1.8 *versehen.*

	Sym. 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*

Wir betrachten ein weiteres Beispiel.

Beispiel 21 *Wir betrachten*

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

Konkret verwenden wir $\varepsilon = .1$. *Die Diskretisierung erfolgt mittels finiter Differenzen. Es werden Randschichten der Größe* $\varepsilon|\ln\varepsilon|$ *berücksichtigt. Bei der Vergröberung sieht man klar die Bevorzugung in* y*–Richtung, was dem Problem wie auch der Konstruktion des AMG entspricht (Abbildung (2.6)).*

Die Vergröberung verhält sich wiederum ähnlich einem geometrischen Mehrgitterverfahren.

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*

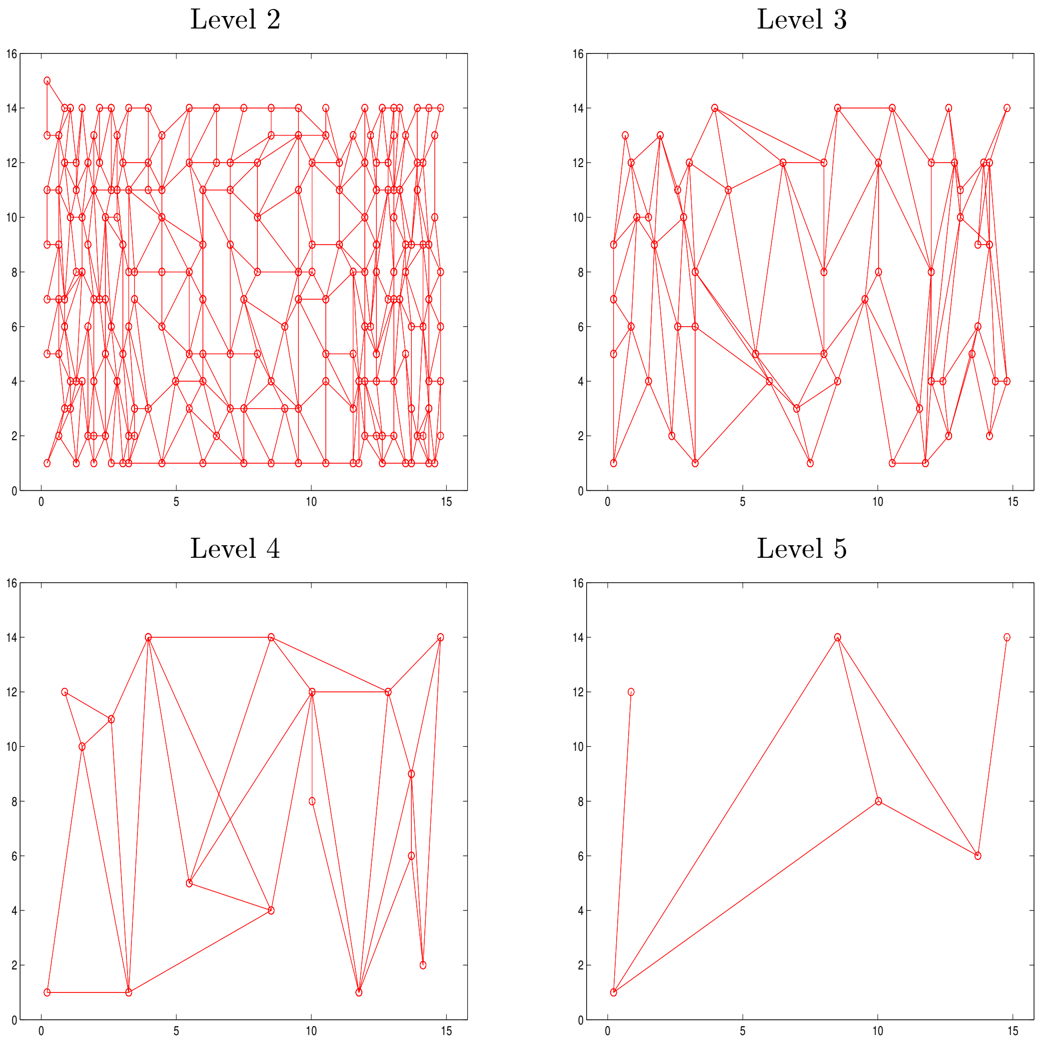
Wir verwenden jetzt als Glätter nur noch das symmetrische Gauß–Seidel–Verfahren. Auf Level 3 und gröber verwenden wir zwei Vor– und Nachglättungsschritte statt einem. Die Grobgitterkorrektur wird mit einem Faktor 1.8 *versehen.*

	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*

Auch hier ist ein befriedigendes Verhalten des AMG zu beobachten, wenn es hier auch nicht ganz so gut skaliert wie im ersten Beispiel.

Die numerischen Beispiele zeigen, dass das Verfahren durchaus recht gute Werte liefert, vorausgesetzt, man verwendet einen Relaxationsparameter für die Grobgitterkorrektur und erhöht die Zahl der Glättungsschritte auf den gröberen Gittern. Dieses Verfahren ist dort

Abbildung 2.6: **Gitterhierarchie**



einsetzbar, wo die zugrunde liegende Matrix keinerlei M–Matrix–Eigenschaft hat aber symmetrisch positiv definit ist. Vernachlässigt wurde bei den numerischen Beispielen die Zeit zur Konstruktion des AMG. Hier ist die Zeit zur Konstruktion eher gering. Laut [8] ungefähr mit drei Schritten des vorkonditionierten cg–Verfahren vergleichbar.

2.5 Unvollst. Block–Eliminationstechniken als AMG

Wir gehen als letztes noch auf unvollständige Block–Eliminationstechniken zur Konstruktion von AMGs ein. Aus Zeitgründen muss auf eine detaillierte Beschreibung dieser Methoden hier verzichtet werden. Eine nette Zusammenstellung dieser Methoden findet man z.B. in [35]. Idee hierbei ist, wie bereits in Abschnitt 2.2 erwähnt, kann man einen Schritt eines exakten Blockeliminationsschrittes als (exaktes) AMG interpretieren. D.h. eine Matrix $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ kann faktorisiert werden als

$$A = \begin{pmatrix} I & \mathrm{O} \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & \mathrm{O} \\ \mathrm{O} & S_{22} \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}A_{12} \\ \mathrm{O} & I \end{pmatrix},$$

wobei $S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$ das Schur–Komplement ist. Mit $P = \begin{pmatrix} -A_{11}^{-1}A_{12} \\ I \end{pmatrix}$, $R = \left(-A_{21}A_{11}^{-1},\ I\right)$ ist $S_{22} = RAP$ und natürlich könnte man z.B. das Verfahren von Ruge und Stüben anwenden um einmal eine Partionierung von A zu bekommen und zum anderen P, R entsprechend auszudünnen. Tatsächlich haben wir gesehen, dass man das AMG von Ruge und Stüben lesen kann als Partionierung der Ausgangsmatrix in der Form

$$\begin{pmatrix} A_{\mathcal{F},\mathcal{F}} & A_{\mathcal{F},\mathcal{C}} \\ A_{\mathcal{C},\mathcal{F}} & A_{\mathcal{C},\mathcal{C}} \end{pmatrix}.$$

Dabei wird durch verschiedene Techniken (bis hin zur Neupartitionierung) der Teil $A_{\mathcal{F},\mathcal{F}}$ durch eine Diagonalmatrix $\tilde{A}_{\mathcal{F},\mathcal{F}}$ ersetzt und $A_{\mathcal{F},\mathcal{C}}$ angepasst zu einer Matrix $\tilde{A}_{\mathcal{F},\mathcal{C}}$. Dabei bleiben die Zeilensummen konstant. Die Prolongationsmatrix ist anschließend ist

$$P = \begin{pmatrix} -\tilde{A}_{\mathcal{F},\mathcal{F}}^{-1}\tilde{A}_{\mathcal{F},\mathcal{C}} \\ I \end{pmatrix}.$$

Man kann natürlich jetzt einen Schritt weitergehen und das Verschieben von Einträgen innerhalb von $\left(\tilde{A}_{\mathcal{F},\mathcal{F}},\ \tilde{A}_{\mathcal{F},\mathcal{C}}\right)$ natürlich als eigenständigen Prozess für die unvollständige Block–Elimination ansehen.

Bei den auf unvollständigen (Block–) Dreieckszerlegungen beruhenden AMGs geht man etwas differenzierter vor. Hierfür gibt es mehrere Gründe. Zum einen muss man bei den Block–Zerlegungen natürlich noch die Behandlung des führenden Diagonalblocks diskutieren. Zum anderen kann man durch schrittweises Vorgehen die Konstruktion des approximativen Schur–Komplements präziser gestalten. Ein Beispiel ist die Erhaltung der Zeilensummen. Oft wird versucht, die Zerlegung $A = LDU$ exakt für gewisse Testvektoren zu halten. Siehe z.B. [12, 36]. Solche Techniken erfordern nicht nur unvollständige Zerlegungen auf einem gewissen Muster, sondern darüber hinaus eine Adaption der Restmatrix (approximatives Schur–Komplement). Ein bekanntes Verfahren, nämlich die hierarchischen Basen [37], lassen sich als unvollständige Dereickszerlegungen interpretieren. Dabei werden im wesentlichen bei der Elimination zunächst die Unbekannten, die zum Feingitter gehören, vorangestellt. Die bei der Elimination entstehende neuen Enträge (fill–in) in der Restmatrix werden nur zwischen Knoten des gröberen Gitters und den Knoten des Feingitters zugelassen. Hierzu werden sogenannte Vater–Knoten eingeführt. Dies sind Nachbarn von Feingitter–Punkten, die selbst zum groben Gitter gehören. Im Falle der Hierarchischen Basis in zwei Raumdimensionen und regulärer Verfeinerung (Viertelung eines Dreiecks durch Verbinden der Seitenmitten) sind das zu einem Feingitterpunkt immer die beiden benachbarten Grobgitterpunkte auf derselben Kante. Genaueres siehe in [35]. Diese Beobachtung kann man natürlich benutzen um algebraisch Vaterknoten und fill–in–Muster zu konstruieren. Siehe z.B. [5, 33, 4].

Eine weitere Methode die auf unvollständiger Blockdreieckszerlegung beruht ist die in [28, 29] vorgestellte Methode der approximative zyklischen Reduktion. Bei blocktridiagonale Matrizen kann man die Blöcke so umordnen, dass zunächst die Blöcke mit ungeraden und dan mit geraden Indizes genommen werden. Diese so partitionierte Matrix hat die Eigenschaft, dass das Schur–Komplement, welches bei der Elimination der ungeraden Blöcke entsteht, wieder blocktridiagonal ist. Diese Beobachtung kann man natürlich verwenden um daraus eine geiegnete unvollständige Dreieckszerlegung und algebraische Gitterhierarchie zu konstruieren.

Eine weitere Klasse von Algorithmen, die teilweise auf unvollständigen (Block–) Eliminationstechniken beruhen, bekommt man, wenn die Approximation des führenden Blocks A_{11}^{-1} in der unvollständigen Blockdreieckszerlegung durch approximative Inverse ersetzt. Siehe z.B. [11, 22, 21, 23].

Literaturverzeichnis

[1] T. Apel. *Anisotropic finite elements: Local estimates and applications.* Advances in Numerical Mathematics. B.G. Teubner Stuttgart, 1999. Habilitation thesis.

[2] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods I. *Numer. Math.*, 56:157–177, 1989.

[3] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Numer. Anal.*, 27:1569–1590, 1990.

[4] R. Bank and R. Smith. The incomplete factorization multigraph method. *SIAM J. Sci. Comput.*, to appear, 1998.

[5] R. E. Bank and C. Wagner. Multilevel ILU decomposition. *Numer. Math.*, to appear, 1999.

[6] A. Berman and R. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Classics in Applied Mathematics. SIAM Publications, 1994.

[7] D. Braess. *Finite Elemente.* Springer-Verlag, 1992.

[8] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[9] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 55:1–22, 1990.

[10] W. Bunse and A. Bunse-Gerstner. *Numerische lineare Algebra.* B.G. Teubner Stuttgart, 1985.

[11] W. Dahmen and L. Elsner. Hierarchical iteration. In Hackbusch, editor, *Lecture Notes on Numerical Fluid Dynamics.* Vieweg, 1988.

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

[13] G. Haase. A parallel amg for overlapping and non-overlapping domain decomposition. *Electr. Trans. Num. Anal.*, 10:41–55, February 2000.

[14] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl. Algebraic multigrid methods based on element preconditioning. *International Journal of Computer Mathematics*, 80(3–4):41–55, to appear.

[15] W. Hackbusch. *Multigrid Methods and Applications.* Springer-Verlag, 1985.

[16] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* B.G. Teubner Stuttgart, 1986.

[17] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme.* B.G. Teubner Stuttgart, 1991.

[18] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme.* B.G. Teubner Stuttgart, second edition, 1993.

[19] V. E. Henson and P. S. Vassilevski. Element-free AMG-e: General algorithms for computing interpolation weights. Technical report UCRL–VG–138290, Lawrence Livermore National Laboratory, March 2000.

[20] J. E. Jones and P. S. Vassilevski. AMGe based on agglomeration. Technical report UCRL–JC–135441, Lawrence Livermore National Laboratory, August 1999. to appear in SIAM J. Sci. Comput.

[21] Y. Notay. Optimal V-cycle algebraic multilevel preconditioner. *Numer. Lin. Alg. w. Appl.*, 5(5):441–459, 1998.

[22] Y. Notay. Using approximate inverses in algebraic multigrid methods. *Numer. Math.*, 80:397–417, 1998.

[23] Y. Notay. A robust algebraic multilevel preconditioner for non symmetric m-matrices. *Numer. Lin. Alg. w. Appl.*, 7:243–267, 2000.

[24] H. Regler and U. Rüde. Layout optimization with algebraic multigrid methods. SFB-Report 342/11/93 A, TU München, Institut für Informatik, 1993.

[25] A. Reusken. Fourier analysis of a robust multigrid method for convection–diffusion equations. *Numer. Math.*, 71:365–397, 1995.

[26] A. Reusken. A multigrid method based on incomplete Gaussian elimination. *Numer. Lin. Alg. w. Appl.*, 3:369–390, 1996.

[27] A. Reusken. On a robust multigrid solver. *Computing*, 56:303–322, 1996.

[28] A. Reusken. Approximate cyclic reduction preconditioning. In W. Hackbusch and G. Wittum, editors, *Multigrid Methods 5, Proceedings of the Fifth European Multigrid Conference*, pages 243–259. Springer-Verlag, 1998.

[29] A. Reusken. On the approximate cyclic reduction preconditioner. *SIAM J. Sci. Comput.*, 21:565–590, 2000.

[30] J. Ruge and K. Stüben. Algebraic multigrid. In S. McCormick, editor, *Multigrid Methods*, pages 73–130. SIAM Publications, 1987.

[31] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[32] H. Schwarz. *Methode der finiten Elemente*. B.G. Teubner Stuttgart, 1991.

[33] A. van der Ploeg, E. Botta, and F. Wubs. Nested grids ILU–decomposition (NGILU). *J. Comput. Appl. Math.*, 66:515–526, 1996.

[34] P. Vanek, J. Mandel, and M.Brezina. Algebraic multigrid by smoothed aggregation for second order and forth order elliptic problems. *Computing*, 56:179–196, 1996.

[35] C. Wagner. Introduction to algebraic multigrid. Course Notes Version 1.0, University of Heidelberg, 1999.

[36] G. Wittum. *Filternde Zerlegungen — Schnelle Löser für Gleichungssysteme*. Teubner Skripten zur Numerik. B.G. Teubner Stuttgart, 1992.

[37] H. Yserentant. On the multisplitting of finte element spaces. *Numer. Math.*, 49:379–412, 1986.