Lösen des Poisson-Problems mittels Finite-Differenzen-Diskretisierung und CG-Verfahren

Marisa Breßler und Anne Jeschke (PPI27)

07.02.2020

Inhaltsverzeichnis

1 Motivation

In unserem vorherigen Berichten haben wir das Poisson-Problem vorgestellt und einen numerischen Lösungsansatz aufgezeigt, der es durch eine Diskretisierung des Gebietes und des Laplace-Operators in das Lösen eines linearen Gleichungssystems überführt. Die dabei entstehende tridiagonale Blockmatrix ist dünn besetzt, d.h. nur wenige Einträge sind ungleich Null. Deswegen ist es sinnvoll, sie als sogenannte *sparse*-Matrix abzuspeichern. Dieser Speicherplatzvorteil geht jedoch beim Lösen des linearen Gleichungssystems mittels Gauß-Algorithmus und LU-Zerlegung verloren, da eine LU-Zerlegung einer dünn besetzten Matrix im Allgemeinen nicht dünn besetzt ist. Aufgrund dieses Umstandes erscheint es sinnvoll, das lineare Gleichungssystem nicht direkt, sondern iterativ zu lösen. Solche iterativen Lösungsverfahren (wie das Gesamtschrittverfahren von Jacobi, das Einzelschrittverfahren von Gauß-Seidel, das SOR-Verfahren und das CG-Verfahren) bieten gegenüber direkten Verfahren außerdem den Vorteil, dass deren Rechenaufwand im Verhältnis zur in der Praxis für gewöhnlich sehr großen Dimension der Blockmatrix recht gering ist. Da das CG-Verfahrens eine heute durchaus übliche Methode zum Lösen großer linearer Gleichungssysteme darstellt, wollen wir dieses im Folgenden im Rahmen unseres bisherigen Settings (Poisson-Problem und Finite-Differenzen-Diskretisierung) vorstellen und untersuchen.

2 Theoretische Grundlagen und Algorithmus

iterative Verfahren bestimmen Näherungslösung, via Grenzwert an exakte Lösung

für CG-Verfahren muss Matrix symmetrisch positiv definit sein
ist für unsere Blockmatrix der Fall, da sie symmetrisch ist, positive Diagonalelemente besitzt und irreduzibel diagonaldominant ist (> positiv definit)

während sich die anderen oben genannten iterativen Verfahren die Methode des Splitting zunutze machen, ist die Idee des CG-Verfahrens die folgende:
Lösung des linearen Gleichungssystems in ein Minimierungsproblem überführen
dazu wird folgende Funktion definiert: ...
diese hat genau an der Stelle x ihr einziges Minimum, wo auch das LGS $Ax = b$ seine Lösung hat
man konstruiert Abstiegsrichtungen so ..., dass Funktionswerte immer kleiner werden
...

3 Experimente und Beobachtungen

Im Folgenden wollen wir verschiedene Experimente präsentieren. Da wir im Gegensatz zu unserer Arbeit vom 03.01.2020 in hiesigen Rahmen das Gleichungssystem nicht direkt mittels Gauß-Algorithmus und LU-Zerlegung, sondern iterativ mittels CG-Verfahren lösen, gilt unser Interesse der Vorstellung der neuen Methode zum Ermitteln einer Lösung des Poisson-Problems sowie der Gegenüberstellung der beiden Verfahrensweisen.

3.1 Entwicklung des absoluten Fehlers pro Iteration

Während direkte Verfahren nach einer endlichen Anzahl an Rechenoperationen „die exakte Lösung“ liefern (dabei handelt es sich in Praxis nicht um die exakte analytische Lösung, da ein Computer aufgrund seiner begrenzten Rechenleistung im Allgemeinen immer bei der Verarbeitung von Zahlen Fehler produziert), ermitteln iterative Verfahren schrittweise eine immer bessere Approximation der gesuchten Lösung.
Diese sukzessive Annäherung an die exakte Lösung lässt sich an den folgenden drei Grafiken erkennen. Diese zeigen jeweils für den ein-, den zwei- und den dreidimensionalen Fall die Entwicklung des absoluten Fehlers pro Iteration. Der Graph schmiegt sich immer mehr der y-Achse an. Dieses asymptotische Fehlerverhalten ist charakteristisch für iterative Verfahren.

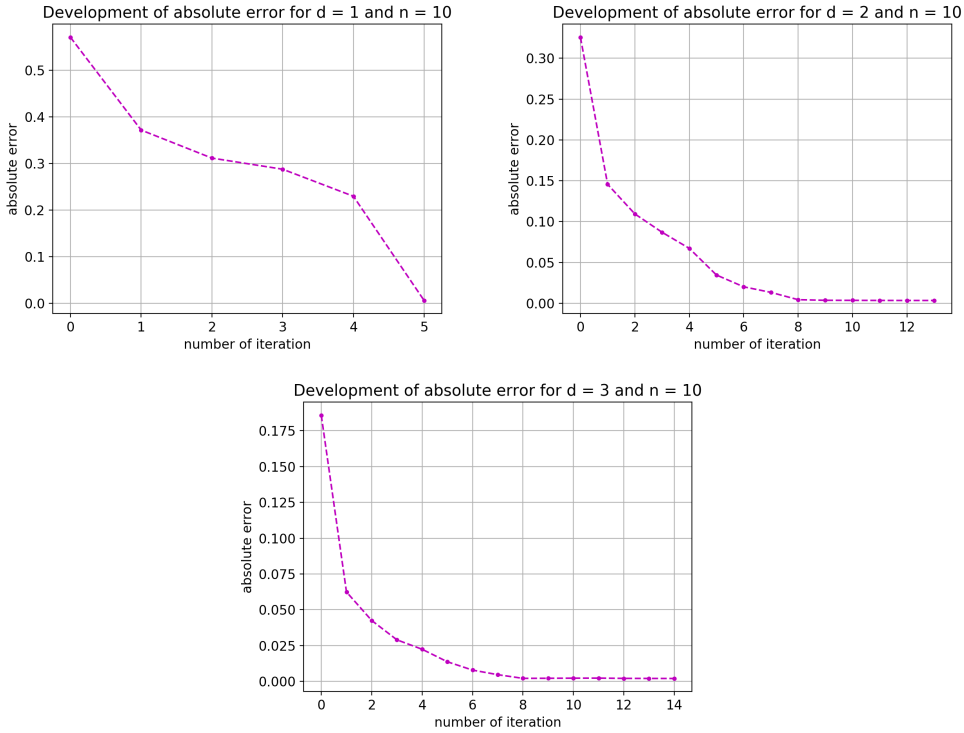

Abbildung 1: Entwicklung des absoluten Fehlers pro Iteration für $d \in \{1, 2, 3\}$ und $n = 10$

Darüber hinaus ist auffallend, dass weitaus weniger als $N = (n-1)^d$ Schritte vonnöten sind, um eine sehr gute Näherungslösung zu errechnen. Während wir im eindimensionalen Fall, $N = 9$, bereits nach der fünften Iteration eine ausreichend genaue Lösung haben, benötigen wir im zweidimensionalen Fall, $N = 81$, nur 13 und im dreidimensionalen Fall, $N = 729$ lediglich eine mehr, nämlich 14 Iterationen. (Die große Konvergenzgeschwindigkeit lässt sich auf die gute Kondition der tridiagonalen Blockmatrix zurückführen.) Doch bei allen ist der Fehler schon nach acht Schritten nur noch sehr gering. Dass unser Programm dennoch weiterrechnet, ist den Umstand geschuldet, dass noch keiner der eingegebenen Abbruchparameter zum Tragen kam. Da die Rechenschritte bei direkten Verfahren wie gesagt im Voraus bekannt und begrenzt sind, bei iterativen jedoch nicht, benötigen letztere Abbruchbedingungen, die den Schleifendurchlauf des Algorithmus zum Ende bringen. Unsere Implementierung der Methode der konjugierten Gradienten (CG) enthält die folgenden drei Abbruchbedingungen. „eps“: Abbruch, falls die Norm des Residuums eine vorgegebene Grenze unterschreitet, „max_iter“: Abbruch, falls eine vorgegebene Anzahl an Iterationen erreicht ist, „min_red“: Abbruch, falls das Verfahren „feststeckt“, d.h. falls der Abstand zwischen den letzten beiden Residuen eine vorgegebene Grenze unterschreitet. ...

3.2 Konvergenzverhalten für ein festes Epsilon

gleiches Konvergenzverhalten wie LU
Konvergenz abh. von Diskretisierung, nicht von Lösungsverfahren

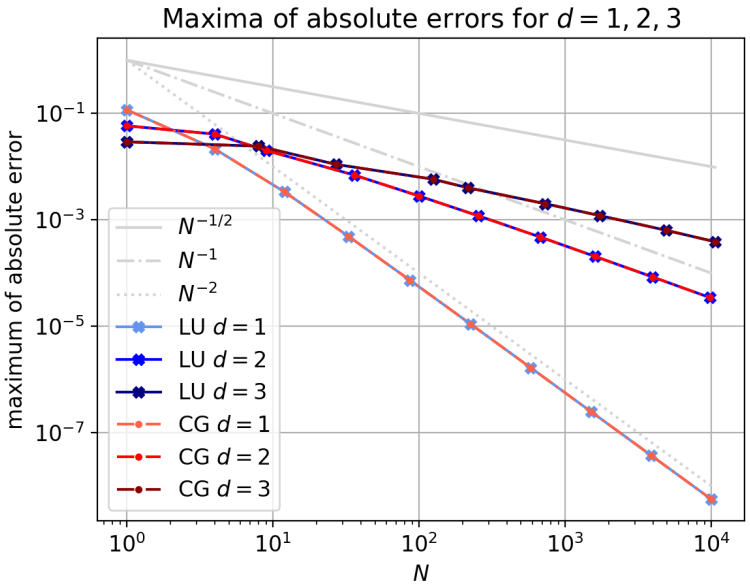

Abbildung 2: Vergleich Konvergenzverhalten LU und CG mit $\epsilon = 10^{-8}$ für $d \in \{1, 2, 3\}$

3.3 Konvergenzverhalten für variable Epsilon

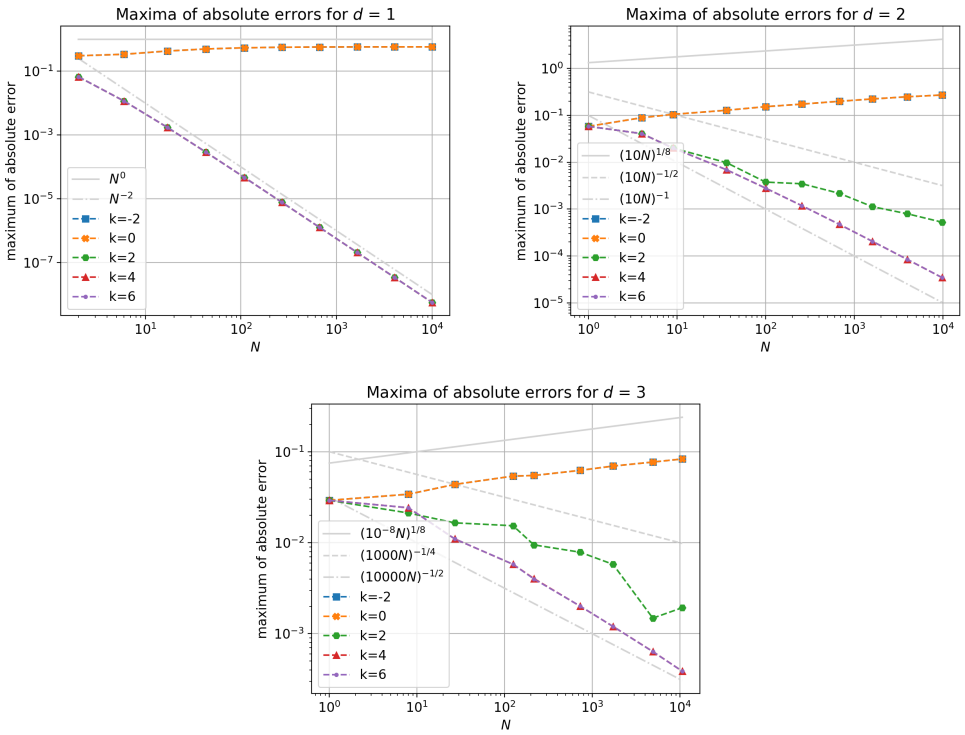

Abbildung 3: Konvergenzverhalten mit $\epsilon^{(k)} = n^{-k}$ für $d \in \{1, 2, 3\}$

4 Abschließende Worte