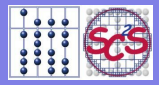
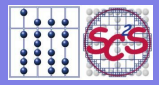
3) Algebraische Mehrgitterverfahren

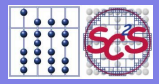


Vorbemerkungen

- **Mehrgitterverfahren** und ihre Verwandten (multi-scale, multi-resolution, multi-something) sind die effizientesten iterativen Lösungsverfahren für große, schwach besetzte lineare Gleichungssysteme. Im Gegensatz zu den klassischen Iterationsverfahren ist bei ihnen die Zahl der für eine bestimmte Fehlerreduktion erforderlichen Iterationsschritte nicht von der Systemgröße abhängig.
- Die Systemmatrizen stammen typischerweise von der Diskretisierung partieller Differentialgleichungen – es gibt also einen geometrischen Kontext (benachbarte Gitterpunkte, grobe und feine Gitter, hohe und niedrige Fehlerfrequenzen). Deswegen nennt man die klassischen Mehrgitterverfahren auch **geometrische Mehrgitterverfahren**.



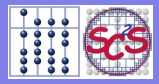
- Was aber tun, wenn dieser Kontext fehlt? Wenn man nur ein System $Ax = b$ hat und nichts über die Herkunft der Matrix weiß? Wenn es also evtl. nie Gitterpunkte etc. gab? Hierfür wurden **algebraische Mehrgitterverfahren (AMG)** entwickelt – quasi als hoch effiziente Black-box-Löser, die sich nur auf die Matrixeinträge abstützen. Mit ihnen wollen wir uns in diesem dritten Kapitel beschäftigen.
- Doch der Reihe nach – zunächst ein kleiner Rückblick (AWR1, Modellbildung und Simulation) auf bzw. Steilkurs über
  - die Diskretisierung von PDEs (wo kommen die Matrizen her?),
  - klassische Iterationsverfahren (wie löste man die Systeme in Vor-Mehrgitterzeiten?) und
  - das (geometrische) Mehrgitterprinzip.



3.1. Das Mehrgitterprinzip

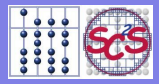
Finite Differenzen Verfahren für PDE

- gegeben: Gebiet
$$\Omega \subseteq \mathbb{R}^d,\ d \in \{1,2,3\}$$
- darauf wird definiert: (regelmäßiges) **Gitter** Ω_h
- Gitterabstand oder **Maschenweite**
$$h = (h_x, h_y, h_z)$$
- ersetze Ableitungen durch Differenzenquotienten:
  - erste Ableitungen: **Vorwärts**-, **Rückwärts**- oder **zentrale** Differenzen
$$\frac{\partial u}{\partial x}(\xi) \doteq \frac{u(\xi + h_x) - u(\xi)}{h_x}, \frac{u(\xi) - u(\xi - h_x)}{h_x}, \frac{u(\xi + h_x) - u(\xi - h_x)}{2h_x}$$
  - zweite Ableitungen: Standard **3-Punkt-Stern**
$$\frac{\partial^2 u}{\partial x^2} \doteq \frac{u(\xi + h_x) - 2u(\xi) + u(\xi - h_x)}{h_x^2}$$
  - Laplace-Operator in 2D oder 3D: 5-Punkt- oder 7-Punkt-Stern
  - Es gibt auch breitere Sterne (Involvierung von mehr Nachbarn).



Finite Differenzen Verfahren (2)

- in jedem inneren Gitterpunkt:
  - setze eine **Differenzengleichung** an
  - Unbekannte (**Freiheitsgrade**): diskrete Näherungswerte für die Funktionswerte in den Gitterpunkten
  - ein Freiheitsgrad pro Gitterpunkt und pro (skalarer) unbekannter Größe
- in Punkten auf dem oder nahe am Rand:
  - konkrete Gestalt der Gleichung hängt von Randbedingungen ab:
    - **Dirichlet**: keine Differenzengleichung in Randpunkten (dort ist Funktionswert ja vorgegeben, somit keine Unbekannte)
    - **Neumann**: spezielle Gestalt der Differenzengleichung am Rand (Einbau der Randbedingungen)
- Diskretisierung führt auf lineares Gleichungssystem (LGS)
  - im Allgemeinen dünn besetzt
  - schnelle (iterative) Lösungsverfahren lebenswichtig

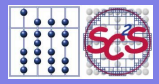


Finite Differenzen Verfahren (3)

- einfaches Beispiel: Poisson-Gleichung auf dem Einheitsquadrat

$$-\Delta u = f \quad \text{on} \quad]0,1[^2$$

- äquidistantes quadratisches Gitter: $h = h_x = h_y = 1/N$
- Anzahl der Freiheitsgrade (Unbekannten): $M = (N-1)^2$
- Standard-5-Punkt-Stern ergibt lineares System $Ax = b$ mit
  - $M \times M$-Matrix A (die punktweisen Differenzengleichungen)
  - M-Vektoren b (rechte Seite) und x (gewünschte Werte von u)
- A ist eine **dünn besetzte Matrix (sparse matrix)** mit Bandstruktur; nach Multiplikation mit h^2 :
  - **Dirichlet Randbedingungen:**
    - alle Diagonalelemente sind 4, pro Zeile 2 bis 4 Nicht-Nullen (-1)
    - Randwerte auf die rechte Seite
  - **Neumann Randbedingungen** (z.B., verschiedene Möglichkeiten):
    - Diagonalelemente sind 2 oder 3 nahe dem Rand und 4 sonst; dementsprechend viele – 1en (2 bis 4) in jeder Zeile
    - Paare $(1,-1)$ entlang dem Rand auf die rechte Seite

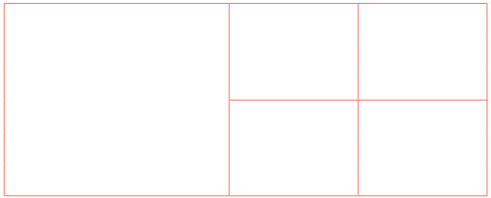


Finite Differenzen Verfahren (4)

- Genauigkeit typischerweise quadratisch:

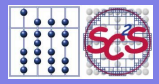
$$\|u^{\text{berechnet}} - u^{\text{exakt}}\| = O(h^2) = O(N^{-2})$$

- **Fluch der Dimension**:
  - es werden $O(N^d)$ Punkte bei d Dimensionen benötigt
- Ansatzpunkte für Verbesserungen:
  - Sterne höherer Ordnung (kubisch, quartisch, ...):
    - mehr als zwei benachbarte Punkte pro Raumrichtung heranziehen
    - Problem: Matrix wird voller (weniger Nicht-Nullen)
  - größere Sparsamkeit mit Gitterpunkten:
    - lokal verfeinerte Gitter verwenden (adaptive Gitter)
    - Problem: Vorgehensweise an den Nahtstellen – welcher Wert?



Standard-Iterationsverfahren für LGS

- **iterative** Lösung großer (dünn besetzter) LGS:
  - eine der zentralen numerischen Aufgabenstellungen in der numerischen Simulation
  - treten bei der Diskretisierung von ODE (RWP) und PDE auf
- **direkte** Löser oft nicht wettbewerbsfähig:
  - Zahl der Unbekannten zu groß (vgl. PDE in 3D)
  - klassische Elimination zerstört die (dünne) Struktur der Matrix
  - wozu exakte Lösung bei Approximationen? (gilt insb. im nichtlinearen Fall, wo ein LGS in jedem äußeren Iterationsschritt zur Behandlung der Nichtlinearität auftritt)
  - Ziel: Performance nach der Art „für 3 Stellen braucht man 10 Schritte“ – unabhängig von der Zahl der Unbekannten
- typisch jedoch für die klassischen Iterationsverfahren:
  - Konvergenzgeschwindigkeit verschlechtert sich mit wachsender Problemgröße!



Grundlegendes zu Iterationsverfahren

- betrachte Iterationsverfahren, starte bei $x^{(0)} \in \mathbb{R}^n$ und ende (hoffentlich) nahe bei Lösung x von $Ax = b$:

$$x^{(0)} \to x^{(1)} \to \ldots \to x^{(i+1)} \to \ldots \to \lim_{i \to \infty} x^{(i)} = x$$

- Konvergenzgeschwindigkeit:

$$\|x - x^{(i+1)}\| < \gamma \cdot \|x - x^{(i)}\|^s$$

  für ein $0 < \gamma < 1$, Konvergenzordnung s

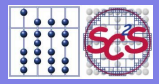
- typisches Verhalten einfacher Iterationsverfahren für LGS:

$$s = 1, \quad \gamma = O(1 - n^{-k}), \quad k \in \{0, 1, 2, ...\}$$

  (n : Anzahl der Gitterpunkte)

- Strategie: suche Verfahren mit
  - nur $O(n)$ arithmetischen Operationen pro Iterationsschritt (Kosten; so viel Aufwand wohl mindestens nötig)
  - Konvergenzverhalten gemäß $\gamma < 1$ – const. (Nutzen)

- zwei große Familien: **Relaxations**- und **Krylov-Raum**-Verfahren

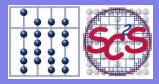


Relaxationsverfahren

- manchmal auch **Glätter** genannt:
  - **Richardson**-Iteration
  - **Jacobi**-Iteration
  - **Gauß-Seidel**-Iteration
  - **Überrelaxation (SOR)** oder **gedämpfte** Methoden
- Ansatzpunkte:
  - **Fehler** e der momentanen Näherung (Ursache): unbekannt
  - **Residuum** r der momentanen Näherung (Wirkung): ermittelbar

$$r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = A(x - x^{(i)}) = -Ae^{(i)}$$

    (mangels Besserem als Fehlerindikator verwendet)
- wie Residuum für verbesserte Approximation verwerten?
  - **Richardson**: Residuum direkt als Korrektur
  - **Jacobi/Gauß-Seidel**: eine Komponente von r zu Null machen
  - **SOR/gedämpft**: wie zuvor, aber Korrektur etwas zu hoch / niedrig



Wichtige Relaxationsverfahren

- **Richardson-Iteration**:

```
for i = 0,1,...
        for k = 1,...,n:
```

$$x_k^{(i+1)} := x_k^{(i)} + r_k^{(i)}$$

  Hier wird einfach das Residuum $r^{(i)}$ komponentenweise als Korrektur für die aktuelle Näherung $x^{(i)}$ herangezogen.

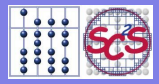
- **Jacobi-Iteration**:

```
for i = 0,1,...
        for k = 1,...,n:
        for k = 1,...,n:
```

$$y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}$$

$$x_k^{(i+1)} := x_k^{(i)} + y_k$$

  - In jedem Teilschritt k eines Schritts i wird eine Korrektur y_k berechnet und gespeichert.
  - Sofort angewendet, würde diese zum (momentanen) Verschwinden der k-Komponente des Residuums $r^{(i)}$ führen (leicht durch Einsetzen zu verifizieren).
  - Gleichung k wäre mit dieser aktuellen Näherung für x somit exakt gelöst – ein Fortschritt, der im folgenden Teilschritt zu Gleichung $k+1$ natürlich gleich wieder verloren ginge.
  - Allerdings werden diese Komponentenkorrekturen nicht sofort, sondern erst am Ende eines Iterationsschritts durchgeführt (zweite k-Schleife).



Wichtige Relaxationsverfahren (2)

- **Gauß-Seidel-Iteration**:

```
for i = 0,1,...
    for k = 1,...,n:
```

$$r_k^{(i)} := b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^{n} a_{kj} x_j^{(i)}$$

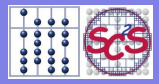
$$y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}, \quad x_k^{(i+1)} := x_k^{(i)} + y_k$$

  - Hier wird also dieselbe Korrektur wie beim Jacobi-Verfahren berechnet, der Update wird jetzt allerdings immer sofort und nicht erst am Ende des Iterationsschritts vollzogen.
  - Damit liegen beim Update von Komponente k für die Komponenten 1 bis $k-1$ bereits die modifizierten neuen Werte vor.

- Manchmal führt in jeder der drei skizzierten Methoden ein **Dämpfen** (Multiplikation der Korrektur mit einem Faktor $0 < \alpha < 1$) bzw. eine **Überrelaxation** (Faktor $1 < \alpha < 2$) zu einem besseren Konvergenzverhalten:

$$x_k^{(i+1)} := x_k^{(i)} + \alpha y_k \,.$$

  - Im Gauß-Seidel-Fall ist vor allem die Version mit $\alpha > 1$ gebräuchlich, man spricht hier von **SOR-Verfahren** (**Successive Over Relaxation**).
  - Im Jacobi-Fall wird dagegen meistens gedämpft.



Diskussion: Additive Zerlegung der Systemmatrix

- Für eine kurze Konvergenzanalyse der obigen Verfahren benötigen wir eine algebraische Formulierung (anstelle der algorithmischen).
- Alle gezeigten Ansätze basieren auf der einfachen Idee, die Matrix A als Summe $A = M + (A - M)$ zu schreiben, wobei $Mx = b$ sehr einfach zu lösen und der Unterschied $A - M$ bzgl. einer Matrixnorm nicht zu groß sein sollte.
- Mit Hilfe eines solchen geeigneten M werden sich Richardson-, Jacobi-, Gauß-Seidel- und SOR-Verfahren schreiben lassen als

$$Mx^{(i+1)} + (A - M)x^{(i)} = b$$

bzw., nach $x^{(i+1)}$ aufgelöst,

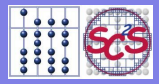
$$x^{(i+1)} := M^{-1}b - M^{-1}(A-M)x^{(i)} = M^{-1}b - (M^{-1}A - I)x^{(i)} = x^{(i)} + M^{-1}r^{(i)}\,.$$

- Darüber hinaus zerlegen wir A additiv in seinen Diagonalteil D_A, seinen strikten unteren Dreiecksteil L_A sowie seinen strikten oberen Dreiecksteil U_A:

$$A =: L_A + D_A + U_A\,.$$

Damit können wir die folgenden Beziehungen zeigen:

- Richardson: $M := I$,
- Jacobi: $M := D_A$,
- Gauß-Seidel: $M := D_A + L_A$,
- SOR: $M := \frac{1}{\alpha} D_A + L_A$.

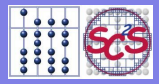


Diskussion: Additive Zerlegung der Systemmatrix (2)

- Bei Betrachtung der algorithmischen Formulierungen für das Richardson- sowie das Jacobi-Verfahren erweisen sich die ersten beiden Gleichungen als offensichtlich:
  - Bei Richardson wird das Residuum direkt als Korrektur genommen, als Vorfaktor ergibt sich somit die Identität I.
  - Bei Jacobi wird das Residuum durch das Diagonalelement dividiert, als Vorfaktor ergibt sich somit die Inverse des Diagonalanteils D_A.
- Weil die Gauß-Seidel-Iteration ein Spezialfall des SOR-Verfahrens ist ($\alpha = 1$), reicht es aus, obige Formel für M für den allgemeinen SOR-Fall zu zeigen. Aus dem Algorithmus folgt unmittelbar

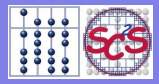
$$
\begin{aligned}
& x_k^{(i+1)} := x_k^{(i)} + \alpha \left(b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^{n} a_{kj} x_j^{(i)} \right) / a_{kk} \\
\Leftrightarrow \quad & x^{(i+1)} := x^{(i)} + \alpha D_A^{-1} \left(b - L_A x^{(i+1)} - (D_A + U_A) x^{(i)} \right) \\
\Leftrightarrow \quad & \frac{1}{\alpha} D_A x^{(i+1)} = \frac{1}{\alpha} D_A x^{(i)} + b - L_A x^{(i+1)} - (D_A + U_A) x^{(i)} \\
\Leftrightarrow \quad & \left(\frac{1}{\alpha} D_A + L_A \right) x^{(i+1)} + \left((1 - \frac{1}{\alpha}) D_A + U_A \right) x^{(i)} = b \\
\Leftrightarrow \quad & M x^{(i+1)} + (A - M) x^{(i)} = b \,,
\end{aligned}
$$

  womit die Behauptung für das SOR-Verfahren bewiesen ist.



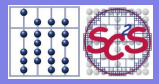
Diskussion: Allgemeines Konvergenzverhalten

- Was die Konvergenz angeht, so gibt es zwei unmittelbare Konsequenzen aus dem Ansatz
$$Mx^{(i+1)} + (A - M)x^{(i)} = b :$$
  - Falls die Folge $\left(x^{(i)}\right)$ konvergiert, dann ist der Grenzwert die exakte Lösung x unseres Systems $Ax = b$.
  - Für die Analyse werde angenommen, dass die Iterationsmatrix $-M^{-1}(A-M)$ (d.h. die Matrix, die auf $e^{(i)}$ angewandt wird, um $e^{(i+1)}$ zu erhalten; s.u.) symmetrisch sei. Dann ist der Spektralradius ρ (d.h. der betragsgrößte Eigenwert) die für das Konvergenzverhalten entscheidende Größe:
$$\left(\forall x^{(0)} \in \mathbb{R}^n : \lim_{i\to\infty} x^{(i)} = x = A^{-1}b\right) \quad \Leftrightarrow \quad \rho < 1 .$$
Um das zu sehen, subtrahiere man $Mx + (A - M)x = b$ von der Gleichung ganz oben:
$$Me^{(i+1)} + (A - M)e^{(i)} = 0 \quad \Leftrightarrow \quad e^{(i+1)} = -M^{-1}(A - M)e^{(i)} .$$
Wenn alle Eigenwerte betragsmäßig kleiner 1 sind und somit $\rho < 1$ gilt, werden alle Fehlerkomponenten in jedem Iterationsschritt reduziert. Im Falle $\rho > 1$ wird sich mindestens eine Fehlerkomponente aufschaukeln.
  - Ziel bei der Konstruktion iterativer Verfahren muss natürlich ein möglichst kleiner Spektralradius der Iterationsmatrix sein (möglichst nahe bei Null).



Diskussion: Konvergenzaussagen

- Es gibt eine Reihe von Resultaten zur Konvergenz der verschiedenen Verfahren, von denen einige bedeutende erwähnt werden sollen:
  - Notwendig für die Konvergenz des SOR-Verfahrens ist $0 < \alpha < 2$.
  - Falls A positiv definit ist, dann konvergieren sowohl das SOR-Verfahren (für $0 < \alpha < 2$) als auch die Gauß-Seidel-Iteration.
  - Falls A und $2D_A - A$ beide positiv definit sind, dann konvergiert das Jacobi-Verfahren.
  - Falls A strikt diagonal dominant ist (d. h. $a_{ii} > \sum_{j \neq i} \mid a_{ij} \mid$ für alle i), dann konvergieren das Jacobi- und das Gauß-Seidel-Verfahren.
  - In bestimmten Fällen lässt sich der optimale Parameter α bestimmten (ρ minimal, so dass Fehlerreduktion pro Iterationsschritt maximal).
- Die Gauß-Seidel-Iteration ist nicht generell besser als das Jacobi-Verfahren (wie man aufgrund des sofort vollzogenen Updates vermuten könnte). Es gibt Beispiele, in denen Erstere konvergiert und Letzteres divergiert, und umgekehrt. In vielen Fällen kommt das Gauß-Seidel-Verfahren jedoch mit der Hälfte der Iterationsschritte des Jacobi-Verfahrens aus.

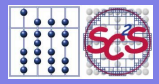


Zum Spektralradius typischer Iterationsmatrizen

- Offensichtlich ist ρ nicht nur entscheidend für die Frage, ob die Iterationsvorschrift überhaupt konvergiert, sondern auch für deren Qualität, also ihre Konvergenzgeschwindigkeit: Je kleiner ρ ist, desto schneller werden alle Komponenten des Fehlers $e^{(i)}$ in jedem Iterationsschritt reduziert.
- In der Praxis haben die obigen Resultate zur Konvergenz leider eher theoretischen Wert, da ρ oft so nahe bei 1 ist, dass – trotz Konvergenz – die Anzahl der erforderlichen Iterationsschritte, bis eine hinreichende Genauigkeit erreicht ist, viel zu groß ist.
- Ein wichtiges Beispielszenario ist die Diskretisierung partieller Differentialgleichungen:
  - Typisch ist, dass ρ von der Problemgröße n und somit von der Auflösung h des zugrunde liegenden Gitters abhängt, also beispielsweise

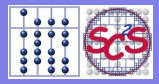
$$\rho \;=\; O(1-h_l^2) \;=\; O\left(1-\frac{1}{4^l}\right)$$

    bei einer Maschenweite $h_l = 2^{-l}$.
  - Dies ist ein gewaltiger Nachteil: Je feiner und folglich auch genauer unser Gitter ist, umso erbärmlicher wird das Konvergenzverhalten unserer iterativen Verfahren. Bessere iterative Löser sind also ein Muss!



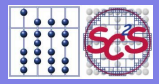
Das Mehrgitterprinzip

- starting point: **Fourier mode analysis** of the errors
  - decompose the error $e^{(i)} = x^{(i)} - x$ into its Fourier components (Fourier transform)
  - observe how they change/decrease under a standard relaxation like Jacobi or Gauß-Seidel (in a two-band sense):
    - The *high* frequency part (with respect to the underlying grid) is reduced quite quickly.
    - The *low* frequency part (w.r.t. the grid) decreases only very slowly; actually the slower, the finer the grid is.
  - This behaviour is annoying
    - the low frequencies are not expected to make troubles, but we can hardly get rid of them on a fine grid;
  - but also encouraging
    - the low frequencies can be represented and, hopefully, tackled on a coarser grid – there is no need for the fine resolution.



Ein einfaches Beispiel

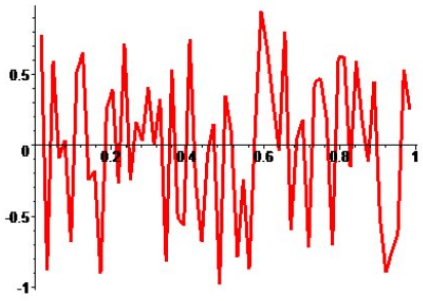
- 1D Laplace equation, $u(0) = u(1) = 0$ (exact solution 0)
- equidistant grid, 65 points, 3-point stencil, damped Jacobi method with damping parameter 0.5
- start with random values in $[0, 1]$ for u in the grid points
- After 100 (!) steps, there is still a maximum error bigger than 0.1 due to low-frequency components!
- therefore the name **smoothers** for relaxation schemes:
  - They reduce the strongly oscillating parts of the error quite efficiently.
  - They, thus, produce a **smooth** error which is very resistent.
- the idea: work on grids of different resolution and combine the effects in an appropriate way



Ein einfaches Beispiel (2)

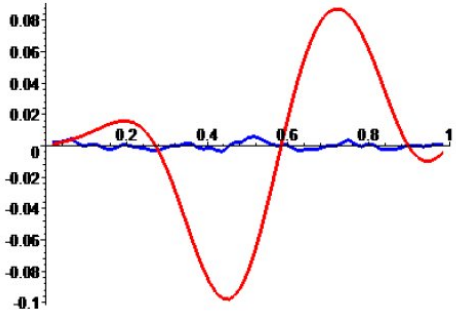
Random-Startfehler

zum Beispielproblem



Fehler nach 100 Jacobi-Schritten (rot)

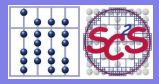
Fehler nach 2 Mehrgitter-Schritten (blau)



Grobgitterkorrektur

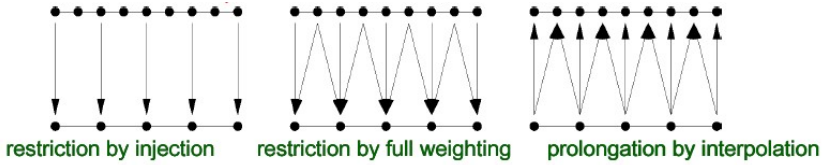
- sequence of equidistant grids on our domain: $\Omega_l,\ l = 1, 2, \ldots, L$, with mesh width $h_l = 2^{-l}$
- let A_l, b_l denote corresponding matrix, right-hand side, ...
- combine work on two grids with a **correction scheme**:

smooth the current solution x_l ;
form the residual $r_l = b_l - A_l x_l$;
restrict r_l to the coarse grid Ω_{l-1} ;
provide a solution to $A_{l-1}\, e_{l-1} = r_{l-1}$;
prolongate e_{l-1} to the fine grid Ω_l;
add the resulting correction to x_l;
if necessary, smooth again ;



Grobgitterkorrektur (2)

- the different steps of this **2-grid algorithm** :
  - the **pre-smoothing**: reduce high-frequency error components, smooth error, and prepare residual for transfer to coarse grid
  - the **restriction**: transfer from fine grid to coarse grid
    - **injection** : inherit the coarse grid values and forget the others
    - **(full) weighting** : apply some averaging process
  - the **coarse grid correction**: provide an (approximate) solution on the coarse grid (direct, if coarse enough; some smoothing steps otherwise)
  - the **prolongation**: transfer from coarse grid to fine grid
    - usually some **interpolation** method



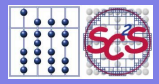
  - the **post-smoothing**: sometimes reasonable to avoid new high-frequency error components

Der V-Zyklus

- recursive application of 2-grid scheme leads to **multigrid methods**
- there, the coarse grid equation is solved by coarse grid correction, too; the resulting algorithmic scheme is called **V-cycle** :

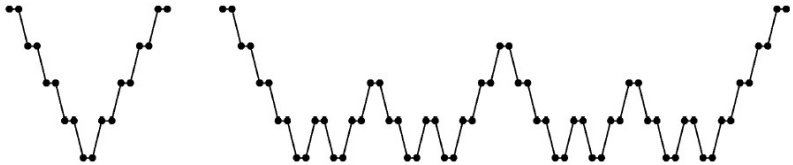
smooth the current solution x_l ;
form the residual $r_l = b_l - A_l x_l$;
restrict r_l to the coarse grid Ω_{l-1} ;
solve $A_{l-1}\, e_{l-1} = r_{l-1}$ by coarse grid correction ;
prolongate e_{l-1} to the fine grid Ω_l;
add the resulting correction to x_l;
if necessary, smooth again ;

- on the coarsest grid: direct solution
- number of smoothing steps: typically small (1 or 2)



Weitere Mehrgitterzyklen

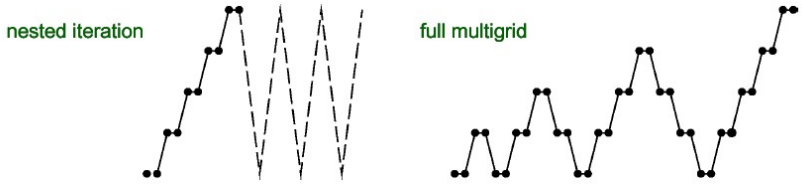
- the V-cycle is not the only multigrid scheme:
  - the **W-cycle**: after each prolongation, visit the coarse grid once more, before moving on to the next finer grid



  - W-cycle is sometimes advantageous with respect to speed of convergence

Nested Iteration und Full Multigrid

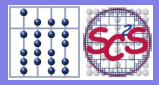
- two more famous multigrid schemes:
  - the **nested iteration**: start on coarsest grid Ω_1 , smooth, prolongate to Ω_2 , smooth, prolongate to Ω_3 , and so on, until finest grid is reached; now start V-cycle
  - **full multigrid**: replace ‘smooth´ steps above by ‘apply a V-cycle´; combination of improved start solution and multigrid solver



- multigrid idea is not limited to rectangular or structured grids: we just need a hierarchy of nested grids (works for triangles or tetrahedra, too)
- also without underlying geometry: **algebraic** multigrid methods

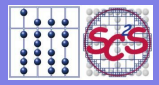
Grundlegende Konvergenzresultate

- **Cost** (storage and computing time):
  - $1D:\quad c \cdot n + c \cdot n/2 + c \cdot n/4 + c \cdot n/8 + \ldots \leq 2c \cdot n = O(n)$
  - $2D:\quad c \cdot n + c \cdot n/4 + c \cdot n/16 + c \cdot n/64 + \ldots \leq 4/3c \cdot n = O(n)$
  - $3D:\quad c \cdot n + c \cdot n/8 + c \cdot n/64 + c \cdot n/512 + \ldots \leq 8/7c \cdot n = O(n)$
  - i.e.: work on coarse grids is negligible compared to finest grid
- **Benefit** (speed of convergence):
  - always significant acceleration compared with pure use of smoother (relaxation method)
  - in most cases even ideal behaviour $\gamma = O(1 - \text{const.})$
  - effect:
    - constant number of multigrid steps to obtain a given number of digits
    - overall computational work increases only linearly with n



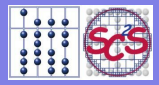
3.2. AMG

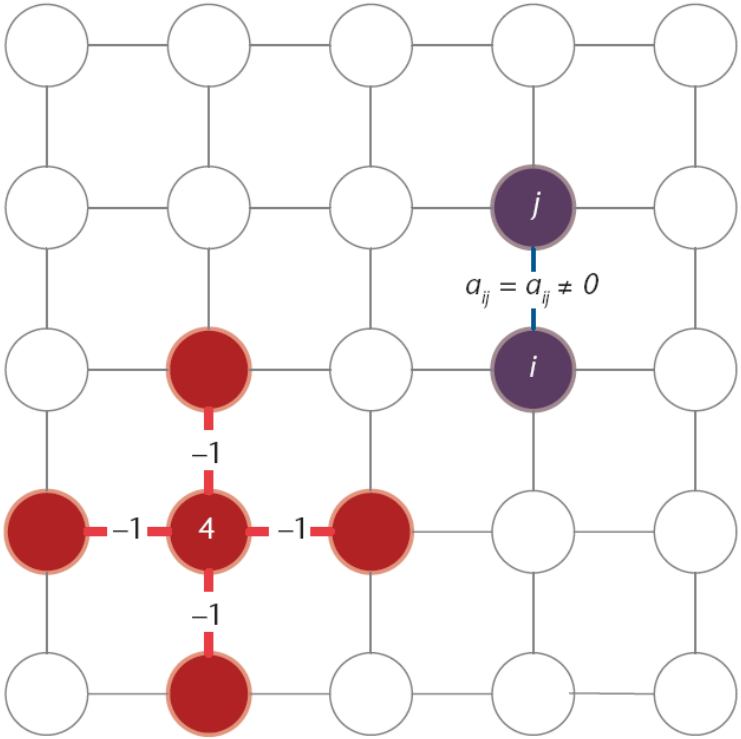
- Wie im geometrischen Fall wollen wir wieder das System $Ax = b$ mit einer reellen, symmetrischen und positiv-definiten Matrix A sowie rechter Seite b lösen. Weitere Kenntnisse über Struktur oder Herkunft von A setzen wir aber nicht mehr voraus.
- Für einen Mehrgitteralgorithmus benötigen wir wieder
  - einen **Glätter** bzw. ein **Relaxationsverfahren** und
  - eine **Grobgitterkorrektur**, bestehend aus den Komponenten **Restriktion**, **Grobgitterlöser** und **Prolongation**.
- Die Begriffe sind noch Geometrie-behaftet („grob“, etc.), die Algorithmen werden es jedoch nicht mehr sein. Insbesondere müssen wir unsere Frequenz-Vorstellung verlassen.
- AMG definiert grobe Gitter, Operatoren zum Gittertransfer sowie Grobgittergleichungen allein auf Grundlage der Matrixeinträge.
- Inzwischen gibt es auch hier zahlreiche Varianten; wir konzentrieren uns auf das „Ur-AMG“.
- Das algorithmische Grundmuster (Zweigitterverfahren, V-Zyklus) sieht dabei zunächst nahezu unverändert aus (siehe den vorigen Abschnitt).



AMG und Geometrie

- Achtung: Obwohl AMG ohne geometrischen Kontext auskommt, wird natürlich auch AMG sehr oft bei partiellen Differentialgleichungen (also mit geometrischem Kontext) eingesetzt. Insofern schlägt auch hier immer wieder eine geometrische Anschauung durch.
- Diese wird auch deutlich, wenn man sich den Adjazenzgraphen von A ansieht:
  - Knoten stehen für Unbekannte, d.h. Komponenten x_i des Lösungsvektors x;
  - Kanten sind eingetragen, falls $a_{i,j} \neq 0$ gilt.
  - Stammt A von einem 2D PDE-Gitter, dann stimmen Gitter und Graph oft unmittelbar überein (bzw. können entsprechend gezeichnet werden).

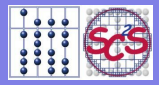




5x5 Gitter und Adjazenzgraph für die Laplace-Gleichung

Algebraische Glattheit

- Was bedeutet **glatt**, wenn der Bezug zu Geometrie und Frequenz fehlt?
  - **geometrisch glatt**: der übliche Glattheitsbegriff (nicht oszillierend etc.)
  - **algebraisch glatt**: schlicht und ergreifend alles, was der Glätter (bei AMG Gauß-Seidel oder Verwandtes) übrig lässt (das könnte jetzt auch geometrisch un-glatt sein!)
- Ein algebraisch glatter Fehler enthält i.A. vor allem Anteile von Eigenvektoren zu kleinen Eigenwerten (so genannten **kleinen Eigenmoden**):
  - Die Glättung dämpft also vor allem die großen Eigenmoden.
  - Die kleinsten Eigenmoden – der Beinahe-Kern von A – erfordern demnach die größte Aufmerksamkeit bei der Grobgitterkorrektur.

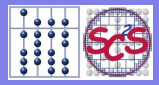


- ein einfaches Beispielproblem:

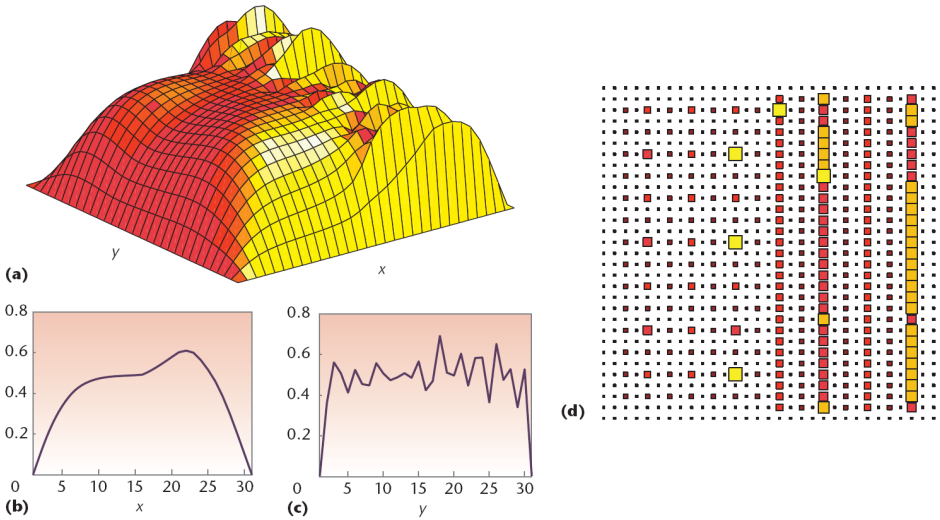
$$-au_{xx} - bu_{yy} = f \quad \text{auf} \quad [0,1]^2, \quad \text{Dirichlet-Rand},$$

wobei $a = b$ in der linken Hälfte ($x < 0.5$) und $a \gg b$ sonst

  - Konstante Funktionen liegen im Beinahe-Kern von A (Matrixstruktur!).
  - Geometrische Glattheit und Beinahe-Kern fallen in diesem Beispiel also zusammen.
  - Folglich sollte AMG hier in Richtung der geometrischen Glattheit vergröbern (siehe nächste Seite).



Beispiel zur algebraischen Glattheit



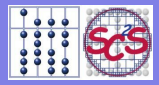
Algebraische Glattheit. (a) Fehler für das Beispielproblem nach sieben Gauss-Seidel-Durchläufen. (b) Der Fehler ist geometrisch glatt in x-Richtung, (c) oszilliert jedoch in y-Richtung in der rechten Hälfte. (d) In diesem Beispiel vergröbert AMG das Gitter in Richtung der geometrischen Glattheit. Größere Quadrate korrespondieren zu gröberen Gittern.

Grundkonzepte im klassischen AMG (C-AMG)

- **Glattheitsheuristik**: Glatte Fehler ändern sich nur langsam in Richtung relativ großer (negativer) Matrixeinträge.
  - Für kleine Eigenmoden v (Länge auf 1 normiert) kann man nämlich zeigen, dass

$$v^T A v \;=\; \sum_{i<j} (-a_{i,j}) \cdot (v_i - v_j)^2 \;\ll\; 1\,.$$

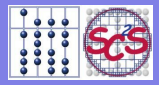
  - Ist $-a_{i,j}$ groß, so können sich v_i und v_j nur unwesentlich unterscheiden.



- **Verbindungsstärke**: Für eine gegebene Schranke $\theta, 0 < \theta \leq 1$, hängt die Variable x_i stark von x_j ab, falls gilt

$$-a_{i,j} \geq \theta \cdot \max_{k \neq i} \{-a_{i,k}\} .$$

  - Die Verbindungsstärke wird also relativ zum größten negativen Außerdiagonalelement in der Zeile gemessen; positive Außerdiagonalelemente gelten immer als schwache Verbindungen.
  - Trotz $A = A^T$ ist es möglich, dass x_i stark von x_j, x_j aber nur schwach von x_i abhängt (es gibt AMG-Varianten, die dies ausschließen; wir gehen von symmetrischer Verbindungsstärke aus).



Grobgitterwahl im C-AMG

- Das grobe Gitter ist immer eine Teilmenge des feinen Gitters.
- Die Wahl der Grobgitterpunkte erfolgt so, dass in Richtungen starker Matrixverbindungen vergröbert wird.
- Struktur des Algorithmus:
  1. ermittle A_s aus A durch Entfernen aller schwachen Verbindungen (bzw. zugehörigen Matrixeinträge)
  2. Durchgang 1: bestimmte eine Menge C von Grobgitterpunkten und eine Menge F (alle anderen Variablen), ausgehend von A_s
  3. Durchgang 2: bessere nach, falls die Interpolation dies erfordern sollte
- ein Beispiel:
  - 2D Laplace-Gleichung
  - Finite-Elemente-Diskretisierung auf uniformem Gitter
  - Standard 9-Punkte-Stern (8 in der Diagonalen, -1 außerhalb)
  - unabhängig von θ sind alle Verbindungen stark, folglich $A_s = A$

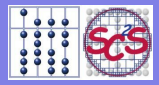
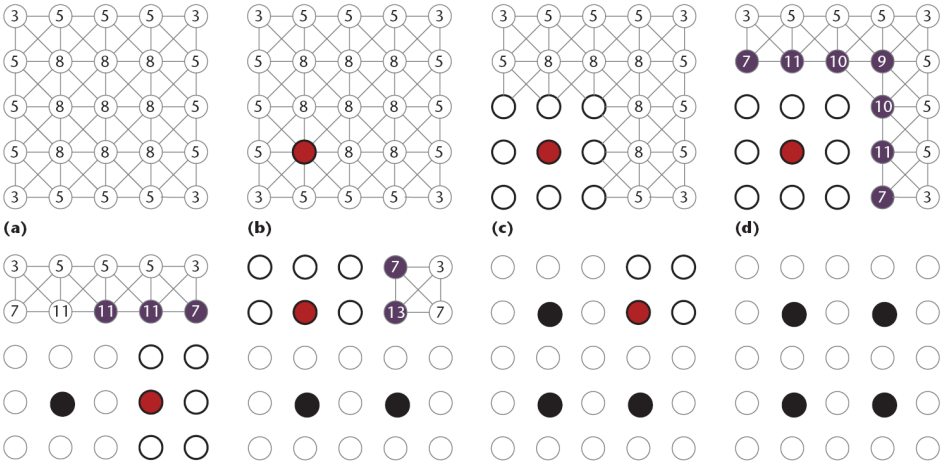
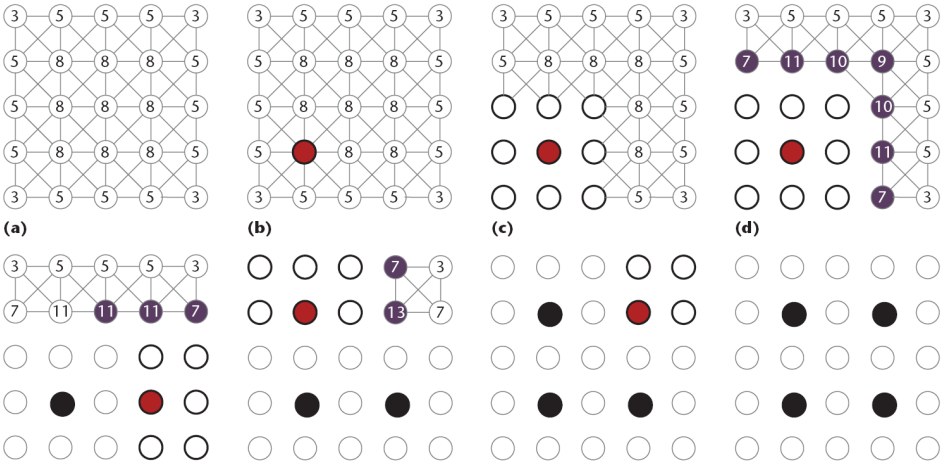


Illustration der C-AMG-Grobgitterwahl (Durchgang 1)



(a) jeder Knoten erhält die Anzahl seiner außerdiagonalen Verbindungen als Gewicht (Zahl der nach Entfernen der kleinen Einträge übrig gebliebenen außerdiagonalen Nicht-Nullen in der jeweiligen Matrixzeile)

(b) ein Punkt maximalen Gewichts wird als C-Punkt gewählt



(c) alle noch nicht F oder C zugeteilten Nachbarn des neuen C-Punkts werden zu F-Punkten

(d) jeder neue F-Punkt verhilft seinen noch nicht zugeteilten Nachbarn zu einem zusätzlichen Gewicht von 1

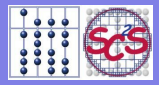
(...) untere Zeile: fahre fort, bis alle Punkte entweder C-Punkte oder F-Punkte sind

Interpolation

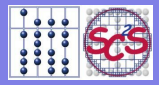
- Ausgangspunkt wieder: algebraisch glatter Fehler bedeutet Dominanz kleiner Eigenmoden v
- wegen $r^T r = e^T A^2 e \approx v^T A^2 v = \lambda^2 \ll 1$ (Länge von v auf 1 nromiert!) bedeutet dies zugleich auch kleine Residuen
- Zur Annäherung an die Interpolation gehen wir einfach mal von $r = Av = 0$ bzw. $r = Ae = 0$ (mit einem i.W. aus kleinen Eigenmoden bestehenden Fehler e) aus und erhalten für einen F-Punkt i

$$a_{i,i}e_i = -\sum_{j \in C_i} a_{i,j}e_j - \sum_{j \in F_i^s} a_{i,j}e_j - \sum_{j \in F_i^w} a_{i,j}e_j \,,$$

  wobei
  - C_i die C-Punkte mit starker Verbindung zu i (nur auf diese wird sich die Interpolation in i abstützen),
  - F_i^s die F-Punkte mit starker Verbindung zu i und
  - F_i^w alle Punkte mit schwacher Verbindung zu i bezeichnen.

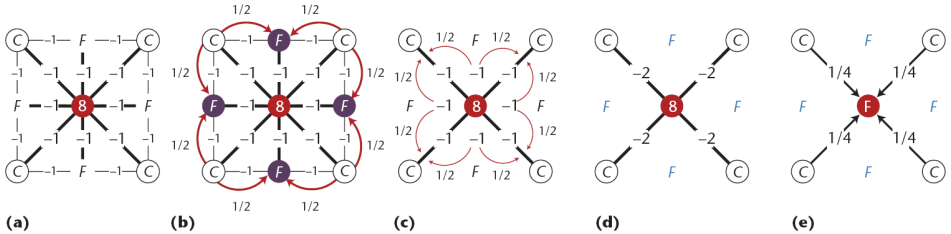


- Die Kunst besteht nun darin, die e_j aus der zweiten und dritten Teilsumme oben entweder den Punkten aus C_i oder gleich dem Punkt i zuzuschlagen.
- dazu zwei Beispiele



Prolongation im C-AMG – Beispiel 1

Standard 9-Punkt Finite-Element-Stern



(a) Standard 9-Punkt Finite-Element-Stern (bzw. Matrixbeziehungen)

(b) stark verbundene F-Punkte werden aus den benachbarten C-Punkten interpoliert (Gewichte ausgehend von den Matrixeinträgen)

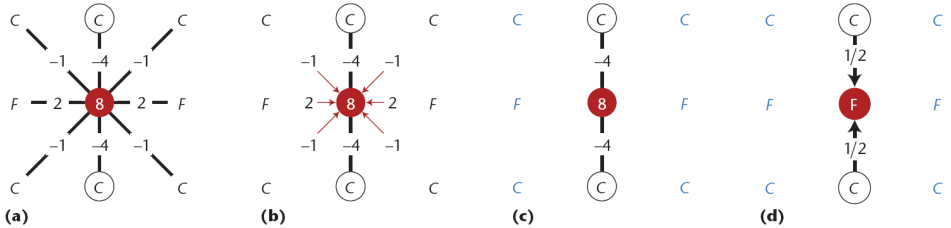
(c) die starken Beziehungen zu den F-Punkten werden dann (anteilig gemäß den Interpolationsgewichten) den Beziehungen zu den C-Punkten zugeschlagen

(d) es ergibt sich der kollabierte Stern ...

(e) ... und daraus die Interpolationsregel

Prolongation im C-AMG – Beispiel 2

anisotroper 9-Punkt Finite-Element-Stern

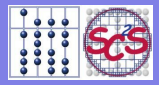


(a) anisotroper 9-Punkt Finite-Element-Stern (bzw. Matrixbeziehungen)

(b) schwache Beziehungen werden dem Diagonalelement zugeschlagen

(c) es ergibt sich der kollabierte Stern ...

(d) ... und daraus die Interpolationsregel



Restriktion und Grobgitteroperator im C-AMG

Hier gibt es nicht viel zu tun:

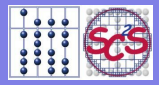
- **Restriktion**: wird im C-AMG als Transponierte des Prolongationsoperators definiert, also

$$R := P^T$$

- **Grobgitteroperator**:
  - Hier hat man weniger Spielraum als beim geometrischen Mehrgitterverfahren, wo man die gegebene PDE ja einfach auf dem groben Gitter diskretisieren kann. Wir wissen aber nicht, wo A herkommt und können den Erzeugungsprozess also auch nicht auf andere Auflösungen übertragen.
  - Deshalb wird im C-AMG stets der so genannte **Galerkin-Operator** gewählt, also

$$A_c := P^T A P .$$

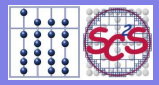
Damit haben wir alle Bausteine zusammen und können, wie gehabt, verschiedene Mehrgitterzyklen bauen.



Leistungsmerkmale

Fine grid	Iterations	Convergence factor	Coarse grids	Grid complexity*	Operator complexity†	Setup time‡	Solve time
31×31	9	0.19	5	1.6	1.7	—	—
61×61	10	0.23	6	1.6	1.6	0.01	0.02
121×121	9	0.23	8	1.6	1.7	0.05	0.07
241×241	9	0.23	9	1.6	1.7	0.25	0.32
481×481	9	0.23	12	1.7	1.7	1.02	1.27
961×961	11	0.29	13	1.7	1.7	4.42	6.28

Note: *The code used strength threshold $\theta = 0.4$ with $\nu_1 = \nu_2 = 1$ step of C-F Gauss-Seidel, and iterated until the relative residual was below 10^{-9}.* *Grid complexity *is the total number of grid points on all grids divided by the number of grid points on the fine grid.* †Operator complexity *is the total number of nonzeroes in the linear operators on all grids divided by the number of nonzeroes in the fine grid operator.* ‡Setup time *is the time required to choose coarse grids and build interpolation, restriction, and coarse-grid operators.*



References

Die Abbildungen sowie die Tabelle aus diesem Kapitel entstammen aus:
Robert D. Falgout. An introduction to algebraic multigrid computing. Computing in Science and Engineering, 8(6):24-33, 2006.