Mehrgittermethode

Grundlage der computergestützten Wissenschaften

Ulrich Rüde

Einleitung

„Die Mehrgittermethode stellt einen der größten Fortschritte im Bereich der Numerik in den letzten Jahrzehnten dar." Dieser Satz stammt von Jens Volkert aus dem „aktuellen Schlagwort" des Informatikspektrums vom April 1989 [13], und er gilt unverändert auch heute, 30 Jahre später. Was sind Mehrgitterverfahren, so dass sie – ganz untypisch für die Informatik – über Jahrzehnte hinweg ein Dauerbrenner sind? Wie schon Jens Volkert vorhergesagt hatte, erklärt sich ihre Bedeutung aus der computergestützten Wissenschaft (engl. Computational Science and Engineering, CSE). Mehrgitterverfahren sind ein zentraler Bestandteil der Algorithmik, die realitätsgetreue Computersimulationen möglich macht.

Wissenschaftliche Erkenntnisse beruhen heute zunehmend auf Computermodellen. Beispiele findet man in der Physik, der Chemie, der Biomedizin, den Geowissenschaften, oder der Astronomie, aber das sind längst nicht alle. Überall helfen Computersimulationen, Prozesse und Systeme zu analysieren, die sich aufgrund der Zeit- und Raumskalen einer direkten menschlichen Beobachtung entziehen. Gleichzeitig sind Computersimulationen Grundlage in vielen Ingenieursdisziplinen, wenn z. B. neue Materialen am Computer entworfen werden oder wenn sichere und leise Flugzeuge entwickelt werden. Computersimulationen werden genutzt, um die Energieausbeute aus Windfarmen zu optimieren und um die Ausbreitung von Schadstoffen im Grundwasser vorherzusagen. Das Anwendungsspektrum ist universell. Nicht zuletzt können mit Computersimulationen belegbare, quantitative Aussagen über das zukünftige Klima gemacht werden. Nur mit Hilfe dieser Simulationsmodelle können dann die verschiedenen gesellschaftlichen und politischen Handlungsalternativen durchgespielt und bewertet werden.

Meist sind wissenschaftliche Modelle als partielle Differentialgleichungen formuliert. Diese werden z. B. mit der Methode der finiten Elemente oder mit finiten Differenzen diskretisiert, so dass große, dünn besetzte Gleichungssysteme entstehen. Die Bedeutung des Mehrgitterverfahrens liegt nun darin, dass diese Gleichungssysteme besonders effizient gelöst werden können, und dass deshalb genauere und zuverlässigere algorithmische Modelle der Realität möglich werden.

Das Mehrgitterverfahren ist dabei kein einzelner Algorithmus sondern vielmehr ein Konstruktionsprinzip für hocheffiziente Lösungsalgorithmen. Hocheffizient heißt hier, dass Mehrgitteralgorithmen asymptotisch optimale Komplexität haben. Im Idealfall benötigen sie nur $C \cdot N$ Gleitpunktoperationen (Floating Point Operations, FLOPS) zur Berechnung von N Unbekannten, wobei C eine Konstante ist. Diese lineare Komplexität zur Lösung von Gleichungssystemen ist dramatisch besser als z. B. das Standardverfahren mit Gauß'scher Elimination, das eine Komplexität $\frac{2}{3} N^3$ besitzt. Die asymptotisch optimale Komplexität ist auch die Grundvoraus-

https://doi.org/10.1007/s00287-019-01154-5

Ulrich Rüde
Lehrstuhl für Systemsimulation,
FAU Erlangen-Nürnberg,
Cauerstrasse 11, 91058 Erlangen, Deutschland
CERFACS, Avenue Gaspard Coriolis 42,
Toulouse 31057, France
E-Mail: ulrich.ruede@fau.de

Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter:
http://www.is.informatik.uni-wuerzburg.de/as

setzung für skalierbare parallele Verfahren. Denn damit ein Verfahren auf einem Parallelrechner mit der Zahl der Prozessoren skaliert, darf der Aufwand nicht schneller als linear mit der Problemgröße ansteigen.

Allerdings sind Mehrgitterverfahren nicht universell für alle Gleichungssysteme einsetzbar. Mehrgitterverfahren müssen auch oft individuell für eine gegebene Problemklasse entwickelt werden, damit die spezifische inhärente Struktur des Problems ausgenützt werden kann. Das Prinzip von Mehrgitterverfahren geht auf Arbeiten von Fedorenko zurück [3], aber praktisch nutzbare Mehrgitterverfahren wurden zuerst von Brandt [1] und Hackbusch [5] vorgeschlagen und analysiert.

Grundlagen der Mehrgitterverfahren

Mehrgitterverfahren beruhen auf stationären Iterationsverfahren. Um die Darstellung kompakt zu halten, nehmen wir an, dass eine gegebene lineare Differenzialgleichung auf einem Gitter mit der Maschenweite h diskretisiert wurde und damit ein Gleichungssystem mit der Form

$$A_h u_h = f_h, \tag{1}$$

mit N_h Unbekannten zu lösen ist. Der Vektor der Unbekannten ist $u_h \in \mathbb{R}^{N_h}$, die nichtsinguläre Matrix $A_h \in \mathbb{R}^{N_h \times N_h}$ ist dünn besetzt. Auch direkte Eliminationsverfahren können hierfür optimiert werden, so dass sie bessere Komplexität als das klassische Gauß-Verfahren erreichen. Jedoch kann mit direkten Verfahren in der Regel keine lineare Komplexität erreicht werden. Die Alternative sind Iterationsverfahren. Wir betrachten zunächst eine stationäre Iteration der Form, die von einem gegebenen u_h^0 startet und dann für $k = 1, 2, 3, \ldots$

$$u_h^{k+1} = u_h^k + \omega D_h^{-1} \left(f_h - A_h u_h^k \right) \tag{2}$$

berechnet. Beim klassischen Verfahren von Jacobi wird D_h als Diagonale von A_h gewählt, und ω ist ein skalarer Relaxationsparameter. Damit ist die Inverse D_h^{-1} zwar trivial berechenbar und braucht auch gar nicht explizit gespeichert zu werden, aber für viele praktisch relevante Fälle konvergiert dieses Verfahren zu langsam. Dies bedeutet, dass zwar jede Ausführung von (2) nur lineare Komplexität hat, die Zahl der erforderlichen Iterationen von (2) aber mit N_h anwächst. Damit ist der Gesamtaufwand schlechter als linear. Beim bekannten Gauß-Seidel-Verfahren und anderen, alternativen Relaxationsverfahren ist die Situation ähnlich.

Die langsame Konvergenz kann mathematisch präzise analysiert werden, in Spezialfällen z. B. mit Fourier-Techniken. Hier soll ein einfacheres, informatisches Argument helfen, das grundlegende Problem zu verdeutlichen. Die dünn besetzte Matrix A_h kann als Graph interpretiert werden, der dem Gitter der Diskretisierung entspricht. Jede Iteration (2) transportiert nun numerische Werte entlang den Kanten dieses Graphen. Um die Information zur Berechnung der Lösung vollständig durch das gesamte Gitter zu transportieren, sind mindestens so viele Iterationsschritte nötig, wie dem Durchmesser des Gitters als Graphen entspricht. Für viele der zu lösenden Probleme, speziell, wenn die Ausgangsdifferenzialgleichung vom elliptischen Typ ist, ist dieser vollständige Austausch unabdingbar. Wir haben damit eine untere Schranke für die Iterationszahl und damit den Rechenaufwand für alle Verfahren der Bauart (2). Es sei noch angemerkt, dass die gleichen Schranken für das häufig verwendete Verfahren der konjugierten Gradienten gelten. Es ist mit dieser Überlegung offensichtlich, dass bei all diesen Verfahren die Zahl der erforderlichen Iterationen wachsen muss, wenn das Gitter für eine bessere Auflösung der Differentialgleichung verfeinert und damit der Durchmesser des Graphen größer wird. Um diese Komplexitätsschranke zu brechen, braucht es eine andere Idee, nämlich die der Mehrgitterverfahren.

Das Mehrgitterverfahren beruht auf einer Rekursion, die unterschiedlich feine Auflösungen der gegebenen Differenzialgleichungen geschickt kombiniert. Im obigen Sinne nutzen Mehrgitterverfahren eine Familie von Graphen, die zusammen einen schnelleren Transport der Information durch das Lösungsgebiet ermöglichen. Nehmen wir hierzu zunächst an, dass es zwei Gitter gibt. Neben dem Gitter mit der feinen Auflösung, h, sei nun auch eines mit gröberer Auflösung, H, gegeben, so dass die gegebene Differenzialgleichung auch durch ein kleineres Gleichungssystem der Form

$$A_H u_H = f_H, \tag{3}$$

auf einem gröberen Gitter mit N_H Unbekannten approximiert werden kann. Durch Interpolation der Daten ist eine Transferabbildung I_H^h vom groben Gitter H auf das feine Gitter h gegeben, sowie in umgekehrter Richtung die Restriktionsabbildung

I_h^H. Die Lösung des Gleichungssystems (3) mit der kleinen Matrix A_H ist nun billiger als die Lösung mit der großen Matrix A_h. Weil A_H eine Näherung an A_h ist, liegt es nahe, diese Relation sowie (3) zu nutzen, um ein iteratives Verfahren für die Lösung von (1) zu konstruieren. Die sogenannte Grobgitterkorrektur hat die Form

$$u_h^{k+1} = u_h^k + I_H^h A_H^{-1} I_h^H \left(f_h - A_h u_h^k\right). \tag{4}$$

Dabei wird die inverse Matrix A_H^{-1} natürlich nicht explizit berechnet (sie wäre dicht besetzt), sondern, dazu äquivalent – aber effizienter – das Gleichungssystem (3) mit rechter Seite $f_H = I_h^H \left(f_h - A_h u_h^k\right)$ gelöst. Es ist sofort klar, dass die exakte Lösung von (1) ein Fixpunkt der Iterationsvorschrift (4) ist. Jedoch ist diese stationäre Iteration für sich allein genommen divergent, weil $I_H^h A_H^{-1} I_h^H$ keinen vollen Rang hat, wenn $N_H < N_h$.

Um ein konvergentes Verfahren zu erhalten, muss die Grobgitterkorrektur (4) mit den Relaxationsverfahren vom Typ (2) kombiniert werden. Diese Kombination, d. h. die abwechselnde Ausführung von (2) und (4), führt zu einem sehr schnell konvergenten Verfahren, denn beide Verfahren haben komplementäre Eigenschaften. Die Relaxationen vom Typ (2) kümmern sich um feine Auflösungsdetails, die Grobgitterkorrektur (4) transportiert die Information global durch das Gitter. Dass die Grobgitterapproximation weniger genau ist, erzeugt keinen Schaden, da dies iterativ wiederholt wird. Im Mehrgitterjargon wird die Relaxation auch als *Glätter* bezeichnet, weil man analysieren kann, dass ihre Rolle die Reduktion des hochfrequenten Fehlers in einer Näherungslösung ist. Alternative Glätter sind das Gauß-Seidel-Verfahren oder unvollständige ILU-Faktorisierungen. Viele weitere Varianten sind möglich. Dabei ist nicht entscheidend, dass der Glätter (2) als Löser schnell konvergiert, nur die Eigenschaft des Glättens ist entscheidend.

Vom bisher beschriebenen Zweigitterverfahren zum Mehrgitterverfahren kommt man, wenn man die Lösung des Grobgitterproblems (2) rekursiv nach dem gleichen Schema berechnet. Diese Rekursion wird fortgesetzt, bis das Grobgitterproblem so klein wird, dass es z. B. auch mit einem direkten Verfahren schnell lösbar ist. Insgesamt erhält man damit den folgenden Grundalgorithmus:

Algorithmus
V-Zyklus $= Vcycle(u_h, h, N_h, f_h, \nu_{\text{pre}}, \nu_{\text{post}})$
  # Löse exakt:
    If (N_h == Coarsest) solve $A_h u_h = f_h$; return u_h
  # Glätte ν_{pre} mal:
    $u_h = Relax(u_h, h, N_h, f_h, \nu_{\text{pre}})$
  # Berechne Residuum:
    $r_h = f_h - A_h u_h$
  # Restringiere das Residuum zum groben Gitter:
    $f_H = I_h^H r_h$
  # Initialisiere das Grobgitter mit 0:
    $u_H = 0$; $N_H = N_h / 2^d$
  # Rekursion:
    $u_H = Vcycle(u_H, H, N_H, f_H, \nu_{\text{pre}}, \nu_{\text{post}})$
  # Grobgitterkorrektur:
    $u_h = u_h + I_H^h u_H$
  # Glätte ν_{post} mal:
    $u_h = Relax(u_h, h, N_h, f_h, \nu_{\text{post}})$
    return u_h

Die mathematische Analyse, warum und unter welchen Bedingungen das Verfahren schnell konvergiert, würde den Platz eines Schlagworts sprengen, so dass wir auf die weiterführende Literatur verweisen müssen [1, 2, 5, 6, 11].

Komplexitätsüberlegungen

Wir können hier aber wenigstens die Kostenanalyse durchspielen. Die Kosten auf einem Gitterlevel, d. h. in einem rekursiven Aufruf von *Vcycle*, ergeben sich aus $\nu_{\text{pre}} + \nu_{\text{post}}$ Relaxationen, plus die Berechnung des Residuums und der Gittertransfers. Dabei ist entscheidend, dass ν_{pre} und ν_{post} typischerweise fixe kleine Zahlen sind, oft 1, 2, oder 3. Damit erzeugt jedes Level nur lineare Kosten mit $C_h \cdot N_h$ FLOPS. In die Rekursion geht die Annahme ein, dass jedes gröbere Level um den Faktor 2^d weniger Aufwand erzeugt, d. h., wir nehmen an, dass die Auflösung h in jeder räumlichen Dimension ($d = 1, 2, 3$) halbiert wird. Der Gesamtaufwand kann somit mit einer geometrischen Reihe als

$$C_h \cdot N_h \left(1 + 2^{-d} + 2^{-2d} + 2^{-3d} + \ldots\right) \leq$$
$$C_h \cdot N_h \left(\frac{1}{1 - 2^{-d}}\right) = C_h \cdot N_h \left(\frac{2^d}{2^d - 1}\right) \tag{5}$$

abgeschätzt werden. Man sieht damit, dass jeder V-Zyklus des Mehrgitterverfahrens Kosten verursacht, die proportional zu N_h sind. Das Verfahren hat damit asymptotisch optimale Komplexität. Ist auch die

schnelle Konvergenz des Verfahrens gegeben, wird eine Lösung mit vorgeschriebener Genauigkeit nach einer konstanten Zahl von V-Zyklen erreicht. Im Idealfall konvergieren Mehrgitterraten mit Konvergenzraten von 0,1 oder besser, so dass z. B. 5 Stellen Genauigkeit bereits nach 5 V-Zyklen erreicht sind.

Es ist sogar noch mehr möglich. Denn mit immer weiterer Verfeinerung des Gitters möchte man natürlich die Genauigkeit jeweils entsprechend erhöhen. Auch dies ist mit linearem Aufwand möglich, wenn man den V-Zyklus in eine weitere Rekursion einbettet. Dies ist dann unter den Namen „Full Multigrid" oder „Nested Iteration" in der Literatur bekannt. Zusammengefasst hat man damit ein Verfahren, bei dem die Lösung eines Gleichungssystems (1) nur um einen konstanten Faktor teurer ist als die Multiplikation eines Vektors mit der Matrix A_h. Für typische Modellprobleme wie das Poisson-Problem in 2D, diskretisiert mit Differenzen zweiter Ordnung, kann so eine Lösung mit weniger als $30\,N_h$ FLOPS erreicht werden [11].

Algorithmische Varianten und Parallelisierung

Die exzellente Effizienz ist äußerst erfreulich, weil man damit auch große Probleme extrem schnell lösen kann. Dies erfordert aber auch eine effiziente Implementierung. Die Lösung einer Poisson-Gleichung in 2D auf einem Gitter von 1024×1024 benötigt nur 30×10^6 FLOPS, kann also auf einer modernen CPU oder auch mit einer GPU in nur Millisekunden erledigt werden. Umso erstaunlicher ist, dass immer wieder Algorithmen vorschlagen werden, die drei, vier oder fünf Größenordnungen weniger effizient als das Mehrgitterverfahren sind. Hier ist Vorsicht angeraten: Nicht alles, was in der theorieorientierten Literatur zu finden ist, hat auch praktische Relevanz.

Neben den geometrischen Mehrgitterverfahren, die die Struktur der partiellen Differenzialgleichung und ihrer Diskretisierung explizit nutzen um die Diskretisierungen A_h und A_H etc. direkt zu erzeugen, gibt es sogenannte algebraische Mehrgitterverfahren. Diese sind dadurch motiviert, dass man gerne die Aufstellung der Matrizen vom algebraischen Lösungsverfahren trennen würde. Algebraische Mehrgitterverfahren müssen deshalb versuchen, die inhärente Vergröberungsstruktur aus der Matrix A_h zu bestimmen. In diesem Fall werden verschiedene Heuristiken eingesetzt um zunächst eine Interpolation I_H^h zu konstruieren. Dann setzt man die Restriktion als Transponierte an $I_H^h = (I_H^h)^T$ und kann damit die sogenannte Galerkin-Approximation $A_H = I_h^H A_h I_H^h$ verwenden. In jedem Fall führt dies jedoch zu signifikanten Overheads im Vergleich zu geometrischen Mehrgitterverfahren.

In die entgegengesetzte Richtung gehen jüngere Überlegungen, dass auf modernen Rechnerarchitekturen die Laufzeit eher durch die Speicherbandbreite begrenzt wird als durch die Ausführung der FLOPS. Deshalb rücken sogenannte matrixfreie Verfahren in den Fokus. Dabei wird die Matrix A_h nicht gespeichert, sondern bei Bedarf „on-the-fly" berechnet. Man erkauft sich damit eine verringerte Speicherbandbreite durch einen erhöhten FLOPS-Aufwand. Andere Optimierungsmöglichkeiten ergeben sich z. B. durch eine bessere Nutzung der Speicherhierarchie, d. h. eine bessere Nutzung der Caches, wie sie z. B. im Projekt „Data-Local Iterative Methods" (DIME) schon früh untersucht wurde. Dieses Projekt wurde von 1998–2005 gemeinsam von dem Lehrstuhl für Rechnerarchitektur der TU München (Prof. Bode) und dem Lehrstuhl für Systemsimulation der Universität Erlangen-Nürnberg durchgeführt.

Die Parallelisierung von Mehrgitterverfahren ist schon seit den 1980er-Jahren ein wichtiges Thema. Im Suprenum-Projekt wurde von schon 1985–1990 an der damaligen Gesellschaft für Mathematik und Datenverarbeitung (GMD) in Kooperation mit mehreren Universitäten eine parallele Rechnerarchitektur speziell für Mehrgitterverfahren entwickelt und realisiert. Dabei wurden viele wichtige Grundlagen für das parallele Hochleistungsrechnen gelegt, die bis heute ausstrahlen. Algorithmisch stellt sich bei den Mehrgitterverfahren das Problem, dass nicht nur sehr große, hochaufgelöste Gitter bearbeitet werden müssen, sondern eine hierarchische Folge von Gittern, die sukzessive kleiner werden. Deshalb müssen Mehrgitterverfahren besonders sorgfältig implementiert werden, damit die Schleifen-Overheads und Kommunikationslatenzen auf großen parallelen Systemen nicht zu stark zu Buche schlagen. Oft ist es so, dass gut implementierte Mehrgitterverfahren den deutlich schnellsten Löser für eine Problemklasse bereitstellen, obwohl sie im Hinblick auf maximale FLOPS-Rate oder Systemauslastung hinter anderen Verfahren zurückstehen. Es ist deshalb wichtig, sich das Problem des Datentransports klar zu machen. Mehrgitterverfahren nutzen eine Hierarchie von Gittern, um die Daten

schnell durch das Lösungsgebiet zu transportieren. Es ist unvermeidlich, dass die Systemauslastung eines großen Parallelrechners sinkt, während die groben Gitter bearbeitet werden. Der Schluss, dass man deshalb die groben Gitter vermeiden müsste, ist aber grundfalsch. Denn wenn man den essenziellen Datenaustausch auf feineren Gittern durchführt, hat man zwar womöglich eine bessere Systemauslastung und höhere FLOPS-Rate, muss aber dafür mit einer unnötig hohen Iterationszahl bezahlen. Die Lösung findet man damit sicher nicht schneller, sondern insgesamt langsamer und mit redundant erhöhtem Rechenaufwand. Forschung zu diesem Themenkomplex findet derzeit in Deutschland gebündelt im DFG-Schwerpunktprogramm „Software für Exascale Computing“ (SPPEXA) [9] statt.

Ein Beispiel aus den Geowissenschaften

Abschließend wollen wir die Leistungsfähigkeit moderner Mehrgitterverfahren mit einem Problem aus der Geophysik illustrieren. Es geht darum, den Erdmantel zu modellieren, also die ca. 3000 km dicke Felsschicht unseres Planeten, die nach außen durch die Kontinentalplatten und die Lithosphäre begrenzt ist und nach innen durch den flüssigen Eisenkern. Auf einer Zeitskala von Jahrmillionen verhält sich der Erdmantel wie eine zähe Flüssigkeit mit Strömungsgeschwindigkeiten von wenigen cm pro Jahr. Die langsame Konvektionsströmung des Erdmantels ist Ursache der Kontinentalverschiebung; sie führt zur Formung von Gebirgen und bedingt die Entstehung von Erdbeben. Der Erdmantel hat ein Volumen von ca. 10^{12} km^3. Weil sich geologische Phänomene auf der Skala von 1 km oder weniger abspielen, würde man gerne die Simulation des Erdmantels mit einer Auflösung von 1 km oder weniger berechnen. Die Diskretisierung des Gebiets führt damit zu Systemen von ca. 10^{12} Gitterzellen, von denen jede durch mehrere Zustandsvariablen wie Temperatur, Geschwindigkeit, und Druck gekennzeichnet ist. In jedem Zeitschritt einer Simulation muss nun ein Gleichungssystem mit diesen $N = 10^{12}$ oder mehr Unbekannten gelöst werden. Würde man dazu unbesehen das Gauß'sche Eliminationsverfahren einsetzen, so müsste man die astronomische Zahl von $2/3\,N^3 = 2/3 \cdot 10^{36}$ FLOPS ausführen. Selbst wenn man die dünne Besetzungsstruktur der Matrix z. B. mit dem Nested-Dissection-Algorithmus für eine Reduktion des Fill-In nutzen würde, kann man den Aufwand hochrechnen und erhält immer noch mehr als 10^{24} FLOPS. Ein PC mit 100 GigaFLOPS Leistung würde hierfür eine Rechenzeit mehr als 100 000 Jahren benötigen. Selbst der aktuell schnellste Rechner in Deutschland, SuperMucNG [10], würde für diese Anzahl FLOPS noch länger als ein Jahr benötigen. Wohlgemerkt: für einen von vielen nötigen Zeitschritten. Dieser gigantische Rechenaufwand rechtfertigt die Klassifizierung des Problems als „Grand Challenge-Anwendung“.

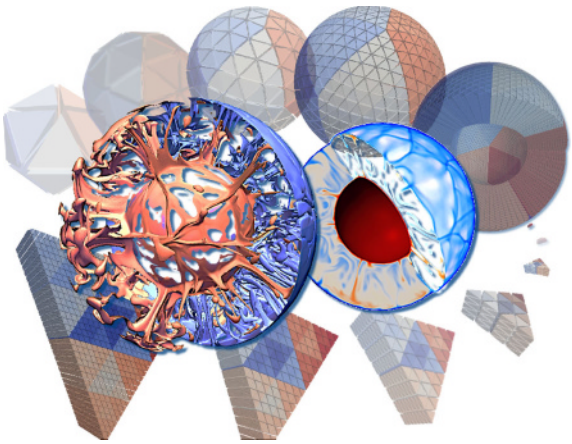

Abb. 1 Gitterhierarchie für die Simulation der Erdmantelkonvektion mit 1 km Auflösung

Im SPPEXA-Projekt TerraNeo wird ein neues paralleles Software-Framework für die Erdmantelkonvektion entwickelt [7]. Die dabei genutzte Mehrgitterhierarchie wird in Abb. 1 zusammen mit einer Visualisierung der auf- und absteigenden Strömungen und Temperaturfelder exemplarisch dargestellt. Für weitere Hintergründe zu den Methoden und Forschungsergebnissen verweisen wir auf die Webseite [7], die dort gelisteten Veröffentlichungen und speziell auf das Teachlet [8]. In diesem interdisziplinären Projekt, das Informatiker und Mathematiker mit Geophysikern zusammenführt, ist es gelungen, die relevanten Gleichungssysteme mit bis zu 10^{13} Unbekannten zu lösen. Ein einzelner Lösungsvektor benötigt dabei bereits 80 TByte Speicherplatz, so dass selbst die größten heute verfügbaren Supercomputer nur wenige von diesen Vektoren gleichzeitig speichern können. Es ist unmöglich, die dazu gehörigen Matrizen zu speichern. Es mussten deshalb neue matrixfreie Mehrgitterverfahren und ein spezieller Glätter entwickelt werden. Mit diesem Mehrgitterverfahren

und bei Verwendung eines der schnellsten Supercomputer [4] kann eine solche Lösung in ca. 13 min berechnet werden. Mehrgitterverfahren und moderne Supercomputer zusammen ermöglichen so geophysikalische Simulationen mit einer Detailtreue und einer Auflösung, die mit anderen Verfahren völlig undenkbar wäre.

Ausblick

Wegen der rapide steigenden Bedeutung von Computersimulationen in Wissenschaft und Technik gewinnt *Computational Science and Engineering* (CSE) als neue Fachdisziplin weiter zunehmend an Bedeutung. Auf der Basis von Grundlagen aus der Mathematik und Informatik können neue Simulationsverfahren entwickelt werden. Schnelle Algorithmen, wie das Mehrgitterverfahren, sind unverzichtbar, um ausreichend genaue Berechnungen durchzuführen und damit fundamentale Fortschritte in vielen Wissenschaftsbereichen zu erzielen. Die Computermodelle des CSE beruhen auf physikalischen Grundgesetzen wie Massen-, Energie-, und Impulserhaltung und bauen darauf kausale Wirkungsketten auf. Diese können präzise formuliert, hinterfragt und überprüft werden. Darin unterscheiden sich CSE-Methoden von Big-Data-Ansätzen, die aus gegebenen Daten Korrelationen bestimmen und diese mithilfe von ausgeklügelten Interpolations- und Extrapolationstechniken nutzen, um Plausibiliätsvorhersagen zu machen. Eine große Zukunftsaufgabe wird es sein, diese beiden Ansätze sachgerecht zu verbinden.

Literatur

1. Brandt A (1977) Multi-level adaptive solutions to boundary-value problems. Math Comp 31:333–390
2. Brandt A, Livne OE (2011) Multigrid techniques: 1984 guide with applications to fluid dynamics. Vol. 67. SIAM
3. Fedorenko RP (1962) A relaxation method for solving elliptic difference equations. U.S.S.R. Comput Math Math Phys 1(5):1092–1096
4. Gmeiner B, Huber M, John L, Rüde U, Wohlmuth B (2016) A quantitative performance study for Stokes solvers at the extreme scale. J Comput Sci 17: 509–521
5. Hackbusch W (1976) Ein iteratives Verfahren zur schnellen Auflösung elliptischer Randwertprobleme. Report 76-12. Institut für Angewandte Mathematik, Universität Köln
6. Hackbusch W (1985) Multi-grid methods and applications. Springer Series in Computational Mathematics 4. Springer, Berlin, New York
7. http://terraneo.fau.de
8. http://terraneo.fau.de/Files/teachlet_v1.0_1080p.mp4
9. http://www.sppexa.de
10. https://doku.lrz.de/display/PUBLIC/SuperMUC-NG
11. Stüben K, Trottenberg U (1982) Multigrid methods: Fundamental algorithms, model problem analysis and applications. In: Hackbusch W, Trottenberg U (eds) Multigrid methods. Lecture notes in mathematics 960. Springer, Berlin, Heidelberg
12. Trottenberg U, Oosterlee CW, Schuller A (2000) Multigrid. Elsevier
13. Volkert J (1989) Mehrgittermethode. Informatik-Spektrum 12(2)