FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Beiträge zum Wissenschaftlichen Rechnen
Ergebnisse des
Gaststudentenprogramms 2003
des John von Neumann-Instituts
für Computing

Rüdiger Esser (Hrsg.)

FZJ-ZAM-IB-2003-10

Dezember 2003

(letzte Änderung: 1. 12. 2003)

Vorwort

Die Ausbildung im Wissenschaftlichen Rechnen ist neben der Bereitstellung von Supercomputer-Leistung und der Durchführung eigener Forschung eine der Hauptaufgaben des John von Neumann-Instituts für Computing (NIC) und hiermit des ZAM als wesentlicher Säule des NIC. Um den akademischen Nachwuchs mit verschiedenen Aspekten des Wissenschaftlichen Rechnens vertraut zu machen, führte das ZAM in diesem Jahr zum vierten Mal während der Sommersemesterferien ein Gaststudentenprogramm durch. Entsprechend dem fächerübergreifenden Charakter des Wissenschaftlichen Rechnens waren Studenten der Natur- und Ingenieurwissenschaften, der Mathematik und Informatik angesprochen. Die Bewerber mussten das Vordiplom abgelegt haben und von einem Professor empfohlen sein.

Die neun vom NIC ausgewählten Teilnehmer kamen für zehn Wochen, vom 4. August bis 10. Oktober 2003, ins Forschungszentrum. Sie beteiligten sich hier an den Forschungs- und Entwicklungsarbeiten des ZAM und wurden jeweils einem Wissenschaftler zugeordnet, der mit ihnen zusammen eine Aufgabe festlegte und sie bei der Durchführung anleitete.

Die Gaststudenten und ihre Betreuer waren:

Jan Albersmeyer	Godehard Sutmann
Matthias Bolten	Thomas Müller
Björn Hagemeier	Herwig Zilken
Mike Hammes	Paul Gibbon
Martin Holtschneider	Artur Baumgärtner, IFF
Christian Kahl	Bernhard Steffen
Kjong-Van Lehmann	Jörg Striegnitz
Eric Lorenz	Holger Dachsel
Natali Zint	Wolfgang Meyer

Zu Beginn ihres Aufenthalts erhielten die Gaststudenten eine viertägige Einführung in die Programmierung und Nutzung der Parallelrechner im ZAM. Um den Erfahrungsaustausch untereinander zu fördern, präsentierten die Gaststudenten am Ende ihres Aufenthalts ihre Aufgabenstellung und die erreichten Ergebnisse. Sie verfassten zudem Beiträge mit den Ergebnissen für diesen Internen Bericht des ZAM.

Wir danken den Teilnehmern für ihre engagierte Mitarbeit - schließlich haben sie geholfen, einige aktuelle Forschungsarbeiten weiterzubringen - und den Betreuern, die tatkräftige Unterstützung dabei geleistet haben.

Ebenso danken wir allen, die im ZAM und der Verwaltung des Forschungszentrums bei Organisation und Durchführung des diesjährigen Gaststudentenprogramms mitgewirkt haben. Besonders hervorzuheben ist die finanzielle Unterstützung durch den Verein der Freunde und Förderer des FZJ und die Firma IBM.
Es ist beabsichtigt, das erfolgreiche Programm künftig fortzusetzen, schließlich ist die Förderung des wissenschaftlichen Nachwuchses dem Forschungszentrum ein besonderes Anliegen.

Weitere Informationen über das Gaststudentenprogramm, auch die Ankündigung für das kommende Jahr, findet man unter http://www.fz-juelich.de/zam/gaststudenten.

Jülich, Dezember 2003 Rüdiger Esser

Inhalt

Parallele Mehrgitterverfahren für die dreidimensionale Poissongleichung

Jan Albersmeyer

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Universität Heidelberg
Im Neuenheimer Feld 368
69120 Heidelberg

E-mail: Jan.Albersmeyer@iwr.uni-heidelberg.de

Zusammenfassung: Es werden, teilweise ausgehend von einer bestehenden seriellen Implementierung, verschiedene Löser 2., 4. und 6. Ordnung für den Einsatz in Mehrgitterverfahren zur Lösung der Poissongleichung untersucht, implementiert und parallelisiert. Dabei wird besonders der dreidimensionale Fall mit Dirichlet-Randbedingungen, wie er z.B. bei Molekulardynamiksimulationen auftritt, betrachtet. Es wird demonstriert, dass die untersuchten Verfahren im Einsatz als einfache iterative Löser eine sehr gute Skalierbarkeit aufweisen, allerdings in Mehrgitterverfahren aufgrund von skizzierten Load-Balancing Problemen noch an Performance verlieren.

Einleitung

Schnelle und genaue Verfahren zur Lösung der Poissongleichung sind in vielen Bereichen von Interesse, insbesondere zum Beispiel in der Molekulardynamiksimulation. Dort dienen sie in einem unserer Anwendungsbeispiele dazu, das von einer (diskreten) elektrischen Ladungsverteilung (wie Fig. 1) erzeugte elektrische Potential zu berechnen, wobei das Potential auf dem Rand vorgegeben ist. Mit Hilfe des so erlangten Potentials können dann die Kräfte auf die Teilchen und deren Bewegung durch Approximation der Ableitung berechnet werden. Dabei ist der rechenintensivere Teil die Lösung der Potentialgleichung. Zur Lösung der Poissongleichung gibt es eine Vielzahl von Ansätzen wie zum Beispiel die Methode der Finiten Elemente. Wir verfolgen in unserem Fall den Ansatz mittels Finiter Differenzen zur Approximation des Laplace-Operators und verwenden zur schnellen Fehlerreduktion Mehrgitterverfahren. Dadurch erhalten wir eine schnelle Methode zur Lösung der Poissongleichung, die theoretisch auch gut auf Mehrprozessormaschinen skaliert.

Theorie

Sei $\Omega \subset \mathbb{R}^3$ ein Gebiet,

$$\Delta = \frac{\partial^2}{\partial x_1^2}\frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2}$$

der Laplaceoperator, sowie $\Phi, f : \Omega \to \mathbb{R}$, $g : \partial\Omega \to \mathbb{R}$.
Dann versteht man unter der dreidimensionalen Poissongleichung die folgende Differentialgleichung

$$-\Delta\Phi(x) = f(x), \quad x \in \Omega.$$

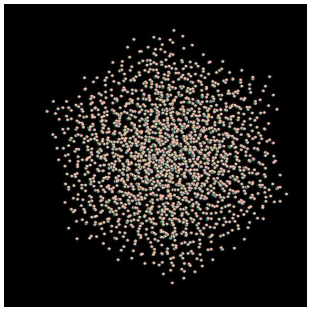


Abbildung 1: Eine Verteilung von geladenen Teilchen im Raum

mit Randbedingungen, z.B. der Dirichlet-Randbedingung

$$\Phi(x) = g(x), \quad x \in \partial\Omega,$$

falls Φ auf dem Rand $\partial\Omega$ von Ω gegeben ist, bzw. mit Neumann-Randbedingungen

$$\frac{\partial\Phi(x)}{\partial n} = g(x), \quad x \in \partial\Omega,$$

falls die Ableitungen nach der jeweiligen äußeren Normalen n von Ω vorgegeben sind. Im folgenden wollen wir unter der Poissongleichung stets die Poissongleichung mit Dirichlet-Randbedingungen verstehen.

Der nun folgende Abschnitt bezieht sich stets auf die Poissongleichung, die meisten Aussagen treffen aber auch allgemeiner für elliptische Differentialoperatoren zu.

Löser für die Poissongleichung

Approximation des Laplaceoperators

Zur Lösung der Poissongleichung mittels der Methode der Finiten Differenzen muss man sich zunächst Gedanken zur Approximation des Laplaceoperators machen. Ab jetzt nehmen wir an, dass wir eine gleichmäßige Diskretisierung des Raumes mit Gitterweite h für eine beliebige, zweimal stetig differenzierbare Funktion ρ vorliegen haben. Wir bezeichnen die Werte von ρ an den Gitterpunkten mit ρ_{ijk}, wobei $0 \leq i, j, k \leq N$.

Zur weiteren Klärung der Notation betrachten wir zunächst den eindimensionalen Fall:
Dort ist eine einfache Approximation 2. Ordnung (in h) für

$$\frac{\partial^2}{\partial x^2}\rho(x)\bigg|_{x=x_0} = \frac{\rho(x_0 - h) - 2\rho(x_0) + \rho(x_0 + h)}{h^2} + \mathcal{O}(h^2),$$

bzw. mit obigen Bezeichnungen

$$\frac{\partial^2}{\partial x^2}\rho_i = \frac{\rho_{i-1} - 2\rho_i + \rho_{i+1}}{h^2} + \mathcal{O}(h^2)$$

Wir definieren nun δ_x^2 als "Differenzenoperator", der auf ein $\rho(x)$ angewandt genau obige Wirkung hat. Dann erhalten wir allgemein eine Approximation beliebig genauer Ordnung durch die Reihe [7]

$$\frac{\partial^2}{\partial x^2}\rho_i = \frac{1}{h^2}\delta_x^2\ (1 - \frac{1}{12}\delta_x^2 + \frac{1}{90}\delta_x^4 - \ldots)\rho_i,$$

aus der sich durch Abbruch nach dem ersten Glied in der Klammer die obige Approximation zweiter Ordnung ergibt, sowie die "klassischen" Approximationen 4. und 6. Ordnung durch Abbruch der Reihe nach dem 2. bzw. 3. Klammerterm. Die Approximation 4. Ordnung hat dann ausformuliert folgende Gestalt:

$$\frac{\partial^2}{\partial x^2}\rho_i = \frac{1}{12h^2}(-\rho_{i-2} + 16\rho_{i-1} - 30\rho_i + 16\rho_{i+1} - \rho_{i+2})$$

Für die Approximation 6. Ordnung würden auch noch die Punkte $\rho_{i\pm3}$ verwendet werden müssen. Man sieht, dass für jede Erhöhung der Ordnung ein weiteres Paar von weiter entfernten Punkten benötigt wird. Dort genau liegt auch, wie wir später sehen werden, die Schwäche bei der Parallelisierung dieser Verfahren. Die Approximation für den dreidimensionalen Fall erhält man schließlich durch einfache Kombination der einzelnen eindimensionalen Approximationen. Zur einfacheren Darstellung des mehrdimensionalen Falls benutzen wir die folgende Notationsform der "Differenzensterne":

$$\Delta\rho_{ijk} = \frac{1}{h^2}\sum_{l,m,n}\delta_{lmn}\rho_{lmn} + \mathcal{O}(h^r),$$

wobei die δ_{lmn} gerade die Einträge der Matrizen der Form

$$\Delta_{ijk} = \begin{pmatrix} \delta_{i,j-1,k+1} & \delta_{i,j,k+1} & \delta_{i,j+1,k+1} \\ \delta_{i,j-1,k} & \delta_{i,j,k} & \delta_{i,j+1,k} \\ \delta_{i,j-1,k-1} & \delta_{i,j,k-1} & \delta_{i,j+1,k-1} \end{pmatrix},$$

sind, bei denen der Matrix-Index dem Index des mittleren Elements entspricht. Die klassischen Approximationen des Laplaceoperators 2. und 4. Ordnung lassen sich dann durch folgende Sterne beschreiben:

2. Ordnung

$$\Delta_{i-1,j,k} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \Delta_{i,j,k} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -6 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \Delta_{i+1,j,k} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

4. Ordnung

$$\Delta_{i\pm2,j,k} = \frac{1}{12}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \Delta_{i\pm1,j,k} = \frac{1}{12}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\Delta_{i,j,k} = \frac{1}{12}\begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 \\ -1 & 16 & -90 & 16 & -1 \\ 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}.$$

Eine weitere Möglichkeit zur Approximation des Laplaceoperators bilden sogenannte kompakte Approximationen, die man beispielsweise durch $Padé$-Approximationen [8, 9] erhalten kann. Diese verwenden zwar zur Approximation auch diagonale Nachbarpunkte und ausserdem verlangen sie Anpassungen auf der rechten Seite der Differentialgleichung, allerdings benötigen sie dann auch für Approximationen 4. und 6. Ordnung nur Punkte aus der nächsten Nachbarschicht, was sich nachher für die Parallelisierung als günstig herausstellt und für Mehrgitterverfahren auf den gröberen Gittern sogar notwendig ist. Bei Verwendung dieser sogenannten High-Order-Compact (HOC) Approximationen haben dann die diskretisierten Differentialgleichungen mit angepaßter rechter Seite und die Differenzensterne die folgende Gestalt [2, 1]:

(HOC 4.Ordnung)

$$-\frac{1}{6h^2}\sum_{l,m,n}\delta_{lmn}\Phi_{lmn}+\mathcal{O}(h^4)=f_{ijk}+\frac{h^2}{12}(\delta_x^2+\delta_y^2+\delta_z^2)f_{ijk}$$

mit

$$\Delta_{i-1,j,k}=\begin{pmatrix}0&1&0\\1&2&1\\0&1&0\end{pmatrix},\ \Delta_{i,j,k}=\begin{pmatrix}1&2&1\\2&-24&2\\1&2&1\end{pmatrix},\ \Delta_{i+1,j,k}=\begin{pmatrix}0&1&0\\1&2&1\\0&1&0\end{pmatrix}.$$

und

(HOC 6. Ordnung)

$$-\frac{1}{30h^2}\sum_{l,m,n}\delta_{lmn}\Phi_{lmn}+\mathcal{O}(h^6)=f_{ijk}+\frac{h^2}{12}\nabla^2 f_{ijk}+\frac{h^4}{360}\nabla^4 f_{ijk}$$

mit

$$\Delta_{i-1,j,k}=\begin{pmatrix}1&3&1\\3&14&3\\1&3&1\end{pmatrix},\ \Delta_{i,j,k}=\begin{pmatrix}3&14&3\\14&-128&14\\3&14&3\end{pmatrix},\ \Delta_{i+1,j,k}=\begin{pmatrix}1&3&1\\3&14&3\\1&3&1\end{pmatrix}.$$

Für dieses Verfahren 6. Ordnung sind allerdings analytische Ableitungen bis zu 4. Ordnung von der rechten Seite f notwendig. In der Implementierung haben wir diese allerdings wieder durch diskrete Verfahren approximiert und dennoch gute Ergebnisse erhalten.

Generell stellt die Approximation der rechten Seite keine große Belastung dar, da sie ja bekannt und auf einem festen Gitter konstant ist und damit die Anpassung nur einmal pro Gitterwechsel berechnet werden muss.

Löser für das lineare Gleichungssystem

Aus den obigen Approximationen für den Laplaceoperator, und gegebenenfalls der rechten Seite, erhalten wir dann ein im allgemeinen sehr grosses lineares Gleichungssystem mit $(N+1)^3$ Gleichungen, das dann aufgrund des Speicher- und Rechenzeitbedarfs normalerweise nicht mehr exakt lösbar ist, so dass hier iterative Verfahren zum Einsatz kommen.

Im folgenden werden im wesentlichen leichte Modifikationen des Gauss-Seidel-SOR Lösers verwendet. Diesen erhält man folgendermaßen:

Man zerlegt die Matrix A des Gleichungssystems $A\Phi = f$ in $A = D * L * U$, wobei D die Diagonale von A ist, und L bzw. R eine strikte untere bzw. obere Dreiecksmatrix ist. Sodann verwendet man die folgende Update-Vorschrift zur Berechnung der nächsten Iterierten:

$$\Phi^{(n+1)}=(D-\omega L)^{-1}((1-\omega)D+\omega U)\Phi^{(n)}+\omega(D-\omega L)^{-1}f,$$

wobei ω der sogenannte Relaxierungsparameter ist, der eine Gewichtung zwischen der alten Iterierten und der Berechnungsvorschrift für die neue bewirkt. Ein Wert von $\omega = 1$ ergibt das klassische Gauss-Seidel-Verfahren.

Wie sich weiter unten zeigen wird, hat der Relaxierungsparameter auch im Einsatz innerhalb eines Mehrgitterverfahrens grossen Einfluss auf die Konvergenzgeschwindigkeit.

Mehrgitterverfahren

Einfache iterative Löser konvergieren zwar in der Theorie, allerdings sind die Konvergenzraten in der Praxis bisweilen sehr schlecht. Hier setzen die Mehrgitterverfahren an, die ausnutzen, dass obwohl die Reduktion des Gesamtfehlers nur sehr schleppend fortschreitet, bei manchen Verfahren (unter anderem dem Gauss-Seidel-SOR) hochfrequente Fehleranteile sehr gut reduziert werden. Man spricht hierbei auch von Glättung und nennt die Verfahren Glätter.

Betrachten wir nun den Fehler e bei der Lösung der Differentialgleichung

$$e = \Phi - w,$$

mit Φ als exakter Lösung und w der Approximation. Dann wird w die Gleichung nicht ganz wahr machen und man erhält als Residuum oder Defekt der Gleichung:

$$r := f + \Delta w.$$

Wenden wir nun auf die obere Gleichung den negativen Laplaceoperator an, so erhalten wir als Beziehung zwischen e und r:

$$-\Delta e = r.$$

Der Fehler bei der Lösung der Differentialgleichung genügt also derselben Differentialgleichung wie die Lösung selbst, nur dass als rechte Seite jetzt das Residuum verwendet werden muss. Man kann nun also dieselben Verfahren, die man zur Lösung der Differentialgleichung verwendet, auch dazu benutzen, den Fehler zu approximieren und das Ergebnis als Korrektur für die ursprüngliche Lösung zu verwenden. Dieses Prinzip benutzt man beim Mehrgitterverfahren, wobei man nun noch geschickt die Glättungseigenschaften der iterativen Verfahren benutzt. Der schematische Ablauf einer einfachen Mehrgitteriteration (einem sogenannten V-Zyklus) sieht dann wie folgt aus:

- Man beginnt auf einem feinsten Gitter mit Glättungsiterationen. Dadurch werden hochfrequente Fehleranteile geglättet.
- Dann berechnet man für jeden Gitterpunkt das Residuum. Dies transferiert man dann als neue rechte Seite auf ein gröberes Gitter (Restriktion). Dort erscheinen die auf dem feineren Gitter niederfrequenten Anteile jetzt hochfrequenter. Die auf dem feineren Gitter hochfrequenten Anteile sind auf dem gröberen Gitter nicht sichtbar.
- Dort glättet man nun die Differentialgleichung für den Fehler. Dies glättet die “neuen” hochfrequenten Anteile.
- Dieses Vorgehen wiederholt man bis zu einem recht groben Gitter, im Extremfall zu einem Gitter mit nur einem Punkt. Dort läßt sich das System auch exakt lösen.
- Anschließend transferiert man die Lösung als Korrektur auf das nächstfeinere Gitter (Interpolation) und führt dort noch einige Glättungsiterationen durch.
- Dies wiederholt man bis zum feinsten Gitter, wo man dann schließlich die eigentliche Lösung der Differentialgleichung korrigiert.

Dabei werden also die einzelnen Komponenten des Fehlers genau auf dem Gitter sehr gut geglättet, auf dem sie hochfrequent erscheinen, sodass man nach einem solchen V-Zyklus eine gleichmäßige Reduktion aller Fehleranteile erwarten kann. Ausserdem findet der größte Teil der Fehlerreduktion auf gröberen Gittern statt, was den Aufwand gegenüber derselben Anzahl Iterationen auf dem feinsten Gitter sehr stark

reduziert. Wenn man nämlich von einer Halbierung der Gittergröße von einem zum nächstem Gitter ausgeht, so ergibt sich mit dem Aufwand auf dem feinsten Gitter als Referenz der folgende Gesamtaufwand für das Mehrgitterverfahren:

$$1 + \frac{1}{8} + \left(\frac{1}{8}\right)^2 + \left(\frac{1}{8}\right)^3 + \ldots.$$

Das bedeutet insbesondere, dass das Mehrgitterverfahren von optimaler Komplexität ist. Ein weiterer Vorzug des Mehrgitterverfahrens liegt darin, dass es zwar aufgrund der seriellen Abarbeitung der verschiedenen Gitterebenen an sich nicht parallelisierbar ist, aber seine Komponenten, also die Gittererzeugung, die Speicherung der Gitter, der Transfer zwischen den Gittern (Restriktion, Interpolation) sowie die Glätter sehr gut parallelisierbar sind.

Parallelisierung und Anwendungen

Ausgehend von der obigen seriellen Variante eines Mehrgitterverfahrens mit Gauss-Seidel-SOR als Glätter gehen wir nun die Parallelisierung an. Dabei lag der Schwerpunkt auf der Implementierung und Parallelisierung der Glätter. Die Parallelisierung erfolgte dabei mit Hilfe des MPI-1-Standards [4], beruht also auf Kommunikation zwischen Prozessoren mit verteiltem Speicher. Als Programmiersprache diente Fortran90.

Parallelisierung

Bei der Parallelisierung und Implementierung der Glätter sind neben den eigentlichen Iterationen besonders die Aspekte der Datenverteilung und der Kommunikation zu berücksichtigen. Da die Iterationsverfahren wie oben gezeigt nur jeweils ein paar Nachbarpunkte benötigen, um die nächste Iterierte zu berechnen, liegt die Idee der Parallelisierung darin, jedem Prozessor einen Teil des aktuellen Gitters zuzuweisen, auf dem er dann die Iterationen ausführt, und in den Randbereichen die benötigten Daten zwischen den Prozessoren nach jeder Iteration auszutauschen.

Verteilung der Daten und Kommunikationsaufwand

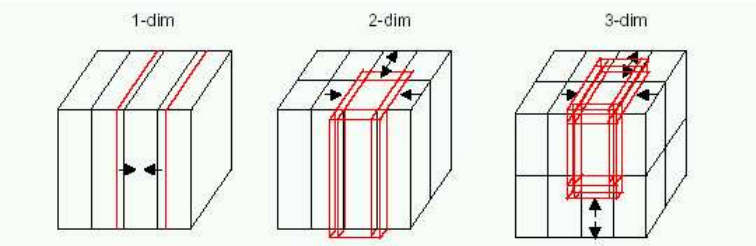


Abbildung 2: Verschiedene Strategien zur Datenverteilung auf die Prozessoren

Es gibt im wesentlichen 3 verschiedene Strategien zur gleichmäßigen Verteilung der Daten auf die einzelnen Prozessoren (s. Fig 2): Scheibchen, Säulen und Quader, welche man auch als 1-,2- und 3-dimensionale Verteilung bezeichnet.

Die jeweils beste Aufteilung für den konkreten Fall ist abhängig von der Gittergröße und der Anzahl der Prozessoren. Generell bestimmt die Datenverteilung den Umfang der später anfallenden Kommunikation mit.

Die Kommunikation ist nämlich bestimmt durch die Anzahl der auszutauschenden Schichten sowie ihrer jeweiligen Größe und der Anzahl der Prozessoren, mit denen Daten getauscht werden müssen.

Die Anzahl der auszutauschenden Schichten ist durch das eingesetzte Approximationsschema bestimmt. Dort haben die von uns verwendeten HOC-Approximationen einen klaren Vorteil gegenüber den klassischen Approximationen, da sie jeweils nur eine auszutauschende Schicht fordern. Demgegenüber benötigen die klassischen Approximationen 4. bzw. 6. Ordnung zweimal bzw. dreimal soviel Kommunikation.

Die Größe der einzelnen Schichten (sprich die Oberfläche der Datenbereiche) ist nun aber von der verwendeten Datenverteilung abhängig, genauso wie die Anzahl der Nachbarn. So besitzt jeder Prozessor bei einer Scheibenaufteilung stets maximal 2 Nachbarn, allerdings ist die Oberfläche, unabhängig von der Anzahl der Prozessoren, gleich gross. Bei einer quaderförmigen Verteilung besitzen die Prozessoren zwar mehr Nachbarn, aber die Oberfläche der Quader nimmt mit steigender Prozessoranzahl ab.

Daher werden die mehrdimensionalen Aufteilungen mit wachsender Problemgröße und Prozessorzahl günstiger.

In unserem Fall ist eine Aufteilung in Scheibchen gewählt worden, einer späteren Änderung in mehrdimensionale Verteilung steht aber nichts im Wege.

Implementierte Glätter und Abwicklung der Kommunikation

Als Glätter wurden leicht abgewandelte Versionen des oben erwähnten Gauss-Seidel-SOR Verfahrens verwendet, und zwar ein gewichteter Red-Black-Gauss-Seidel für die klassische Approximation 2. Ordnung und die HOC4-Approximation sowie ein gewichteter 8-Farben-Gauss-Seidel für die HOC6-Version.

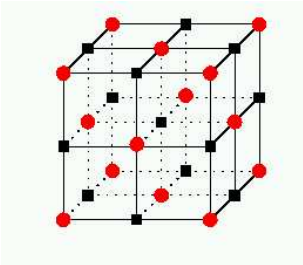


Abbildung 3: Rot-Schwarz-Aufteilung der Gitterpunkte beim Red-Black-Gauss-Seidel

Beim Gauss-Seidel-SOR wird die nächste Iterierte zur Hälfte aus schon aktualisierten Nachbarpunkten und zur Hälfte aus noch nicht aktualisierten Nachbarpunkten berechnet. Dieses Vorgehen wird bei den Mehrfarben-Gauss-Seidel-Schemata leicht modifiziert, indem die Gitterpunkte gefärbt werden (s. Fig. 3) und eine Iteration in Farbphasen unterteilt wird. In jeder dieser Phasen werden dann nur die Punkte einer Farbe aktualisiert. Beim Red-Black-Gauss-Seidel werden in der ersten Phase von allen roten Punkten nur nicht aktualisierte schwarze Punkte als Nachbarn verwendet, während in der zweiten Phase die schwarzen Punkte die bereits aktualisierten roten Punkte als Nachbarn besitzen. Analog läuft es beim 8-Farben-Gauss-Seidel Algorithmus in 8 Phasen ab.

Also muss bei der Kommunikation nicht nach jeder Iteration eine Schicht ausgetauscht werden, sondern nach jeder Farbphase eine Hälfte bzw. ein Achtel einer Schicht.

Realisiert wurde dies mittels der MPI-Routinen für persistente Kommunikation, so dass die Kommunikationen nicht in jeder Phase neu initialisiert werden müssen.

Zu Beginn der Glättungsiterationen wird die Datenaufteilung festgelegt. Dann werden je nach Farbschema Datentypen für die zu übergebenden Halb- bzw. Achtelschichten mittels `MPI_TYPE_HVECTOR` definiert, so dass die Daten direkt aus den entsprechenden Feldern verschickt und empfangen werden können, ohne dass ein Zwischenspeichern notwendig ist. Dann werden alle Kommunikationen einmal mittels `MPI_RECV_INIT` und `MPI_SENT_INIT` initialisiert und die Request-Handles generiert. Schließlich wird zwischen den Farbphasen nur noch die Kommunikation über die Request-Handles mittels `MPI_STARTALL` gestartet und auf die Komplettierung mittels `MPI_WAITALL` gewartet.

Theoretisch wäre dies dann die einzig nötige Kommunikationsart im Glätter. Aufgrund des seriellen Frameworks des Mehrgitterprogramms müssen aber momentan noch am Ende der Glättungsiterationen auf einem Gitter die kompletten Werte mit allen Prozessoren synchronisiert werden und alle Residuen auf allen Prozessoren berechnet werden, da die Gitterspeicherung und der Transfer zwischen den Gittern noch nicht parallelisiert ist. Wenn dies in Zukunft einmal der Fall ist, entfällt die Notwendigkeit hierzu und jeder Prozessor muss nur auf seinem Gebiet die Residuen berechnen und beim Transfer ist auch nur noch höchstens an den Gebietsrändern eine Kommunikation erforderlich.

Momentan beeinträchtigt dieser Zustand aber, wie wir weiter unten sehen werden, die Skalierung beim Einsatz der Glätter im Mehrgitterverfahren erheblich.

Zu erwähnen ist noch, dass man beim Mehrgitterverfahren auf den groben Gittern teilweise weniger Punkte iteriert als Prozessoren zur Verfügung stehen. Im vorliegenden Fall wurde ab dem Gitterlevel, ab dem die Anzahl der Punkte entlang einer Kante gleich oder größer der Anzahl der Prozessoren ist, seriell weitergearbeitet. D.h. hierbei werden keine Daten zwischen den Prozessoren ausgetauscht, die alle dasselbe rechnen. Hierbei entfällt dann auch die normalerweise durchgeführte Synchronisation der Residuen.

Testprobleme

Die implementierten Verfahren wurden auf einer Reihe von Testproblemen verifiziert und auf ihre Skalierung untersucht. Für die Testprobleme war im allgemeinen die analytische Lösung bekannt, so dass die Fehler der Verfahren berechnet werden konnten.

Coulomb-Problem

Dieser Fall kommt dem späteren Einsatz am nächsten: Gegeben ist eine diskrete zufällige Verteilung von geladenen Teilchen im Raum, deren Ladungen auf das feinste Gitter verschmiert werden. Dann wird die Poissongleichung mit den verschmierten Ladungen als rechter Seite gelöst.

Testproblem 1

Es wird die Poissongleichung mit der Dirichlet-Randbedingung eines auf dem Rand verschwindenden Potentials und dem Quellterm

$$f_{ijk} = 3\pi^2 \sin(\pi i h) \sin(\pi j h) \sin(\pi k h)$$

gelöst.

Testproblem 2

Es wird die Laplacegleichung, d.h. $f_{ijk} = 0$, mit Dirichlet-Randbedingung behandelt.

$$\begin{aligned} \phi_{ijk} = &\quad \sin(\pi jh)\,\sin(\pi kh) &\quad i = 0 \\ \phi_{ijk} = &\quad 2\sin(\pi jh)\,\sin(\pi kh) &\quad i = N \\ \phi_{ijk} = &\quad 0 &\quad j,k \in \{0, N\} \end{aligned}$$

Skalierung der Verfahren als Glätter auf dem feinsten Gitter (Eingitterverfahren)

Die folgenden Grafiken zeigen die Skalierung der Verfahren 2. und 4. Ordnung (s. Fig 4,7), den Einfluss des Relaxierungsparameters ω auf die Konvergenz (des Mehrgitterverfahrens) (s. Fig 6,8), sowie exemplarisch das Verhältnis zwischen der Kommunikations- und der Rechenzeit (s. Fig 5). Die Rechnungen wurden auf dem ZAMpano [3] (32 Pentium-III-Xeon CPUs auf 8 Knoten) durchgeführt und zwar auf einem Gitter der Schrittweite $h = \frac{1}{64}$ mit $\Omega = [0, 1]^3$ und den Testproblemen 1 und 2. Das Kriterium für den Abbruch war ein Residuum pro Gitterpunkt kleiner als 10^{-10}.

Beim Einsatz der Verfahren als Glätter auf dem feinsten Gitter zeigen diese eine gute Skalierung. Die relativ schlechte Skalierung beim Einsatz vom 4 Prozessoren auf einem Knoten ist darauf zurückzuführen, dass das Kommunikationsnetzwerk des ZAMpano in diesem Fall einfach überfordert ist und Engpässe im Datentransfer entstehen.

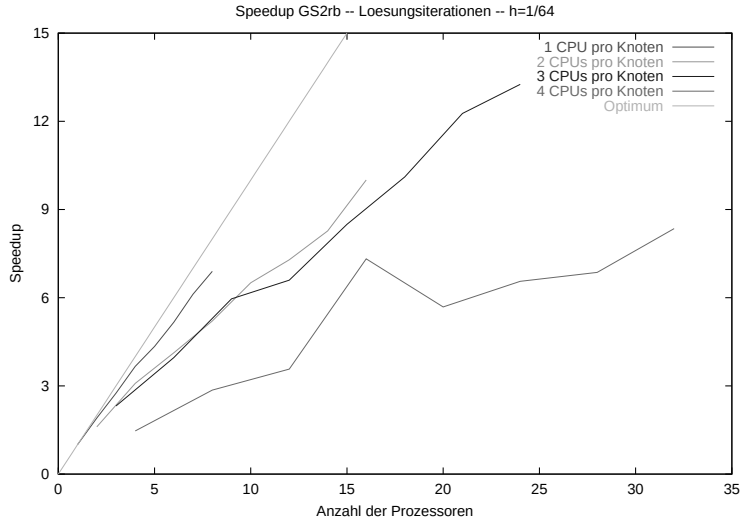


Abbildung 4: Klasssiche Approximation 2. Ordnung mit Gauss-Seidel-Red-Black

Skalierung beim Mehrgitterverfahren

Die folgende Skalierung (s. Fig 9) ist auf dem ZAMpano bei der Anwendung der HOC4-Approximation im Mehrgitterverfahren mit 3 Glättungsschritten auf jeder Ebene entstanden. Das feinste Gitter hat Schrittweite $h = \frac{1}{128}$ und es ist wieder $\Omega = [0, 1]^3$.

Hier sieht man deutlich, wie die notwendige Synchronisation vor einem Gitterwechsel die gute Skalierung des Glätters wieder zunichte macht, da im Mehrgitterfall die Transfers zwischen den Gittern über die Iterationen auf den einzelnen Gittern dominieren.

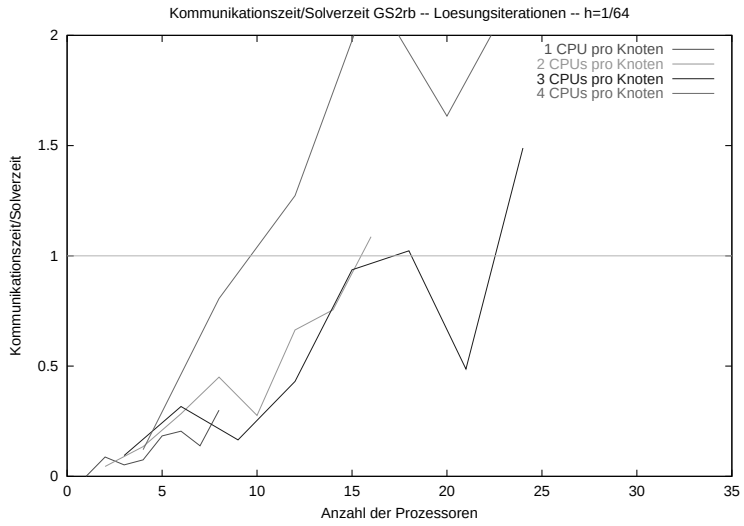


Abbildung 5: Klassische Approximation 2. Ordnung mit Gauss-Seidel-Red-Black

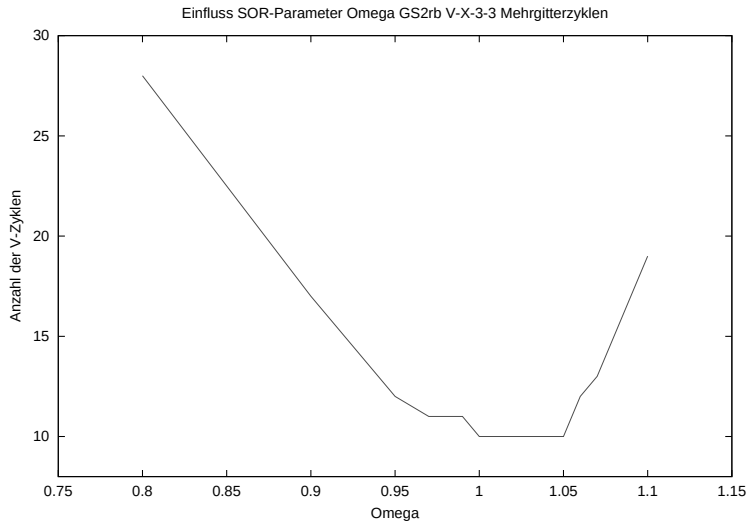


Abbildung 6: Klassische Approximation 2. Ordnung mit Gauss-Seidel-Red-Black

Zusammenfassung und Ausblick

Es wurde gezeigt, dass man bei der Verwendung der HOC-Verfahren und Mehrfarben-Gauss-Seidel-Iterationen sehr gut skalierende Glätter erhält. Diese werden momentan im Mehrgitterverfahren noch durch das serielle Framework ausgebremst, sind aber auch für das Mehrgitterverfahren sehr vielversprechend.

Als Ausblick wollen wir nun noch ein paar Punkte betrachten, mit denen man sich in der Zukunft noch beschäftigen könnte.

Zum einen hat man den bereits erwähnten Punkt der parallelen Datenverteilung des Gitters sowie und vor allem die Parallelisierung des Transfers zwischen den Gittern, die einen hohen Performancegewinn verspricht.

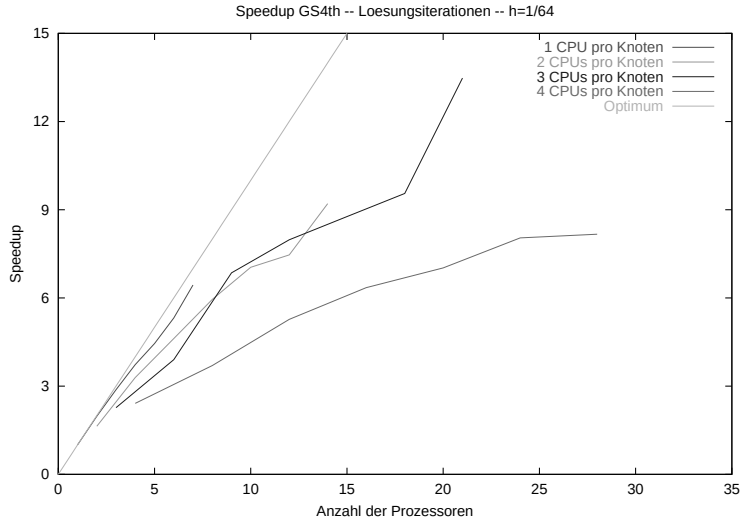


Abbildung 7: HOC Approximation 4. Ordnung mit Gauss-Seidel-Red-Black

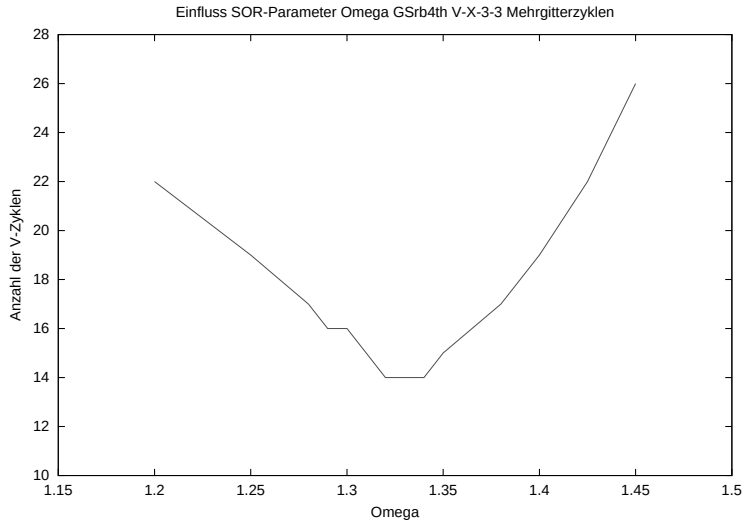


Abbildung 8: HOC Approximation 4. Ordnung mit Gauss-Seidel-Red-Black

Weiterhin wäre im Hinblick auf große Probleme und größere Rechner mit mehr Prozessoren eine Implementierung von mehrdimensionaler Kommunikation interessant, die wie oben erwähnt den Kommunikationsaufwand für diese Fälle verringern könnte.

Ein weiterer Punkt wäre die Approximation der rechten Seite im HOC6-Verfahren. Wie oben erwähnt, verlangt dort die Approximation der rechten Seite Ableitungen 4. Ordnung von f. Wenn man diese approximiert bleibt einem nur der Weg über die klassischen Verfahren, die dann auch die übernächsten Nachbarpunkte benötigen. Dies stellt am Rand von Ω ein Problem dar. Momentan wurde es so gelöst, dass einfach eine Dummy-Randschicht durch Duplikation der vorhandenen Randschicht erzeugt wird. Dies erzeugt aber besonders auf den gröberen Gittern relativ viel zusätzliche Last und ist auch nur bedingt korrekt. Interessant wäre dort die lokale Approximation der Ableitungen von f am Rand durch Verfahren 2. Ordnung, die wiederum nur eine Randschicht benötigen.

Abschliessend wäre noch zu überlegen, bis zu welchem Gitterlevel sich die Parallelisierung lohnt. Of-

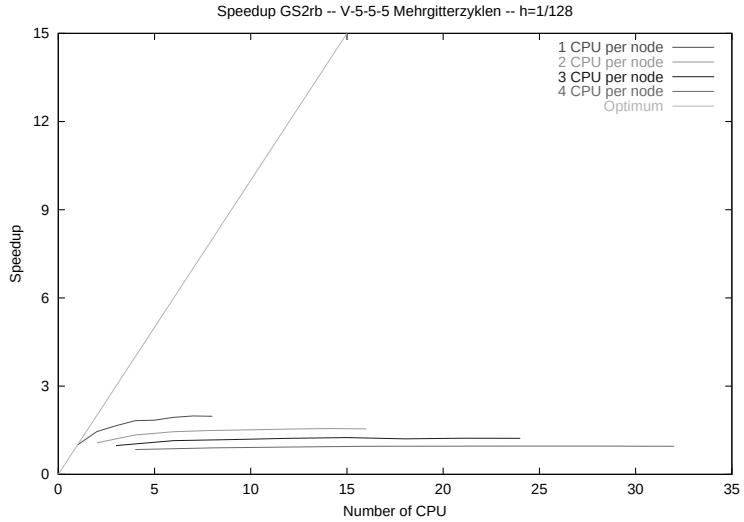


Abbildung 9: HOC Approximation 4. Ordnung mit Gauss-Seidel-Red-Black

fensichtlich wäre z.B. bei der Aufteilung des Raumes in Scheiben eine Parallelisierung nicht sinnvoll, wenn jeder Prozessor nur noch eine Ebene oder weniger zu berechnen hätte, da der Kommunikationsoverhead dort sehr zu Buche schlagen würde. Dieser spezielle Fall wird in der obigen Implementierung auch schon auf einfache Weise gelöst, indem dann alle Prozessoren das Problem seriell bearbeiten, bis wieder auf ein gröberes Gitter gewechselt wird. So spart man sich zumindest die Kommunikation in diesem Bereich. Diese Behandlung liesse sich aber mit Sicherheit noch verfeinern. Insbesondere wenn die Gitterspeicherung und der Transfer parallelisiert werden, müßte man sich hierüber noch Gedanken machen.

Literatur

1. J. Zhang, Fast and high accuracy multigrid solution of the three dimensional Poisson equation.
   J. Comput. Phys., 143:449-461 (1998).
2. W. F. Spotz and G. F. Carey, A high-order compact formulation for the 3D Poisson equation.
   Numer. Methods Partial Differential Equations, 12:235-243 (1996)
3. ZAMpano: ZAM Parallel Nodes
   http://zampano.zam.kfa-juelich.de
4. MPI: Message Passing Interface Standard
   http://www.mpi-forum.org
5. William L. Briggs, A Multigrid Tutorial.
   SIAM (1987)
6. U. Trottenberg, C.W. Oosterlee and A. Schüller, Multgrid.
   Academic Press, San Diego (2001)
7. W.F.Ames, Numerical Methods for Partial Differential Equations.
   Academic Press, New York (1977)
8. Y. Gu, W. Liao and J. Zhu, An effi cient high order algorithm for solving systems of 3-D reaction-diffusion equations.
   J. Comp. Appl. Math., 155:1-17 (2003)
9. G. Sutmann and B. Steffen, High-Order Compact Solvers for the Three Dimensional Poisson Equation
   to be published

Parallele lineare Algebra für blockdiagonale Matrizen

Matthias Bolten

Universität zu Lübeck

E-mail: bolten@informatik.uni-luebeck.de

Zusammenfassung: Beim Einsatz von numerischen Berechnungen in der theoretischen Chemie treten blockdiagonale Matrizen auf. Ein Ausnutzen dieser Struktur stellt eine massive Rechenzeitersparniss dar. Da die bekannten Pakete zur parallelen Verarbeitung keine Funktionen speziell zur Behandlung dieses Matrixtyps zur Verfügung stellen, wurden im Rahmen dieser Arbeit Routinen entwickelt, um diese Matrizen mit Hilfe von ScaLAPACK zu verarbeiten. Die hier vorgestellten Unterprogramme machen die Arbeit mit allen in ScaLAPCK für voll besetzte Matrizen vorhanden Funktionen möglich und zeigen insbesondere für aufwändige Probleme, wie die Lösung des Eigenwertproblems, eine zufriedenstellende Geschwindigkeitssteigerung.

Einleitung

Die Computational Chemistry ist – obwohl sehr jung – inzwischen zu einem wichtigen Zweig der theoretischen Chemie geworden. Die hierbei angestellten Berechnungen sind sehr aufwendig und daher nur für kleinere Moleküle sinnvoll auf Workstations durchzuführen. Beim Einsatz von Supercomputern ist eine Parallelisierung erforderlich. Die bei symmetrischen Molekülen auftretenden blockdiagonalen Matrizen (siehe Abbildung 1) gestalten die parallelen Verarbeitung deutlich aufwändiger.

Theoretischer Hintergrund

Zuerst ein kleiner Überblick über die theoretischen Hintergründe der Berechnungen. Eine weitergehende Einführung in die theoretische Chemie findet sich zum Beispiel in [1], [2] oder [3].

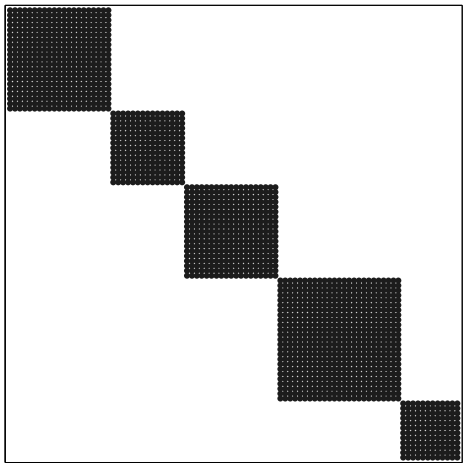


Abbildung 1: Beispiel für eine blockdiagonale Matrix.

Quantenmechanik und die Schrödingergleichung

Bei quantenmechanischer Betrachtung eines N-Teilchensystems, z.B. eines Moleküls, wird dieses durch die Zustandswellenfunktion $\Psi(x_1, y_1, z_1, \ldots, x_N, y_N, z_N, t)$ der Ortskoordinaten aller Teilchen (x_i, y_i, z_i, $(i = 1, \ldots, N)$) und der Zeit (t)beschrieben. Das Betragsquadrat $|\Psi|^2$ wird hierbei als Wahrscheinlichkeit interpretiert, dass die N Teilchen zum Zeitpunkt t die Koordinaten $x_i, y_i, z_i, (i = 1, \ldots, N)$ haben.

Die Wellenfunktion Ψ kann nun als Lösung der *Schrödingergleichung*

$$\mathbf{H}\Psi = E\Psi = i\hbar\frac{\partial\Psi}{\partial t} \tag{1}$$

gewonnen werden, wobei $\mathbf{H}$ der *Hamilton-Operator* zur Observablen E, der Energie, ist. Der Hamilton-Operator ist abhängig von den berücksichtigten Teilchenwechselwirkungen, der für ein Molekül mit ruhenden Kernen lautet beispielsweise:

$$\mathbf{H} = -\frac{1}{2}\sum_i \triangle_j - \sum_A \sum_k \frac{Z_A}{r_{Ak}} + \sum_{A<B} \frac{Z_A Z_B}{r_{AB}} + \sum_{i<j} \frac{1}{r_{ij}} = \mathbf{T} + \mathbf{V}_{ne} + \mathbf{V}_{nn} + \mathbf{V}_{ee}$$

Hierbei beschreibt die erste Summe die kinetische Energie der Elektronen, die Zweite die Wechselwirkung der Kerne und der Elektronen, die dritte Summe die der Kerne untereinander und die Vierte die der Elektronen untereinander.

Da der Hamilton-Operator nicht zeitabhängig ist, ist man oft an der Lösung der *stationären* oder *zeitunabhängigen Schrödingergleichung* interessiert. In diesem Fall ist die Gesamtenergie zeitunabhängig und (1) ergibt sich zu:

$$\begin{aligned} \mathbf{H}(\mathbf{x})\Psi(\mathbf{x},t) &= E\Psi(\mathbf{x},t) \\ \Leftrightarrow \mathbf{H}(\mathbf{x})\psi(\mathbf{x})\chi(t) &= E\psi(\mathbf{x})\chi(t) \\ \Leftrightarrow \mathbf{H}(\mathbf{x})\psi(\mathbf{x})e^{-i\frac{E}{\hbar}t} &= E\psi(\mathbf{x})e^{-i\frac{E}{\hbar}t} \\ \Leftrightarrow \mathbf{H}(\mathbf{x})\psi(\mathbf{x}) &= E\psi(\mathbf{x}) \end{aligned}$$

Die Schrödingergleichung ist im allgemeinen nicht geschlossen lösbar.

Numerische Näherungslösung der Schrödingergleichung

Die gesuchte Lösung Ψ der Schrödingergleichung muß geeignet approximiert werden, will man die Schrödingergleichung numerisch lösen. Zunächst ist eine endliche Basis zu finden, in der man Ψ approximiert; dies ist in so fern schwierig, da der Raum, aus dem Ψ stammt unendlich dimensional ist.

Bei der *Hartree-Fock*- oder *SCF*-Näherung geht man zur Lösung der elektronischen Schrödingergleichung eines Moleküls mit N Elektronen folgendermaßen vor: Die Spinorbitale φ_j werden als Linearkombination der gewählten Basis, z.B. atomzentrierten Gauss-Funktionen (GTOs - Gauss Type Oribtals) $\chi_\mu(x, y, z)$ entwickelt:

$$\varphi_j = \sum_\mu c_{j\mu}\chi_\mu(x, y, z)\alpha_j$$

Dann wird die N-Teilchen-Wellenfunktion ψ des Moleküls als antisymmetrisiertes Produkt von N Einteilchen-Spinorbitalen φ_j ausgedrückt:

$$\psi = \mathcal{A} \prod_{i=1}^{N} \varphi_i(i)$$

Dieses antisymmetrisierte Produkt lässt sich auch als *Slater-Determinante* schreiben.

Die auf dem vorgehend beschriebenen Hamiltonoperator basierende exakte Energie dieser Eindeterminantennäherung der Wellenfunktion ergibt sich zu:

$$E = \sum_{i=1}^{N} (\varphi_i, \mathbf{h}\varphi_i) + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left\{ (\varphi_i(1)\varphi_j(2), \frac{1}{r_{12}} \varphi_i(1)\varphi_j(2)) - (\varphi_i(1)\varphi_j(2), \frac{1}{r_{12}} \varphi_j(1)\varphi_i(2)) \right\}$$

Hierbei ist $\mathbf{h}$ der Einelektronenoperator. Die elektronische Gesamtenergie wird nun minimiert, indem man die Koeffizienten $c_{j\mu}$ der Spinorbitale φ_i variiert. E ist, falls die φ_i orthogonal sind, gerade dann stationär, wenn die Spinorbitale die Hartree-Fock-Gleichungen erfüllen(siehe [2]):

$$\mathbf{F}\varphi_i = \epsilon_i \varphi_i \tag{6}$$

Hierbei sind die ϵ_i die Orbitalenergien und der *Fock-Operator* $\mathbf{F}$ ist gegeben als

$$\mathbf{F} = \mathbf{h} + \sum_j (\mathbf{J}^j - \mathbf{K}^j),$$

wobei

$$\mathbf{J}^j(1) = \int \frac{|\varphi_j(2)|^2}{r_{12}} \mathrm{d}\tau_2, \tag{8}$$

$$\mathbf{K}^j(1)\vartheta(1) = \int \frac{\varphi_j^*(2)\varphi_j(1)\vartheta(2)}{r_{12}} \mathrm{d}\tau_2. \tag{9}$$

Die Gesamtenergie wird nun iterativ minimiert. Zuerst werden Startorbitale geeignet gewählt. Danach wird zunächst die Dichtematrix und dann die *Fockmatrix* als Diskretisierung des Fock-Operators berechnet. Die neuen Orbitale sind dann als Eigenvektoren der Fockmatrix (entspricht Eigenfunktionen φ_i zu den Eigenwerten ϵ_i in (6)) gegeben. Falls die Konvergenzkriterien (u.a. keine Änderung der Energiebeiträge) erfüllt sind, bricht das Verfahren ab. Ein Ablaufdiagramm für den Algorithmus findet sich in Abbildung 2. Ein weitergehender Überblick über die *Ab initio*-Methoden der theoretischen Chemie und tiefergehende einführende Literatur findet sich z.B. in [4].

Symmetrie

Beim Aufbau der Fockmatrix spielt Symmetrie eine Rolle. Bei einem symmetrisches Molekül, wie z.B. dem Cyclopropan (siehe Abbildung 3), ist es möglich verschiedene Operationen durchzuführen, die das Molekül wieder in eine deckungsgleiche Lage bringen. Im Beispiel kann man das Molekül in der Ebene,

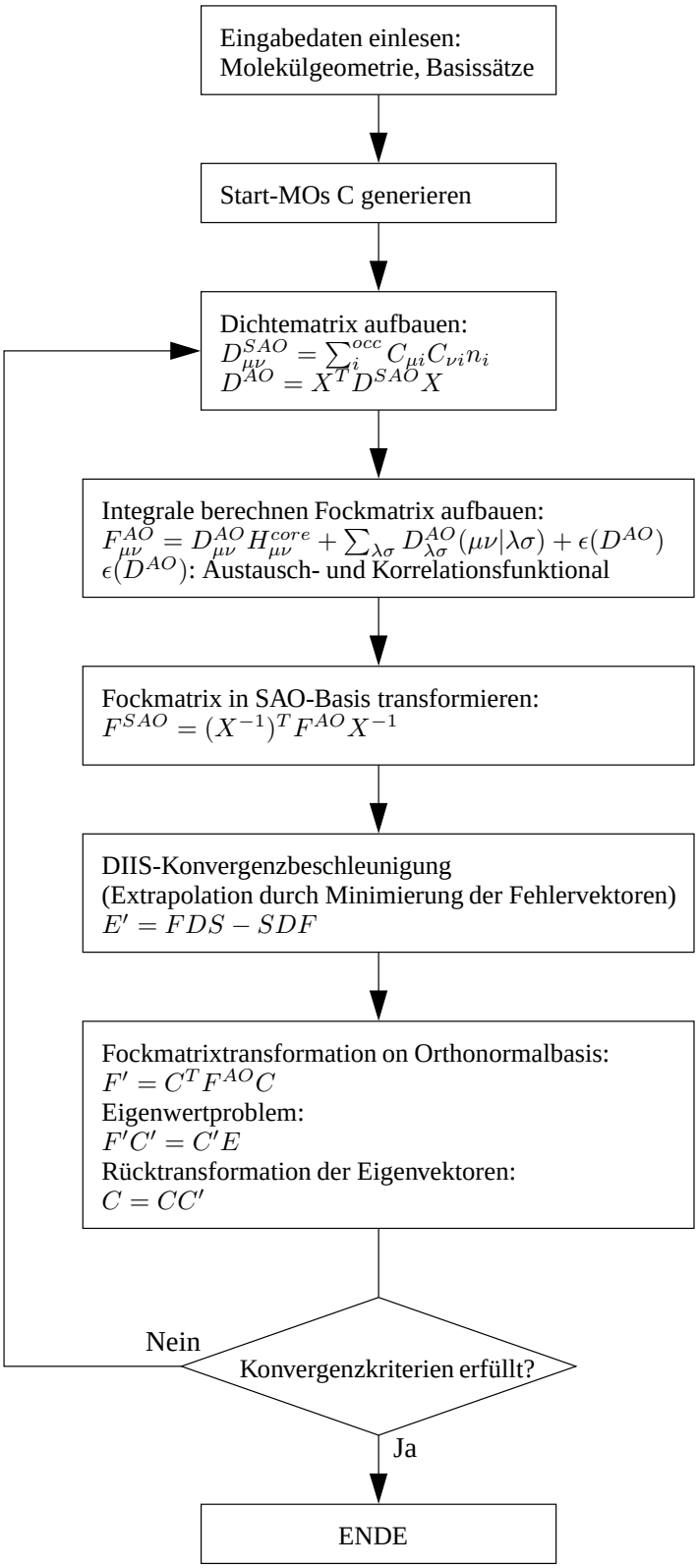


Abbildung 2: Ablauf des Hartree-Fock-Verfahrens.

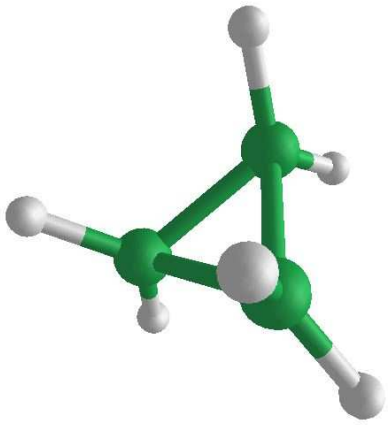


Abbildung 3: Cyclopropan: Molekül aus der Punktgruppe D_{3h}, eine Symmetriegruppe der Ordnung 12.

die durch die drei C-Atome aufgespannt wird, z.B. um 120° oder 240° drehen, ohne dass sich am Bild etwas ändert. Die Menge der verschiedenen Symmetrieoperationen, die ein Molekül erlaubt, ist eine Gruppe (siehe [1]). Die Kardinalität dieser Menge wird auch als Ordnung der Gruppe bezeichnet. Im Beispiel in Abbildung 3 hat die Symmetriegruppe des Moleküls die Ordnung 12, d.h. es neben den bereits genannten Drehungen noch 10 andere Operationen, die das Cyclopropan in eine deckungsgleiche Lage bringen.

Der Fock-Operator kommutiert mit den Symmetrieoperationen, daher folgt, dass die Einträge $F_{\mu\sigma}$ der Fockmatrix verschwinden, wenn die Orbitale μ und σ zu verschiedenen irreduziblen Darstellungen gehören. Das bedeutet, dass bei der entstehenden Matrix die Diagonale mit Blöcken besetzt ist und die restlichen Elemente der Matrix verschwinden, die Anzahl der Blöcke ist dabei gleich der Anzahl der irreduziblen Darstellungen.

Numerische Komplexität

Um zu verdeutlichen, warum diese Verfahren parallelisiert werden, nun zur numerischen Komplexität. In Tabelle 1 sind einige Zeiten aufgetragen, die für eine Iteration auf mehreren Prozessoren eines Supercomputers benötigt werden. Neben Diamant (zur Diamantstruktur: siehe Abbildung 4) und Graphit (Abbildung 5), die symmetrisch sind, finden sich auch Zeolith und das Coenzym B_{12} (siehe hierzu Abbildung 6), die keine Symmetrie aufweisen. Offensichtlich kann mit Ausnutzung der Symmetrie viel Rechenzeit eingespart werden, so ist die Lösung des Eigenwertproblems beim $C_{165}H_{100}$ ungefähr 300-mal schneller als beim $Si_{48}O_{112}H_{32}$, obwohl die beiden eine ähnliche Dimension haben, auch die DIIS-Konvergenzbeschleunigung ist noch ca. 35-mal schneller.

Parallele lineare Algebra

Wenn man Probleme der numerischen linearen Algebra auf einem Parallelrechner lösen möchte, muss man sich um eine Reihe weiterer Probleme Gedanken machen, die man im seriellen Fall nicht hat. Drei wichtige Punkte sind bei der Auswahl und Implementierung eines parallelen Algorithmus für die numerische lineare Algebra zu beachten. Eine Einführung in die parallele Numerik findet sich zum Beispiel in [5].

Einige Algorithmen, die man etwa auf einer Workstation zur Lösung benutzt, sind überhaupt nicht parallelisierbar. Als Beispiel sei hier der Thomas-Algorithmus zur LU-Zerlegung einer tridiagonalen Ma-

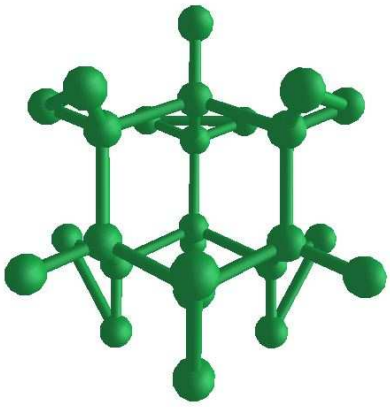


Abbildung 4: Diamantstruktur.

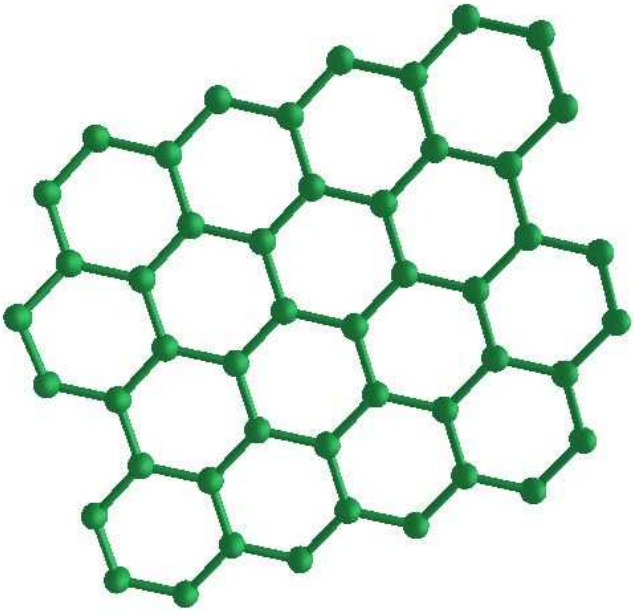


Abbildung 5: Graphitstruktur.

	Diamant	Diamant	Zeolith	Graphit
	$C_{165}H_{100}$	$C_{455}H_{196}$	$Si_{48}O_{112}H_{32}$	$C_{150}H_{30}$
Sym., Dim., #Blöcke	T_d, 2810, 5	T_d, 7350, 5	C_1, 2592, 1	D_{6h}, 2250, 12
Dichtematrix (AO)	0.5	3.9	1.5	0.3
Fockmatrix (AO)	5.7	24.1	30.0	2.8
Fockmatrix (AO → SAO)	0.2	2.5	0.7	0.2
DIIS	0.1	1.4	3.5	0.6
Eigenwertproblem	0.2	3.8	62.4	0.2
Prozessoren	4	4	8	4

	Graphit	Graphit	Graphit	Coenzym B_{12}
	$C_{294}H_{42}$	$C_{384}H_{48}$	$C_{600}H_{60}$	$C_{63}N_{14}O_{14}H_{88}PCo$
Sym., Dim., #Blöcke	D_{6h}, 4326, 12	D_{6h}, 5616, 12	D_{6h}, 8700, 12	C_1, 4362, 1
Dichtematrix (AO)	1.1	2.4	6.2	-
Fockmatrix (AO)	8.3	13.0	26.9	50.2
Fockmatrix (AO → SAO)	0.7	1.1	2.9	-
DIIS	0.3	0.9	5.0	2.6
Eigenwertproblem	1.2	1.4	5.0	61.8
Prozessoren	4	4	4	12

Tabelle 1: Zeiten in Sekunden für eine Iteration des Hartree-Fock-Verfahrens. Ausgeführt auf dem IBM p690 Cluster JUMP, ohne die hier vorgestellten Erweiterungen zur speziellen Behandlung blockdiagonaler Matrizen.

trix genannt. Dieser Algorithmus benötigt in jedem Schritt das Ergebnis aus dem direkt vorangehenden und ist damit inhärent sequentiell. Andere Algorithmen sind nur mit gewissen Änderungen parallelisierbar, wie zum Beispiel das Gauss-Seidel-Verfahren als Glätter bei Mehrgitterverfahren zur Lösung von partiellen Differentialgleichungen mittels finiter Differenzen. Bei lexikographischer Ordnung der Gitter-

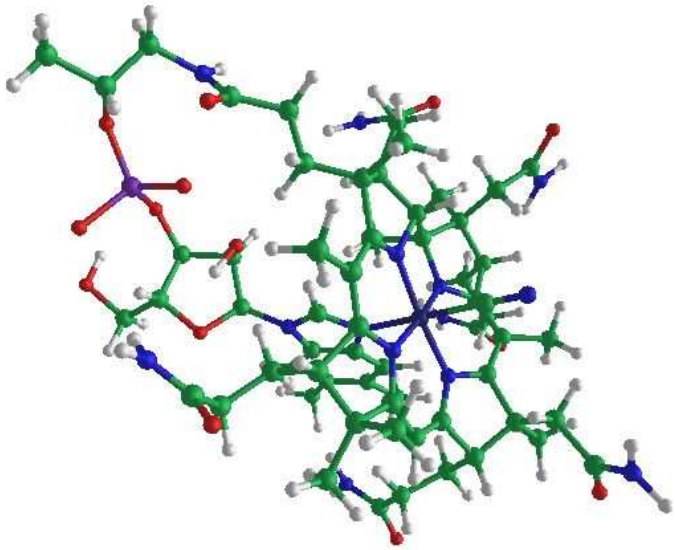


Abbildung 6: Coenzym B_{12}.

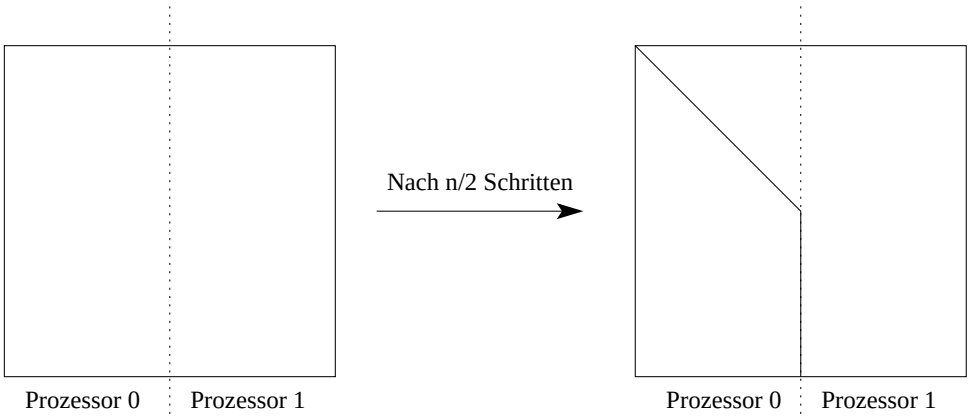


Abbildung 7: Problem der Datenverteilung beim Gauss-Verfahren auf zwei Prozessoren.

punkte ist der Glätter aus den selben Gründen wie der Thomas-Algorithmus nicht parallelisierbar, bei Schachbrett- oder Rot-Schwarz-Ordnung der Gitterpunkte ist er parallelisierbar. Dabei wirkt der Glätter jedoch etwas anders.

Weiter gibt es – wie meist beim Übergang vom seriellen zum parallelen Programm – Algorithmen, die zwar auf einer Uniprozessormaschine eine gute Laufzeit bieten, die aber schlecht skalieren und somit auf einem Mehrprozessorsystem eine schlechtere Wahl darstellen als ein gut skalierender Algorithmus, der kein so gutes serielles Laufzeitverhalten hat.

Ein wichtiger Punkt ist die Datenverteilung, ein Problem, das man auf einem Einprozessorsystem natürlich gar nicht hat. Zunächst ein Beispiel, wie sich dieses Problem äußert: Wenn man das Gauss'sche Eliminationsverfahren auf zwei Prozessoren verteilen will und hierzu die $N \times N$-Matrix einfach in zwei Blöcke zerteilt, so hat Prozessor 0, der den ersten Block bearbeitet, nach $N/2$ Schritten seine Arbeit erledigt, während Prozessor 1 die restlichen $N/2$ Schritte alleine ausführen muss (siehe Abbildung 7).

Die Lösung dieses Problem ist recht einfach: Anstelle einer Zerlegung in p Blöcke auf einer p-Prozessormaschine wählt man eine zyklische Verteilung in mehr Blöcke als Prozessoren. Bei einer Zweiprozessormaschine führt dies zu einer spalten- oder zeilenzyklischen Verteilung, bei mehr Prozessoren auf eine blockzyklische Verteilung. Die Blockgröße ist dabei ein weiterer Parameter, mit dem man den Algorithmus optimieren kann (kleinere Blöcke bedeuten unter Umständen eine bessere Lastverteilung aber mehr Kommunikation). Die blockzyklische Verteilung ist dabei so, dass jeder Block einer Matrix der Reihe nach auf ein Prozessorgitter verteilt wird, wobei das Prozessorgitter periodisch fortgesetzt wird. Ein Beispiel für eine blockzyklische Verteilung auf vier Prozessoren ist in Abbildung 8 zu finden.

Implementierung

Die Implementierung der Routinen zur Bearbeitung der blockdiagonalen Matrizen wurde so gehalten, dass sie sich möglichst einfach in die Bibliothek, die am Zentralinstitut für Angewandte Mathematik zur Parallelisierung des Pakets *TURBOMOLE* verwendet wird, einbinden lassen.

Verwendete Programmpakete

Zur Parallelisierung von TURBOMOLE wird das *Global Arrays Toolkit* benutzt, daher sollte es auch bei den erstellten Routinen verwendet werden. Weiter sollten die Blöcke der blockdiagonalen Matrizen mit Hilfe von *ScaLAPACK* bearbeitet werden, da in diesem Paket eine Vielzahl von Routinen der numerischen linearen Algebra in einer Variante für Parallelrechner mit verteiltem Speicher implementiert

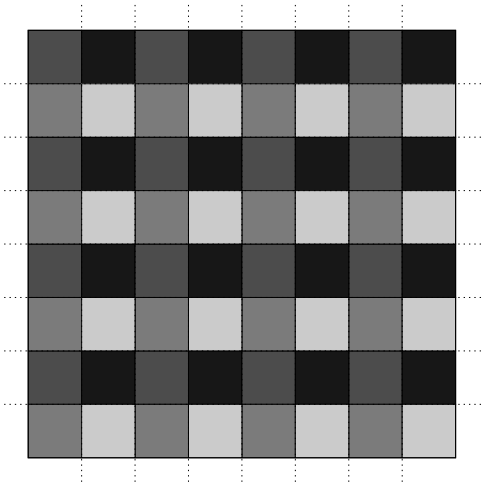


Abbildung 8: Blockzyklische Verteilung einer Matrix auf vier Prozessoren.

sind.

Global Arrays Toolkit

Das Global Arrays (GA) Toolkit wurde am Pacific Northwest National Laboratory (PNNL) entwickelt. Das GA Toolkit stellt verteilte (n-dimensionale) Felder bereit, um deren Verteilung und Verwaltung der Benutzer dieses Toolkits sich nicht zu kümmern braucht. Es sind sowohl reguläre als auch irreguläre Verteilungen möglich. Der Zugriff auf diese Felder erfolgt dabei von jedem Knoten aus gleich, egal wo die Daten physikalisch gespeichert sind, mittels einfacher Operationen wie *put*, *get* etc. Für den Nutzer dieser Bibliothek sind diese Operationen als einseitige Kommunikation implementiert. Weiter gibt es auch Operationen um etwa die Verteilung abzufragen oder auf den lokalen Speicher direkt zuzugreifen. Die Benutzung des GA Toolkit ist zudem aus Fortran, C, C++ und Python möglich, so dass man leicht in heterogenen Programme mit denselben Daten arbeiten kann, ein Feature, welches hier jedoch nicht benötigt wurde.

ScaLAPACK

Das ScaLAPACK-Packet (siehe $[6]$) wurde am Oak Ridge National Laboratory (ORNL) und der Rice University, der University of California, Berkeley, der University of California, Los Angeles, der University of Illinois und der University of Tennessee, Knoxville entwickelt. Es stellt eine parallele Version vieler LAPACK-Routinen für Parallelrechner mit verteiltem Speicher zur Verfügung. Dabei setzt ScaLAPACK auf den Bibliotheken BLAS (Basic Linear Algebra Subprograms), BLACS (Basic Linear Algebra Communication Subprograms) und PBLAS (Parallel BLAS) auf, so dass direkt von etwaigen Optimierungen dieser Bibliotheken profitiert wird. Insbesondere von der BLAS-Bibliothek stehen viele Hersteller-optimierte Versionen (z.B. die Intel BLAS) oder auch eine automatisch für Maschinen mit Cache optimierte Version in der ATLAS (Automatically Tuned Linear Algebra Software) bereit.

Idee

Bei der Parallelisierung von TURBOMOLE werden die auftretenden Matrizen Block für Block ganz streifenweise (bei unsymmetrischen Blöcken) bzw. nur der obere Teil streifenweise (bei symmetrischen Blöcken) hintereinander in einem Vektor abgespeichert. Die Grundidee war nun, die Blöcke nebeneinander in ein großes Feld der Größe $N \times \max(N_i)$, wobei die N_i die Blockgrößen der einzelnen Blöcke sind,

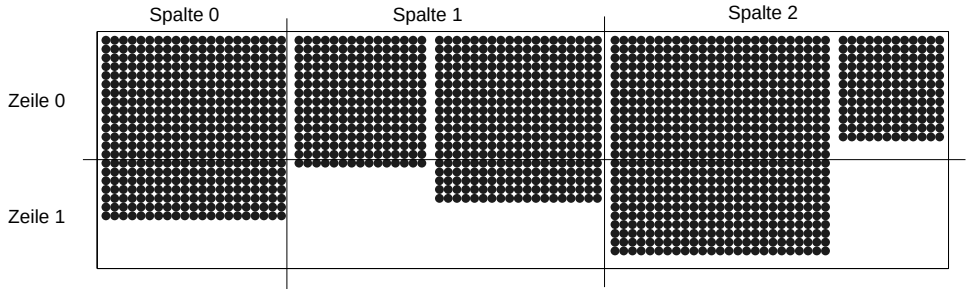


Abbildung 9: Abspeicherung der Blöcke der Beispielmatrix aus Abbildung 1 in einem Feld und Verteilung dieses Feldes auf ein 2×3-Prozessorgitter.

zu speichern. Dieses Feld muss dann nur so noch auf ein geeignetes Prozessor-Gitter verteilt werden, wie ScaLAPACK es erwartet, und die Matrix kann dann mit verschiedenen PBLAS- oder ScaLAPACK-Routinen bearbeitet werden. Die Bearbeitung der Blöcke kann dann parallel erfolgen, da die Blöcke ja entkoppelt sind, und jeder Block für sich kann, durch die Verwendung von ScaLAPACK, parallel verarbeitet werden.

Details

Bei der Umsetzung der skizzierten Idee entstanden im wesentlichen zwei Routinen, die für den Benutzer von Bedeutung sind, `matrix_block` und `matrix_unblock`, sowie eine Reihe von Hilfsroutinen. Die Implementierung erfolgte in Fortran 77 mit einigen Fortran 90-Erweiterungen.

Die Routine `matrix_block` nimmt hierbei ein Global Array, in dem die Matrix wie beschrieben abgespeichert ist und Informationen über die Größe der einzelnen Blöcke als Eingabe. Mit Hilfe dieser Daten wird dann zunächst berechnet, wie das zu erzeugende Prozessorgitter aussehen soll, dann werden die einzelnen Spalten des Prozessorgitters den Blöcken zugeordnet, wobei die Grenzen so verschoben werden, dass eine Blockgrenze gleichzeitig Prozessorgrenze ist. Nun wird ein Global Array der Größe $N \times \max(N_i)$ erzeugt, wobei eventuell zusätzlich durch die blockzyklische Verteilung benötigter Speicher berücksichtigt wird. Ein Beispiel wie dieses Feld für die Beispielmatrix aus Abbildung 1 aussehen würde ist in Abbildung 9 zu finden. Dann werden für jeden Block ein Array Descriptor und ein BLACS-Context, dieser enthält die Informationen über das für diesen Block zuständige Subgitter, für ScaLAPACK erzeugt und die Daten werden aus dem Eingabevektor in das neue Feld kopiert. Bei dieser Kopie wird zudem die blockzyklische Zerlegung berücksichtigt, so dass jeder Prozessor auf seinem Teil des Global Arrays die Daten zur Verfügung hat, die er benötigt, siehe hierzu Abbildung 10.

Die Routine `matrix_unblock` kopiert diese geblockte Matrix wieder zurück in die Ausgangsform.

Ablauf einer PBLAS-/ScaLAPACK-Routine auf einer blockdiagonalen Matrix

Die Durchführung einer Operation der linearen Algebra auf einer blockdiagonalen Matrix erfolgt nun einfach in der folgenden Reihenfolge:

1. Erzeugen der ScaLAPACK-Kopien der beteiligten Matrizen mit Hilfe der Routine `matrix_block`.

2. Jeder Prozessor prüft für jeden Block ob er diesen bearbeiten muss, wenn ja, wird die entsprechende ScaLAPACK-Routine angestossen. Durch die verschiedenen Kontexte ist dabei jedem Prozessor bekannt, mit welchen anderen Prozessoren er zusammenarbeiten muß.

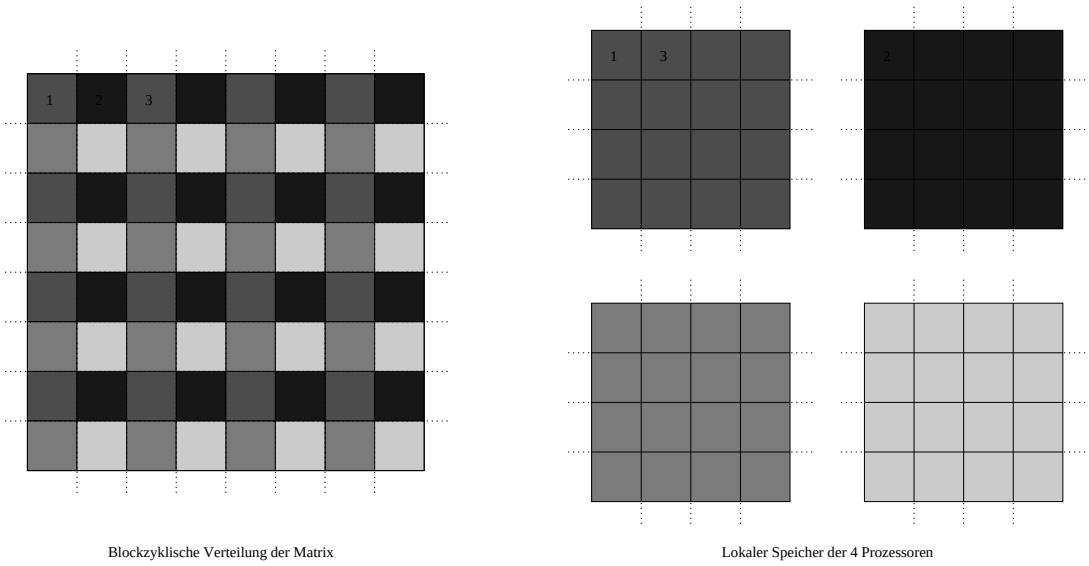


Abbildung 10: Zusammenhang zwischen lokalem Speicher eines Prozessors und blockzyklischer Vertielung.

3. Zurückkopieren der Ergebnisse mit der Routine `matrix_unblock`; ScaLAPACK-Kopien von Matrizen, die nicht verändert worden sind, können natürlich einfach gelöscht werden.

In dieser Weise können beliebige ScaLAPACK- oder PBLAS-Routinen mit den hier vorgestellten Hilfsmitteln durchgeführt werden. Für die Tests wurden in dieser Weise die Ähnlichkeitstransformation ($U^T AU$) und die Lösung des Eigenwertproblems implementiert.

Ergebnisse

Die implementierten Routinen wurden einigen Tests unterzogen. In den Tabellen 2 und 3 finden sich die Ausführungszeiten für eine kleinere und eine größere Matrix auf dem IBM p690 Cluster JUMP. Die dazugehörigen Speedup- und Effizienzkurven sind in de Abbildungen 11, 12, 13 und 14 zu finden. Wie man sieht, sind die erreichten Speedups für die kleinere Matrix besser. Bei der großen Matrix wäre es besser, die großen Blöcke parallel mit allen Prozessoren zu bearbeiten. Wenn mehr Prozessoren die größere Matrix bearbeiten, sollten auch hier höhere Speedups erreichbar sein. Messungen mit einer 4000×4000-Matrix mit 16 Prozessoren haben zum Beispiel gezeigt, dass die hier implementierten Routinen schneller sind, als die serielle Abarbeitung der Einzelblöcke parallel mit allen Prozessoren. Die beiden Ansätze sind jedoch nicht genauer verglichen worden.

Zusätzlich wurde die Lastverteilung auf dem ZAMpano Linux-SMP-Cluster gemessen. In Abbildung 15 ist das Verhältniss von Anlegen der geblockten Kopie, Matrixmultiplikationen bei der Ähnlichkeitstransformation und Zurückschreiben der Ergebnisse bei einer 4000×4000-Matrix mit 7 Blöcken zu sehen. Man erkennt, dass das Kopieren und im Vergleich zur eigentlichen Arbeit nicht ins Gewicht fällt, die Lastverteilung auf die Einzelprozessoren ist jedoch nicht optimal, da nur drei der acht Knoten optimal ausgelastet sind.

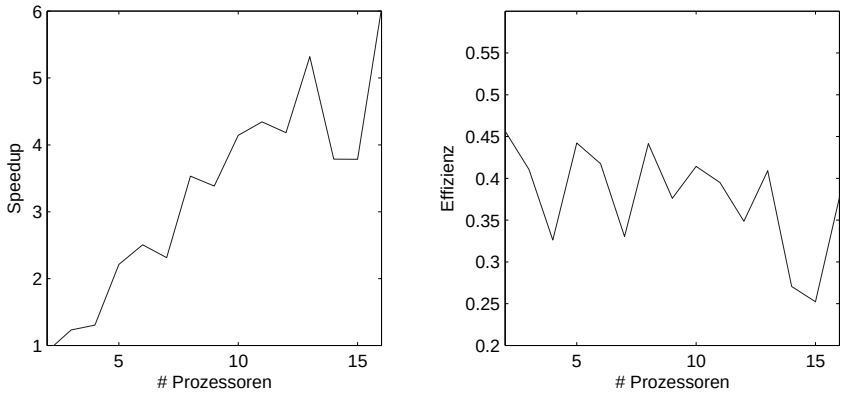


Abbildung 11: Speedup und Effizienz der Ähnlichkeitstransformation der 8000×8000-Matrix mit 7 Blöcken (vgl. Tabelle 2).

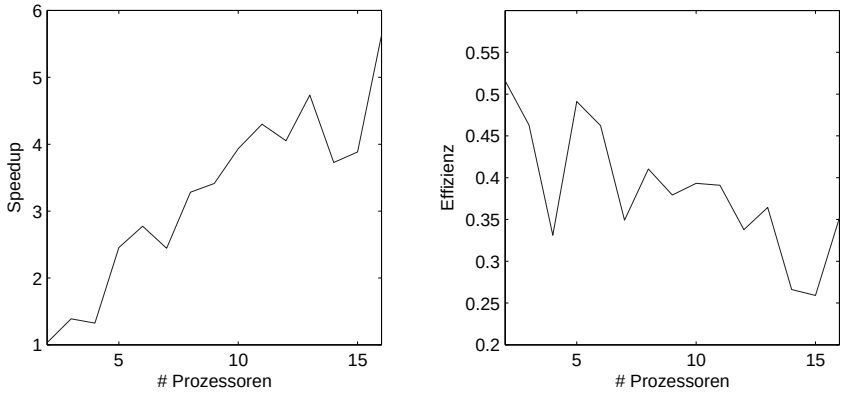


Abbildung 12: Speedup und Effizienz der Lösung des Eigenwertproblems der 8000×8000-Matrix mit 7 Blöcken (vgl. Tabelle 2).

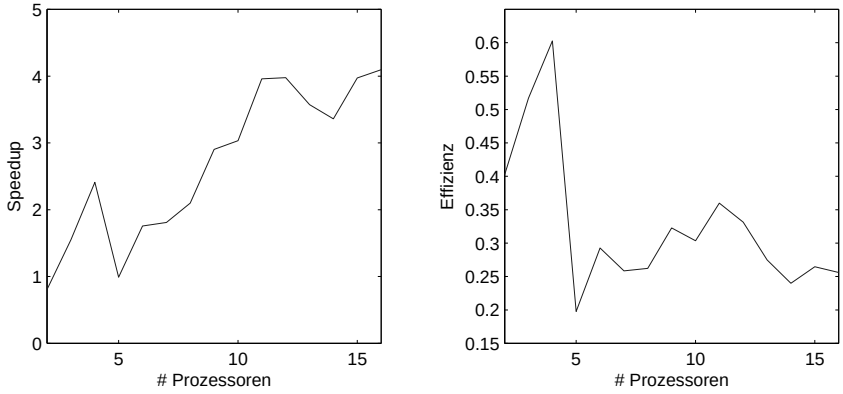


Abbildung 13: Speedup und Effizienz der Ähnlichkeitstransformation der 12000×12000-Matrix mit 5 Blöcken (vgl. Tabelle 3).

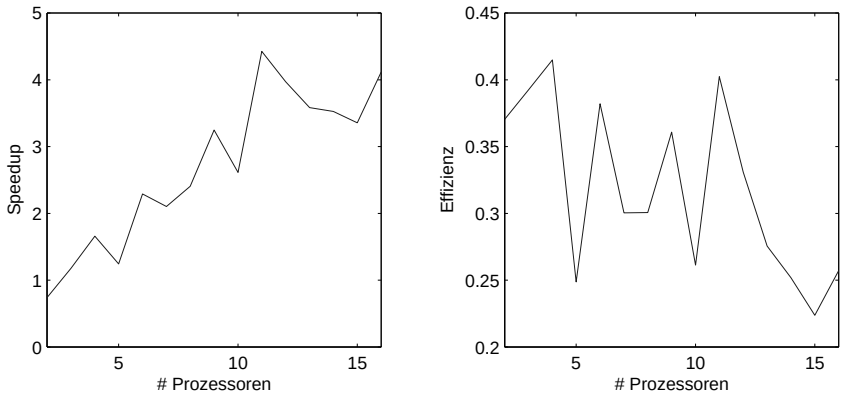


Abbildung 14: Speedup und Effizienz der Lösung des Eigenwertproblems der 12000×12000-Matrix mit 5 Blöcken (vgl. Tabelle 3).

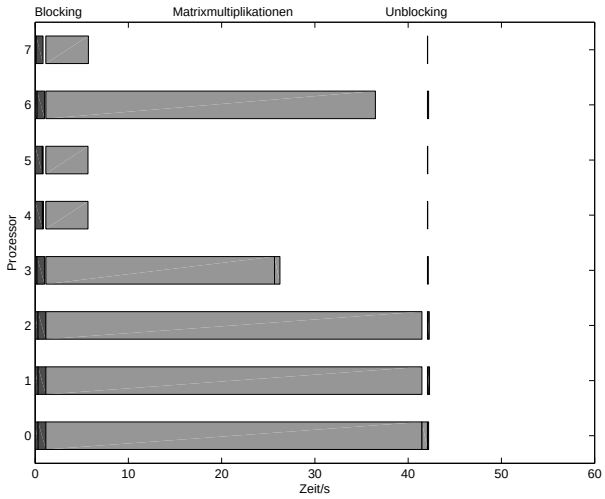


Abbildung 15: Verteilung der Arbeit auf dem ZAMpano Linux-SMP-Cluster bei Ähnlichkeitstransformation einer 4000×4000 Matrix mit 7 Blöcken, Blockgrößen 200, 1200, 700, 200, 500, 800 und 400.

# Prozessoren	Ähnlichkeitstransformation	Eigenproblem
2	24.0781	47.3761
3	17.8414	35.2001
4	16.8485	36.9100
5	9.9367	19.9040
6	8.7735	17.6247
7	9.5030	19.9967
8	6.2221	14.8909
9	6.4946	14.3190
10	5.3047	12.4288
11	5.0590	11.3656
12	5.2527	12.0607
13	4.1316	10.3214
14	5.8024	13.1165
15	5.8063	12.5817
16	3.6420	8.6778

Tabelle 2: Ausführungszeiten für Ähnlichkeitstransformation und Lösung des Eigenwertproblems auf dem IBM p690 Cluster JUMP. Die Matrizen haben die Größe 8000×8000 mit 7 Blöcken der Größen 400, 2400, 1400, 400, 1000, 1600 und 800.

# Prozessoren	Ähnlichkeitstransformation	Eigenproblem
2	123.0226	233.6711
3	63.8953	147.0640
3	41.1102	104.3285
4	100.2094	139.2100
5	56.4433	75.5372
6	54.7514	82.3323
7	47.2319	71.9792
8	34.1246	53.3105
9	32.6561	66.2625
10	25.0324	39.1122
11	24.9195	43.6040
12	27.7323	48.3246
13	29.4962	49.0926
14	24.9525	51.5826
15	24.1927	42.0717

Tabelle 3: Ausführungszeiten für Ähnlichkeitstransformation und Lösung des Eigenwertproblems auf dem IBM p690 Cluster JUMP. Die Matrizen haben die Größe 12000×12000 mit 5 Blöcken der Größen 2400, 1300, 4000, 2500 und 1800.

Fazit

Mit den hier vorgestellten Routinen ist es möglich blockdiagonale Matrizen parallel zu verarbeiten. Die Routinen sind hierbei so konzipiert, dass sie da noch einen Geschwindigkeitsvorteil bringen, wo die serielle Abarbeitung der Einzelblöcke mit allen Prozessoren keine gute Effizienz mehr hat. Es sind viele Erweiterungen der hier vorgestellten Routinen denkbar, z.B. eine zusätzliche Gewichtung der Blöcke bei der Verteilung – bisher werden sie einfach nur linear gewichtet – oder der Einbau der seriellen Abarbeitung der Blöcke, falls die Blöcke eine gewisse Grösse, die noch zu bestimmen ist, in Abhängigkeit von der Prozessorzahl haben.

Danksagung

Zu guter Letzt möchte ich mich noch bei Herrn Dr. Thomas Müller von Zentralinstitut für Angewandte Mathematik des Forschungszentrums Jülich bedanken. Ohne seine Betreuung wäre die Arbeit im Rahmen des Gaststudentenprogrammes sicher nur halb so erfolgreich gewesen und ich hätte keinen Einblick in die Methoden der theoretischen Chemie bekommen.

Literatur

1. W. Kutzelnigg : *Einführung in die theoretische Chemie, Band 1: Quantenmechanische Grundlagen.* VCH, Weinheim, 1992.
2. W. Kutzelnigg : *Einführung in die theoretische Chemie, Band 2: Die chemische Bindung.* VCH, Weinheim, 1994.
3. C. A. Coulson : *Die chemische Bindung.* S. Hirzel Verlag, Stuttgart, 1969.
4. R. Ahlrichs, S. D. Elliott, U. Huniar : *Ab Initio Treatment of Large Molecules.* In: J. Grotendorst (Editor) : Modern Methods and Algorithms of Quantum Chemistry. NIC, Jülich, 2000.
5. G. Golub, J.M. Ortega : *Scientifi c Computing: An Introduction to Parallel Computing.* Academic Press, New York, 1993.
6. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. C. Whaley : *ScaLAPACK User's Guid.* SIAM, Philadelphia, 1997.

Interaktive Visualisierung großer Moleküle

Björn Hagemeier

Universität Paderborn

bjoernh@uni-paderborn.de

Einleitung

Eine interaktive Visualisierung von Molekülen mit mehreren Tausend Atomen ist mit herkömmlichen Methoden nicht mehr möglich. Wird z. B. die so genannte Ball-and-Stick Technik verwendet, in der Atome als Kugeln und Bindungen als Zylinder gezeichnet werden, so liegen die zu erzielenden Frame-Raten oft unter einem Frame pro Sekunde. Dies liegt daran, dass für die Visualisierung jede Kugel und jeder Zylinder aufwändig durch viele Dreiecke repräsentiert werden müssen. Aus diesem Grund sollen Methoden angewandt werden, mit denen Moleküle mit einer geringeren geometrischen Komplexität dargestellt werden können, ohne die visuelle Qualität der Graphik gravierend zu verschlechtern. Im Rahmen dieser Arbeit wurden existierende Verfahren untersucht und eines exemplarisch implementiert.

Visualisierung von Molekülen

Es gibt verschiedene Arten und Weisen, Moleküle zu visualisieren. RasMol, ein beliebtes Tool für die Visualisierung, unterstützt acht verschiedene Darstellungen: Wireframe, Backbone, Sticks, Spacefill, Ball and Stick, Ribbons, Strands und Cartoons. Während einige dieser Modi die Struktur eines Moleküls auf einer relativ abstrakten Ebene wiedergeben, wird in anderen jedes einzelne Atom in der dem Modus entsprechenden Form dargestellt. Erstere verringern durch die Art der Darstellung die Komplexität der Szene und können so schneller gerendert werden. Mindestens werden durch die Abstraktion weniger Objekte gerendert. Diejenigen Modi, die jedoch alle Atome und Bindungen einzeln darstellen, bringen einen erheblichen Mehraufwand allein auf Grund der großen Anzahl von Atomen mit sich. Sollen dann noch Atome als Kugeln und Bindungen als Zylinder dargestellt werden, kann das Rendern der Szene sehr aufwändig werden.

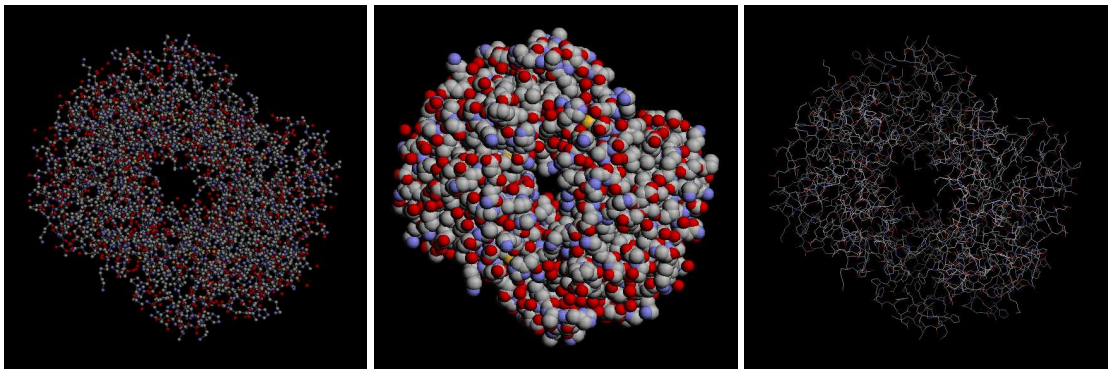


Abbildung 1: Verschiedene Darstellungen eines Moleküls. Von Links nach Rechts: Ball and Stick, Spacefill, Bonds.

Probleme

Im Folgenden sollen die Probleme, die beim Rendern von großen Molekülen auftreten, beleuchtet werden. Anschließend werden Lösungsansätze präsentiert. In erster Linie liegt die Schwierigkeit bei der Darstellung großer Moleküle in der Anzahl der beteiligten Atome. Zumindest wenn eine vollständige Darstellung der Moleküle gewünscht ist kann an der Anzahl nichts geändert werden. Daher muss man sich überlegen, wie man die Darstellung eines einzelnen Atoms möglichst effizient realisieren kann.

Wie zuvor erwähnt ist das Rendern von Kugeln im Vergleich zu anderen Objekten aufwändig. Um eine gute Annäherung an die tatsächliche Kugeloberfläche zu bekommen, muss die Oberfläche einer Kugel in viele Dreiecke eingeteilt werden. Die Darstellung der Kugeln mit Hilfe einer Tesselierung (Einteilung der Oberfläche in Dreiecke) ist von einer guten Schattierung abhängig, da dadurch die vorhandenen Kanten kaschiert werden und die Kugel in gewisser Weise glatt wirkt. Für dieses Shading ist die Berechnung eines Normalenvektors für die Eckpunkte eines jeden Dreiecks notwendig. Mit dessen Hilfe wird die endgültige Farbe für den Eckpunkt berechnet und in der Fläche dann die Farben von einem Eckpunkt in Richtung des nächsten interpoliert. Dadurch sieht die Kugeloberfläche relativ glatt aus, obwohl sie aus planaren Dreiecken besteht. Man kann sich vorstellen, dass die Interpolation der Farbwerte zwischen den Eckpunkten teuer ist. Für Zylinder und andere Körper mit “runden” Oberflächen gilt diese Aussage genau so.

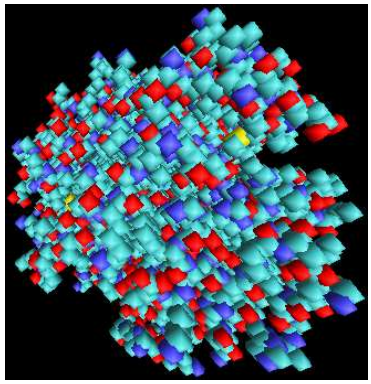


Abbildung 2: Die parallele Ausrichtung der mit wenigen Dreiecken gerenderten Kugeln deutet eine in der realen Welt nicht vorhandene Struktur an.

Die gesamte Geometrie einer Szene kann auf verschiedene Art und Weise vereinfacht werden. Zum einen wäre es möglich, die Kugeln durch weniger Dreiecke darzustellen. Diese wirken dadurch aber trotz Shading “kantig” und kleiner als echte Kugeln. Im Extremfall sehen diese Kugeln Würfeln ähnlich. Darüber hinaus ginge in der Spacefill-Darstellung die Information über die Überlappung der Kugeln verloren. Die parallele Ausrichtung der wenigen Kanten im Raum deutet zudem eine Struktur in den visualisierten Molekülen an, die überhaupt nicht vorhanden ist (Abbildung 2).

Die Anzahl der dargestellten Objekte kann nur verringert werden, wenn die Information über einzelne Objekte nicht relevant ist. Dadurch werden Abstraktionen oder Auslassungen möglich. Auslassungen könnten z. B. interaktiv ausgewählte Atome oder ganze Teilbereiche des Moleküls sein, die für die aktuelle Betrachtung nicht relevant sind. Diese würden in einem solchen Fall aus der Szene entfernt. Atome, die zu einer Abstraktion gehören sollen, könnten ebenso interaktiv ausgewählt und so ihre Abstraktion erzeugt werden. Ebenso wäre es möglich, Abstraktionen automatisch oder halb-automatisch zu erzeugen. [2] schlägt vor, jeweils benachbarte Kugeln zu einer zusammen zu fassen. Das Verfahren kann in mehreren Schritten rekursiv angewendet werden und erhält die Struktur des Moleküls sehr gut. Der Benutzer kann dann interaktiv steuern, bis zu welcher Ebene dieses Verfahren angewendet werden soll.

Visualisierung in RasMol

RasMol bietet auch für sehr große Moleküle eine sehr schnelle Visualisierung. Das funktioniert aber nur auf Grund einiger Vereinfachungen, die für VR-Systeme nicht verwendbar sind. Die Darstellung in RasMol basiert auf einer Orthogonalprojektion. Dadurch geht ein gewisses Maß an Tiefeninformation verloren. In einer Stereo-Darstellung würden die weiter entfernten Objekte größer als die näheren erscheinen. Weiterhin benutzt RasMol eine Integer-Tabelle, in der für Kugelradien von 0 bis 255 die Rasterpunkte entlang der Kugelachsen vorberechnet sind. Diese Methode bringt zwar eine enorme Beschleunigung, da teure Operationen nur einmal zur Initialisierung der Tabelle ausgeführt werden müssen. Diese Methode ist aber nicht mehr besonders Flexibel. RasMol führt die Berechnung der gesamten Szene im Hauptspeicher durch. Auf diese Weise kommt RasMol auch ohne Unterstützung von OpenGL aus. Das Protein Groel (PDB-Id: 1AON; 58688 Atome) wird mit akzeptabler Frame-Rate in der Spacefill- und Ball-and-Stick-Darstellung visualisiert.

Billboarding

Eine weitere Methode, um den Aufwand pro dargestelltem Atom zu verringern, ist die Verwendung von Billboards. Die Technik des Billboarding wird deshalb so genannt, weil sie dem Aufkleben eines Plakates auf eine Plakatwand (auf Englisch billboard) sehr ähnlich ist. Die Plakatwand ist in diesem Fall ein Rechteck, das entweder dem Benutzer zugewandt ist oder zumindest parallel zur Bildfläche ausgerichtet ist. Auf dieses Rechteck wird dann eine Textur gerendert.

Billboards sind Rechtecke, deren Flächen zum Benutzer oder zum Bildschirm ausgerichtet sind. Im ersten Fall zeigt der Normalenvektor der Fläche zum Projektionszentrum, im zweiten Fall ist er parallel zur Blickrichtung. Auf diese Flächen werden Texturen gerendert, so dass der Eindruck eines 3-dimensionalen

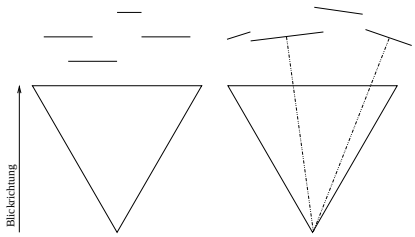


Abbildung 3: Am Bildschirm und am Betrachter ausgerichtete Billboards

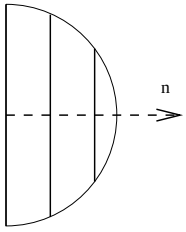


Abbildung 4: Darstellung einer Kugel durch hintereinander liegende Billboards verbessert den 3D-Effekt

Objektes entsteht. Ein großer Vorteil dieser Technik ist, dass eine sehr einfache Geometrie benutzt werden kann, um den 3D-Effekt zu erzeugen. Eine planare Fläche lässt sich deutlich schneller rendern als z. B. eine Kugel.

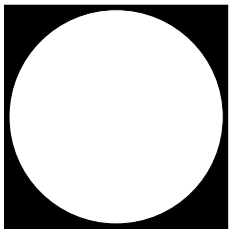


Abbildung 5: Bereiche der Textur, in denen das Objekt, hier ein Kreis, nicht erscheint, werden transparent gemacht. Im gezeigten Fall wäre der schwarze Bereich transparent.

```
int id;
id = glGenLists(1);
glNewList(id, GL_COMPILE);
...
glEndList();
...
glCallList(id);
```

Abbildung 6: Erzeugung und nachfolgender Aufruf einer Display-Liste in OpenGL

Das Billboard wird in den meisten Fällen größer sein, als das darzustellende Objekt. Die Bereiche der Textur, in denen das Objekt nicht zu sehen ist, werden daher transparent geschaltet (Abbildung 5).

Es gibt aber auch Nachteile bei der Verwendung von Billboards. Wenn man von einem am Bildschirm ausgerichteten Billboard und einer Zentralprojektion ausgeht, wirken alle Billboards, die am Rand des Bildes liegen, flach, da man schräg auf die Fläche blickt. Objekte, die durch zum Benutzer ausgerichtete Billboards dargestellt werden, scheinen aus jeder Blickrichtung gleich auszusehen. Für Kugeln, um die es bei der Visualisierung von Molekülen meistens geht, stellt dies wg. der Symmetrie kein Problem dar. Ein anderes Problem ergibt sich für Kugeln aber insbesondere bei der Spacefill-Darstellung von Molekülen. Für den Betrachter ist es durchaus interessant, wie die Kugeln sich überschlappen. Da die Kugeln beim Billboarding aber nur durch planare Flächen dargestellt werden, geht diese Information verloren. Eine Fläche liegt immer vor einer anderen. Das letztgenannte Problem lässt sich durch einen Trick lösen. Man zeichnet pro Kugel mehrere Billboards hintereinander. Durch die Verzahnung der einzelnen Flächen sich überschneidender Kugeln kann die Verschränkung der Kugeln näherungsweise dargestellt werden (Abbildung 4).

SVT

Das Scientific Visualization Toolkit (SVT) ist ein portables Toolkit für wissenschaftliche Visualisierung, das am Forschungszentrum Jülich entwickelt wird. Als C++-Klassenbibliothek ist es leicht erweiterbar und kann schnell neuen Bedürfnissen angepasst werden. Es bietet Unterstützung für Stereoansichten, wie sie in VR-Umgebungen benötigt werden.

Szenen-Graphen

SVT speichert alle Objekte in einem sogenannten Szenen-Graphen ab. Das ist eine Baumstruktur, die dem Programmierer viele Aufgaben vereinfacht. So ist es z. B. möglich, eine Transformation auf einen tieferliegenden Teilbaum anzuwenden, so dass ganze Gruppen von Objekten mit nur einer Transformation beeinflusst werden können. Diese können wieder andere Operationen auf anderen Teilbäumen enthalten. Abhängigkeiten zwischen Objekten können so sehr einfach abgebildet werden.

Display-Listen

Display-Listen sind Gruppen von OpenGL-Befehlen, die unter einer Id zusammengefasst und als Gesamtheit ausgeführt werden können. Eine Display-Liste lässt sich erzeugen, indem man eine freie Id von OpenGL anfordert und mit dieser eine neue Liste anlegt. Abbildung 6 zeigt, wie der Quellcode dazu in C/C++ aussieht. Es gibt einige OpenGL-Befehle, die nicht in eine Display-Liste aufgenommen werden können. Diese werden beim Anlegen der Liste direkt ausgeführt. Die genaue Spezifikation lese man in [1] nach.

Interessant zu wissen ist es, dass man den Aufruf einer Display-Liste, also den Befehl `glCallList(id)` in eine Display-Liste aufnehmen kann. Diese zweite Liste muss beim Zusammenstellen der Befehle noch nicht existieren und kann auch unabhängig von der ersten verändert werden. In SVT wird eine solche Verschachtelung von Display-Listen verwendet. Eine "äußere" Liste ist für die Transformationen, also die Positionierung und Skalierung der Atome, zuständig und ruft dann eine "innere" Display-Liste auf, die ein einzelnes Atom darstellt. Die Eigenschaft, dass die "innere" Display-Liste zwischen zwei Aufrufen der äußeren Liste verändert werden kann, wird in der später beschriebenen Implementierung der Billboards benutzt.

Implementierung

Die zuvor beschriebene Technik des Billboarding wurde in SVT implementiert. Im Folgenden wird diese Implementierung beschrieben.

Texturierte, planare Flächen stellen in OpenGL keine Herausforderung dar. Mit wenigen Zeilen Code lässt sich ein Quadrat der Kantenlänge 1 mit einer Textur erzeugen. Etwas schwieriger gestaltet sich die Ausrichtung der Billboards. Aus Gründen der Effizienz wurden bei der Implementierung am Bildschirm ausgerichtete Billboards benutzt. Dadurch entfällt die Berechnung der Ausrichtung für jede einzelne Kugel, was wieder einigen Rechenaufwand pro Frame mit sich bringen würde.

Es wurde zuvor gesagt, dass SVT mit verschachtelten Display-Listen arbeitet, um die Objektdarstellung möglichst Effizient zu gestalten. Diesen Ansatz kann man, mit einer kleinen Änderung, auch für Billboards übernehmen. In der normalen Darstellung der Atome als Kugeln gibt es eine Display-Liste K, die eine Kugel darstellt, und eine Display-Liste T, die Transformationen für die einzelnen Atome durchführt und dann K aufruft. K kann also durch eine beliebige Display-Liste ersetzt werden, die die Darstellung eines Atoms ist. Insbesondere kann hier auch eine Display-Liste aufgerufen werden, die ein Atom als Billboard darstellt. Die einzige Änderung zur vorherigen Situation ist, dass die Display-Liste für das Billboard pro Frame ein Update erfahren muss, damit die Ausrichtung des Billboards noch stimmt.

Ausrichtung der Billboards

$$\begin{bmatrix} r_1 & u_1 & m_{13} & m_{14} \\ r_2 & u_2 & m_{23} & m_{24} \\ r_3 & u_3 & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

Abbildung 7: Up- und Right-Vektor lassen sich leicht aus der Modelview-Matrix extrahieren

Für am Bildschirm ausgerichtete Billboards kann man die Eckpunkte der Rechtecke einfach aus der Modelview-Matrix berechnen. Für ein Quadrat der Kantenlänge 1, das zudem noch in der xy-Ebene liegt, kann man den Up- und den Right-Vektor benutzen, die beide im Rotations-Teil der Modelview-Matrix stehen [3]. Die Vektoren sind orthogonal zu einander. Abbildung 7 zeigt, wo die Vektoren extrahiert werden können. Seien der Up-Vektor $u = (u_1, u_2, u_3)$ und der Right-Vektor $r = (r_1, r_2, r_3)$ normiert. Dann berechnen sich die Positionen der Eckpunkte p_i des Rechtecks um dem Ursprung wie folgt:

$$\begin{aligned} p_1 &= -u - r \\ p_2 &= -u + r \\ p_3 &= u + r \\ p_4 &= u - r \end{aligned}$$

Die p_i liegen in Bildkoordinaten alle in der xy-Ebene. Um je eine Kugel durch mehrere Kugelscheiben darzustellen, müssen die Scheiben entlang der z-Achse in Richtung des Betrachters positioniert werden. Dazu berechnet man den Vektor n, der entlang der z-Achse zeigt als das Kreuzprodukt aus u und r.

$$n = r \times u$$

Sei s die Anzahl der Kugelscheiben, so ergeben sich die Positionen der Eckpunkte zu

$$p_{ij} = p_i + n * \frac{j}{s} \text{ für } j = 0, .., s-1$$

Die Display-Liste zur Darstellung einer Kugel kann in einer Schleife erzeugt werden, in der die s Kugelscheiben als Textur auf die Rechtecke gerendert werden. Man sieht, dass die Rechtecke alle die gleiche Größe haben. Kreisscheiben, die näher am Benutzer sind, müssen aber kleiner sein als die entfernteren. Das wird dadurch erreicht, dass verschiedene Texturen verwendet werden. Für Kugeln, die mit s slices dargestellt werden, gibt es auch s Texturen. Effektiv wird durch diese Technik nur eine Halbkugel angenähert. Da diese aber immer am Bildschirm ausgerichtet wird, fällt das nicht auf.

Erstellen der Texturen

Wie oben erwähnt, erhält jedes slice einer Kugel eine eigene Textur. Da die Anzahl der Texturen vor Ausführung des Programms nicht bekannt ist, werden die Texturen zur Laufzeit erzeugt. Dabei treten Schwierigkeiten auf, die im Folgenden erläutert werden.

Grundsätzlich ist es in OpenGL möglich, Daten aus dem Framebuffer direkt in eine Textur zu übernehmen. Der Befehl `glCopyTexImage2D(...)` kopiert Daten aus dem aktuellen Read-Buffer in eine Textur. Man kann also mit den üblichen OpenGL-Befehlen das Bild einer Kugel erzeugen und dieses dann in eine Textur kopieren. Es stellt sich aber heraus, dass trotz entsprechender Parametrisierung der Funktion die Alpha-Werte aus dem aktuellen Read-Buffer nicht mit in die Textur übernommen werden. Diese werden benötigt, weil die Randbereiche der Textur transparent sein sollen und diese Transparenz mit einem Alpha-Filter realisiert wird. Die erzeugte Textur bedarf also einer Korrektur der Alpha-Werte. Aus diesem und einem weiteren leicht einzusehenden Grund wird die Textur noch einmal mittels `glGetTexImage(...)` aus dem Texturspeicher in ein Array gelesen und dort manipuliert. Dies ist die Textur, die später auf das Slice 0 gerendert wird. In dieser müssen lediglich die schwarzen Randbereiche, in denen die Kugel nicht zu sehen ist, einen Alpha-Wert von 0 erhalten. Ist dies geschehen, wir die Textur mittels `glTexImage2D(...)` wieder in den Texturspeicher kopiert. Die Texturen für die übrigen Slices werden erzeugt, indem in der bestehenden Textur immer größere Randbereiche "abgeschnitten" werden, d. h. deren Alpha-Wert auf 0 gesetzt wird.

Sei $r = cos(i * \pi/(s * 2)) * TEX\_SIZE/2.0$ der Radius des Slice i in Pixel , wobei s die Anzahl der Slices und TEX_SIZE die Kantenlänge der Textur in Pixel ist. Dann bekommen genau die Pixel der Textur einen Alpha-Wert von 0, für die der Abstand vom Mittelpunk der Textur größer als der Radius r ist. Zu SVT wurde eine neue Klasse `svt_basic_billboard_sphere` hinzugefügt, die von `svt_basic_object` abgeleitet ist und deshalb in einem `svt_basic_container` gehalten werden kann. Dies wurde so realisiert, da in `svt_pdb` alle Atome, Bindungen etc. durch Objekte der Klasse `svt_basic_object` dargestellt werden. Auf diese Weise gab es an `svt_pdb` nicht viel zu ändern und die wenigen Änderungen passten sauber in das bestehende Konzept.

Änderungen

In `svt_pdb` wurde der Visualisierungsmodus `SVT_PDB_VDW_BILLBOARD` hinzugefügt. Die Aktivierung dieses Modus veranlasst die Speicherung von Objekten der Klasse `svt_basic_billboard`

_sphere in einem svt_basic_container. Gleichzeitig wird veranlasst, dass nicht mehr das gesamte svt_pdb-Objekt mit Display-Listen dargestellt wird. Bei der Benutzung von Billboards ist es notwendig, dass die Billboards in jedem Frame neu am Bildschirm ausgerichtet werden. Diese Ausrichtung geschieht durch einen Aufruf der statischen Methode svt_basic_billboard_sphere::updateSphereDL(...) aus svt_pdb::drawGL() heraus. Dieser kann nur erfolgen, wenn die automatische Benutzung von Display-Listen deaktiviert ist. Jede Klasse, die svt_basic_billboard_sphere verwendet, hat also dafür Sorge zu tragen, dass das Update der Billboards an entsprechender Stelle geschieht.

Versuchsweise wurden auch zum Benutzer ausgerichtete Billboards implementiert. Auf Grund der vielen Berechnungen, die bei dieser Art von Billboards notwendig sind, konnte die Performance nicht überzeugen. Für diese Art von Billboards macht die Verwendung von Display-Listen auch wenig Sinn, da für jeden Frame die Ausrichtung jedes einzelnen Billboards neu berechnet werden muss. Die Klasse svt_atom wurde um den Modus SVT_ATOM_VDW_BILLBOARD erweitert, der ein Atom als Billboard visualisiert.

Ergebnis

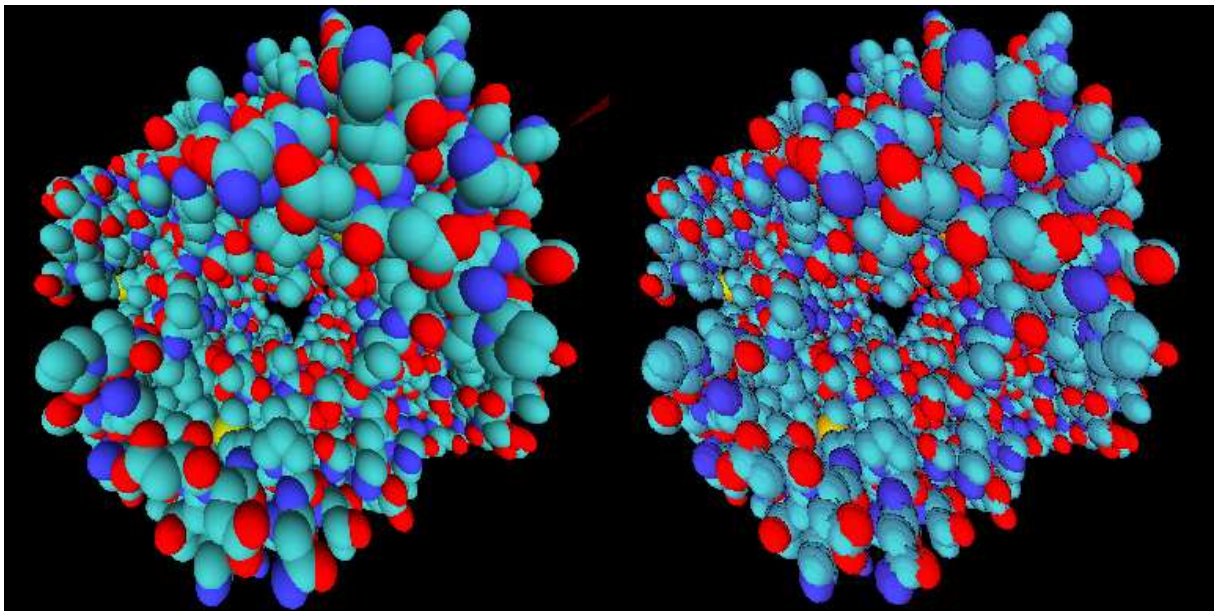


Abbildung 8: Echte Kugeln (links) im Vergleich mit den Billboard-Kugeln

Unter Verwendung der Billboards konnte, je nach Anzahl der verwendeten Slices, eine Beschleunigung der Visualisierung um den Faktor 3-4 festgestellt werden. Je mehr Slices für die Visualisierung verwendet werden, desto geringer ist der Speedup. Bei 6 Slices pro Kugel war die Frame-Rate augenscheinlich ungefähr so groß wie bei “echten” OpenGL-Kugeln mit 16 stacks und 16 slices. Abbildung 8 zeigt die beiden Darstellungen im Vergleich.

Ausblick

Vor der Implementierung der Billboard-Spheren wurde auch überlegt, Point-Sprites zu verwenden. Diese sind aber nicht im OpenGL-Standard festgeschrieben und werden nur von vereinzelten Herstellern als Erweiterung angeboten. Sie werden auch nicht von jeder OpenGL-Bibliothek unterstützt. Diese Idee

wurde daher wieder verworfen. Billboards, wie sie hier vorgestellt wurden, sind vollständig mit dem OpenGL-Standard kompatibel.

Eine weitere Überlegung war, die Moleküle ähnlich wie in RasMol zu rendern und dann in den Graphik-Speicher zu kopieren. Diese wurde am Ende nicht mehr realisiert, da die Billboards den Vorzug erhalten hatten. Diese Methode hätte einiger Anpassung für eine perspektivische Darstellung bedurft. Ob sie dann immer noch effizienter gewesen wäre als die echten OpenGL-Kugeln konnte nicht abgeschätzt werden.

In [4] wird das parallele Rendern von Kugeln behandelt. Im Vergleich zu einer SGI Onyx mit Reality-Engine2 Graphik konnte maximal eine Beschleunigung um den Faktor 90 festgestellt werden. Auf einer Cray T3D und einer CM-5 wurden 660000 bzw. 330000 Kugeln pro Sekunde gerendert. Mit der SGI Onyx konnten 19000 tesselierte Kugeln bzw. 87000 Bitmap-Kugeln pro Sekunde erreicht werden. Die Bitmap-Kugeln unterliegen aber einigen Einschränkungen, die für VR-Umgebungen nicht akzeptabel sind.

Auf Grund technischer Probleme konnte die Brauchbarkeit der implementierten Methode auf der Holobench des ZAM nicht verifiziert werden. Somit kann nur vermutet werden, dass die durch Billboards dargestellten Kugeln dem Betrachter einen relativ guten 3D-Eindruck geben.

Literatur

1. OpenGL specifi cation and manual pages. http://www.opengl.org/developers/documentation/specs.html.
2. Chandrajit L. Bajaj, Valerio Pascucci, Ariel Shamir, Robert J. Holt, and Arun N. Netravali. Multiresolution molecular shapes, 1999.
3. Kevin Hawkins and Dave Astle. *OpenGL Game Programming*. Prima Publishing, 2001.
4. Michael Krogh, James Painter, and Charles Hansen. Parallel sphere rendering. *Parallel Computing*, 23(7):961–974, 1997.

Equilibration of enclosed particle systems interacting with a finite-power-series potential using the N-body code PEPC

Mike Hammes

Bergische Universität Wuppertal
Fachbereich C (Physik)
Gaußstraße 20
42097 Wuppertal

E-mail: mike.hammes@gmx.net

Abstract:

This article describes two new features added to the N-body code Pretty Efficient Parallel Coulomb-solver (PEPC) [1]. The first addition allows modelling of many-particle-systems constrained within containers formed by planes, spheres, ellipsoids and (elliptic) cylinders. The second addition allows easy implementation of any finite-power-series potentials with integer exponents and real coefficients. This makes it possible to use the potential from LENNARD and JONES or a fit to the MORSE potential in place of the COULOMB potential. A number of simulations were made using these additions with the goal of finding electrostatic equilibrium configurations.

Introduction.

Recent research in plasma physics deals with the understanding of the effects short-pulse lasers can create on solid targets. This research is of very high interest since there have been observations of high energy protons emitted from the rear surface of pulsed-laser-radiated foil targets. Generating high energy protons with the help of a pulsed laser may lead to future construction of cheap, compact proton beam sources, which can be useful for many applications. Many numerical simulations have been made to help understand the acceleration of the protons (for recent accounts see [2, 3, 4]). But the constitutive assumptions which have been made for these simulations are not able to reconstruct *all* effects that have been observed in experiments (recent accounts of experiments can be found in [5, 6]). Most numerical simulations to date have used a rectangular foil as a target. But we would like to be able to use a variety of geometries within the plasma simulation, for which PEPC (Pretty Efficient Parallel Coulomb-solver) is more suited - for a short introduction see [1]. Until now, PEPC initially spreads particles randomly into a given geometry, but does not constrain them inside this geometry and does not use periodical boundary conditions (which is PEPC's main advantage over other plasma simulation codes). So, PEPC has to be upgraded by a constraint routine which can deal with various geometries. Another difficulty comes from the fact that randomly spread particles in restricted space are not in electrostatic equilibrium. But because the target has to be solid, we need start configurations in such equilibria. Trying to find electrostatic equilibria with the already implemented COULOMB potential seems to be unpromising, because such equilibria are unstable. This leads us to a second problem: We have to implement interactions which

allow stable static equilibrium configurations to be found. And because the force summation in PEPC is based on a multipole expansion, this should ideally be valid for these interactions.

According to the problems noted above, this document is divided into three parts. The first part describes the implementation of the constraint routine to restrict the available space for the particles. In the second part a multipole expansion for finite power series potentials is described. This is then used to implement the LENNARD-JONES potential and a fit to the MORSE potential. The last part describes how to find static equilibrium configurations dynamically with a modified LENNARD-JONES potential and shows some examples.

Restriction of the available space for the particles.

In most cases in simulating huge systems of particles, the use of periodic boundary conditions is customary, particularly if the theoretical description of the system assumes a large - even infinite - number of particles and the geometrical boundary conditions are not of interest. However, it is sometimes necessary to deal with systems of particles which are enclosed into limited and geometrically well defined containers, e. g. cuboids, spheres, cylinders, prisms or others. So an addition to PEPC which was investigated here allows a fluid of a finite number of particles to be constrained within such a volume.

The constrain routine.

The basic container model.

The basic container model taken, assumes totally elastic collisions between the particles and the walls, so that a particle which reaches the wall gets reflected at the tangent plane associated with the collision point. This assumption leads to three problems:

1. Find the point where the particle came into collision with a wall of the container.
2. Find the tangent plane associated with this collision point.
3. Reflect the particle at this tangent plane.

The problems are solved by defining the inside V of a container as follows:

$$V := \{\mathbf{r} \in \mathbf{R}^3 : f(\mathbf{r}) \leq 0\},$$

whereby $f : \mathbf{R}^3 \to \mathbf{R}$ is a parametrization for the surface of the container. The following parametrizations are implemented:

Planes:

$$f(\mathbf{r}) = \mathbf{n} \cdot (\mathbf{r}_0 + \mathbf{r}_s - \mathbf{r}) \tag{1}$$

with $\mathbf{n} \in \mathbf{R}^3$: normal vector, $\mathbf{r}_s \in \mathbf{R}^3$: position vector and $\mathbf{r}_0 \in \mathbf{R}^3$: translation of the origin.

Ellipsoids, spheres, cylinders:

$$f(\mathbf{r}) = (\mathbf{r} - \mathbf{r}_0) \cdot \begin{pmatrix} a^{-2} & 0 & 0 \\ 0 & b^{-2} & 0 \\ 0 & 0 & c^{-2} \end{pmatrix} \cdot (\mathbf{r} - \mathbf{r}_0)^\top - R^2 \tag{2}$$

with $\mathbf{r}_0 \in \mathbf{R}^3$: translation of the origin and $a, b, c, R \in \mathbf{R}_{>0}$.

The following table shows the conditions for the parameters a, b and c to realize different containers with parametrization (2).

Container	**Expectations on** $M := \{a, b, c\} \subset \mathbf{R}_{>0}$
sphere	$M = \{1\}$
cylinder	$M = \{1, \infty\}$ and $a^{-2} + b^{-2} + c^{-2} = 2$
ellipsoid	$\sharp M \geq 2$ and $\infty \notin M$
elliptic cylinder	$\sharp M = 3$ and $\infty \in M$

Table 1: Realizing different surfaces with the parametrization (2). $x = \infty$ has to be interpreted as $x \to \infty$ so that $x^{-2} \to 0$. The operator $\sharp$ counts the elements in a set.

The algorithm to calculate the tangent plane.

After a container is parametrized, we can start to constrain the particles to stay into it. We assume, that at time n all particles are inside V, and that at time $n + 1$ at least one particle with the parametrization $\mathbf{r} : \mathbf{N} \to \mathbf{R}^3; n \mapsto \mathbf{r}(n)$ of its discrete trajectory is outside V. The point where this particle came into collision with one of the walls, has to get found within the stretch

$$\{\mathbf{r} \in \mathbf{R}^3 : \exists u \in [0, 1] : \mathbf{r}(n) + u(\mathbf{r}(n + 1) - \mathbf{r}(n))\}.$$

Because inside of V, f is less or equal to zero and outside of V, f is greater then zero, we are looking for a root $\mathbf{r}_c$ with alternating sign. Such a root can be found very fast and intuitively by using a bisection algorithm [7]. If we use a bisection algorithm to find a root, we have to define a truncation condition such as

`if (`$|f(\mathbf{r}_{test})| \leq \varepsilon$`) then` $\mathbf{r}_c \leftarrow \mathbf{r}_{test}$`; exit bisection;`

Such a procedure causes an error $\Delta\mathbf{r}_c = \pm\varepsilon'(\mathbf{r}(n + 1) - \mathbf{r}(n))$ for the resulting root and it may happen that the $\mathbf{r}_c$ found does *not* end on the surface of the container. Because of this problem it is good to know exactly where the walls of the container are located, and what their normal vectors are at each point of their surface. If we know this, we can cut $\mathbf{r}_c$ down to the correct length.

If we have a plane wall parametrized via (1), we know exactly where it is located and what its normal vector is. The following considerations deal with calculating tangent planes on parametrizations like (2) next to $\mathbf{r}_c$. We first transform the ellipsoid to the simpler geometry of a ball. This can be done by fixing one of the axes (e. g. the axis in x-direction with length Ra, if $a \ll \infty$) and normalize the others to the same length. This is done by the matrix $T := \mathrm{Tr}(1, \frac{a}{b}, \frac{a}{c})$. Then the vector $T \cdot \mathbf{r}_c$ has to get cut to the radius Ra of this ball. After using T^{-1}, this procedure results in the position vector

$$\mathbf{r}_s = Ra\frac{\mathbf{r}_c}{\|T \cdot \mathbf{r}_c\|}$$

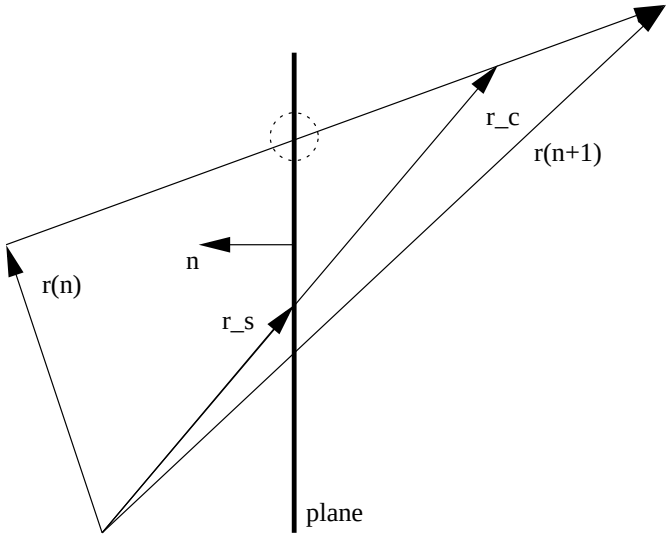


Figure 1: Construction of the position vector $\mathbf{r}_s$ of the tangent plane. Consider a course of one particle. At time n it is inside of the container and at time $n+1$ it is outside of the container (the normal vector $\mathbf{n}$ points towards the inside of the container). After a bisection algorithm along the vector $\mathbf{r}(n+1) - \mathbf{r}(n)$ we end at the vector $\mathbf{r}_c$. Its length gets corrected to $\mathbf{r}_s$ so that it ends within the surface of the container. Without numerical deviations coming from the bisection algorithm, $\mathbf{r}_s$ (and $\mathbf{r}_c$) should end at the centre of the drawn circle.

of the tangent plane. It has the correct length and its only remaining error coming from $\mathbf{r}_c$ is that of its direction. Now we obtain the normal vector $\mathbf{n}$ of the tangent plane. For this we use the fact that the tangent in $\mathbf{r}_s$ on the ellipsoid and this in $T \cdot \mathbf{r}_s$ on the ball intersect in one point of the fixed axis. This is clear because an ellipse is a 2D-projection of a rotated circle around one axis in 3D-space. And we only have to consider ellipses and circles. A reflection happens exactly in one plane.

Figure 2 illustrates that the angle γ between these two tangents is obtained by substracting φ form ϕ, where ϕ is given by

$$\tan\phi = \frac{x_s}{\sqrt{\mathbf{r}_s'^2 - x_s^2}}$$

and φ, noting that $x_s' = x_s$, is given by

$$\tan\varphi = \frac{x_s\sqrt{\mathbf{r}_s^2 - x_s^2}}{\mathbf{r}_s'^2 - x_s^2}.$$

These formulas are easily proved by using the intersection theorems together with PHYTHAGORAS' theorem and the tangent definition. The normal vector $\mathbf{n}$ gets finally obtained as

$$\mathbf{n} = -R_{\mathbf{r}_s\times\mathbf{r}_s'}(\gamma) \cdot \frac{\mathbf{r}_s'}{\|\mathbf{r}_s'\|},$$

where $R_{\mathbf{r}_s\times\mathbf{r}_s'}(\gamma)$ is a matrix which rotates a vector with the angle γ around the axis spanned by the vector $\mathbf{r}_s \times \mathbf{r}_s'$. The negative sign is a convention in this work. The normal vector should point towards the inside of the container.

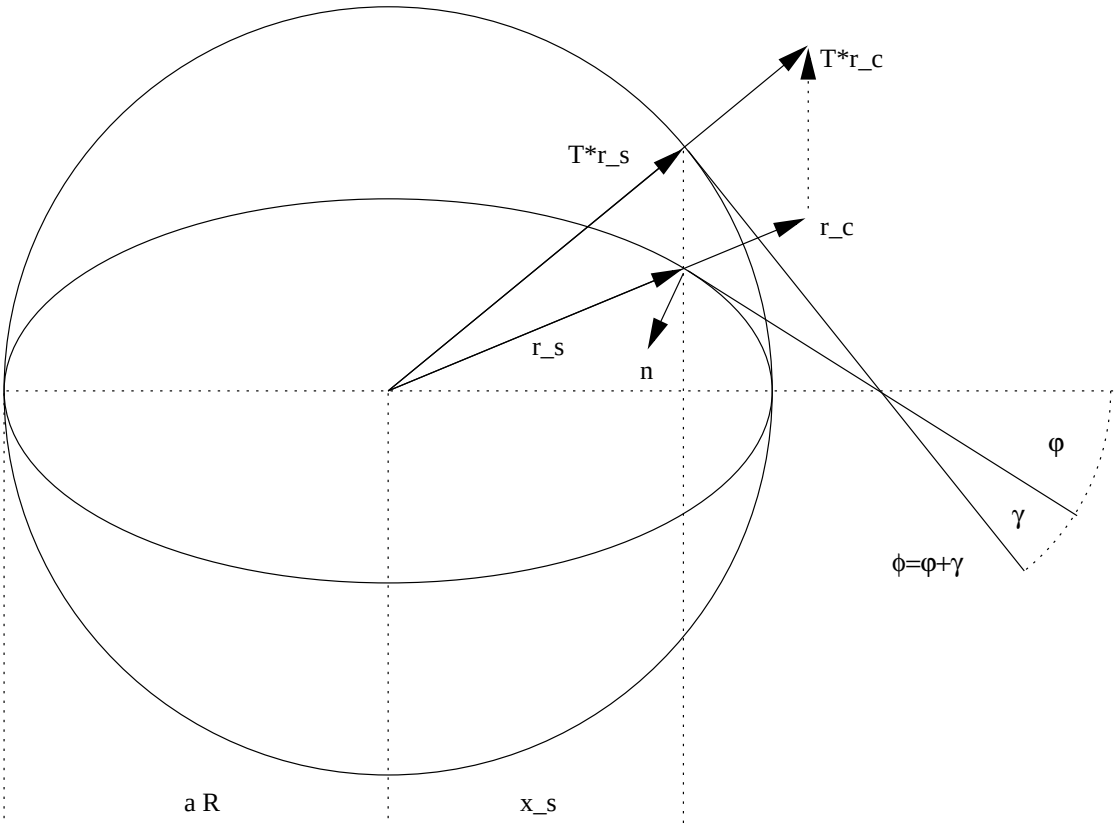


Figure 2: Construction of a normal vector $\mathbf{n}$ on a given point $\mathbf{r}_s$ which is ending within the surface of an ellipsoid. The first step is to transform the ellipsoid with the help of T to a ball so that $\mathbf{r}_c$ becomes to $T \cdot \mathbf{r}_c$. The next step is to cut this vector down to $T \cdot \mathbf{r}_s$. This vector has the length Ra. Back transformation with T^{-1} leads to the position vector $\mathbf{r}_s$. γ is the angle between the tangent plane on the ball in $T \cdot \mathbf{r}_s$ and this on the ellipsoid in $\mathbf{r}_s$. It is needed to calculate the normal vector $\mathbf{n}$.

Reflection of the particle on the tangent plane.

The tangent plane t is now described by

$$t(\mathbf{r}) = \mathbf{n} \cdot (\mathbf{r}_0 + \mathbf{r}_s - \mathbf{r})$$

for both, a plane and an ellipsoid. Now we can complete the reflection by the simple calculations

$$\mathbf{v}' = \mathbf{v} - 2(\mathbf{v} \cdot \mathbf{n})\mathbf{n},$$

$$\mathbf{r}'(n+1) = \mathbf{r}_s + \|\mathbf{r}(n+1) - \mathbf{r}_s\| \frac{\mathbf{v}'}{\|\mathbf{v}'\|},$$

where $\mathbf{v}$ means the velocity of the particle and primed vectors are corrected vectors after reflection (see also figure 3).

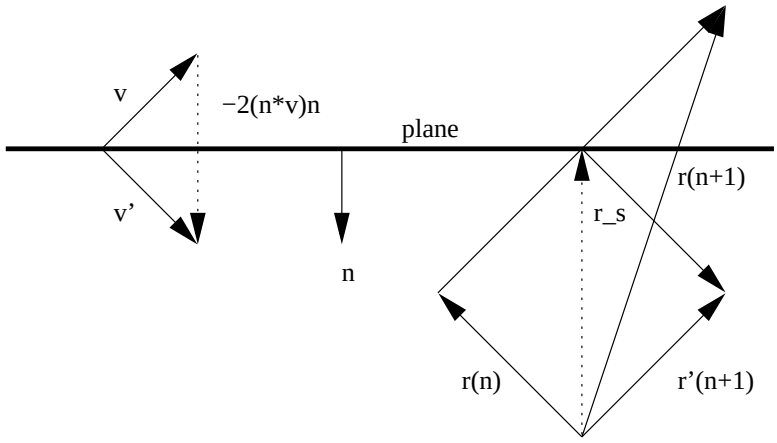


Figure 3: Reflection of a particle on a plane. The horizontal line represents the plane. The normal vector $\mathbf{n}$ points to the inside of the container. Primed vectors label vectors after reflection. On the left hand side the reflection of the velocity $\mathbf{v}$ is shown. Only the normal component has to be modified. On the right hand side the correction of the location $\mathbf{r}$ at time $n+1$ is shown. Here is assumed that the particle was inside the container at time n. Note that the stretch $\mathbf{r}_s - \mathbf{r}(n)$ has to be parallel to $\mathbf{v}$ (before reflection) and $\mathbf{r}'(n+1) - \mathbf{r}_s$ has to be parallel to $\mathbf{v}'$ (after reflection).

Remarks.

At first sight, in case of an ellipsoid the above procedure seems rather complicated, because to obtain the normal vector, only the JACOBIan is needed. On the other hand, if the JACOBIan is used, a large number of expensive trigonometrical calculations is needed to obtain the position vector $\mathbf{r}_s$ for the tangent plane. Also the number of simple algebraic calculations to get the normal vector is much larger in that way then in the demonstrated one, because the JACOBIan delivers just one tangent vector. From this, a further tangent vector must be calculated via a cross product and from these two tangent vectors, the normal vector is finally obtained, again via a cross product. All in all, the demonstrated way is much more efficient than that over the JACOBIan.

One should mention that in the code, the expensive procedure above is only used for real ellipsoids (see also table 1). For perfect balls or cylinders there is no need to rotate the normal vector or to transform the root vector. In these cases, the root vector has just to get cut to the correct length, and the normal vector is, with respect to a real factor, equal to the cut root vector (ball) or to the projection of this vector down to the ground plane (cylinder).

Further more, in case of a plane the root vector gets cut to the correct length, too. The procedure used here is based on the intersection theorems whereby the root vector $\mathbf{r}_c$ defines one of the intersecting straight lines, and the normal vector $\mathbf{n}$ spans the other one (see also figure 4).

The implemented geometries.

In total, there are nine different container geometries available in PEPC. Three of them were already implemented before the generalization of the constraint algorithm and have now been rewritten in terms of this algorithm, based on the normal vector $\mathbf{n}$ and the position vector $\mathbf{r}_s$ of the tangent plane. So now all container geometries are realized with the same constraint algorithm. The three translated geometries are the cuboid and two sorts of cylinders called "disc"and "wire", whereby the disc has cylindrical sym-

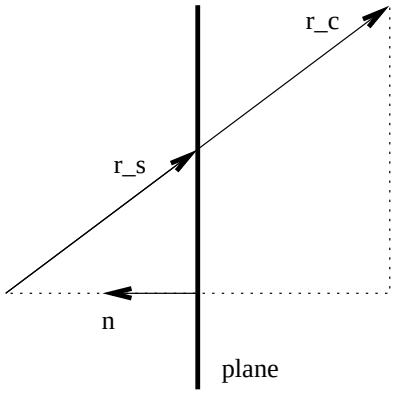


Figure 4: Cutting a root vector $\mathbf{r}_c$ on a plane. We get the equation $\frac{\|\mathbf{r}_s\|}{\|\mathbf{r}_c\|} = \frac{|\mathbf{r}_s \cdot \mathbf{n}|}{|\mathbf{r}_c \cdot \mathbf{n}|}$ from the intersection theorem. This means $\|\mathbf{r}_s\| = \frac{|\mathbf{r}_s \cdot \mathbf{n}|}{|\mathbf{r}_c \cdot \mathbf{n}|}\|\mathbf{r}_c\|$, so that $\mathbf{r}_s = \|\mathbf{r}_s\|\frac{\mathbf{r}_c}{\|\mathbf{r}_c\|} = \frac{|\mathbf{r}_s \cdot \mathbf{n}|}{|\mathbf{r}_c \cdot \mathbf{n}|}\mathbf{r}_c$.

metry around the x-axis and the wire has cylindrical symmetry around the z-axis. This distinction is for convenience when adding an external field or particle source. A sphere was also available but was not able to constrain the particles dynamically. So this was the first container implemented using the new scheme.

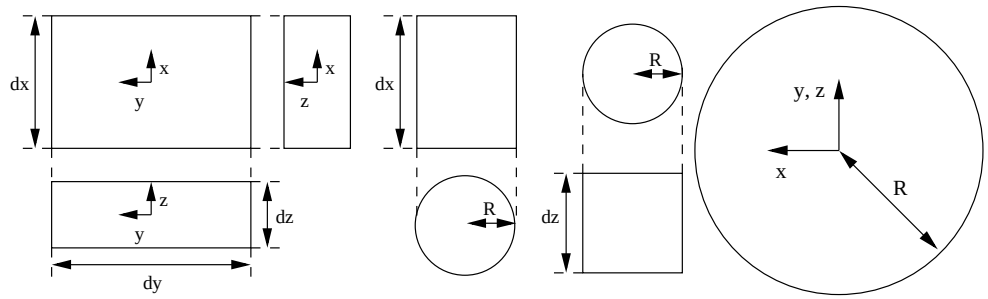


Figure 5: The already implemented containers (left) and the sphere (right) and their parameters. From the left to the right: The cuboid with the parameters dx, dy and dz. The disc with the parameter R for the radius and the length dx. The wire with the radius R and the height dz. The only parameter of the sphere is the radius R.

The next, more complicated container to be implemented was the ellipsoid. Further containers are built up from planes and spheres. Examples now available are hemispheres, hollow balls, hollow hemispheres and prisms. The following figures 6 to 8 on the next pages describe these containers and their parameters in detail.

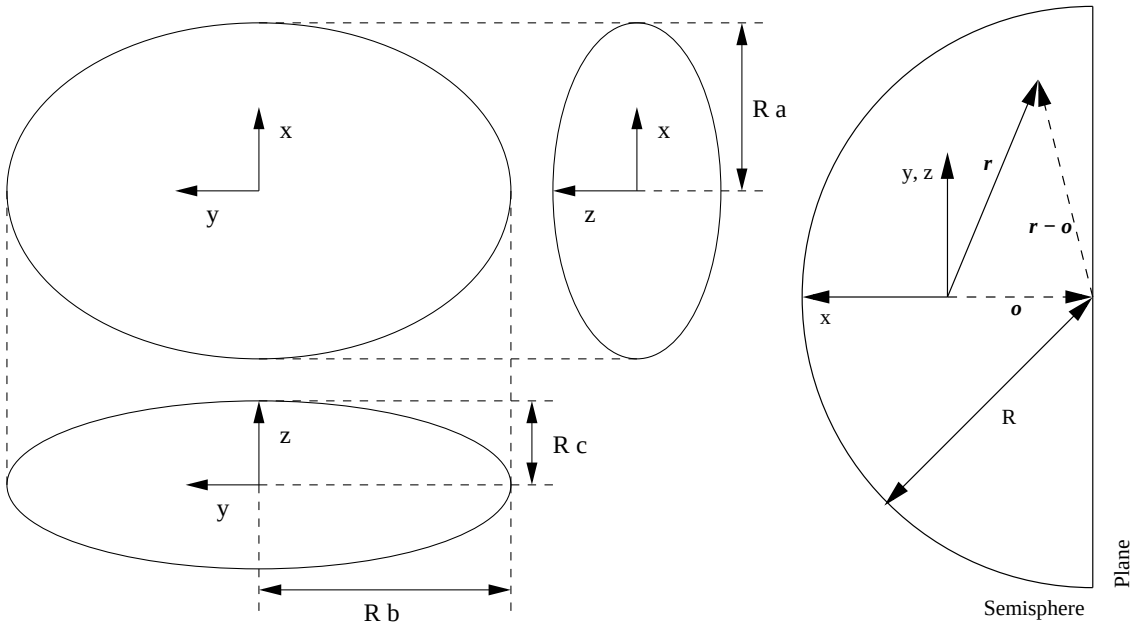


Figure 6: The ellipsoid (left) and the hemisphere (right). The parameters of the ellipsoid are the scale factors a, b and c and the radius R. The only parameter of the hemisphere is the radius R. Shown is the offsetvector $\mathbf{o}$, which defines the position of the centre of the sphere. Such a non vanishing vector is necessary to make it possible to cut a vector to its correct length. To see this, if an arbitrary vector $\mathbf{r}$ ends within the container, one has to take the formula $(\mathbf{r}-\mathbf{o})^2 - R^2 = 0$.

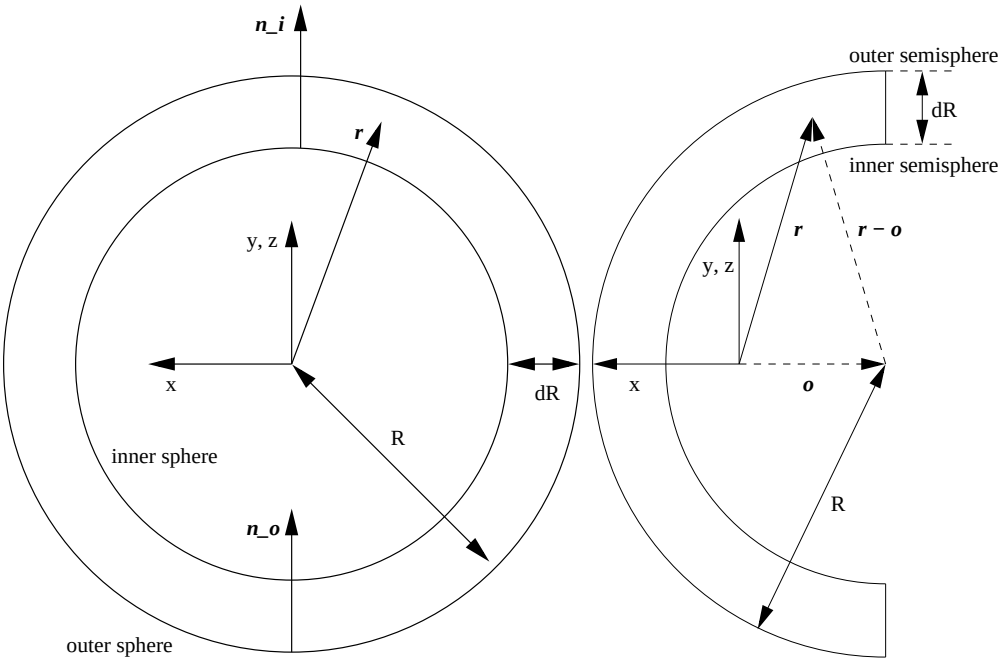


Figure 7: The hollow sphere (left) and the hollow hemisphere (right). The parameters of the hollow sphere are the radius R of the outer sphere and the shell thickness dR. Shown is an arbitrary vector $\mathbf{r}$ which ends within the container; $\mathbf{n}_o$ is a normal vector on the outer sphere and $\mathbf{n}_i$ is one on the inner sphere. They fulfill the requirement of pointing towards the inside of the container. The parameters of the hollow hemisphere are the radius R of the outer sphere and the shell thickness dR. Shown is an arbitrary vector $\mathbf{r}$ which ends within the container. For more explanations see also figure 6 (right).

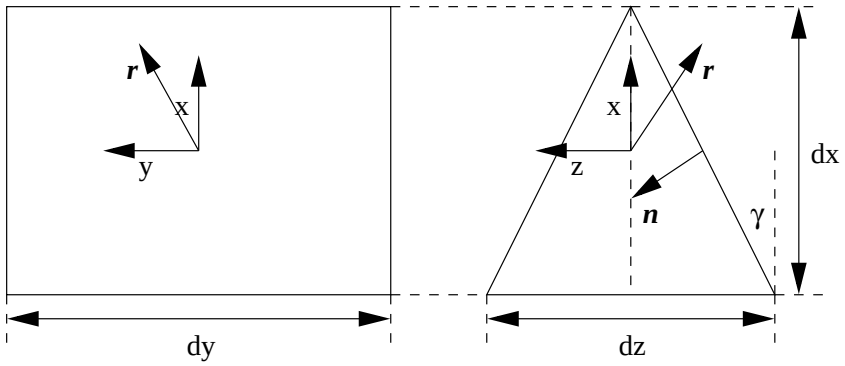


Figure 8: The prism. The parameters are dx, dy and dz. Shown is an arbitrary vector $\mathbf{r}$ which ends outside the container. The right figure shows a normal vector $\mathbf{n}$ on an inclined plane and the inclination angle γ. Here it is $\mathbf{n} = (-\sin\gamma, 0, \cos\gamma)$. The position vector $\mathbf{r}_s$ (undrawn) equals $(-dx/2, 0, -dz/2)$.

All the parameters of the containers can be defined within the file `run.h` within the project directory. In this file they have names which are different from these used in this document. The following table lists the corresponding names.

In this document	**In the file** `run.h`
dx, a, dR	`x_plasma`
dy, b	`y_plasma`
dz, c	`z_plasma`
R	`r_sphere`

Table 2: Parameters for the geometries in this document and in the file `run.h`.

The required container can be chosen with the help of a flag, also set in the file `run.h`. Its name is `initial_config`. The following table lists the value of `initial_config` and the corresponding container.

Value of `initial_config`	**Container**
`0`	cuboid
`1`	sphere
`2`	disc
`3`	wire
`4`	ellipsoid
`5`	prism
`6`	hemisphere
`7`	hollow ball
`8`	hollow hemisphere

Table 3: The flag for the required container.

New interactions for PEPC.

PEPC was originally developed to simulate plasma systems, so therefore only a COULOMB potential had been implemented to date. To explicit the capability of PEPC for modelling other types of force-law, we first generalize existing concepts used in the code following the pattern of Part I. The relevant concept to

generalize in this part is the multipole expansion.

After this generalization, we will implement two further potentials beside the COULOMB potential, namely the LENNARD-JONES- and a fit to the MORSE-potential. Finally, one of these two potentials can be used to build up promising start configurations.

Multipole expansion for finite-power-series potentials.

Calculus.

The following calculations base on these in [8] and [9]. Assume we have a potential like

$$\varphi(\mathbf{r}) = e\|\mathbf{r} - \mathbf{r}_0\|^k$$

with $\mathbf{r} = (X_1, X_2, X_3) \in \mathbf{R}^3$, an exponent $k \in \mathbf{Z}$, a generalized charge $e \in \mathbf{R}$ and the location $\mathbf{r}_0 = (x_1, x_2, x_3) \in \mathbf{R}^3$ of this charge. This potential can be expanded around $\mathbf{r}_0 = \mathbf{0}$ following TAYLOR:

$$\varphi(\mathbf{r}) = \varphi_0(\mathbf{r}) + \varphi_1(\mathbf{r}) + \varphi_2(\mathbf{r}) + \mathcal{O}(\mathbf{r}_0^3)$$

with

$$\varphi_0(\mathbf{r}) = e\|\mathbf{r}\|^k,$$

$$\varphi_1(\mathbf{r}) = e\mathbf{r}_0 \cdot \nabla_{\mathbf{r}}\|\mathbf{r}\|^k = ke\mathbf{r}_0 \cdot \mathbf{r}\|\mathbf{r}\|^{k-2},$$

$$\varphi_2(\mathbf{r}) = \frac{1}{2} \sum_{(\alpha,\beta)\in\{1,2,3\}^2} ex_\alpha x_\beta \frac{\partial^2}{\partial X_\alpha \partial X_\beta}\|\mathbf{r}\|^k =$$

$$= \frac{1}{2} \sum_{(\alpha,\beta)\in\{1,2,3\}^2} ex_\alpha x_\beta \left(k(k-2)X_\alpha X_\beta\|\mathbf{r}\|^{k-4} + \delta_{\alpha\beta}k\|\mathbf{r}\|^{k-2}\right)$$

and $\mathcal{O}(\mathbf{r}_0^3)$ as a function depending on terms of order three and higher and from which is assumed, that it is negligable in comparison to the summands φ_0 to φ_2.

Because φ should fulfill the LAPLACEian equation

$$\triangle_{\mathbf{r}}\varphi(\mathbf{r}) = 0,$$

we have to substract the terms from φ_2 where α equals to β. This is done by writing $\sum e\left(x_\alpha x_\beta - \frac{1}{3}\mathbf{r}_0^2\delta_{\alpha\beta}\right)$ instead of $\sum ex_\alpha x_\beta$. Then φ_2 gets written as

$$\varphi_2(\mathbf{r}) = \sum_{(\alpha,\beta)\in\{1,2,3\}^2} \frac{e}{6}\left(3x_\alpha x_\beta - \mathbf{r}_0^2\delta_{\alpha\beta}\right)\left(k(k-2)X_\alpha X_\beta\|\mathbf{r}\|^{k-4} + \delta_{\alpha\beta}k\|\mathbf{r}\|^{k-2}\right).$$

φ_0 with $q := e$ correspondents to the monopole term; φ_1 with $\mathbf{d} := e\mathbf{r}_0$ to the dipole term and φ_2 with $(Q_{\alpha\beta})_{(\alpha,\beta)\in\{1,2,3\}^2} := \frac{e}{6}\left(3x_\alpha x_\beta - \mathbf{r}_0^2\delta_{\alpha\beta}\right)$ to the quadrupole term.

To get the forces from the multipole terms, we have to take the negative gradient of φ in $\mathbf{r}$-coordinates. The following terms will occur: From the monopole term:

$$-\nabla_{\mathbf{r}} e\|\mathbf{r}\|^k = -ke\mathbf{r}\|\mathbf{r}\|^{k-2},$$

from the dipole term:

$$-\nabla_{\mathbf{r}} k\mathbf{d}\cdot\mathbf{r}\|\mathbf{r}\|^{k-2} = -k\mathbf{d}\|\mathbf{r}\|^{k-2} - k(k-2)(\mathbf{d}\cdot\mathbf{r})\mathbf{r}\|\mathbf{r}\|^{k-4}$$

and from the quadrupole term:

$$-\nabla_{\mathbf{r}}\delta_{\alpha\beta}k\|\mathbf{r}\|^{k-2} = -\delta_{\alpha\beta}k(k-2)\mathbf{r}\|\mathbf{r}\|^{k-4},$$

$$-\nabla_{\mathbf{r}} k(k-2)X_\alpha X_\beta\|\mathbf{r}\|^{k-4} = -k(k-2)\left(X_\alpha\mathbf{e}_\beta + X_\beta\mathbf{e}_\alpha\right)\|\mathbf{r}\|^{k-4} - k(k-2)(k-4)X_\alpha X_\beta\mathbf{r}\|\mathbf{r}\|^{k-6}.$$

Because of the linearity of the differential operators, the demonstrated expansion is valid for any finite power series potential like

$$\varphi(\mathbf{r}) = \sum_{j=1}^{M} s_j \sum_{i=1}^{N_j} e_{ij}\|\mathbf{r} - \mathbf{r}_{ij}\|^{k_j},$$

where $j \in \{1,...,M\} \subset \mathbf{N}$ labels the form of the potential and $i \in \{1,...,N_j\} \subset \mathbf{N}$ labels a single generalized charge $e_{ij} \in \mathbf{R}$ which forms a potential of the form j around $\mathbf{r}_{ij} \in \mathbf{R}^3$. $s_j \in \mathbf{R}$ is just a scale factor for the resulting force. In case only one form of potential is given, the multipole terms has simply to get summed over i. If there is more than one form of potential, then they have to be calculated form by form.

Remarks.

From a theoretical point of view, finite power series potentials are also just special potentials for which the multipole expansion is valid. In fact the same is true for all two times total differentiable potentials depending on $\mathbf{r} - \mathbf{r}_0$. However, non finite power series potentials such as exponential or trigonometrical functions are too expensive to get implemented in PEPC, particularly since the force calculation takes sixty to seventy percent of the total computational effort.

The above generalized multipole expansion for finite power series potentials is now implemented into PEPC. To add a new force, all that is needed is to set an integer identifier for the new force and define the number of terms, the scale factors and the exponents. Some consideration about the scaling of the charges may be necessary. The already implemented routine to calculate the multipole moments for a COULOMB interaction ($k = -1$), can be retained as the mathematical expressions above suggest.

Implementation of the LENNARD-JONES- *and the* MORSE*-potential.*

In physics, three forms of potential are frequently used. Gravitational forces between masses and forces between electrical charges are usually described with the COULOMB potential, which is currently implemented in PEPC for the electrical case. A commonly used short-range potential is the LENNARD-JONES potential and is the most important one in molecular dynamics simulations. A third commonly used potential to describe bonds between atoms or molecules is called MORSE potential.

The LENNARD-JONES *potential.*

In this section we discuss how to implement the LENNARD-JONES- and the MORSE potential into PEPC, using the generalized multipole expansion for finite power series potentials as developed in the last section. We start with the LENNARD-JONES potential, commonly expressed as [10]

$$\varphi(\mathbf{r}) = 4\varepsilon \left[\left(\frac{\sigma}{\|\mathbf{r} - \mathbf{r}_0\|} \right)^{12} - \left(\frac{\sigma}{\|\mathbf{r} - \mathbf{r}_0\|} \right)^{6} \right], \tag{3}$$

where $\varepsilon = 4.06 \cdot 10^9 \frac{\text{TeV}}{\text{mol}}$ is a characteristic energy and σ in the range of nanometers is a cross section of the used particles. The figure on the next page shows the qualitative form of the LENNARD-JONES potential.

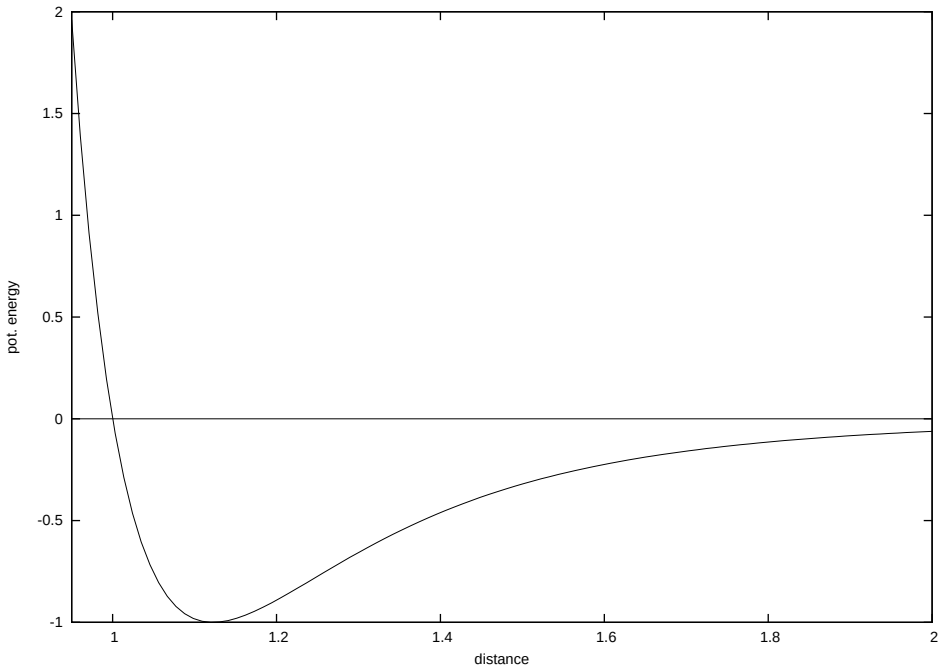


Figure 9: Qualitative form of the LENNARD-JONES potential in the range $0.95 < \|\mathbf{r} - \mathbf{r}_0\| < 2$. Here is set $\varepsilon = \sigma = 1$.

This is implemented into PEPC by defining $s_1 = 4\varepsilon\sigma^{12}, k_1 = -12$ and $s_2 = -4\varepsilon\sigma^6, k_2 = -6$. The charge for the LENNARD-JONES potential is set to one.

The MORSE *potential.*

We now consider how to implement the MORSE potential into PEPC, starting with its mathematical description [11]

$$\varphi(\mathbf{r}) = D\left[1 - \exp\left(-a(\|\mathbf{r} - \mathbf{r}_0\| - r_e)\right)\right]^2 \tag{4}$$

with D as the potential energy for bond formation and a again a kind of cross section to control the width of the potential well. r_e is the equilibrium bond length.

The MORSE potential is obviously not a finite power series function. To use it in PEPC, we have to expand it around r_e following TAYLOR. If we do so, we get to third order:

$$\varphi(\mathbf{r}) \approx D\left[1 + a^2(\|\mathbf{r} - \mathbf{r}_0\| - r_e)^2 - a^3(\|\mathbf{r} - \mathbf{r}_0\| - r_e)^3\right]. \tag{5}$$

A visible improved fitting approximation with order three seems to be

$$\varphi(\mathbf{r}) \approx D\left[\frac{3}{2}a^2(\|\mathbf{r} - \mathbf{r}_0\| - r_e)^2 - \frac{4}{5}a^3(\|\mathbf{r} - \mathbf{r}_0\| - r_e)^3\right]. \tag{6}$$

The figure on the next page compares the exact MORSE potential (solid line) with its TAYLOR expansion to third order (points) and the visible improved fit (points on solid line).

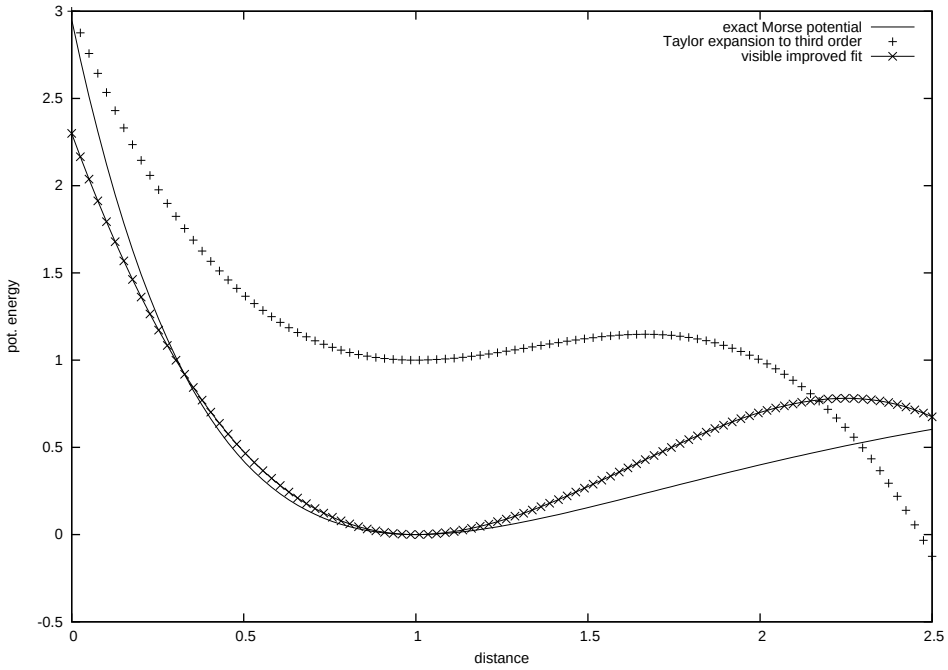


Figure 10: Comparison between the MORSE potential (4) (solid line), its TAYLOR expansion to third order (5) (points) and a visible improved fit using a series potential (6) (points on solid line). The parameters are set to $D = a = r_e = 1$.

The terms of this approximation have to be sorted in orders of the distance before implementing it in PEPC. Rearranging, we have

$$\varphi(\mathbf{r}) \approx D\left[\left(\frac{3}{2}a^2r_e^2+\frac{4}{5}a^3r_e^3\right)+\left(-3a^2r_e-\frac{12}{5}a^3r_e^2\right)\|\mathbf{r}-\mathbf{r}_0\|+\right.$$
$$\left.+\left(\frac{3}{2}a^2+\frac{12}{5}a^3r_e\right)\|\mathbf{r}-\mathbf{r}_0\|^2+\left(-\frac{4}{5}a^3\right)\|\mathbf{r}-\mathbf{r}_0\|^3\right].$$

Because φ is a potential, the first constant term is not of interest. So the suggested approximation of the MORSE potential is implemented into PEPC as follows:

$$s_1=-D\left(3a^2r_e+\frac{12}{5}a^3r_e^2\right),k_1=1,$$
$$s_2=D\left(\frac{3}{2}a^2+\frac{12}{5}a^3r_e\right),k_2=2,$$
$$s_3=-\frac{4}{5}Da^3,k_3=3.$$

The charge for the MORSE potential is set to one.

Remarks.

As when choosing a geometry, the interaction between the particles can also be chosen in the file `run.h`, again by setting a flag named `force_variation`. The following table lists up the value of `force_variation` and the corresponding interaction:

Value of `force_variation`	**Interaction**
`0`	none
`1`	COULOMB potential
`2`	LENNARD-JONES potential
`3`	the visible improved fit to MORSE potential

Table 4: Setting the flag `force_variation` to get an interaction.

Before using the LENNARD-JONES or the MORSE potential, one has to define the parameters of these potentials. For the LENNARD-JONES potential the parameters are ε, called `lj_eps` in `run.h` and σ, called `lj_sigma` in `run.h`. For the MORSE potential, the parameters are D (`morse_d` in `run.h`), a (`morse_a` in `run.h`) and r_e (`morse_r` in `run.h`).

Towards static equilibrium configurations.

Now we are in the position to find a static equilibrium configuration for an N-particle system. The particles get spread randomly inside a geometrically well defined container. Taking a potential $\varphi:\mathbf{R}_{>0}\to\mathbf{R}$ with $\|\mathbf{r}-\mathbf{r}_0\|$ as the argument and one global minimum, we can try to find such a stable configuration dynamically.

Theoretical preparations.

Selection of the potential.

The LENNARD-JONES potential as well as the MORSE potential, implemented in PEPC as described in the last section, seems to be a potential like we are looking for. But naively taking the exact LENNARD-JONES potential (3), the particle system suffers numerical heating. Because we are not interested in

the mechanism how static equilibrium configurations arise in nature but only need them for the actual simulation, we are free to modify the LENNARD-JONES potential so that the heating of the system will be suppressed. We take the following potential:

$$\varphi(\mathbf{r}) = 4\varepsilon \left[\left(\frac{\sigma}{\|\mathbf{r} - \mathbf{r}_0\| + \delta} \right)^{12} - \left(\frac{\sigma}{\|\mathbf{r} - \mathbf{r}_0\| + \delta} \right)^{6} \right], \tag{7}$$

where the modifying of (3) is composed by the new parameter $\delta \in \mathbf{R}_{>0}$ which shifts the minimum nearer to $\|\mathbf{r} - \mathbf{r}_0\| = 0$. Experimentally we found δ to be best as $\delta = 0.96\sigma$. The following figure shows the comparison between the exact LENNARD-JONES potential (3) and our modified version (7).

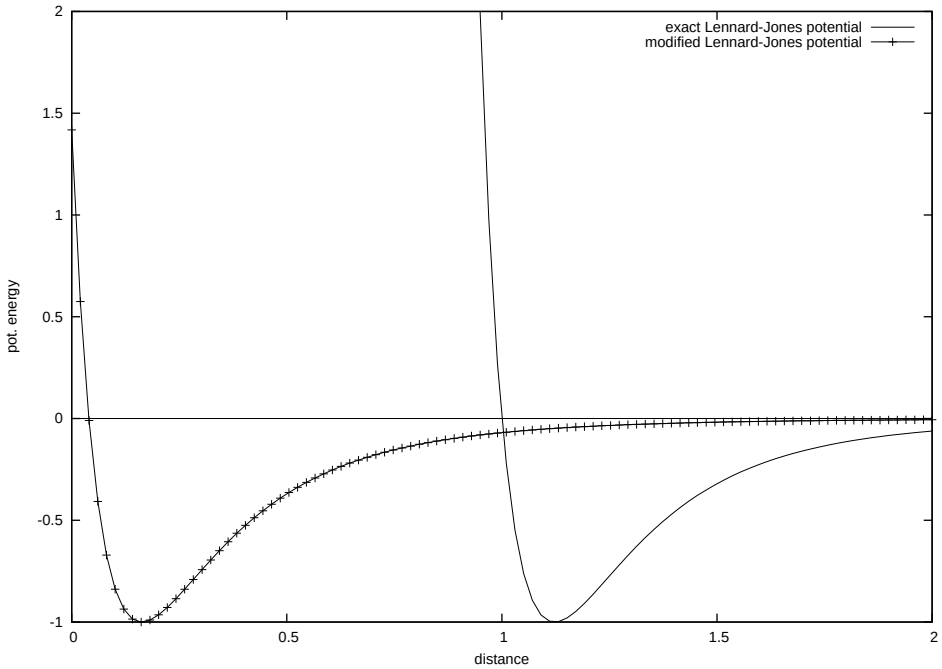


Figure 11: Comparison between the exact LENNARD-JONES potential (3) and the modified version (7) used in our simulations to find static equilibrium configurations.

One can see that the modified potential do not raise infinity but reaches the maximum in magnitude of the minimum (equal to ε) as an ordinate intersect. This allows the particles to interpenetrate each other, but we have mentioned that the correct physical picture of the potential is not of interest to date. Next we have to find the range of σ. Therefore, we first ascertain the distance of the minimum. With $\delta = 0.96\sigma$ this can be calculated to

$$\|\mathbf{r} - \mathbf{r}_0\|_{\text{min}} = \left(\sqrt[6]{2} - 0.96 \right) \sigma$$

and leads to

$$\sigma = \frac{\|\mathbf{r} - \mathbf{r}_0\|_{\text{min}}}{\sqrt[6]{2} - 0.96}.$$

On the other hand, the average distance l between the particles in the range of

$$l \approx \sqrt[3]{\frac{V}{N}}$$

with V as the volume of the restricted space and N as the number of particles, should overcome with $\|\mathbf{r} - \mathbf{r}_0\|_{\min}$. Thus we get

$$\sigma \approx \frac{1}{\sqrt[6]{2} - 0.96} \sqrt[3]{\frac{V}{N}}.$$

Our simulations had shown that σ should be smaller then the above approximation. Otherwise we observe the appearance of clusters. The parameter ε defines the maximum acceleration and should be coordinated with the length of the time steps.

Selection of the ensemble.

From Part I we expect no interactions of the particle system inside the container with its surrounding. So we prefer to assume, that the volume V, the energy E and the number of particles N are conservation values. This means we study a microcanonical ensemble and perform (N, V, E)-simulations to get the static equilibrium configuration. But this assumption brings us in conflict to the second law of thermodynamics.

Let us look at the example of a cuboid. Its ideal start configuration (final configuration f) seems to be a fcc-lattice-configuration but at startup the particles are spread randomly inside the cuboid (initial configuration i). To describe the initial configuration, we have to list up all N position vectors $(\mathbf{r}_j)_{j\in\{1,\dots,N\}}$ of the particles so that the entropy is

$$S_i \propto \ln(N) + c.$$

To describe an ideal cubic lattice we only need one position vector, three vectors for the lattice basis and one vector to describe the size of the lattice. Here the entropy is

$$S_f \propto \ln(5) + c.$$

We try to simulate the transition $i \to f$ and in our simulations (and in natural N particle systems) we have $N \gg 5$. So we get

$$\Delta S_{i\to f} = S_f - S_i \propto \ln(5) - \ln(N) < 0$$

in conflict to the second law of thermodynamics

$$\Delta S_{i\to f} \geq 0.$$

The way out is to simulate a canonical ensemble instead of the microcanonical one. This means to conserve the temperature T of the system instead of the whole energy E. We do such (N, V, T)-simulations by substracting average speed increases per particle and component from each particle and component.

The picture behind a canonical ensemble is that the N particle system Σ is embedded within a much larger system Σ' at the same temperature as Σ which can, because of its size, absorb thermal energy coming from Σ without heating up significantly. Thus for the system Σ we have $\Delta S < 0$ but for the whole system $\Sigma + \Sigma'$ we have $\Delta S \geq 0$.

Quality of the configuration.

We make two assumptions for the final configuration:

1. The system develops towards a lattice configuration.
2. The potential energy of the system develops towards a minimum.

To test the first assumption, we can calculate the structure factor [12, 13]

$$m_{\text{cub}} := \frac{2}{N(N-1)} \sum_{i<j} \cos\left(\mathbf{r}_{ij} \cdot \mathbf{g}_{\text{cub}}\right) \forall (i,j) \in \{1, ..., N\}^2 \tag{8}$$

with

$$\mathbf{g}_{\text{cub}} := \frac{4\pi}{d}(1,1,1)$$

and d as the lattice constant.

The value of m_{cub} equals one if the configuration is an ideal cubic (sc-, bcc-, fcc-) lattice and fluctuates around zero if the configuration is a random one. Obviously, this value is only usable if the container is built up from cuboids and the number of particles is adjusted to the size of the container. For round containers, m_{cub} at most may be used as a *hint* for a promising configuration. We will see that m_{cub} is a suitable quantifier for promising configurations for small N only.

To prove the second assumption, we can calculate the potential energy.

Both the potential energy and the structure factor get calculated from PEPC periodically after a defined number of timesteps in `run.h` and get written into a file so that their time evolution can be studied.

Examples.

A "big"cuboid.

The following figure shows the time evolution of a system of $4096 = 16^3$ particles constrained in a $1\left(\frac{c}{\omega_p}\right)^3$ sized cuboid (c is the speed of light and ω_p is the plasma frequency). The temperature is held at 0.5keV and is distributed among the particles via a MAXWELL distribution. The LENNARD-JONES parameter σ is set to $0.05\frac{c}{\omega_p}$ and ε is set to 0.01.

The first picture (top-left) shows the initial configuration. The particles are randomly spread inside the cuboid. After 10 timesteps (top-right), the cuboid is completely filled out with the particles. The system has cooled down, which can be seen by the color coded speed in the pictures. A regular spacing between

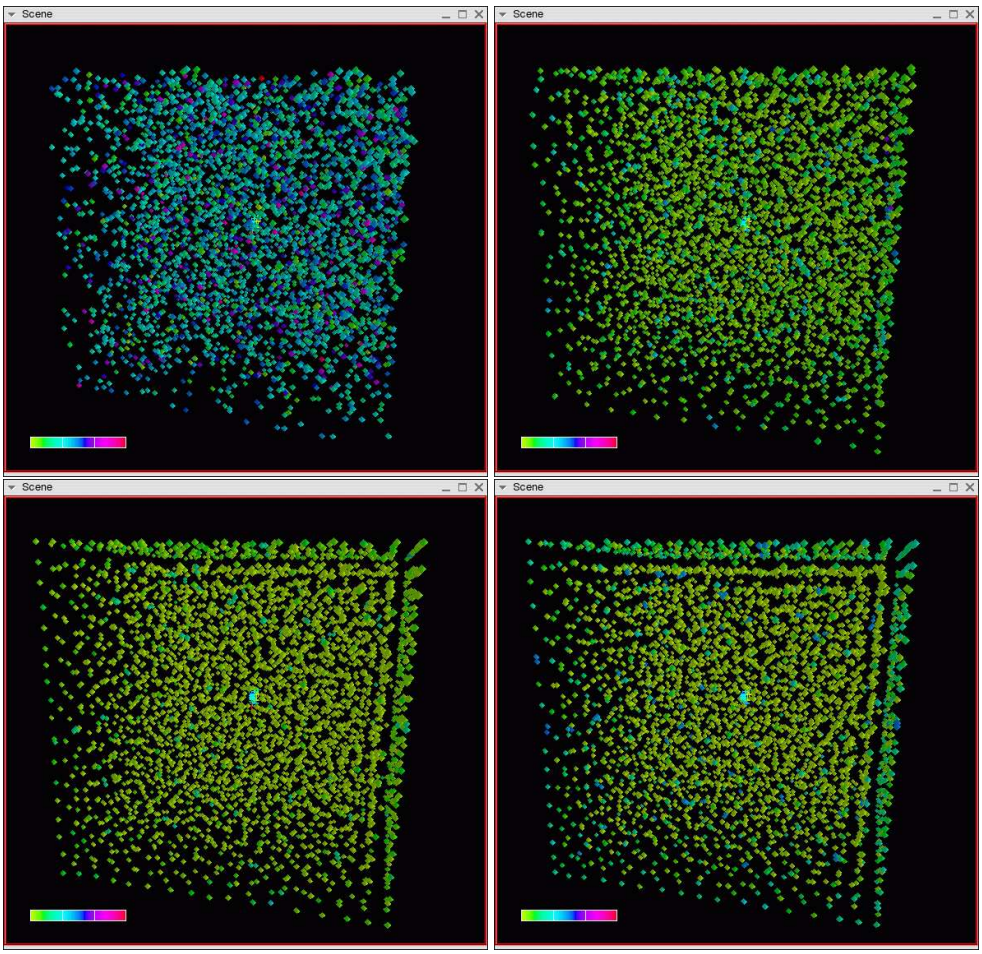


Figure 12: Time evolution of a 4096 particle system in a cuboid. Initial configuration top-left; 10 timesteps top-right; 50 timesteps bottom-left; 200 timesteps bottom-right.

two layers of particles next to the container walls can be observed. After 50 timesteps (bottom-left), these spacings can be seen more clearly. Also one or two more spacings are visible. After 200 timesteps (bottom-right), the spacings are even sharper. The particles next to the walls are faster than these near the middle of the cuboid. We can count 17 particles at front, bottom edge of the cuboid. Because we have 16^3 particles in the cuboid, we expected 16 particles at each edge.

It is interesting to look at the quality of the configuration. The following figure shows the time evolution of the structure factor and the potential energy.

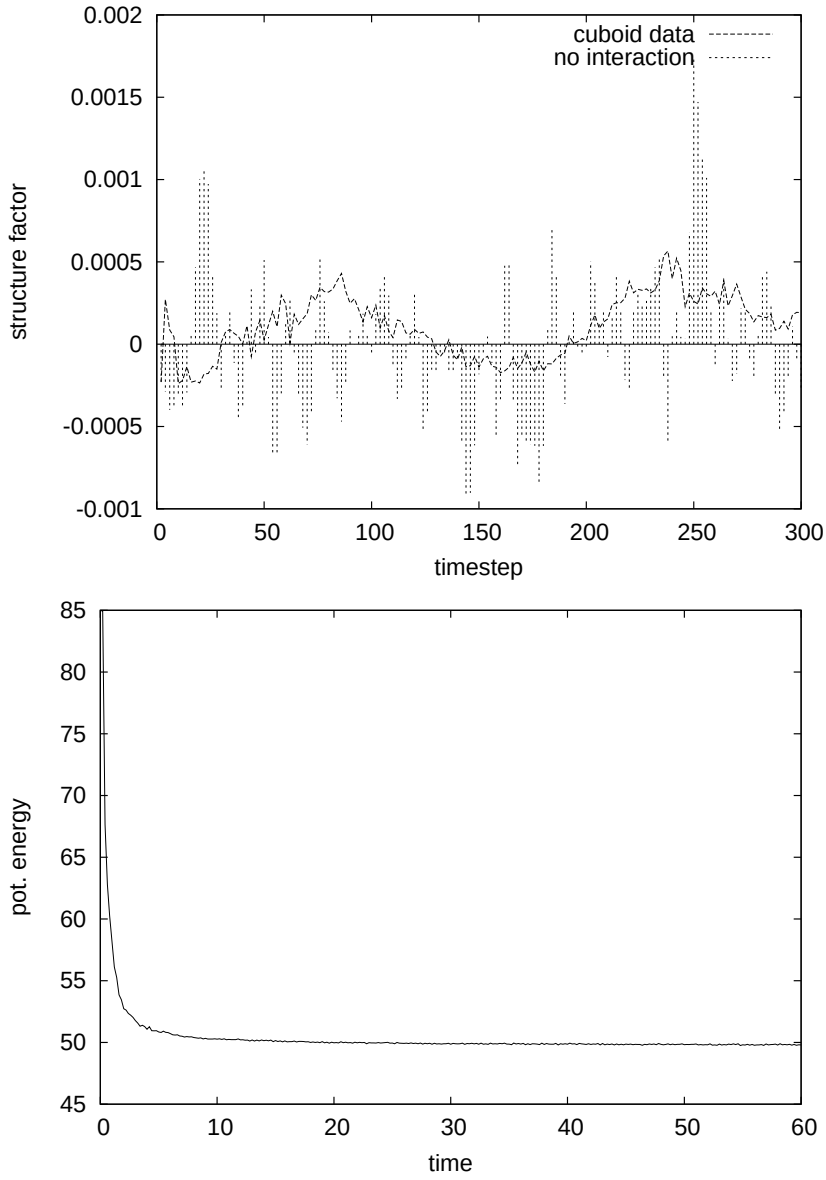


Figure 13: The time evolution of the quality of the big cuboids configuration. Shown is the structure factor (top) (the impulses come from a system with no interaction) and the potential energy (bottom).

First we determine that the potential energy has reached a minimum after about 100 timesteps. Our second determination is that the structure factor seems less suitable to identify promising configurations. The pictures show that a structure develops in time. But this structure could not be captured by the structure factor. The next example will give a hint why this is so.

A "small"cuboid.

In this example we deal with $64 = 4^3$ particles, again inside a cuboid of the size $1\left(\frac{c}{\omega_p}\right)^3$. σ is set to $0.3\frac{c}{\omega_p}$ and ε is set to 1.0. The following figure shows the time evolution.

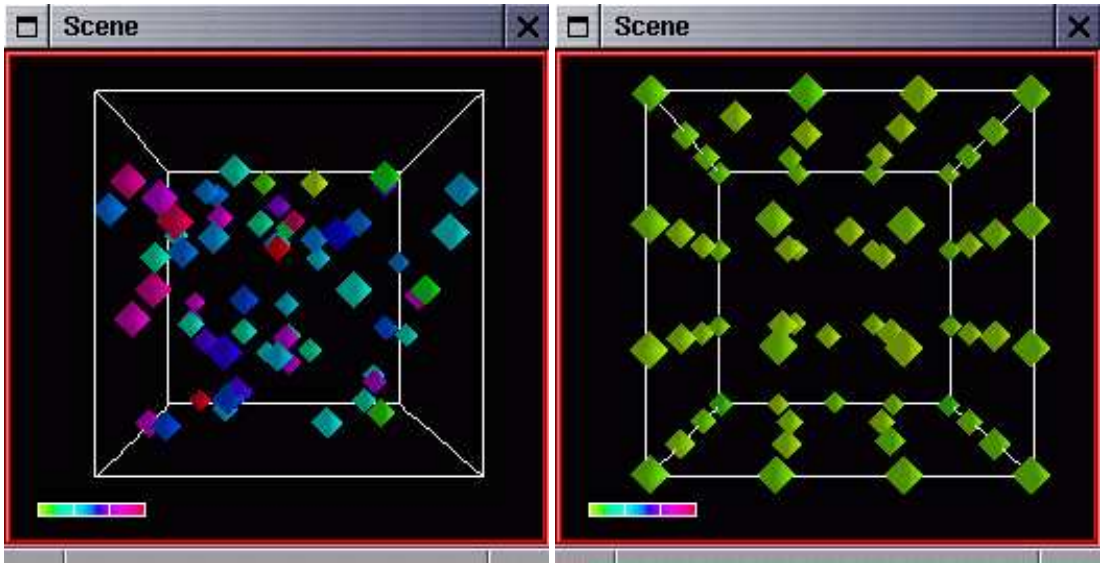


Figure 14: Time evolution of a 64 particle system constrained within a cuboid. On the left hand side is shown the initial configuration. The right hand side shows the system after 200 timesteps.

The developed structure is obvious. We will look at the quality indicators for this system.

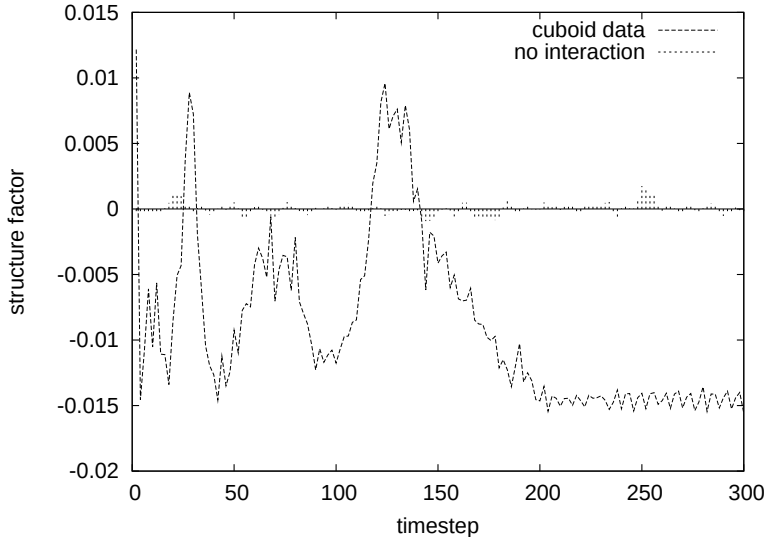


Figure 15: The time evolution of the structure factor of the small cuboid. The impulses come from a system with no interaction again.

Here our determinations are as follows: First, the equilibrium value of the structure factor is about two magnitudes better then this of the "big"cuboid. Second, the system seems to develop towards a local minimum. This can be verified by looking at right picture of figure 14 in detail. Four of the particles are not at their ideal positions if the end configuration should be a cubic lattice. But they do not move from their recent position significantly. They have found a local minimum. To see that the system has reached a promising configuration we will, for completeness, look at the time evolution of the potential energy.

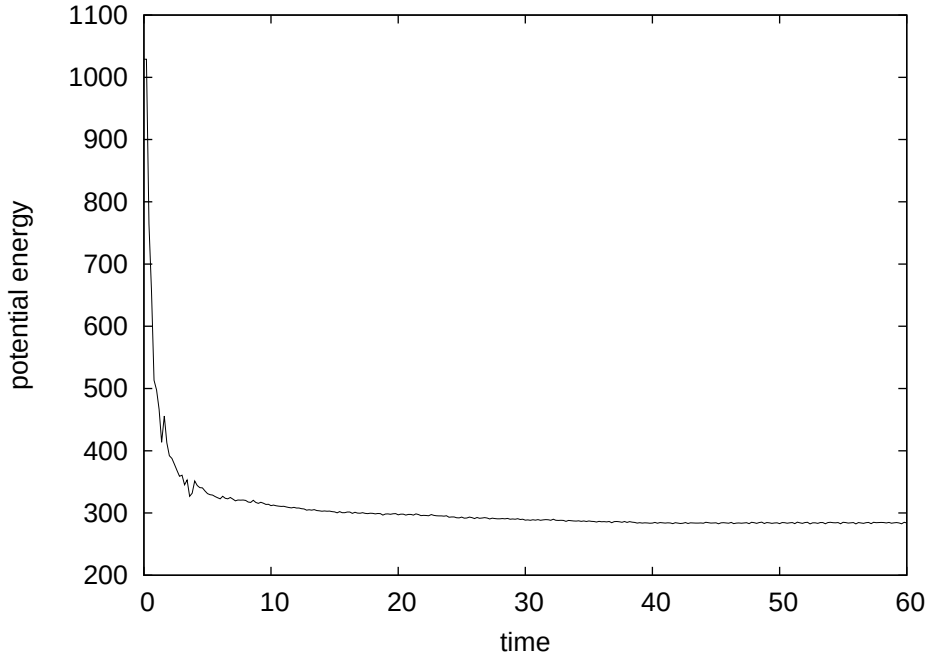


Figure 16: Time evolution of the potential energy of the small cuboid system.

A hollow hemisphere.

A more complex geometry is the hollow hemisphere, a system which would normally be difficult to set up as a static equilibrium configuration. Nevertheless, we tried to perform a simulation according to the procedure described above. 5000 particles were placed in a hollow hemisphere of thickness $0.5\frac{c}{\omega_p}$. The temperature of the system was 0.5keV. The parameters of the LENNARD-JONES potential were set to $\sigma = 0.03\frac{c}{\omega_p}$ and $\varepsilon = 0.1$. The figures 17 and 18 on the next page show the time evolution of this system.

The process shown in figure 17 is similar to that for the "big"cuboid system (compare figure 12). Starting with a random configuration (top-left) the systems cools down (after 10 timesteps, top-right) and regular spacings develop. After 50 timesteps (bottom-left), three layers parallel to the surfaces of the hollow hemisphere have arisen. After 200 timesteps (bottom-right), the particles next to the walls have high kinetic energy. Also some particles get arranged into nearly equidistant rings around the symmetry axis. The equilibration value of the potential energy is reached after 100 timesteps as in the first example.

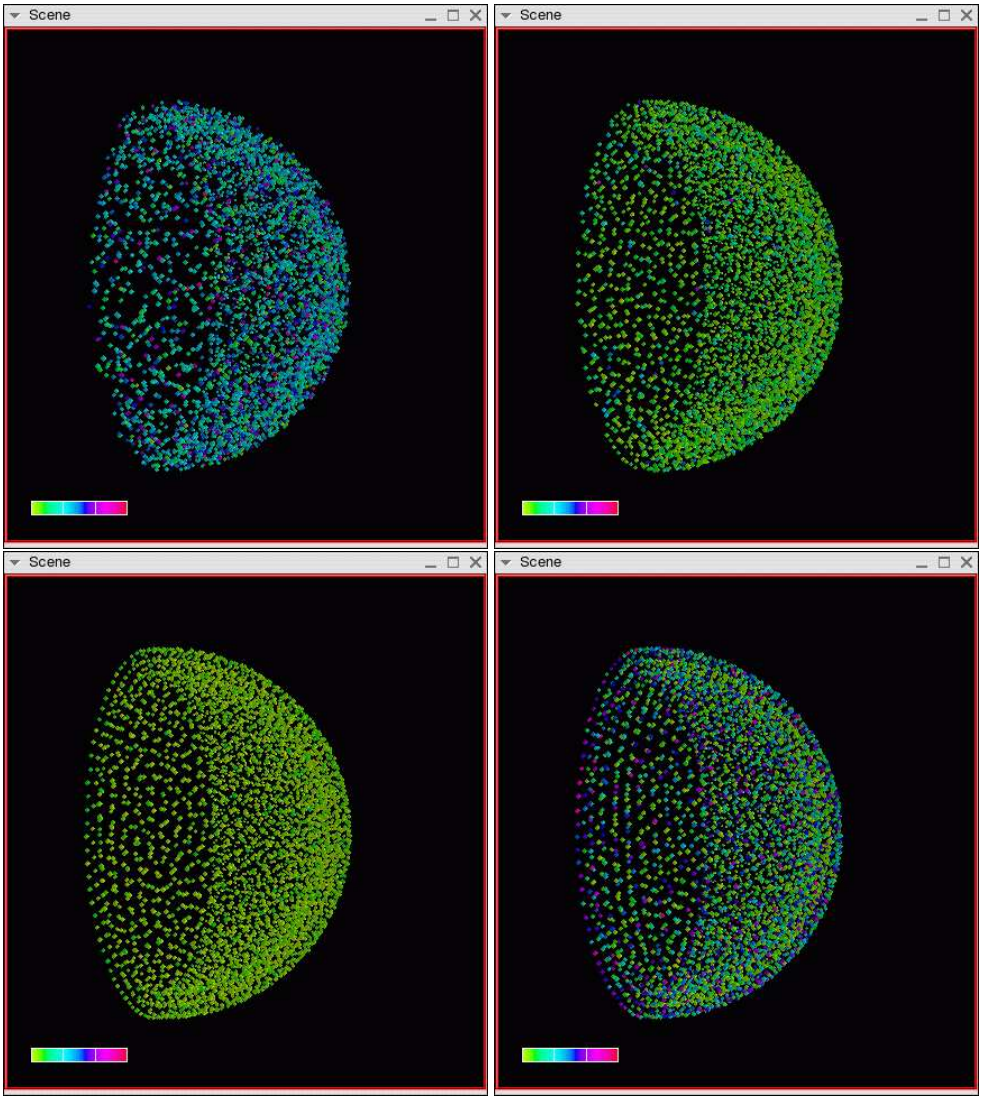


Figure 17: Time evolution of a 5000 particles system inside a hollow hemisphere. Shown are snapshots of the initial configuration (top-left), after 10 timesteps (top-right), after 50 timesteps (bottom-left) and after 200 timesteps (bottom-right).

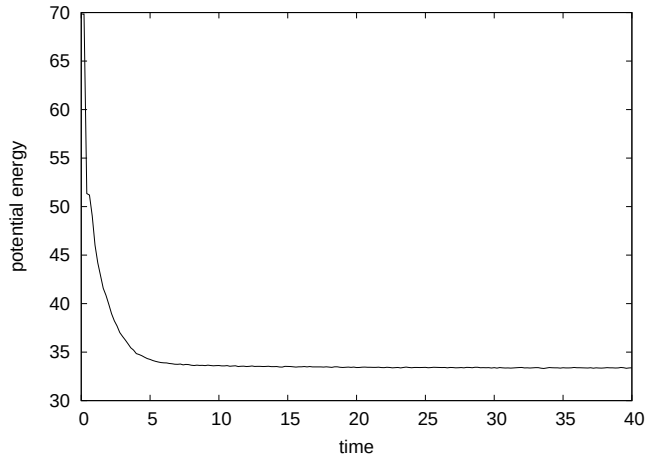


Figure 18: Time evolution of the potential energy of the system in a hollow hemisphere.

Conclusions and outlook.

We have developed a constraint routine which can deal with various more or less complicated geometries parameterized by planes, spheres, ellipsoids and (elliptic) cylinders. Nine of such geometries are now implemented into PEPC. Further we could show that the multipoles for any finite-power-series potentials are equal to that of a COULOMB potential. This was used to implement the LENNARD-JONES- and a fit to the MORSE potential into PEPC which uses a kind of multipole code to compute forces between particles. These two additions were used to find static equilibrium configurations for N-body systems constrained in complex containers.

The configuration will be immediately put to use in simulations of beam-plasma interactions as mentioned in the introduction. It still needs to be tested whether one could get better configurations by using the implemented fit to the MORSE potential instead of the used modified LENNARD-JONES potential. Further developments of PEPC will include finding a more suitable quantifier instead of the structure factor to identify promising configurations automatically. The idea would be to implement a more general pair correlation function. Finally, one could refine start configurations by heating up a system if it has reached a local, rather than a global minimum.

Added and modified source-code files.

The author has added and modified many source-code files for PEPC. The following list gives a short description of each added and modified source-code file:

`run.h`: Configuration file of PEPC. Here are added multiple flags and parameters to chose the geometry and the interaction and to configure the interaction.

`constrain.f90`: Reimplementation of the constrain routine, now according to Part I of this document.

`cut_vector.f90`: Calculation of the normal vector and the position vector of the tangent plane.

`face.f90`: Implementation of the parametrizations of the geometries.

`randion.f90`: The randomly spreading of the particles is implemented for all geometries.

`sum_forces.f90`: The implementation of the force summation is modified according to Part II of this document.

`forces_bal.f90`: The call of `sum_forces` is implemented for all interactions in PEPC.

`structure.f90`: Implementation of the structure factor (8).

`treevars.f90`: Definition and initialization of the additional needed global variables.

`setup.f90`: Additions to the namelist reading in `run.h`; initial calculations and definitions for the new geometries and interactions.

`makefile`: Addition of the new files `cut_vector.f90`, `face.f90` and `structure.f90`.

References

1. P. Gibbon: PEPC: Pretty Effi cient Parallel Coulomb-solver, Research Centre Jülich, 2003, available from `http://www.fz-juelich.de/zam/docs/autoren2003/gibbon.html`
2. A. Pukhov: Three-Dimensional Simulations of Ion Acceleration from a Foil Irradiated by a Short-Pulse-Laser, Phys. Rev. Lett **86** 16, 2001
3. H. Ruhl et. al.: Computer Simulation of the Three-Dimensional Regime of Proton Acceleration in the Interaction of Laser Radiation with a thin Spherical Target, Plasma Physics Report **27** 5, 2001
4. Y. Sentoku et. al.: High-energy ion generation in interaction of short laser pulse with high-density plasma, Appl. Phys. B **74**, 2002
5. R. A. Snavely et. al.: Intense High-Energy Proton Beams from Petawatt-Laser Inrradiation of Solids, Phys. Rev. Lett. **85** 14, 2000
6. M. Zepf et. al.: Proton Acceleration from High-Intensity Laser Interactions with Thin Foil Targets, Phys. Rev. Lett. **90** 6, 2003
7. K. Meyberg, P. Vachenauer: Höhere Mathematik I, Springer-Verlag, 2001
8. L. D. Landau, E. M. Lifschitz: Lehrbuch der theoretischen Physik II - Klassische Feldtheorie, Akademie-Verlag Berlin, 1987
9. A. Lindner: Grundkurs Theoretische Physik, Teubner, 1997
10. From the sites of the Center of Polymer Studies, Boston University `http://polymer.bu.edu/Wasser/robert/work/node8.html`
11. From the sites of the Institute of Physical Chemistry, University Connecticut `http://www.sp.uconn.edu/ ch351vc/pdfs/morse.pdf`
12. H. Vogel: Gerthsen Physik, Springer-Verlag, 1995
13. W. Nolting: Grundkurs Theoretische Physik 7 - Vielteilchentheorie, Springer-Verlag, 2001

Monte Carlo-Simulationen von Ising-Magneten mit periodischem Pinning mobiler Defekte im externen Feld

Martin Holtschneider

Institut für Theoretische Physik
RWTH Aachen

holtschneider@physik.rwth-aachen.de

Zusammenfassung: Ein zweidimensionales Ising-Modell mit kurzreichweitigen Wechselwirkungen und mobilen Defekten, das die Entstehung, Fluktuation und Zerstörung eindimensionaler Defektstreifen beschreibt, wird aus experimentellen Daten motiviert und mit Hilfe von Monte Carlo-Simulationen studiert. Von herausgehobenem Interesse sind die Auswirkungen eines externen magnetischen Feldes und eines lokalen Pinning-Potentials, das die Anordnung der Defekte in geraden äquidistanten Linien energetisch bevorzugt. Das Pinning der Defekte führt bei niedrigen Temperaturen und moderaten Feldstärken zu einer langreichweitig geordneten Phase. Die kritischen Temperaturen, bei denen die geordnete in eine ungeordnete Hochtemperatur-Phase übergeht, werden für verschiedene Feld- und Pinningstärken bestimmt. Die mikroskopischen Mechanismen des Phasenübergangs werden diskutiert.

Einführung

Seit der Entdeckung der Hochtemperatur-Supraleitung ist das Wechselspiel zwischen Spins und Ladung in dotierten Kupraten von zentralem Interesse. Ein wichtiger Aspekt ist dabei das Auftreten eindimensionaler Streifen, die antiferromagnetische Bereiche voneinander trennen. Neuere Messungen, die in der Gruppe von Prof. Büchner an der RWTH Aachen durchgeführt werden, deuten auf die Streifenbildung in sogenannten "Telefonnummern-Verbindungen", $(Sr,La,Ca)_{14}Cu_{24}O_{41}$, hin [1, 2]. Das Material ordnet bei tiefen Temperaturen in Kupferoxid-Ebenen, zwischen denen sich die Strontium-, Lanthan- oder Calcium-Atome befinden (vgl. Abbildung 2). Die Experimente legen nahe, dass Kupfer- und Sauerstoff-Ionen sich in jeder Ebene in einer von zwei periodischen Bindungsstrukturen arrangieren, den Lei-

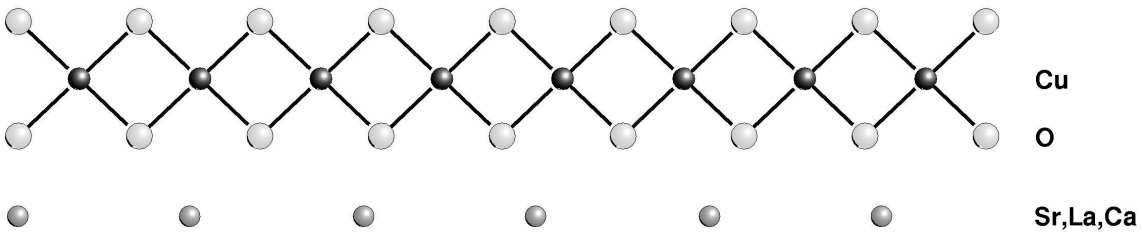


Abbildung 1: Kupferoxidkette in $(Sr,La,Ca)_{14}Cu_{24}O_{41}$. Gezeigt wird die Aufsicht auf einen Ausschnitt einer Kettenebene, ergänzt durch die Positionen der Strontium-, Lanthan- oder Calcium-Atome.

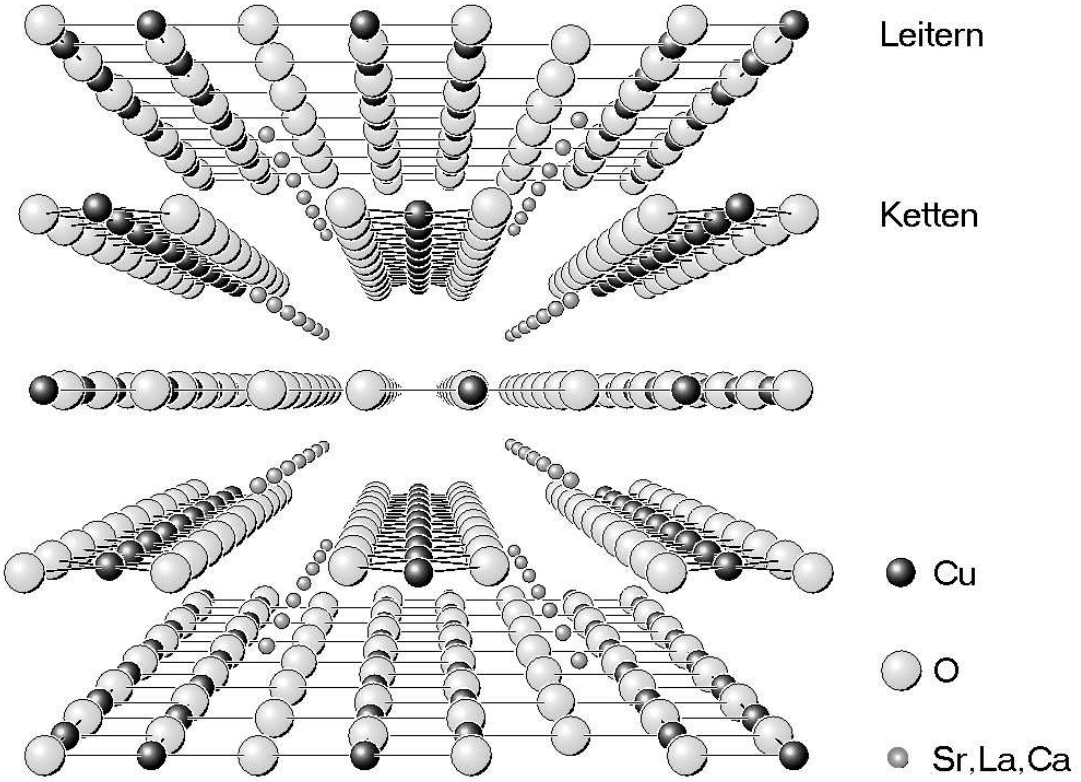


Abbildung 2: Dreidimensionale Darstellung der Ebenenstruktur von $(Sr,La,Ca)_{14}Cu_{24}O_{41}$

tern und den Ketten. Die beiden Ebenentypen alternieren im dreidimensionalen Festkörper (vgl. Abbildung 2). Energieabschätzungen ergeben, dass die Kupferoxid-Ketten im interessierenden Temperaturbereich die magnetischen Eigenschaften bestimmen und deshalb die Streifenbildung moderieren könnten [3]. Innerhalb der in Abbildung 1 detaillierten Ketten trägt jedes Cu^{2+}-Ion einen Spin, $S = 1/2$, der mit einem effektiv ferromagnetischen Austauschintegral, $J > 0$, an die nächstbenachbarten Kupfer-Spins koppelt. Durch die (Sr,La,Ca)-Dotierung wird ein Teil der magnetischen Cu^{2+}-Ionen in unmagnetische Defekte, Cu^{3+}-Ionen mit Spin $S = 0$, umgewandelt [4], die in den Ketten wandern können. Die Wechselwirkung innerhalb der Kette ist dann durch eine stark antiferromagnetische Kopplung, $J_0 < 0$, zu ergänzen, die zwischen Kupfer-Spins vermittelt, die durch einen Defekt voneinander getrennt sind. Eine dritte Wechselwirkung mit antiferromagnetischem Austauschintegral, $J_a < 0$, verbindet in den Ebenen die parallelen Ketten untereinander. Der Mechanismus der (Sr,La,Ca)-Dotierung zwischen den Kupferoxid-Ebenen induziert eine erhöhte Aufenthaltswahrscheinlichkeit der Defekte in der Nähe der Lanthan- oder Calcium-Ionen. Dadurch werden Defektpositionen in den Ketten energetisch ausgezeichnet, d. h. die Defekte pinnen an bestimmte Gitterplätze. Experimentelle Studien an $La_5Ca_9Cu_{24}O_{41}$ dokumentieren ein ungewöhnliches Verhalten der Verbindung im Magnetfeld: Das externe Feld scheint die langreichweitige magnetische Ordnung zu unterdrücken [5].

Die skizzierten experimentellen Resultate und ihre mikroskopische Interpretation motivieren ein einfaches zweidimensionales Ising-Modell mit mobilen Defekten, das die Arbeitsgruppen von Prof. Büchner und Prof. Selke an der RWTH Aachen gemeinsam mit Prof. Pokrovsky (Texas A&M University) untersuchen und weiterentwickeln [6]. Die Modellierung basiert auf dem Ising-artigen Charakter der "Telefonnummern-Verbindungen" und respektiert die experimentell nahegelegten Werte der drei Kopplungsparameter J, J_0 und J_a. Dieses Modell wird im folgenden Abschnitt definiert und auf eine vereinfachte Variante reduziert. Die Simulationstechniken, mit deren Hilfe die thermodynamischen Eigenschaften

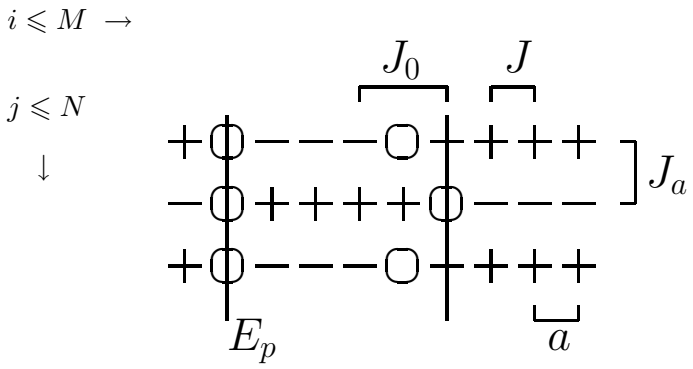


Abbildung 3: Skizze der Wechselwirkungen im Ising-Modell mit periodischem Pinning mobiler Defekte, a bezeichnet die in beiden Gitterachsen gleiche Gitterkonstante. Der 'Spin-Up'-Zustand, $S_{ij} = +1$ wird hier verkürzend mit '+', der 'Spin-Down'-Zustand, $S_{ij} = -1$ mit '-' bezeichnet. Defekte, $S_{ij} = 0$ werden mit dem Symbol '0' dargestellt.

der Ising-Magnete untersucht werden, und ihre effiziente Parallelisierung auf Rechner-Systemen mit gemeinsamem Speicher beschreibt das anschließende Kapitel. Danach werden die bereits publizierten Erkenntnisse kurz zusammengefasst und anhand neuer Daten im externen magnetischen Feld erweitert. Der Bericht schließt mit einem kurzen Ausblick.

Das Modell

Das Modell besteht aus Ising-Spins, die auf einem Quadratgitter der Dimension $M \times N$ mit der Gitterkonstante $a = 1$ angeordnet sind (vgl. Abbildung 3). Jeder Gitterplatz (i, j) ist entweder von einem Spin, $S_{i,j} = \pm 1$, oder einem Defekt, $S_{i,j} = 0$, besetzt. Die Defekte sind entlang einer der Achsen des Gitters beweglich, solange sie einen Mindestabstand von zwei Gitterkonstanten nicht unterschreiten. Dies garantiert, dass zwei benachbarte Plätze (i, j) und $(i + 1, j)$ nicht gleichzeitig von Defekten okkupiert werden. Die ausgezeichnete Gitterachse, entlang derer sich die Defekte bewegen, wird als Kettenrichtung bezeichnet. Innerhalb jeder Kette ist die Defektkonzentration ϑ und damit auch die Anzahl der Defekte gleich, alle Untersuchungen sind bei dem experimentell begründeten Wert $\vartheta = 0.1$ durchgeführt. Zwischen zwei nächstbenachbarten Spins in den Ketten, $S_{i,j}$ und $S_{i+1,j}$ wirkt eine ferromagnetische Kopplung, $J > 0$. Werden zwei Spins, die in einer Kette übernächste Nachbarn sind, $S_{i,j}$ und $S_{i+2,j}$, durch einen Defekt, $S_{i+1,j} = 0$, voneinander getrennt, wechselwirken sie mit einem antiferromagnetischen Austauschintegral, $J_0 < 0$. Die nächstbenachbarten Spins unterschiedlicher Ketten, $S_{i,j}$ und $S_{i,j+1}$ koppeln ebenfalls antiferromagnetisch, $J_a < 0$. Das Modell kann durch ein lokales Pinning-Potential ergänzt werden [7], das einzelne Gitterplätze (i_p, j_p) auszeichnet. Besetzt ein Defekt einen solchen Platz, wird die Energie des Systems um den Betrag E_p abgesenkt. Die Pinning-Positionen liegen auf geraden äquidistanten Linien senkrecht zu den Spin-Ketten, ihre Anzahl entspricht der der Defekte. Ein externes Feld kann in der üblichen Weise dem Hamiltonoperator des Systems hinzugefügt werden:

$$\begin{aligned} \mathcal{H} \quad = \quad -\sum_{j=1}^{N}\sum_{i=1}^{M} \Big[& J S_{i,j} S_{i+1,j} \; + \; J_0 S_{i,j} S_{i+2,j} (1 - S_{i+1,j}^2) \; + \; J_a S_{i,j} S_{i,j+1} \\ & + E_p (1 - S_{i,j}^2) \delta_{i,i_p} \; + \; H S_{i,j} \Big] \end{aligned} \tag{1}$$

Aus experimentellen Daten lassen sich die Kopplungskonstanten J, J_0 und J_a abschätzen. Für die Wechselwirkung zwischen den Spin-Ketten J_a ergibt sich ein Wert, der betraglich die beiden anderen Austauschintegrale, J und J_0, um mindestens eine Größenordnung unterschreitet. Da die schwächste Kopplung das Tieftemperaturverhalten bestimmt, kann eine vereinfachte Variante des Ising-Modells mit mobilen Defekten definiert und untersucht werden [6]. Die Wechselwirkungen in den Kette, J und J_0, werden im sogenannten Minimalmodell als unendlich stark angenommen. Die Bedingung erfordert intakte ferromagnetische Cluster zwischen zwei benachbarten Defekten derselben Kette und einen Vorzeichenwechsel der Spins über einen Defekt hinweg. Der Hamiltonoperator (1) erhält mit diesen Annahmen die folgende, kürzere Form.

$$\mathcal{H} \quad = \quad -\sum_{j=1}^{N}\sum_{i=1}^{M}\Big[\; J_a S_{i,j} S_{i,j+1} \;+\; E_p(1-S_{i,j}^2)\delta_{i,i_p} \;+\; H S_{i,j} \;\Big] \tag{2}$$

Im folgenden Text werden ausschließlich Simulationsergebnisse für das Minimalmodell gezeigt und ausgewertet, für eine Diskussion der thermischen Eigenschaften des vollen Modells sei auf [5], [6] und [8] verwiesen.

Simulationstechnik

Monte Carlo-Simulationen helfen in vielen Gebieten der Statistischen Physik, die thermodynamischen Eigenschaften eines Modellsystems zu studieren. Mehrere Lehrbücher widmen sich ausführlich der Simulationstechnik [9, 10], hier findet nur eine knappe Einführung ihren Platz, die die Anwendung der Methode auf das Ising-Modell mit mobilen Defekten motiviert.
Eine Monte Carlo-Simulation generiert eine Folge von Konfigurationen eines Modellsystems endlicher Größe. Die Folge soll als repräsentativer Pfad durch den Konfigurationsraum aufgefasst werden können, so dass durch Mittelung über die Folgenglieder thermodynamische Größen im Gleichgewicht berechnet werden. Dabei folgt das Modell nicht notwendig einer Trajektorie, die einer exakten Lösung der Bestimmungsgleichungen entspricht. Die Verkettung der Folge beruht stattdessen auf einem stochastischen Algorithmus: Ausgehend von einer vorzugebenden Startkonfiguration $X^{(0)}$ wird jedes Folgenglied $X^{(l)}$ durch die zufällige Änderung seines direkten Vorgängers $X^{(l-1)}$ erzeugt. Die Wahrscheinlichkeiten W_{ji}, mit denen der Algorithmus in einem beliebigen Schritt l eine Konfiguration, $X^{(l-1)} = S^j$, in einen vorgebenen Nachfolger, $X^{(l)} = S^i$, transformiert, läßt sich als bedingte Wahrscheinlichkeit formulieren:

$$W_{ji} \quad = \quad W(S^j \to S^i) \quad = \quad P(X^{(l)} = S^i \mid P^{(l-1)} = S^j)$$

Für die Übergangswahrscheinlichkeiten gilt natürlich

$$W_{ji} \quad \geq \quad 0 \qquad\qquad \text{und} \qquad\qquad \sum_i W_{ji} \quad = \quad 1 \qquad .$$

Die Wahrscheinlichkeit, mit der sich das System nach n Iterationen des Algorithmus in der Konfiguration S^i befindet, bezeichne $P(S^i, n) \;=\; P(X^{(n)} = S^i)$. Im thermodynamischen Gleichgewicht ist $P(S^i, n)$ stationär. Fasst man die diskrete Variable n, die die Monte Carlo-Schritte zählt, als kontinuierlichen Zeitparameter auf, lässt sich die Bedingung durch die verschwindende Ableitung beschreiben. Gleichzeitig muss dieselbe Ableitung einer Kontinuitätsgleichung genügen, die dem Erhaltungssatz der totalen Wahrscheinlichkeit, $\sum_i P(S^{(i)}, n) \equiv 1$, entspricht. Dies führt auf folgende Voraussetzung, der jeder Monte Carlo-Algorithmus im thermodynamischen Gleichgewicht genügt.

$$0 \quad = \quad \frac{dP(S^{(i)}, n)}{dn} \qquad \Leftrightarrow \qquad 0 \quad = \quad \sum_j \Big[\; W_{ji}\, P(S^{(j)}, n) \;-\; W_{ij}\, P(S^{(i)}, n) \;\Big]$$

Die Gleichung wird bereits erfüllt, wenn jeder einzelne Summand verschwindet:

$$W_{ji}\ P(S^{(j)})\ =\ W_{ij}\ P(S^{(i)})$$

Auf dieser einfacheren Bedingung, die als "detailed balance" bezeichnet wird, basiert auch der Metropolis-Algorithmus [11], mit dem das Ising-Modell mit mobilen Defekten simuliert wird. Seine Übergangswahrscheinlichkeiten errechnen sich aus dem Energieunterschied, $\Delta E_{ij} = \mathcal{H}(S^{(j)}) - \mathcal{H}(S^{(i)})$, zwischen aufeinander folgenden Konfigurationen, S^i und S^j:

$$W_{ij} \quad = \quad \begin{cases} 1 & \Delta E_{ij} \leq 0 \\ e^{-\frac{\Delta E_{ij}}{k_B T}} & \text{sonst} \end{cases}$$

Die zufällige Modifikation einer Konfiguration erfolgt lokal, so dass die Energiedifferenz ebenfalls lokal und damit effizient bestimmt werden kann. Dazu ist in einer Umgebung die Änderung der Beiträge der Spin-Wechselwirkung, des Pinning-Potentials und des externen Feldes so auszuwerten, wie es der Hamiltonoperator des Systems (vgl. Gleichung 2) vorschreibt. Da die minimale Variante des Modells ausschließlich Defektbewegungen innerhalb der Ketten erlaubt, folgt ein einzelner Metropolis-Schritt dem folgenden Schema:

1. Ein Defekt $S_{i,j}$ und eine der beiden Richtungen innerhalb der Kette wird zufällig gewählt.
2. Der Defekt wird um einen Gitterplatz in der gewählten Richtung verschoben. Wird dabei der Mindestdefektabstand unterschritten, kehrt man mit der alten Konfiguration zurück zu Schritt 1.
3. Die alte Defektposition (i, j) wird mit einem Spin ersetzt, dessen Typ die Nebenbedingung intakter Spin-Cluster zwischen zwei Defekten respektiert.
4. Aus der Energiedifferenz zwischen der alten und der neuen Konfiguration ergibt sich die Übergangswahrscheinlichkeit, mit der die Defektbewegung akzeptiert wird. Wird die Konfigurationsänderung abgelehnt, kehrt man mit der alten Konfiguration zurück zu Schritt 1.

Dieses Schema wird iteriert, nach üblicherweise $1/5\, M \cdot N$ Metropolis-Schritten erfolgt die Berechnung thermodynamischer Größen für den jeweils aktuellen Status des Gitters. Dies garantiert geringe Korrelationen der ausgewerteten Konfigurationen, da zwischen je zwei aufeinander folgenden Auswertungen durchschnittlich zwei Bewegungen pro Defekt versucht werden. Erwartungswerte der Größen werden typischerweise über 10^5 bis 10^6 Konfigurationen gemittelt, so dass die Fehlerbalken der gezeigten Daten kleiner als die Symbolgrößen sind. Um Effekte der endlichen Systemgröße in den Simulationen weitestgehend zu vermeiden, werden stets periodische Randbedingungen angenommen, bei denen gegenüberliegende Ränder des Gitters miteinander wechselwirken.
Die Parallelisierung der Monte Carlo-Simulationen des Ising-Modells mit mobilen Defekten folgt verbreiteten Konzepten [12]. Auf Systemen mit gemeinsamem Speicher erfolgt eine geometrische Zerlegung des Spin-Gitters in m Streifen konstanter Größe, die auf die gleiche Anzahl Prozessoren verteilt werden. Die Grenzen der Gebiete verlaufen parallel zu den Spinketten, so dass durch die identische Zahl von Defekten pro Prozessor eine gleichmäßige Lastverteilung auch bei statischer Dekomposition des Systems garantiert wird. Jeder einzelne Prozessor führt unabhängig von den Nachbarn auf seinem Streifen Metropolis-Schritte durch. Da über die Gebietsgrenzen lediglich Nächste-Nachbar-Wechselwirkungen bestehen, reduziert sich der lesende Zugriff auf Gitterplätze angrenzender Streifen, der bei der Energieberechnung von Spins am Rand der Gebiete notwendig wird, auf ein Minimum. Vor der Auswertung einer Konfiguration werden die Prozessoren synchronisiert. Die Parallelisierung der Programmteile, die physikalische Größen berechnen, führt zu weiterer Lastverteilung und erhöht die Skalierbarkeit der Simulation. Die parallele Simulation des Modells erschließt auf aktuellen Hochleistungsrechnern Systemgrößen bis zu 640×640 Gitterplätzen, so dass genauere Extrapolationen im thermodynamischen Limes, $N, M \to \infty$ ermöglicht werden.

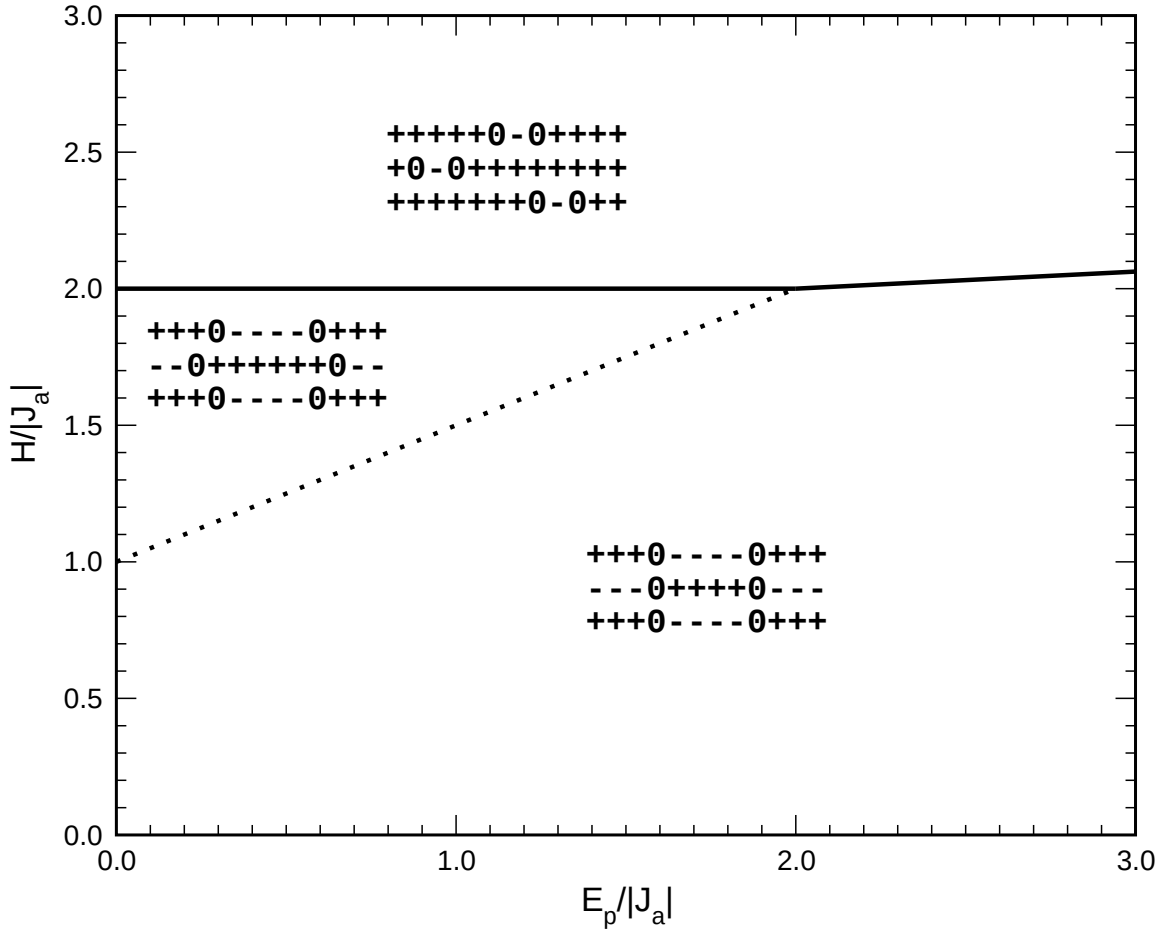
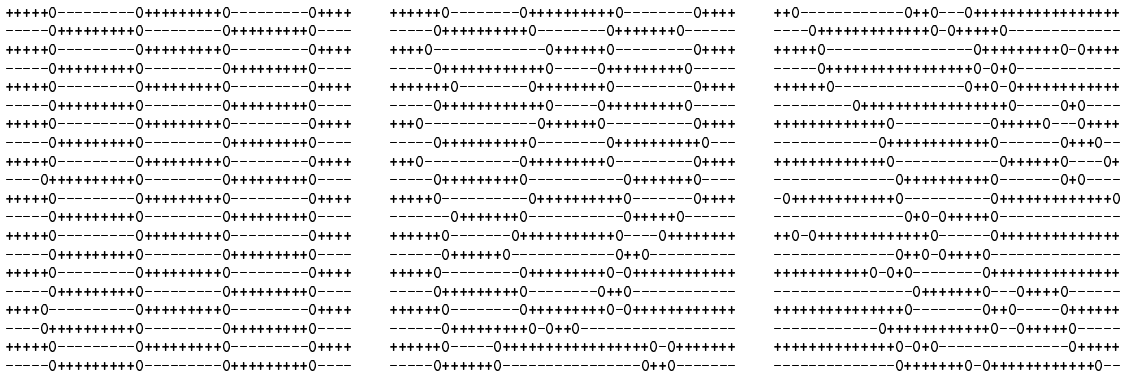


Abbildung 4: Grundzustände des Minimalmodells in Abhängigkeit von der Stärke des externen Feldes H und der des Pinning-Potentials E_p, das externe Feld favorisiert hier den 'Spin-Up'-Zustand.

Ergebnisse

Die minimale Version des Ising-Modells mit mobilen Defekten befindet sich je nach Stärke des Pinning-Potentials und des externen Feldes am Temperatur-Nullpunkt, $T = 0$, in einem der Grundzustände, die in Abbildung 4 skizziert sind. In schwachen externen Feldern bilden die Defekte gerade, eindimensionale Streifen. Ohne ein Pinning, $E_p = 0$, ist keine Position der Defektlinien energetisch ausgezeichnet, der Grundzustand ist entartet. Die Spin-Spin-Korrelationsfunktion in der Kettenrichtung, $G_1(r) = < S_{i,j} S_{i+r,j} >$, fällt dann exponentiell mit dem Abstand r ab, dieselbe Abhängigkeit gilt auch im Grenzwert hoher Temperaturen, $T \to \infty$, [6]. Verschwindet das Potential nicht, $E_p > 0$, pinnen die Defektstreifen im eindeutigen Grundzustand an die parallelen, äquidistanten Pinning-Linien. Dies führt zu langreichweitiger Ordnung: In den Ketten wechseln die ferromagnetischen Spin-Cluster an jedem Defekt das Vorzeichen, senkrecht dazu sind die perfekt antiferromagnetisch geordneten Domänen von eindimensionalen Defektstreifen begrenzt. Starke externe Felder, $H > \max\{2\,|J_a|, {}^{15}\!/_{8}\,|J_a| + {}^{1}\!/_{16}\,E_p\}$, zerstören im Grundzustand die Streifenstruktur. Je zwei Defekte ordnen sich in den Ketten im Mindestabstand an, wobei sie genau einen Spin einschließen, dessen Zustand nicht vom Feld favorisiert wird (vgl. Skizze in Abbildung 4). Die Defektpaare befinden sich in einer ferromagnetischen Struktur. Für moderates Pinning, $E_p < 2\,|J_a|$, existiert ein weiterer Grundzustand, in dem die Defekte in Zick-Zack-Linien ordnen (vgl. Abbildung 4).
Die Analyse der Grundzustände des Minimalmodells erweitern die Monte Carlo-Simulationen im Be-



(a) (b) (c)

Abbildung 5: Typische Monte Carlo-Konfigurationen im thermodynamischen Gleichgewicht für das Minimalmodell ohne externes Feld, $H = 0$, bei einem Pinning-Potential der Stärke $E_p = |J_a|$. Ausschnitte aus Systemen der Größe $M = N = 40$ bei den Temperaturen $k_B T/|J_a| = 0.8$ (a), 2.3 (b) und 2.9 (c).

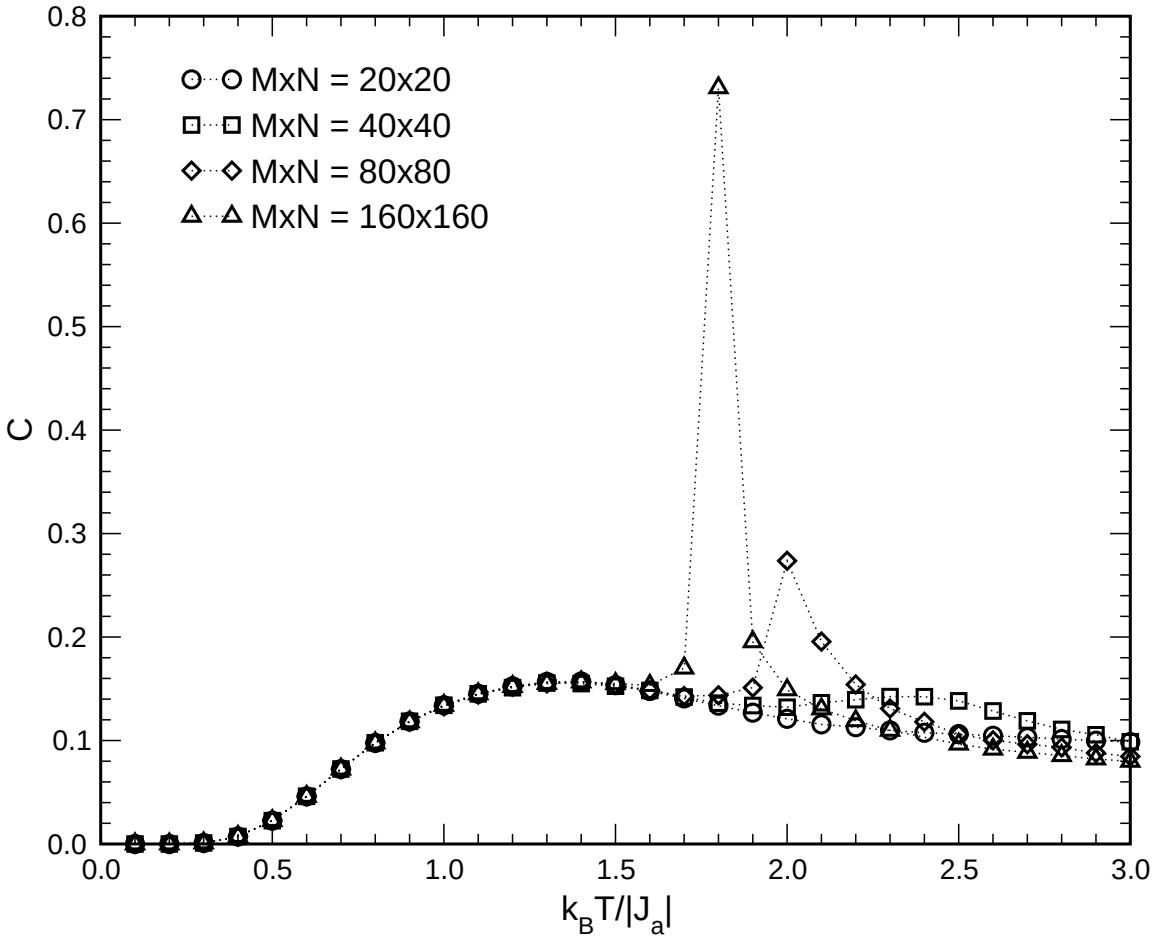


Abbildung 6: Spezifische Wärme C im Minimalmodell ohne externes Feld, $H = 0$, bei einem Pinning-Potential der Stärke $E_p = |J_a|$.

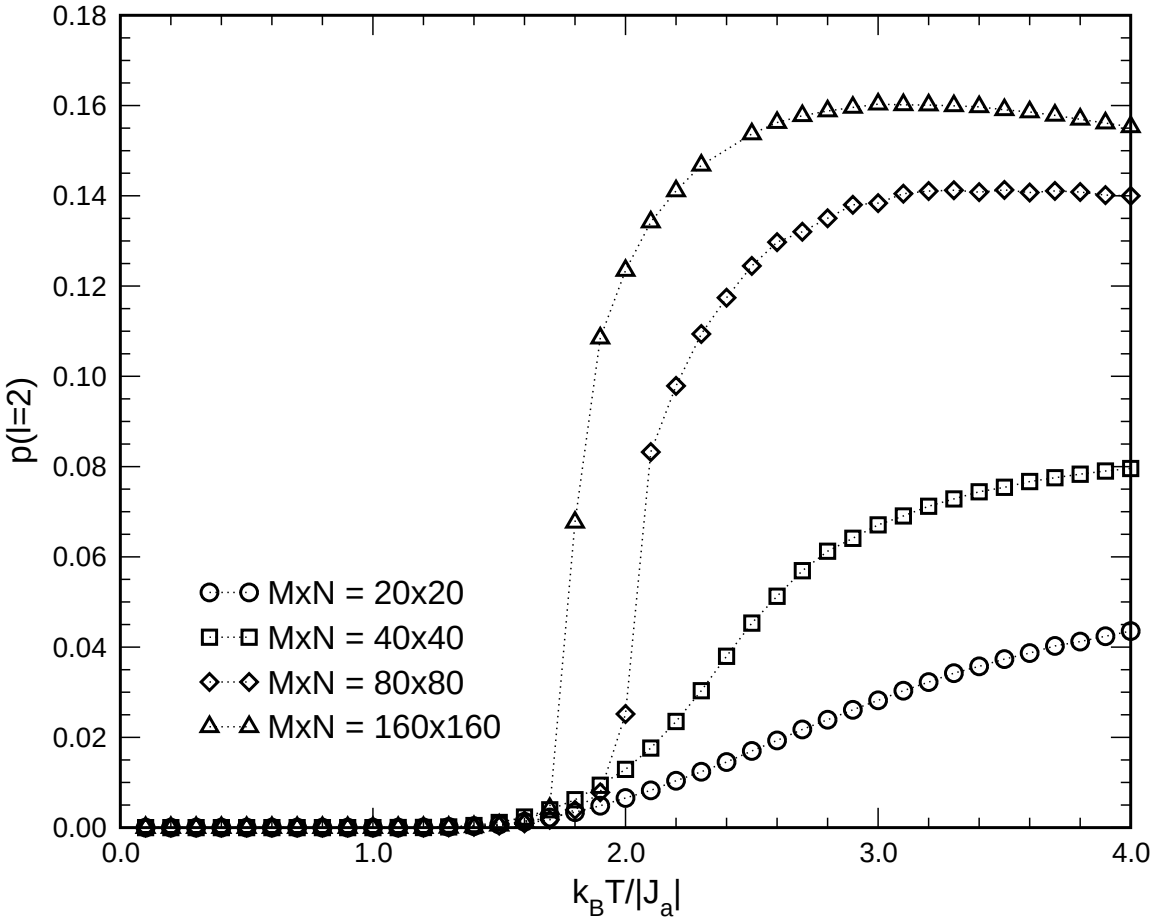


Abbildung 7: Defektpaar-Häufigkeit, $p(l = 2)$, im Minimalmodell ohne externes Feld, $H = 0$, bei einem Pinning-Potential der Stärke $E_p = |J_a|$.

reich nicht-verschwindender Temperaturen, $T > 0$. Bereits die Betrachtung typischer Konfigurationen im thermodynamischen Gleichgewicht, die der Algorithmus generiert, ermöglicht qualitative Aussagen über Modelleigenschaften. Das Tieftemperatur-Verhalten ist durch gerade Streifen charakterisiert, deren Ordnung von Auslenkungen einzelner Defekte gestört wird (vgl. Abbildung 5 a)). Die thermischen Anregungen können als Kinken bzw. Kink-Antikink-Paare aufgefasst werden [13]. Vergleichende Simulationen zeigen, dass das Minimalmodell bei tiefen Temperaturen auf Terassen-Stufen-Kinken-(TSK-)Modelle mit modifizierten Anregungsenergien abgebildet werden kann [6, 8]. Die Defektlinien entsprechen dabei den Stufen, die auf vizinalen Oberflächen fluktuieren. Die Annahme, dass benachbarte Defekte einen Mindestabstand von zwei Gitterkonstanten nicht unterschreiten dürfen, führt auf das ausführlich diskutierte Phänomen der entropischen Repulsion [14]: Die mäandernden Streifen neigen dazu, einen mittleren Abstand zu ihren direkten Nachbarn einzunehmen, der der inversen Defektkonzentration entspricht. Die Statistik des mittleren Abstandes benachbarter Defekte, die in den Monte Carlo-Simulationen gewonnen wird, weist dieses Verhalten nach [8]. Die Fluktuationen der Streifen verursachen ein erstes Maximum in der Kurve der spezifischen Wärme (vgl. Abbildung 6). Wie aus den Studien zu TSK-Modellen bekannt, ist dieses Maximum weitgehend systemgrößenunabhängig und zeigt keinen Phasenübergang an.

Wird die Temperatur weiter gesteigert, erhöht sich die Zahl der Kinkenanregungen und damit die Amplitude der Streifen-Fluktuationen, bis die Defektlinien aufreißen. In den typischen Konfigurationen bei hohen Temperaturen (vgl. Abbildung 5 c)) fällt der große Anteil jener Defekte auf, die sich mit einem Nachbarn innerhalb der Spinketten im Mindestabstand angeordnet, d. h. Paare gebildet haben. Die nor-

mierte Häufigkeit der Defektpaare,

$$p(l=2) \quad = \quad \frac{1}{2\vartheta NM} \sum_{j=1}^{N} \sum_{i=1}^{M} \left(1 - S_{i,j}^2\right)\left(1 - S_{i+2,j}^2\right) \qquad ,$$

markiert das Aufreißen der Defektstreifen mit einem steilen Anstieg (vgl. Abbildung 7). Ein mikroskopischer Mechanismus, der sowohl die Destabilisierung der Streifenstruktur als auch die Paarbildung erklären kann, beruht auf effektiv attraktiven Wechselwirkungen der Defekte. Starke Fluktuationen können zwei aufeinanderfolgende Defekte in einer Kette so positionieren, dass sie in den benachbarten Ketten von perfekten Spin-Clustern eingeschlossen werden, die aus Spins desselben Typs bestehen wie jene, die die beiden Defekte trennen:

```
------------          ++++++++++++
+++0----0+++   bzw.   ---0++++0---
------------          ++++++++++++
```

Durch die Wechselwirkungen zwischen den Ketten tendieren die beiden Defekte dazu, sich einander anzunähern und ein Paar zu bilden.
Die Destabilisierung der Defektstreifen manifestiert sich auch in anderen physikalischen Größen [6], z. B. zeigt die spezifische Wärme ein Maximum bei derselben Temperatur $T_{\max}$, bei der auch die Defektpaar-Häufigkeit stark ansteigt (vgl. Abbildung 6). Dieses zweite Maximum der spezifischen Wärme hängt von der Systemgröße, $M \times N$, ab, wobei in dieser Arbeit ausschließlich quadratische Systeme, $M = N$, betrachtet werden. Für größere Spingitter prägt es sich immer stärker aus, während sich seine Position $T_{\max}(M)$ zu tieferen Temperaturen verschiebt (vgl. Abbildung 6). Die Daten können als Zeichen eines Phasenübergangs interpretiert werden, bei dem das System von der geordneten Streifen- in eine ungeordnete Hochtemperatur-Phase wechselt. Um die kritische Temperatur, $T_c = T_{\max}(M = N = \infty)$ zu bestimmen, ist in Abbildung 8 die Position des zweiten Maximums der spezifischen Wärme, $T_{\max}(M)$, gegenüber der inversen Kettenlänge $1/M$ aufgetragen. Aus einer linearen Extrapolation ergeben sich in Abwesenheit des externen Feldes für verschiedene Pinning Stärken folgende Werte im thermodynamischen Limes:

E_p	T_c
$0.0\ \lvert J_a\rvert$	$1.02(05)\ \lvert J_a\rvert$
$0.5\ \lvert J_a\rvert$	$1.32(05)\ \lvert J_a\rvert$
$1.0\ \lvert J_a\rvert$	$1.54(10)\ \lvert J_a\rvert$
$2.0\ \lvert J_a\rvert$	$2.04(10)\ \lvert J_a\rvert$

Die weitere Analyse der Korrelationsfunktionen und -längen im Bereich der Übergangstemperatur T_c identifiziert einen Phasenübergang erster Ordnung.
Die Monte Carlo-Simulationen im externen Feld, $H > 0$, vervollständigen die Studie des Ising-Modells mit mobilen Defekten. Ein wichtiger Aspekt ist das Auftreten von Zick-Zack-Strukturen, wie sie bereits die Grundzustands-Analyse feststellt (vgl. Abbildung 4). Die Strukturen am Temperatur-Nullpunkt besitzen jeweils unterschiedliche Magnetisierungen

$$< M > \quad = \quad \frac{1}{NM} \sum_{j=1}^{M} \sum_{i=1}^{N} S_{i,j} \quad .$$

Die thermodynamische Größe ändert sich demnach sprunghaft, wenn durch Variation des externen Feldes oder des Pinning-Potentials das System zwischen zwei Grundzuständen wechselt. Für den Übergang zwischen Streifen- und Zick-Zack-Struktur verschwindet dieses Verhalten bereits bei kleinen Temperaturen, $T > 0$, die Magnetisierung ändert sich dann stetig. Andere Observable des Systems zeigen ebenfalls kein unstetiges oder singuläres Verhalten, so dass kein zusätzlicher Phasenübergang nachgewiesen werden kann. Die in Abbildung 4 gestrichelt dargestellte Phasengrenze setzt sich nicht in den

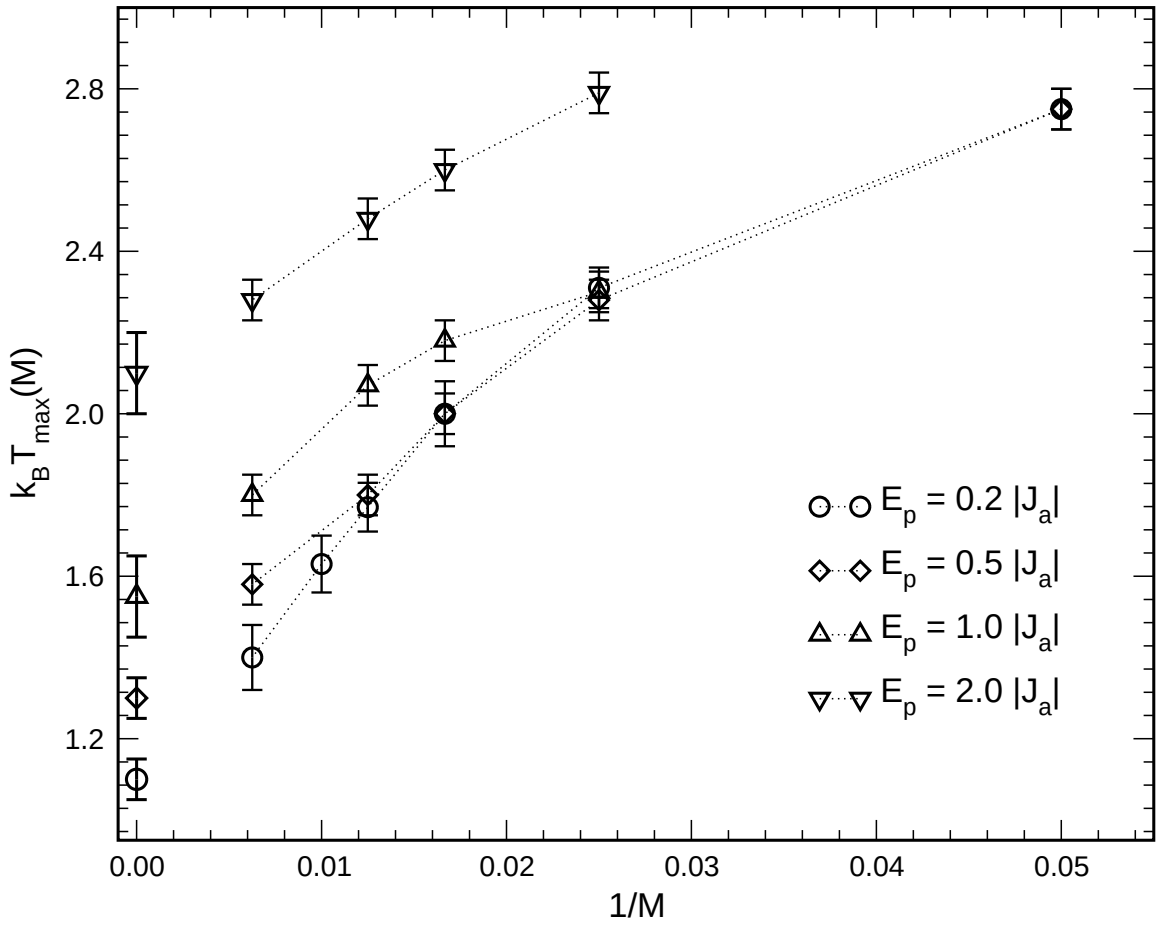


Abbildung 8: Extrapolation der Position des zweiten Maximums der spezifischen Wärme $T_{\max}$ in Abhängigkeit von der inversen Systemlänge $1/M$ für verschiedene Pinning-Stärken E_p im Minimalmodell ohne externes Feld, $H = 0$.

Bereich nicht-verschwindender Temperaturen fort.

Der Übergang von der magnetisch geordneten Streifen- in die Hochtemperatur-Phase bleibt in moderaten Feldern bestehen. Oberhalb einer kritischen Feldstärke, $H = 2\,|J_a|$, charakterisieren Defektpaare die Systemeigenschaften auch bei tiefen Temperaturen (vgl. Abbildung 4), die Streifenphase verschwindet genau wie der Phasenübergang. Analog zur Auswertung der feldfreien Simulationen werden die Daten bei verschiedenen Stärken des Pinning-Potentials und des externen Feldes extrapoliert, um die kritische Temperatur T_c im thermodynamischen Limes zu bestimmen. Stärkere Felder verschieben den Übergang zu kleineren Temperaturen, es ergibt sich das in Abbildung 9 skizzierte Phasendiagramm in der H-T-Ebene.

Ausblick

Die Monte Carlo-Simulationen zeichnen zusammen mit der Grundzustandsanalyse, der Methode der 'Freien Fermionen' [6] und Transfermatrixrechnungen [7] ein umfassendes Bild der minimalen Variante des Ising-Modells mit mobilen Defekten. Das für das Modell fundamentale Wechselspiel zwischen Defekten und Spins scheint in den genannten Kupratmagneten die wesentliche Physik zu bestimmen. Simulationen des Modells, die alle drei Kopplungen berücksichtigen, sind Gegenstand aktueller Arbeiten [8]. Eine substantielle Erweiterung des Modells durch abweichende Gittergeometrien, unterschiedliche

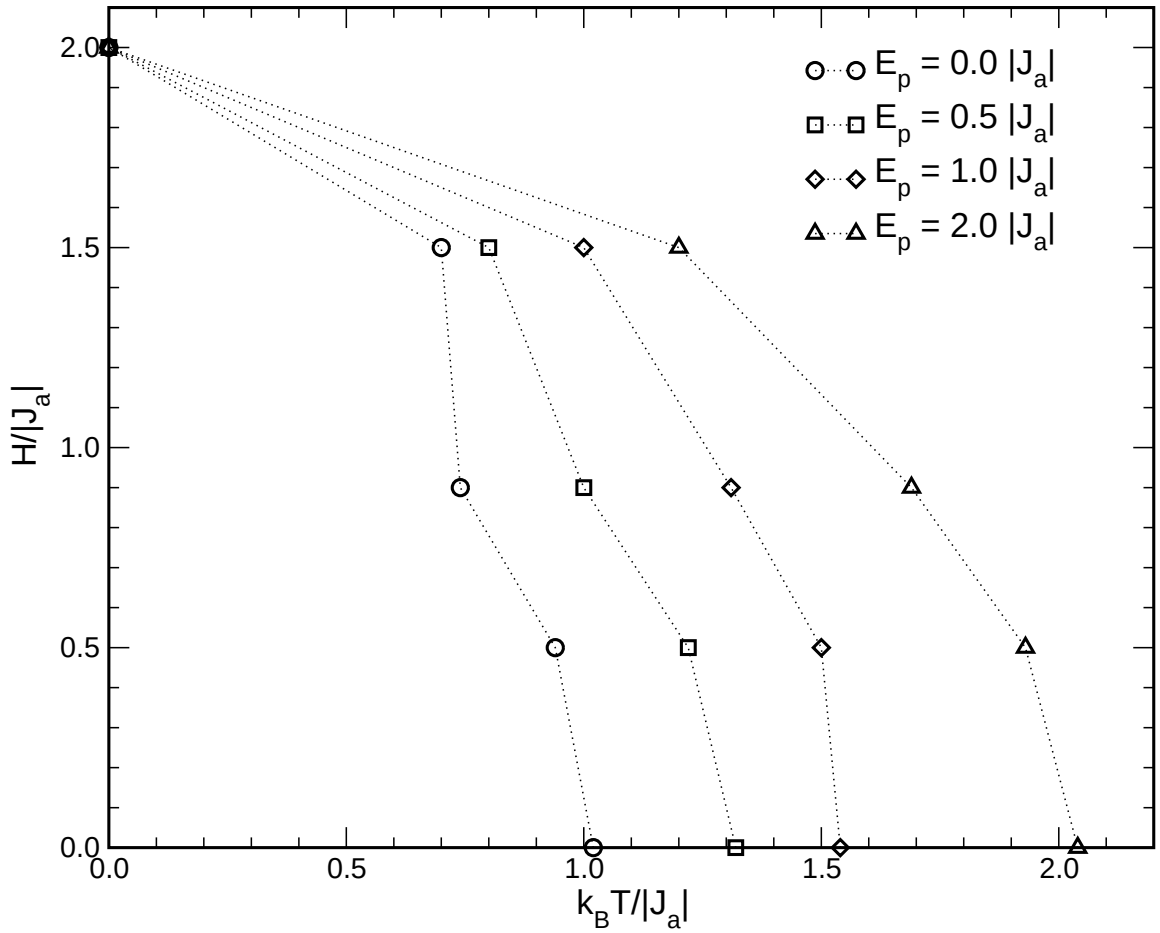


Abbildung 9: Phasengrenzen des Minimalmodell im thermodynamischen Limes, $M, N \to \infty$ für verschieden starke Pinning-Potentiale in Abhängigkeit von der Temperatur T und der Stärke des externen Feldes H

Defektpotentiale und Wechselwirkungen längerer Reichweite könnte klären, welche Charakteristika robust gegenüber Variationen des Modells sind bzw. welche zusätzlichen Phänomene auftreten können. Weiterhin könnten Modifikationen einen direkteren Vergleich der Simulationsergebnisse mit den experimentellen Daten zu “Telefonnummern“-Verbindungen ermöglichen.

Danksagung

Die präsentierten Ergebnisse entstanden im Rahmen des Gaststudentenprogramms “Wissenschaftliches Rechnen“, das das Zentralinstitut für Angewandte Mathematik im Forschungszentrum Jülich ausrichtet. Herrn Dr. Esser, der mir die Teilnahme an der Veranstaltung ermöglicht hat, möchte ich ebenso herzlich danken wie Herrn PD Dr. Baumgärtner, der meine Arbeit am Institut für Festkörperforschung betreut hat. Besonderer Dank gilt Herrn Prof. Dr. Selke, der meine Diplomarbeit begleitet und die Untersuchungen der Ising-Modelle mit mobilen Defekten fördert. Die Simulationen sind u. a. auf den Parallelrechnern des John von Neumann-Institutes für Computing und des Rechen- und Kommunikationszentrums der Rheinisch-Westfälischen Technischen Hochschule Aachen ausgeführt worden.

Literatur

1. U. Ammerahl, B. Büchner, C. Kerpen, R. Gross, A. Revcolevschi, Phys. Rev. B 62, R3592 (2000)
2. A. Goukasov, U. Ammerahl, B. Büchner, R. Klingeler, T. Kroll, A. Revcolevschi, in Vorbereitung
3. R. Klingeler, Doktorarbeit, RWTH Aachen (2003)
4. F. C. Zhang, T. M. Rice, Phys. Rev. B 37, 3759 (1988)
5. T. Kroll, Diplomarbeit, RWTH Aachen (2003)
6. W. Selke, V. L. Pokrovsky, B. Büchner, T. Kroll, Eur. Phys. J. B 30, 83 (2002)
7. M. Holtschneider, W. Selke, Phys. Rev. E 68, 026120-6 (2003)
8. M. Holtschneider, Diplomarbeit, in Vorbereitung
9. D. P. Landau, K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics (Cambridge University Press, 2000)
10. K. Binder, D. W. Heermann, The Monte Carlo Method in Statistical Physics, 4th edition (Springer Verlag, 2002)
11. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, J. Chem. Phys. 21, 1087 (1953)
12. D. W. Heermann, A. N. Burkitt, Parallel Algorithms in Computational Science, (Springer Verlag, 1992)
13. J. Villain, D. R. Grempel, J. Lapujoulade, J. Phys. F 15, 809 (1985)
14. B. N. J. Persson, Surf. Sci. Rep. 15, 1 (1992)

Lanczos Verfahren zur Berechnung innerer Eigenwerte von Anderson Matrizen

Christian Kahl

Bergische Universität,
Gaußstrasse 20,
42097 Wuppertal

E-mail: ckahl@wmpi01.math.uni-wuppertal.de

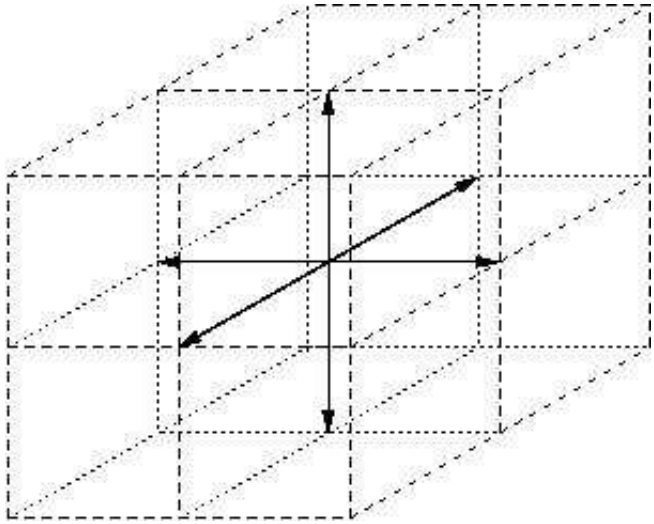
Zusammenfassung: Dieser Artikel behandelt die Berechnung innerer Eigenwerte großer, dünn besetzter Matrizen, insbesondere der aus der theoretischen Festkörperphysik stammenden Anderson Matrizen. Dabei werden verschiedene Lanczos Methoden miteinander verglichen, insbesondere in Hinsicht auf ihre Konvergenzgeschwindigkeit gegenüber dem Algorithmus von *Cullum/Willoughby*. Es wird mit einem Restarted-Lanczos Verfahren gearbeitet um insbesondere das Verfahren der Residuuenminimierung möglich zu machen, wie es bereits in Verbindung mit einem Jacobi-Davidson Verfahren verwendet wurde.

Einleitung

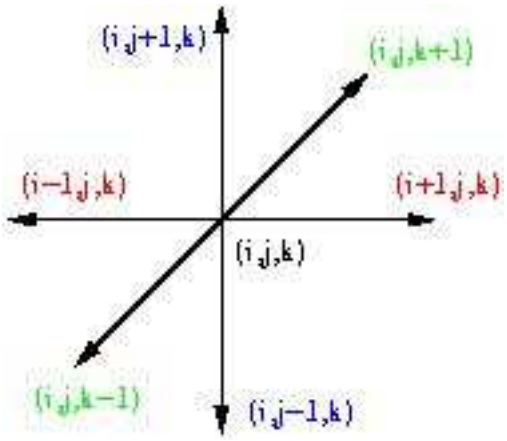
Das Problem der Berechnung innerer Eigenwerte ist ein Problem der numerischen linearen Algebra, welches generell einen sehr direkten Bezug zur Praxis besitzt. Dabei stammen die Aufgaben aus verschiedensten wissenschaftlichen Bereichen. Speziell sollen in diesem Artikel nun die Anderson Matrizen untersucht werden. Dabei ist das wesentliche Problem, dass durch die zufallsverteilten Einträge, sich das Spektrum der Matrix sowohl bezüglich der Lage als durch die innere Struktur sehr unvorteilhaft gestaltet.
Generell kann man die Probleme der Eigenwerteberechnung grob dadurch unterscheiden, ob man an allen Eigenwerten der Matrix interessiert ist, oder ob man Eigenwerte in einem bestimmten Intervall bestimmen möchte. Um einen geeigneten Algorithmus auszuwählen, ist es weiterhin von großer Bedeutung wie die Matrix strukturiert ist. Dabei entstehen bei dem Anderson Modell symmetrische Matrizen mit nur 7 Einträgen pro Zeile, wobei der Diagonaleintrag eine gleichverteilte Zufallsvariable ist. In diesem Artikel wird primär versucht, mittels einer Ritzprojektion auf einen Krylov Unterraum die Eigenwerte zu bestimmen, wie es auch in dem zur Zeit für dieses Problem am besten funktionierenden Algorithmus von Cullum/Willoughby geschieht. Dieses Verfahren ist das sogenannte Lanczos Verfahren, die Konvergenzgeschwindigkeit des Ritzverfahrens soll durch eine Residuuenminimierung beschleunigt werden, dazu wird mit einem Restarted Lanczos Verfahren gearbeitet. Das Verfahren der Residuuenminimierung ist insbesondere im Zusammenhang mit dem Jacobi-Davidson Verfahren [1, 6] bereits erfolgreich eingesetzt worden, der wesentliche Aspekt ist jedoch, dass eine Näherungsinverse gebildet werden kann, um eine geeignete Suchraumerweiterung durchzuführen, und für die Anderson Matrizen ist auf Grund der besonderen Struktur noch keine approximative Inverse gefunden worden.
Leider werden die numerischen Experimenten zeigen, dass die Residuuenminimierung, gepaart mit einem Lanczos-Verfahren und harmonischer Ritzprojektion, auch nicht den durchschlagenden Erfolg hinsichtlich der Konvergenzgeschwindigkeit bringt, was insbesondere auf Instabilitäten beim Restart zurückzuführen ist.

Anderson Matrizen

Das Problem der Anderson Matrizen stammt aus der theoretischen Physik, genauer gesagt dem Teil der Quantenphysik. Das quantenmechanische Problem wird durch einen Hamilton Operator A beschrieben und die quantenmechanischen Wellenfunktionen werden durch die Eigenvektoren der Matrix gegeben. Zur Modellbildung geht man von einem kubischen Gitter der Größe $M = N \times N \times N$ mit periodischen Randbedingungen aus



Die Matrix A entsteht nun dadurch, dass die Korrespondenz eines Gitterpunktes zu seinen direkten Nachbarn die Wahrscheinlichkeit darstellt, mit der ein Elektron sich in dem Gitter bewegen kann. Der Einfachheit halber wird in der Diskretisierung die nächste Nachbar Kopplung auf 1 gesetzt, s.d. nur die Diagonaleinträge von A variieren. Wenn man die Kartesischen Koordinaten mit i, j, k bezeichnet, so bekommt man folgenden 7-Punkte Stern

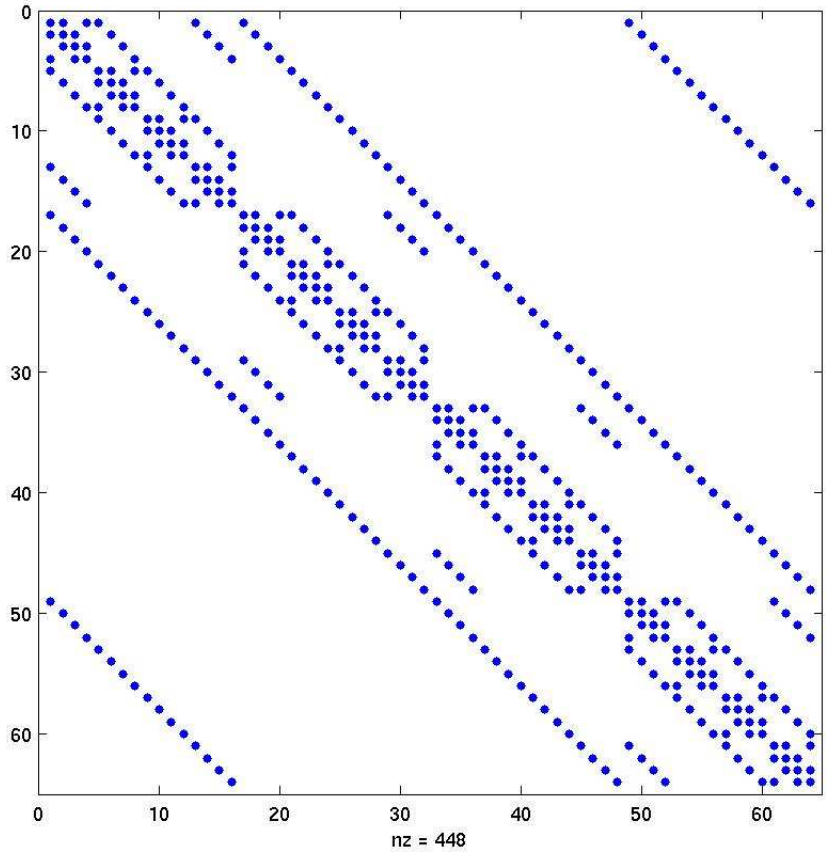


aus dem sich folgende charakteristische Gleichung ergibt

$$x_{i-1,j,k}+x_{i,j-1,k}+x_{i,j,k-1}+x_{i+1,j,k}+x_{i,j+1,k}+x_{i,j,k+1}black+\epsilon_{i,j,k}x_{i,j,k} = \lambda x_{i,j,k} \qquad (1)$$

dabei ist $\epsilon_{i,j,k}$ eine gleichverteilte Zufallsvariable in dem Intervall $[-\omega/2, \omega/2]$. Offensichtlich werden dadurch die Eigenwerte nach dem Theorem von Gerschgorin [4] in dem Intervall $[-\omega/2 - 6, \omega/2 + 6]$

beschränkt. Das Verhalten der Eigenvektoren hängt in prägnanter Weise von der Wahl von ω ab, so sind die Eigenvektoren für $w \ll 16.5$ ausgedehnt, dagegen hat man es für $w \gg 16.5$, auf Grund der dominierenden Diagonalen, mit lokalen Eigenvektoren zu tun. Für eine genauere physikalische Beschreibung sei an dieser Stelle auf [3] verwiesen. Dabei ist der Bereich um 16.5 der Interessanteste insofern, als dass der Übergang von lokalen zu ausgedehnten Eigenvektoren gerade den Wechsel eines Leiters (ausgedehnte Eigenvektoren) zu einem Nichtleiter (lokale Eigenvektoren) beschreibt.
Da man bei dem Modell der Anderson Matrizen mit zyklischen Randbedingungen arbeitet, erhält man folgendes Besetzungsmuster der Matrix



Mathematik

Es sollen an dieser Stelle sowohl die mathematischen Grundlagen gelegt werden, als auch insbesondere auf die Probleme bei der Berechnung von inneren Eigenwerten der Anderson Matrix hingearbeitet werden. Dabei wird dementsprechend das Verfahren von Cullum/Willoughby vorgestellt, welches sich zum derzeitigen Stand der Forschung als das schnellste erweist, als auch der Versuch das Problem mittels eines Lanczos Verfahrens mit Residuuenminimierung zu lösen, besprochen. Einleitend werden nun ein paar grundlegende Begriffe aus der numerischen linearen Algebra definiert.

Eigenwertberechnung und Krylov Unterräume

Definition 1. *Sei $A \in \mathbb{R}^{n\times n}$, dann heißt ein Paar (λ_i, v_i) mit $\lambda_i \in \mathbb{R}$ und $v_i \in \mathbb{R}^n$ Eigenwert bzw. Eigenvektor von A, falls folgendes gilt*

$$Av_i = \lambda_i v_i \tag{2}$$

mit dem Spektrum einer Matrix bezeichnet man die Menge aller Eigenwerte

$$spec(A) = \{\lambda_1, \lambda_2 \ldots, \lambda_n\} \tag{3}$$

Da es sich bei Anderson Matrizen um symmetrische Matrizen handelt ist folgender Satz hilfreich

Satz 2. *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch dann sind alle Eigenwerte reell*

Beweis: siehe [4]

Damit ist man bei Anderson schon mal in der angenehmen Lage, ausschließlich reelle Eigenwerte zu suchen. Da bei großen, dünn besetzten Matrizen ein direkter Eigenlöser viel zu aufwendig wäre, löst man ein solches Eigenwertproblem iterativ. Die Idee ist es, das Ausgangsproblem durch eine orthogonale Projektion zu vereinfachen, dazu verwendet man geeigneter Weise als Basis des Projektionsraumes einen Krylov Unterraum, welchen man mit Hilfe eines Lanczos-Verfahrens orthogonalisiert. Generell wird das Verfahren der orthogonalen Projektion auch *Ritzprojektion* genannt.

Definition 3 (Krylov Unterraum). *Sei $A \in \mathbb{R}^{n\times n}$ und $v \in \mathbb{R}^n$ dann ist der* Krylov-Unterraum *der Stufe* m *definiert durch*

$$K_m(A, v) := span\{v, Av, A^2v, \ldots, A^{m-1}v\} \tag{4}$$

Das bedeutet, dass jede Erweiterung des Krylov Unterraumes genau eine Matrix-Vektor Multiplikation kostet. Nun ist man an einer passenden Orthonormalbasis interessiert, hierzu bietet sich das wohlbekannte Verfahren von Gram-Schmidt [4] an, dabei entsteht sukzessive eine Orthonormalbasis

$$V_m := span\{K_m(A, v)\}$$

Definition 4 (Ritzwert). *Sei $V \in \mathbb{R}^{n\times m}$ eine beliebige Orthonormalbasis, dann heißt ein Eigenpaar (λ, v) von $H = V^TAV$*

$$Hv = \lambda v \tag{5}$$

Ritzpaar *von A bezüglich V*

Bemerkung 5. *Äquivalent zu obiger Definition ist die sogenannte* Galerkin *Bedingung*

$$Av - \lambda v \perp V \Leftrightarrow w^T(Av - \lambda v) = 0\,, \forall w \in V \tag{6}$$

Da man an den Eigenwerten der Matrix H interessiert ist, kann man den folgenden Sachverhalt ausnutzen, dass bei der Orthogonalisierung nach Gram-Schmidt die Reorthogonalisierungskoeffizienten genau die Matrix H bilden. Zusätzlich ist zu beachten, dass auf Grund der Symmetrie von A das Reorthogonalisieren zu einer Drei-Term-Rekursion degeneriert

$$\beta_i v_i = Av_{i-1} - \alpha_{i-1}v_{i-1} - \beta_{i-1}v_{i-2} \tag{7}$$

damit erhält man auch schon das sogenannte *Lanczos-Verfahren*

$v_1 = \frac{v_1}{\|v_1\|},\ v_0 = 0,\ h_{1,0} = 0$
for $i = 2, 3, \ldots$ **do**
  $v_i = Av_{i-1}$
  $v_i = v_i - h_{i-1,i-2}v_{i-2}$
  $h_{i-1,i-1} =< v_i, v_{i-1} >$
  $v_i = v_i - h_{i-1,i-1}v_{i-1}$
  $h_{i,i-1} = \|v_i\|$
  $v_i = \frac{1}{h_{i,i-1}} \cdot v_i$
end for

Bemerkung 6. *Da man direkt die Matrix H erzeugt, dient die Basis V_m praktisch nur der Rückprojektion der Ritzvektoren. Sei nämlich (λ, u) ein Ritzpaar, dann ist die zugehörige Eigenpaarapproximation (λ, v) gegeben durch*

$$v = V_m u \tag{8}$$

Definition 7 (Harmonisches Ritzpaar). *Sei $A \in \mathbb{R}^{n\times n}$ und V eine Orthogonalbasis und $W = AV$, dann ist die Eigenlösung (λ, v) von*

$$\frac{1}{\lambda}V^T A^T AVy = V^T A^T Vy \tag{9}$$

harmonisches Ritzpaar *von A und V*

Bemerkung 8. *Falls $W = AV$ ein Orthogonalsystem ist wird (9) zu*

$$\frac{1}{\lambda}y = V^T A^T Vy = W^T A^{-1} W \tag{10}$$

also approximiert ein harmonisches Ritzpaar ein Eigenpaar von A^{-1}. Nachdem jetzt die wesentlichen mathematischen Grundlagen gelegt wurden, können nun die auf dem Lanczos Prozess basierenden Verfahren vorgestellt werden.

Lanczos-Verfahren

Ausgehend von dem Lanczos Algorithmus hat man verschiedene Möglichkeiten, um die gesuchten Eigenwerte zu ermitteln, dabei wird insbesondere von der günstigen Struktur von H Gebrauch gemacht, denn durch Lanczos Verfahren ist H eine Tridiagonalmatrix. Da eine Tridiagonalmatrix eine sogenannte *Sturmsche Kette* bildet, hat dies den Vorteil, dass ein Eigenwert mit einem Aufwand von $O(N)$, N die Dimension von H, zu berechnen ist. Genau diesen Sachverhalt nutzt der Algorithmus von *Cullum/Willoughby* aus

Bemerkung 9 (Cullum/Willoughby). *Sei M die Dimension der Anderson Matrix A, dann erzeugt der Algorithmus von Cullum und Willoughby mittels des Lanczos Prozesses für dieses Problem eine projizierte Matrix H der Dimension $N \approx 4M$. Die Ritzwerte der Matrix H, sind dann sehr gute Näherungen der gesuchten Eigenwerte. Da offensichtlich einige Eigenwerte mehrfach approximiert werden, ist noch eine geeignete Auswahl notwendig, um sogenannte "spurios values" herauszufiltern.*

Der wesentliche Vorteil dieses Verfahrens ist, dass man auf Grund der Tatsache, dass man prinzipiell 4 mal so viele Ritzwerte wie Eigenwerte hat, es sehr wahrscheinlich ist, dass auch im Inneren des Spektrums ein Großteil der Eigenwerten approximiert wird. Dies wird auch durch numerische Experimente bestätigt und man bekommt zu meist genügend interessante Eigenpaare.

Der Nachteil ist allerdings, dass man nicht die Möglichkeit hat das Verfahren zu steuern, um Eigenwerte ganz gezielt in einem ausgesuchten Intervall zu approximieren. Deshalb liegt es auf der Hand hierfür ein *Restarted-Lanczos* Prozess zu versuchen; dabei nutzt man aus, dass man bei dem Lanczos Verfahren einen Startvektor wählen kann.

wähle Starvektor v_0
for $i = 1, \ldots$ **do**
  for $j = 0, \ldots,$ *Inner* **do**
    erweitere v_j zu V nach (7)
  end for
  berechne die Ritzwerte von H
  berechne die Residuuen und Eigenpaar Approximationen
  wähle neuen Startvektor v_{j+1}
end for

Residuuenminimierung

Um die Konvergenzeigenschaften des Restarted Lanczos Algorithmus zu verbessern, war nun die Idee, anstelle der Berechnung der Ritzwerte nach Aufbau des Krylov Unterraums eine Residuuenminimierung durchzuführen. Der wesentliche Aspekt, der sich bei den Anderson Matrizen besonders auszeichnet, ist die Tatsache, dass der Aufbau des Krylov Unterraums auf Grund der Dünnbesetztheit der Matrix, sehr günstig ist.

Um nun die Residuuenminimierung im Zusammenhang mit einem Krylov Unterraum Verfahren zu konkretisieren, wird ausgehend von einer Eigenpaarapproximation (λ, v) ein Vektor $w \in V$ nach folgender Bedingung gesucht

$$||Aw - < w, Aw > w||_2 = \min_{v \in V, ||v||_2 = 1} ||Av - < v, Av > v||_2 \tag{11}$$

Dabei wählt man folgenden geeigneten Ansatz

$$w = v + V\alpha \tag{12}$$

Wenn man nun noch den sogenannten *Rayleigh-Quotienten* $\frac{<w,Aw>}{||w||_2}$ durch λ ersetzt, kann man mit (12), α als Lösung folgender Gleichung bestimmen

$$C\alpha = d \tag{13}$$

wobei die Unbekannten sich wie folgt zusammensetzen

$$\begin{aligned} C &= ((A - \lambda I)V)^T((A - \lambda I)V) \\ d &= -((A - \lambda I)V)^T r \\ r &= Av - \lambda v \end{aligned}$$

Numerische Test werden aber zeigen, dass ein Harmonisches Ritz-Verfahren basierend auf dem Lanczos Algorithmus, gepaart mit einer Residuuenminimierung nicht *monoton* konvergiert und darüber hinaus zu Instabilitäten neigt.

Spektralverschiebung

Ein weiterer Ansatz, der sich im Zusammenhang mit einem Krylov Unterraumverfahren anbietet, ist eine s.g. Spektralverschiebung. Dabei löst man prinzipiell folgendes Eigenwertproblem

$$f(A)v = f(\lambda)v \tag{14}$$

mit einer geeigneten Funktion f. Es bieten sich zwei verschiedene Varianten an

1. Shift-and-Invert
2. Polynombeschleunigung

Shift-and-Invert

Bei dieser Art der Konvergenzbeschleunigung wird die Funktion f wie folgt gewählt

$$f(x) = (x - s)^{-1} \tag{15}$$

wenn man s in der Nähe der gesuchten Eigenwerte wählt, so werden diese Eigenwerte durch das invertieren an den Rand des Spektrums abgebildet. Da bei dem Anderson Modell primär Eigenwerte nahe 0 relevant sind, könnte man mit $f(x) = \frac{1}{x}$ arbeiten, damit stellt sich allerdings das Problem, die Matrix zu invertieren oder zumindestens während der Iteration immer ein Gleichungssystem zur Suchraumerweiterung lösen zu müssen. Dies erweist sich als äußerst ineffektiv, da für die Anderson Matrizen keine geeignete approximative Inverse bekannt ist und somit eine solche Spektralverschiebung als Konvergenzbeschleunigung ausscheidet.

Polynombeschleunigung

Bei der Polynombeschleunigung wird das Spektrum der Matrix mittels eines geeigneten Polynoms verschoben, auf Grund der speziellen Lage des Spektrums von Anderson Matrizen, bietet sich insbesondere folgende Spektralverschiebung an

$$f(x) = x^2 \tag{16}$$

Damit werden die Eigenwerte, welche am nächsten bei Null sind, zwar nicht zu den größten Eigenwerten, wie bei der Shift-and-Invert Technik, allerdings liegen diese Eigenwerte nun am Rande des Spektrums und das Lanczos Verfahren tendiert dazu gegen diese äußeren Eigenwerte zu konvergieren [6].

Generell bietet sich eine Polynombeschleunigung an, welche das Spektrum nicht nur in günstiger Weise bezüglich der Lage verändert, sondern zusätzlich auch noch den Abstand der gesuchten Eigenwerte von einander erhöht, um damit eine bessere Konvergenz zu erzielen.

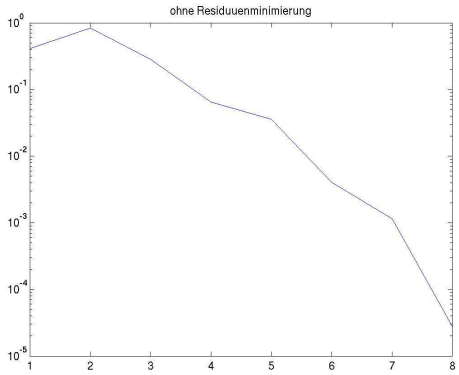
Numerische Test

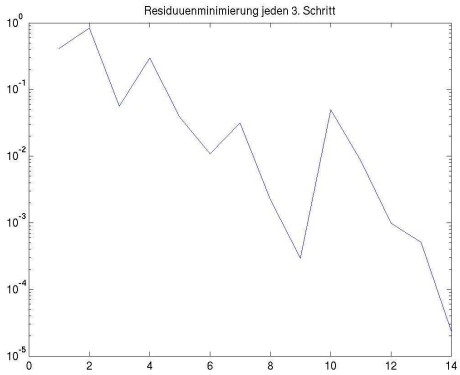
An dieser Stelle sollen nun einige numerische Resultate präsentiert werden, wobei ein Schwerpunkt der Betrachtung, die Gegenüberstellung der getesteten Verfahren mit dem Algorithmus von *Cullum/Willoughby* ist.

Harmonisches Ritzverfahren mit Residuuenminimierung

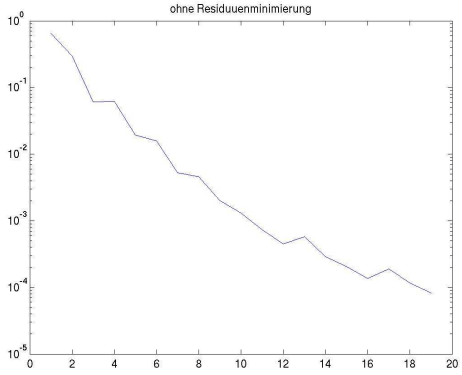
Es wurde ein Restarted-Lanczos zum Aufbau des Krylov Unterraumes verwendet, um eine harmonische Ritzprojektion (9) durchzuführen, dabei wurde zusätzlich in gewissen Abständen die Suchraumerweiterung dazu genutzt eine Residuuenminimierung zu machen.

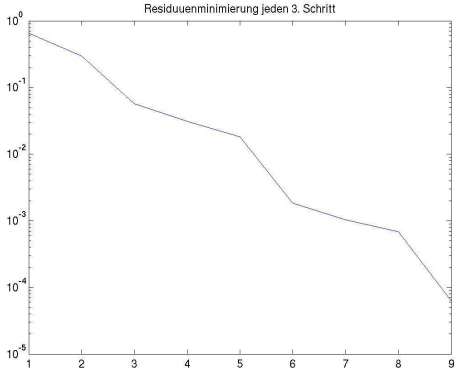
Als ein wesentliches Problem stellte sich die Instabilität des Verfahrens heraus, so waren numerische Test nur begrenzt aussagekräftig, da dass das Konvergenzverhalten mehr oder minder chaotisch ausfiel.





So zeigt der linke Konvergenzverlauf das Harmonische Ritzverfahren ohne Residuuenminimierung, während bei dem rechten alle 3 Schritte eine Residuuenminimierung durchgeführt wurde. Dabei wird das wesentliche Problem sehr gut dargestellt, die Residuuenminimierung schafft zwar immer wieder eine deutliche Verbesserung, allerdings eignet sich der Vektor, dessen Residuum minimiert wurde, im allgemeinen nicht als Ausgangspunkt für den Restart. Man könnte nun vermuten, dass die Residuuenminimierung an sich zu keiner Konvergenzbeschleunigung beitragen kann, dafür nun folgendes Gegenbeispiel





Nun schafft die Residuuenminimierung genau den erwünschten Effekt und insbesondere fällt der neue Restart Vektor während der gesamten Iteration sehr günstig aus.

Bemerkung 10. *Es ist vollkommen klar, dass diese numerischen Resultate kein Beweis für die Instabilitäten der Residuuenminimierung sind, nur sollen an Hand von Beispielen die Probleme hinsichtlich der Unvorhersagbarkeit des Konvergenzverlaufes illustriert werden.*

Spektralverschiebung

Implementiert wurde eine Spektralverschiebung, welche die Polynombeschleunigung mit $f(x) = x^2$ ausnutzt, dabei wurde auch hier ein Restarted-Lanczos Verfahren verwendet. Die wesentlichen Aspekte, unter denen die Konvergenzgeschwindigkeit hier betrachtet werden soll sind

- Matrix-Vektor-Multiplikationen
- Vektor-Vektor-Multiplikationen

Es sind folgende Iterationszahlen für $\omega = 16$ und eps $= 10^{-4}$ ermittelt worden

N	innere It.	äußerer It.	MV-mult.	VV-mult.
5	125	2.9	**725**	1086
	64	11.7	1497	2247
	32	59.7	3701	5552
8	512	3.1	**3174**	4721
	256	20.7	10598	15897
	128	92.6	23706	35559
10	1000	2.6	**5200**	7800
	500	11.4	11400	17100
12	1728	3.3	**11405**	17108
	864	11.8	20390	30585
15	3375	2	**13500**	20250
	1687	5.2	17544	26316

An dieser Stelle lässt sich schon eine deutliche Tendenz feststellen und zwar neigt der Algorithmus dazu, dass mit abnehmender Zahl an inneren Iterationen, der Gesamtaufwand für den Algorithmus immer größer wird. Damit wird auch das Prinzip des Restarted Lanczos insofern uninteressanter, als dass eine große Zahl innerer Iterationen, genau dem Schema des *Cullum/Willoughby* Algorithmus entspricht. Diese Beziehung bzgl. inner bzw. äußerer Iterationen ist ein weiterer Grund, um das Verfahren der Residuuenminimierung basierend auf einem Krylov Unterraum Verfahren auszuschließen, denn nur bei niedriger innerer Iterationszahl kommt die Konvergenzbeschleunigung vollendens zum Zuge.

CW vs. A^TA

Um einen Eindruck davon zu gewinnen, mit welchem Aufwand der Algorithmus von Cullum/Willoughby arbeitet, ist an dieser Stelle ein Vergleich mit der Spektralverschiebung aufgeführt, wieder ist $\omega = 16$ und eps $= 10^{-4}$

N	CW	Ritz mit A^TA
5	500	750
10	4000	5200
15	13500	13500

diese Werte erscheinen bzgl. dieses Vergleiches noch als sehr günstig, doch wenn man bedenkt, dass das Verfahren von *Cullum/Willoughby* nicht nur einen Eigenwert liefert, sondern mit großer Wahrscheinlichkeit genügend viele Innere, muss man das Verhältnis wieder relativieren. Dennoch bleibt ein Vorteil des Restarted-Lanczos Verfahren, dass man die Suche hinsichtlich des gewünschten Eigenwertes steuern kann und auch bzgl. des Speicheraufwandes ein Restart günstiger ist.

Fazit

Ausgehend von den numerischen Test kann man folgendes Resümee ziehen

1. das Verfahren von *Cullum/Willoughby* ist zur Suche von $5 - 10$ inneren Eigenwerte der schnellste Algorithmus
2. eine Residuuenminimierung ist für das Verfahren der Krylov Unterräume zu instabil hinsichtlich der Konvergenz
3. die Spektralverschiebung ist im Bezug auf die Konvergenzgeschwindigkeit nur Beschränkt einsetzbar
4. falls eine geeignete Näherungsinverse existieren sollte, könnten sowohl das Shift-and-Invert, als auch das Jacobi-Davidson Verfahren interessant werden
5. die Residuuenminimierung eignet sich nur, um zum Abschluss der Rechnung die Genauigkeit der Eigenvektoren zu erhöhen

Literatur

1. A. Göke:*Parallele Teilraumverfahren zur Bestimmung von Eigenwerten im Inneren des Spektrums*, Berichte des Forschungszentrums Jülich
2. B. Steffen:*Subspace Methods for Sparse Eigenvalue Problems*, International Journal of Differential Equations and Applications, Volume 3,3,2001, pp. 339 - 351
3. P.W. Anderson:*Absence of diffusion in certain random lattices*, Phys. Rev.,109 (1958), pp. 1492-1505
4. H.R. Schwarz:*Numerische Mathematik*, B.G.Teubner,Stuttgart
5. J. Cullum, R.A. Willoughby:*Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1: Theory* Birhäuser, Boston, 1985
6. Z. Bai, J. Demmel, J. Dongarra, A. Ruhem, H. van der Vorst:*Templates for the Solution of Algebraic Eigenvalue Problems*
7. E. Hofstetter, M. Schreiber:*Statistical properties of the eigenvalue spectrum of three-dimensional Anderson hamiltonian*, Phys. Rev. B, 48 (1993), pp. 16979-16985
8. F. Milde, R.A. Römer, M. Schreiber:*Multifractal analysis of the metal insulator transition in anisotropic systems* , Phys. Rev. B, 55 (1997), pp. 9463-9469
9. A. Frommer:*Iterationsverfahren* Vorlesung, Sommersemester 2003, home.telebel.de/miblumb
10. R.G. Grimes, J.G. Lewis, H.D. Simon:*A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems* ,SIAM J.Matrix Anal. Appl. Vol 15, No. 1, pp 000-000, January 1994
11. Y. Saad:*Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms* Wiley, New York 1992

Parallelising a Parameter Estimation Method for Substitution Models

Kjong-Van Lehmann

Friedrich-Schiller-Universität Jena, Institut für Bioinformatik
E-mail: kjong.lehmann@gmx.de

Abstract: A basic process in the evolution of DNA-sequences is the substitution of nucleotides during evolutionary time. The process of nucleotide substitution is influenced by many different factors. Substitution models describe these changes as stochastic processes depending on parameters regarding these factors. Different methods for estimating these parameters by two given sequences are developed. This work deals with parallelising a parameter estimation method for a substitution model regarding transitions, transversions and interaction between neighbours [1]. This method involves high computational effort, due to the fact that it includes a very time-consuming Monte-Carlo Simulation and a particular time-consuming Maximum likelihood estimation. Parallelising this method improved the speed of estimating the parameters nearly linear to the number of processes.

Motivation

Since Charles Darwin founded his Theory of Evolution in 1859 [4] it is a major task to the biology to reveal the relationship between all existing species. As a tool for visualizing evolutionary relations between different species, Charles Darwin introduced the evolutionary tree.

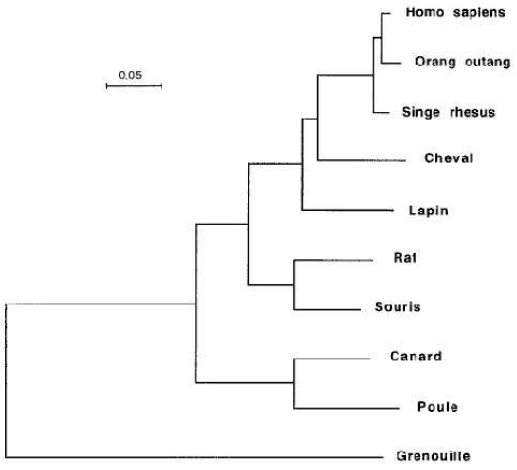


Figure 1: Evolutionary tree: the leaves represent certain species. Parental nodes represent the common ancestor. The branch length represents the time the species need ro reach their evolutionary state relative to their ancestors.

Whereas the construction of such trees was based on morphological attributes until a short time ago,

today we have the possibilities to reconstruct evolutionary trees by molecular data as for example DNA-sequences. The observation that the number of differences of aminoacid-sequences is proportional to the time of speciation of species [2] forms the basis for tree-reconstructing-methods derived from DNA-sequences. Different tree reconstructing methods have been developed. To find the accurate evolutionary tree involves high computational effort, because the number of possible trees increases exponentially with the number of species to analyze. Usually it is not possible to evaluate all possible trees, so heuristic methods are used to reconstruct a certain tree. Some tree reconstruction methods are based on the number of differences of DNA-sequences, which however perform rather poor. The neighbour joining algorithm is a clustering algorithm which searches for a tree with minimal total length (sum of all branch lengths). The branch length depends on the number of differences of DNA-sequences. A controversial tree reconstruction method is “Maximum-Parsimony”. This method is based on the assumption that the evolution of a sequence occurs by a minimal number of nucleotide substitutions. Tree reconstruction methods based on the “Maximum Likelihood” principle use substitution models to find an evolutionary tree, which leads with highest probability to the observed sequences.

Substitution models describe changes in DNA-sequences as a stochastic Markov Process and are used to calculate the transfer probability from one sequence to another. The rate of substitution depends on many different factors, such as the base itself or the position in the sequence. In the substitution model these rates are described by special parameters. The simplest substitution model is the Jukes-Cantor Model which treats each substitution with the same rate.

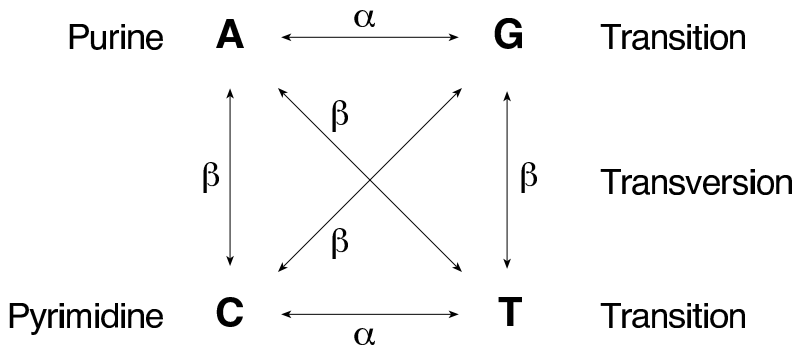


Figure 2: Substitution Model: Kimura Model: A substitution model which differentiates between the substitution rates of transitions and transversions

The Kimura Model (Fig. 2) is normally used to model the process of substitution. Of course, this model doesn’t fit the reality, but suffices an adequate accuracy in most cases. Nevertheless in some special cases this model fails, so more parameters are introduced. In this special work we dealt with a substitution model depending on seven parameters to regard different rates between transitions (Purin(A)↔Purin(G); Pyrimidin(C)↔Pyrimidin(T)) and transversions (Purin↔Pyrimidin) and also to regard substitution rates in CpG-isles (Hotspots of substitution). CpG-isles are high repetitive sequences in DNA, which tend to mutate frequently. The need of an appropriate parameter estimation method seems evident, if one consider the number of parameter. This work deals with the parameter estimation of two given DNA-sequences.

Parameter Estimation Method

Some Definitions

- **path:** a path is a random variable(vector) which describes the way of a sequence x evolving to sequence y (Fig. 3).

- **single-path:** a single-path is a path with sequence length 1 (Fig. 3).

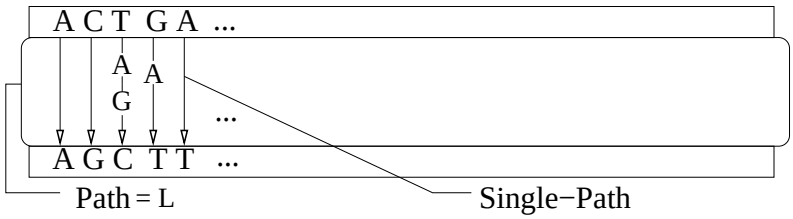


Figure 3: Path and Single-path: A path consists of all interstates a sequence x passes, when it evolves to sequence y. A single-path describes the evolution of one single nucleotide to another.

- **probability density:** if one considers an evolutionary process starting from sequence x and time t = 1, the probability density over all possible paths ending in any sequence is given by:

$$\int_{L\epsilon\chi} f(L,\theta)d\mu(L,t) = 1 \tag{1}$$

L = path

θ = set of parameters

χ = set of all possible paths

- **transfer probability:** the transfer probability is then:

$$p_{x\to y}(\theta) = \int_{L\epsilon\chi|x\to y} f(L,\theta)d\mu(L,t) \tag{2}$$

$\chi_{x\to y}$ = all paths which transfer sequence x to sequence y

The "Maximum Likelihood" principle

The "Maximum Likelihood" method is a parameter-estimation method for a given probability density depending on a set of parameters. For example:

$$f(u,\theta) \qquad \theta = \theta_1,...,\theta_m \qquad u\epsilon U \tag{3}$$

f is a certain probability density, θ a set of parameters and u a random variable. Now one holds a concrete random sample of r random variables ($u_1,..u_r\epsilon U$). The likelihood function is then defined as follows:

$$H(u_1,u_2,...,u_r;\theta) = \prod_{i=1}^{r} f(u_i,\theta) \tag{4}$$

Now the task is to determine the set of parameters such that the function becomes a maximum. For mathematical convenience the likelihood is usually evaluated by the logarithmic transformation which transforms the multiplication into summation.

"Maximum Likelihood" function

Obviously the transfer probability is not easy to calculate, as in our case it depends on seven parameters. So we embark on another strategy. We try to calculate the likelihoodratio. Starting from the defined probability density we can establish the following connections:

$$\frac{p_{x\to y}(\theta)}{p_{x\to y}(\theta_0)} = \int_{L\epsilon\chi|x\to y} \frac{f(L,\theta)}{f(L,\theta_0)} g(L,\theta_0) d\mu(L,t) \tag{5}$$

g is defined as:

$$g(L,\theta_0) = \frac{f(L,\theta_0)}{p_{x\to y}(\theta_0)} \tag{6}$$

g represents the probability density of paths which transfer sequence x to sequence y

$$\frac{p_{x\to y}(\theta)}{p_{x\to y}(\theta_0)} \approx \frac{1}{r} \sum_{i=1}^{r} \frac{f(L_i,\theta)}{f(L_i,\theta_0)} \tag{7}$$

L = $(L_1 ... L_r) \subseteq \chi_{|x\to y}$

θ_0 = initial set of parameters

r = number of sampled paths (next chapter)

L_i = paths sampled by Markov-Chain-Monte-Carlo Simulation from $g(L,\theta_0)$ (next chapter)

If we maximize this function we still maximize the transfer probability following the maximum likelihood principle. It is essential for this method to have a random sample of paths of high probability. This sample is designed by a Monte-Carlo Simulation. Using this sample, we can search a set of parameters θ which maximize the likelihoodratio.

M*arkov-***C***hain-***M***onte-***C***arlo Simulation*

We need to sample a set of paths which transform sequence x to sequence y with highest propability. This can be done by a Monte-Carlo Simulation[5]. In this simulation one generates a Markov-Chain of paths which are mainly of high probability. The idea is to accept every path with a higher probability than the predecessor path, but also to jump with a certain probability to paths with low probability. This ensures not to get stuck in a local maximum.

Fig. 4 clarifies this method. Sampling paths can lead into the local maximum by following the paths with higher probability. As we jump randomly to paths with low probability, we are able to free ourselves from the local maximum. The sampling-method we use is the Metropolis-Hastings algorithm.

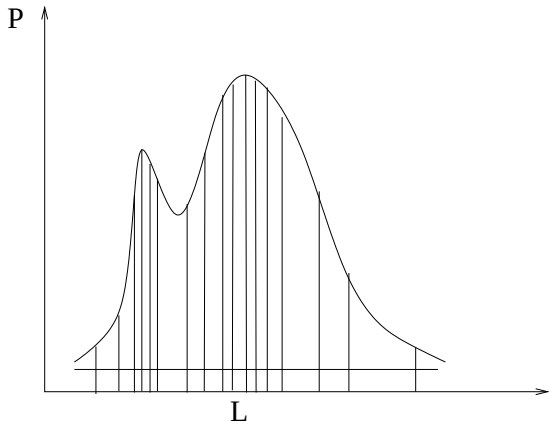


Figure 4: Principle of MCMC-Simulation. One can see the chosen paths and their probability

Implementation

Fig. 5 gives an overview of the parameter-estimation method. First a certain number of paths are sampled. Tests have shown that 2000 paths satisfy a sufficient accuracy. The Powell function is the implementation of a numerical multidimensional maximisation function [3], which main task consists of calculating the likelihood with a certain set of parameters.

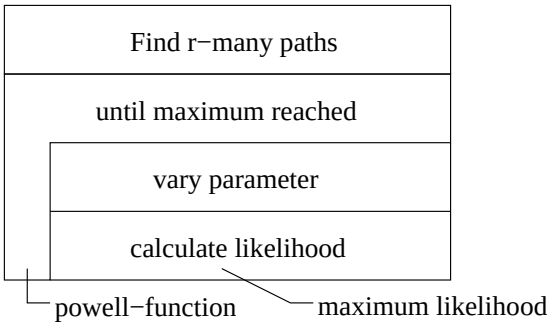


Figure 5: Implementation of parameter estimation

Parallelisation

As already mentioned this parameter estimation method is particularly time-consuming. So a parallelisation makes sense, especially because reconstructing evolutionary trees needs the evaluation of about 400 sequences. The Message Passing Interface (MPI) was used to parallelise the program.

Monte-Carlo-Simulation

There are two obvious possibilities to parallelise the Monte-Carlo Simulation.

1. One possibility is splitting the two given sequences in p parts (p=number of processes) and send one part of the pair of sequences to each process. Every process should now be able to sample a certain number of single-paths, which could be fused together to one sample of paths. This method seems practical, but it turned out that it is not. The matter is that we try to evaluate a substitution model which regards substitution rates depending on direct neighbours (parameter for CpG-isles).

Therefore the next single-path depends on its neighbour. This would cause problems at the borders of each part, involving much interprocess communication.

2. Another possible method is to send each process a complete copy of the pair of sequences. Now every process generates a certain number of paths (Fig. 6). Afterwards every process posses a random-sample of paths which are treated as one random sample in the following calculations. This method was finally implemented. There are some points to consider, which are discussed later.

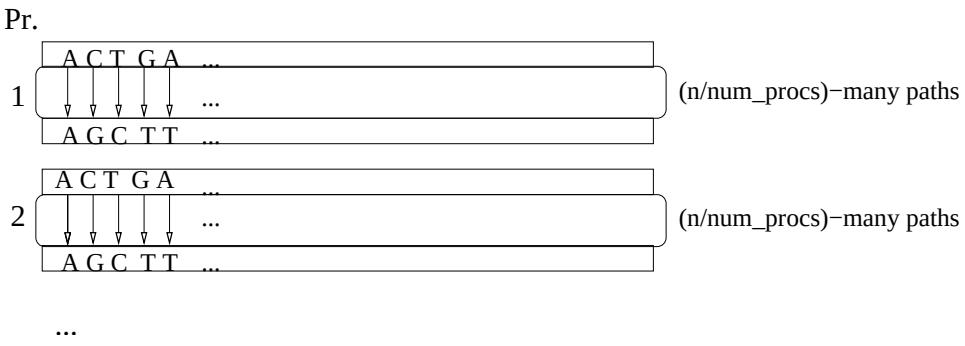


Figure 6: Parallelizing MCMC-Simulation. n = number of paths which have to be sampled; num-procs = number of processes; n/num-procs = number of paths sampled by one process

Monte-Carlo simulations rely on a huge amount of random numbers. So random number generators with long periods are needed. Parallelising such a simulation only makes sense, if one can generate independent random numbers on each process. To fulfill these demands we've used the Scalable Parallel Random Number Generator Library (SPRNG [6]). This package implements different parallel random number generators with a sufficient periodicity and ensures independent random number streams on all processes.

Likelihood calculation

The decision to parallelize the likelihood calculation is evident if one consider the likelihood function (7).

$$\frac{p_{x\to y}(\theta)}{p_{x\to y}(\theta_0)} \approx \frac{1}{r}\sum_{i=1}^{r}\frac{f(L_i,\theta)}{f(L_i,\theta_0)}$$

As every process holds a random sample of paths, every process calculates the likelihood function with a given parameter set and afterwards a reduce operation calculates the likelihood of all random samples. Due to the fact that the major task of the Powell function is composed of calculating the likelihood with different set of parameter this strategy makes sense.

Speedup Analysis

In this section we measure the speedup of the MCMC-Simulation and the Powel function. All tests have been done on ZamPaNo.

Speedup MCMC-Simulation

One could expect that the speedup of this simulation using the described parallelising strategy scales linear with the number of processes. There is little interprocess communication because the only in-

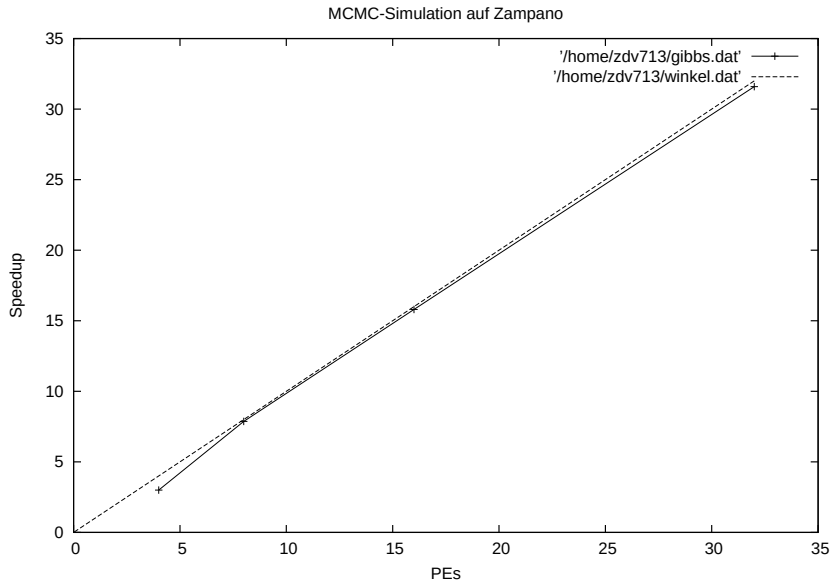


Figure 7: Speedup of the MCMC-Simulation on ZamPaNo;5000 Paths; Sequence length = 2000

formation every process needs are the DNA-sequences. The tests confirm our assumption. A special load-balancing was not necessary, because one could observe that sampling paths climbs nearly the same time on each process.

Speedup Powell function

The speedup of the Powell function is surprising, because only one part of the Powell function was parallelised. This speedup could be explained by the fact that the remaining due of sequential work is negligible small. Further tests should be done to validate this results which haven't been possible in that time because of technical problems. But this speedup seems probable if one consider that the major task of the Powel function consists of calculating the likelihood.

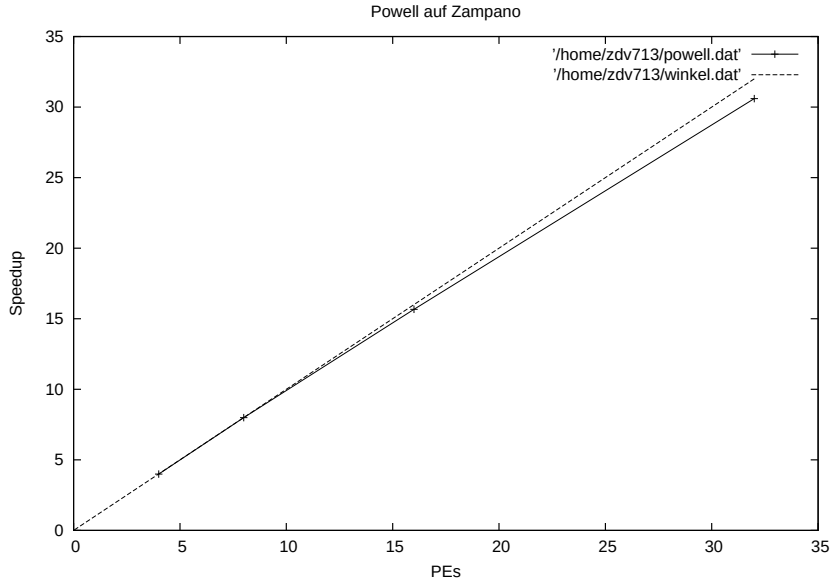


Figure 8: Speedup of the Powell-function on ZamPaNo;5000 Paths; Sequence length = 2000

Discussion and Future Work

There are some points to regard and to validate in future tests.

- After parallelising the MCMC-Simulation this program constructs p Markov Chains (p = number of processes) of paths. First tests showed that this seems to work well, but further tests have to be done to analyze:
  - the behaviour of the results with high number of processes
  - the behaviour of the results with different sequence lengths and sequences
  - how much paths have to be sampled, so that one process has a random sample of high probable paths.
- This tests have been very successful, showing that this method scales nearly linear with increasing process number. All tests have been done on ZamPaNo with sequence length 2000. It would be interesting to make some tests on th new Jülich Supercomputer Jump. In addition tests with different sequence length and different number of paths would be informative.

References

1. T. Schlegel
   Sequenzevolution mit lokalen Abhängigkeiten
2. D. Graur,W.-H. Li
   Fundamentals of Molecular Evolution; Sinauer
3. Press, Willian H., Teukolsky, Saul A., Vetterling, William T., Flannery, Brian T
   (1992) Numerical Recipes in C; Cambridge University Press
4. Ch. Darwin
   On The Origin of Species; Penguin Books
5. W.R. Gilks,S. Richardson,D.J. Spiegelhalter
   (1997) Markov Chain Monte Carlo in Practice; Chapman and Hall
6. M. Mascagni et al. http://sprng.cs.fsu.edu/

Parallelization of the Fast Multipole Method

Eric Lorenz

Institute for Theoretical Physics, University of Leipzig

E-mail: lorenz@itp.uni-leipzig.de

Abstract: The Fast Multipole Method (FMM) is introduced including the mathematical basics. Because of its efficiency in calculating energies and forces for a given charge distribution an implementation on massively parallel computers is highly desirable. The first steps in parallelizing the FMM approach had already been done by Jürgen Wieferink. Our aim is the parallelization of the radix sort algorithm using dynamic load balancing. We have achieved acceptable speedups for realistic systems of point charges.

Motivation

Many scientific applications such as molecular dynamics require the computation of Coulomb energies and forces. The direct method to compute the Coulomb energy requires the calculation of all pairwise interactions.

$$E_C = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{q_i q_j}{|\underline{r}_i - \underline{r}_j|} \tag{1}$$

N is the number of particles, i and j are the indices of two interacting charges, and $\underline{r}_i$ and q_i are position and charge of particle i. The direct computation requires a summation which shows $O(N^2)$ scaling. Due to the quadratic scaling computable systems are very limited in size. The use of a cutoff-radius is not applicable because of the long range behavior of the Coulomb potential.
Another possibility to calculate the Coulomb potential is given by the Fast Multipole Method which was published by Leslie Greengard [1] in 1988. White and Head-Gordon [2] have implemented the ideas of FMM to treat charge distributions. They have also introduced the so-called rotation based FMM which shows a better scaling with regard to the expansion length [3]. Because the FMM requires $O(N)$ work systems consisting of a billion of point charges can now be treated.

Mathematical basics

The electrostatic potential must satisfy the Laplace equation $\nabla^2\phi = 0$. The potential can be expanded in spherical harmonics which allows the factorization of the shperical coordinates.

$$\phi(r,\vartheta,\varphi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}[A_{lm}r^l + B_{lm}r^{-(l+1)}]Y_{lm}(\vartheta,\varphi) \tag{2}$$

The expansion of the potential is the sum of a multipole and a Taylor expansion. The spherical harmonics are defined as

$$Y_{lm}(\vartheta,\varphi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_{lm}(\cos\vartheta)e^{im\varphi} \tag{3}$$

The P_{lm} are the associated Legendre polynomials defined as

$$P_{lm}(x) = (-1)^m(1-x^2)^{\frac{m}{2}}\frac{d^m}{dx^m}P_l(x) \tag{4}$$

using the ordinary Legendre polynomials which can be defined in a explicit or implicit way

$$P_l(x) = \frac{1}{2^l l!}\frac{d^l}{dx^l}(x^2-1)^l \qquad m = -l,...,l \tag{5}$$

$$\frac{1}{\sqrt{1+\varrho^2-2\varrho x}} = \sum_{l=0}^{\infty}P_l(x)\varrho^l \tag{6}$$

The Legendre polynomials are the solutions of the ordinary Legendre differential equation for $\cos\vartheta = x$. Together with the addition theorem for spherical harmonics

$$P_l(cos\gamma) = \frac{4\pi}{2l+1}\sum_{m=-l}^{l}Y^*_{lm}(\alpha,\beta)Y_{lm}(\vartheta,\varphi) \tag{7}$$

where γ is the angle between (α,β) and (ϑ,φ) such that $\cos\gamma = \cos\alpha\cos\vartheta + \sin\alpha\sin\vartheta\cos(\alpha-\varphi)$ we can use the implicit definition of ordinary Legendre polynomials to obtain a relation which is the basis for further definitions.

$$\begin{aligned}\frac{1}{|\underline{r}-\underline{a}|} &= \frac{1}{\sqrt{r-a-2ra\cos\gamma}}\\ &= \sum_{l=0}^{\infty}\frac{a^l}{r^{l+1}}P_l(\cos\gamma)\\ &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(l+|m|)!}{(l-|m|)!}\frac{a^l}{r^{l+1}}P_{lm}(\cos\alpha)P_{lm}(\cos\vartheta)e^{-im(\beta-\varphi)}\end{aligned} \tag{8}$$

Equation (8) is the complete factorization of the inverse of two distant points $\underline{a} = (a,\alpha,\beta)$ and $\underline{r} = (r,\vartheta,\varphi)$, which is the substantial part of the Coulomb potential.

Coulomb energy of far field interaction

Multipole expansion

A multipole expansion allows to express the electrostatic potential in an expansion over spherical harmonics. The expansion is valid in case point $\underline{a}$ is well separated from point $\underline{a}$ as shown in fig. 1

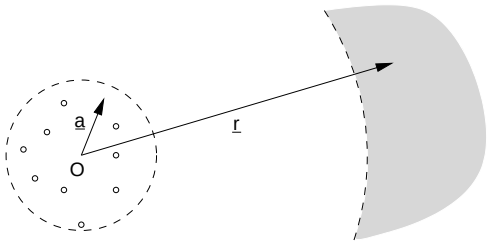


Figure 1: A multipole expansion is used to express the far field of the electrostatic potential of a charge pattern near the origin. $|\underline{r}| \gg |\underline{a}|$.

The multipole-like expansion is defined by

$$\phi(\underline{r}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \omega_{lm}(q, \underline{a}) \frac{(l-|m|)!}{r^{l+1}} P_{lm}(\cos\vartheta) e^{im\varphi} \tag{9}$$

The coefficients ω_{lm} are the multipole moments which can be calculated by

$$\omega_{lm}(q, \underline{a}) = q \frac{a^l}{(l+|m|)!} P_{lm}(\cos\alpha) e^{-im\beta} = q O_{lm}(\underline{a}) \tag{10}$$

$O_{lm}(\underline{a})$ are chargeless multipole moments as introduced by White and Head-Gordon. Multipole moments of the same order corresponding to several charges of the pattern can be summed.

Taylor-like expansion

We can obtain a taylor-like expansion from (8) as an expression of the potential near an origin caused by a charge pattern located far away from the origin as shown in fig. 2.

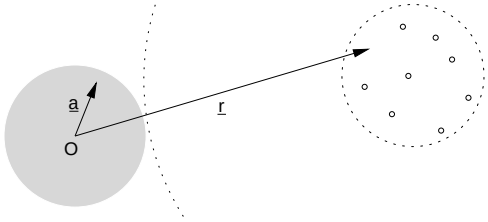


Figure 2: A taylor-like expansion is used to express the electrostatic potential around the origin caused by a charge pattern remote from the origin. $|\underline{r}| \gg |\underline{a}|$.

The relation we obtain is given by

$$\phi(\underline{a}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \mu_{lm}(q, \underline{r}) \frac{a^l}{(l+|m|)!} P_{lm}(\cos\alpha) e^{-im\varphi} \tag{11}$$

The Taylor coefficients are given by

$$\mu_{lm}(q, \underline{r}) = q \frac{(l-|m|)!}{r^{l+1}} P_{lm}(\cos\vartheta) e^{im\varphi} = q M_{lm}(\underline{r}) \tag{12}$$

The Taylor coefficients $\mu_{lm}(q, \underline{r})$ of the same order can be summed.
The definitions of multipole moments and Taylor coefficients allows the conclusion $\omega_{l,-m} = \omega^*_{l,m}$ (and same for μ_{lm}). Only the coefficients $\omega_{l,m}$ must be stored which reduces the memory requirement by a factor of two.

Coulomb energy of far field interaction

Using the above derived expressions for the Coulomb potential related to different origins we can now express the Coulomb energy of two spatial seperated charge patterns according to fig. 3 .

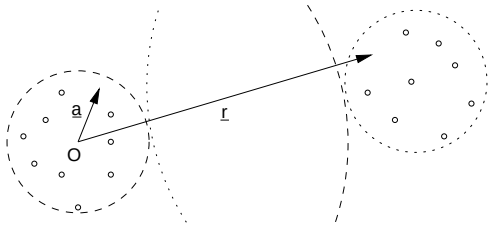


Figure 3: The Coulomb energy of two charge patterns can be written in terms of multipole moments and taylor-like coefficients.

The interaction energy of two charge patterns satisfying the condition $|\underline{r}_i| \gg |\underline{a}_j| \quad \forall q_i \in \varrho_r, q_j \in \varrho_a$ can now be written as

$$E_{ia} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \omega_{lm}(q, \underline{a}) \mu_{lm}(q, \underline{r}) \tag{13}$$

or without taking charges into account

$$\frac{1}{|\underline{r} - \underline{a}|} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} O_{lm}(\underline{a}) M_{lm}(\underline{r}) \tag{14}$$

In numerical calculations the infinit sum over all orders of ω_{lm} and μ_{lm} has to be truncated. The highest multipole moment $l = L$ is user-defined. The maximum error of this approximation can be obtained analytically by taking the absolute difference between the complete and truncated expansion.

$$\begin{aligned} \left| \frac{1}{|\underline{r} - \underline{a}|} - \sum_{l=0}^{L} \frac{a^l}{r^{l+1}} P_l(\cos\gamma) \right| &= \left| \sum_{l=L+1}^{\infty} \frac{a^l}{r^{l+1}} P_l(\cos\gamma) \right| \\ &\leq \frac{a^{L+1}}{r^{L+2}} \sum_{l=0}^{\infty} \left(\frac{a}{r}\right)^l = \frac{1}{r-a} \left(\frac{a}{r}\right)^{L+1} \end{aligned} \tag{15}$$

$a < r$ is necessary for convergence. It is possible to construct a sphere around the origin such that ϱ_a is located completely inside and ϱ_r outside of the sphere. The approximation is more accurate for larger L and better separation of the interacting charges $\frac{a}{r}$.

Transformation operators

The FMM is based on several transformation operators. There are three operators which translate a given expansion about one origin to another. Shifting expansions in space is the most time consuming part of the FMM. In case the expansions are rotated first the translation can be done along the quantization axis z. The whole transformation requires now $O(L^3)$ instead of $O(L^4)$ work (fig. 4).

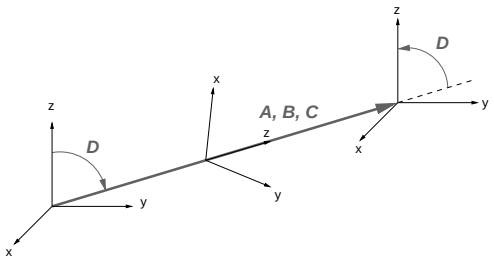


Figure 4: Rotation based FMM: To achieve $O(L^3)$ scaling expansions are rotated such that translation can be done along the quantization axis z.

Rotation

The translation operators are derived in general and simplified for rotations about the z axis ($\vartheta = 0$). All associated Legendre polynomials with $m \neq 0$ vanish.

$$P_{lm}(\cos 0) = P_{lm}(1) = \delta_{m,0} \tag{16}$$

Multipole moments and Taylor coefficients have to be rotated first followed by the tranlation about the z axis. Finally the moments and coefficients must rotated again to the primary positions in space. Each rotation encloses a rotation about the z axis by the angle γ and a rotation about the y axis by δ. Corresponding to [3] the rotation operator can be splitted in these two parts.

$$\hat{R} = \hat{R}_\delta \hat{R}_\gamma = e^{-i\delta J_z} e^{-i\gamma J_y} \tag{17}$$

J_z and J_y are the angular moment operators about z and y. The Wigner rotation matrices can be written as a product of two rotation matrices.

$$D^l_{km}(\delta, \gamma) = d^l_{km}(\delta) e^{-ik\gamma} \tag{18}$$

The d's are defined analytically as

$$\begin{aligned} d^l_{km}(\delta) =& \frac{1}{2^l} \sqrt{\frac{(l-m)!(l+m)!}{(l-k)!(l+k)!}} (1 + sgn(k)\cos\delta)^{|k|} (\sin\delta)^{m-|k|} \\ & \times \sum_{n=0}^{l-m} (-1)^{l-m-n} \binom{l-k}{n} \binom{l+k}{l-m-n} (1+\cos\delta)^n (1-\cos\delta)^{l-m-n} \end{aligned} \tag{19}$$

and can be determined by two recurrence relations. The first one is given by

$$d^l_{k+1,m}(\delta) = \frac{-(k+m)}{\sqrt{l(l+1) - k(k+1)}} \frac{\sin\delta}{(1+\cos\delta)} d^l_{k,m}(\delta) + \sqrt{\frac{l(l+1) - m(m-1)}{l(l+1) - k(k+1)}} d^l_{k,m-1}(\delta) \tag{20}$$

with starting point

$$d^l_{0,m}(\delta) = \begin{cases} \sqrt{\frac{(-m)!}{(l+m)!}} P_{lm}(\cos\delta) & \text{for } m \geq 0 \\ (-1)^m \sqrt{\frac{(-m)!}{(l+m)!}} P_{lm}(\cos\delta) & \text{for } m < 0 \end{cases} \tag{21}$$

This recurrence relation becomes unstable if δ approaches π. The numerical instability can be avoided by the combination of the first recurrence formula with a second recurrence relation given by

$$d^l_{k,m-1}(\delta) = \sqrt{\frac{l(l+1) - k(k+1)}{l(l+1) - m(m-1)}} d^l_{k+1,m}(\delta) - \frac{k+m}{\sqrt{l(l+1) - m(m-1)}} \frac{\sin\delta}{(1+\cos\delta)} d^l_{k,m}(\delta) \tag{22}$$

with starting point

$$d^l_{k,l}(\delta) = \frac{1}{2^l}\sqrt{\frac{(2l)!}{(l-k)!(l+k)!}}(\sin\delta)^{l-k}(1+\cos\delta)^k \tag{23}$$

Because of the property

$$d^l_{k,m}(\pi-\delta) = (-1)^{l+k}d^l_{m,k}(\delta) \tag{24}$$

angles greater than $\frac{\pi}{2}$ need not to be calculated. Because the FMM uses a rigid box scheme the number of possible angles is limited.
Finally we are able to rotate spherical harmonics by a linear combination of spherical harmonics of all orders.

$$Y_{lm}(\vartheta+\delta,\varphi+\gamma) = \sum_{k=-l}^{l} D^l_{km}Y_{lk}(\vartheta,\varphi) \tag{25}$$

The chargeless moments are defined by

$$\begin{aligned} O_{lm}(a,\alpha+\delta,\beta+\gamma) &= \sum_{k=-l}^{l}\sqrt{\frac{(l-k)!(l+k)!}{(l-|m|)!(l+|m|)!}}D^l_{km}O_{lk}(a,\alpha,\beta) \\ M_{lm}(r,\vartheta+\delta,\varphi+\gamma) &= \sum_{k=-l}^{l}\sqrt{\frac{(l-|m|)!(l+|m|)!}{(l-|k|)!(l+|k|)!}}D^l_{km}O_{lk}(a,\alpha,\beta) \end{aligned} \tag{26}$$

Operator A

The Operator A tanslates a given multipole expansion into another one about a different origin. Imagine a coordinate system with a given multipole expansion is shifted by $-\underline{b}$ to another one and the expansion is unchanged. It is equivalent to a shift by $\underline{b}$ of the multipole expansion in the original coordinate system. Another addition theorem is needed to obtain a transformation relation. In the notation of White and Head-Gordon it can be written as

$$O_{lm}(\underline{a}+\underline{b}) = \sum_{j=0}^{l}\sum_{k=-j}^{j} A^{lm}_{jk}(\underline{b})O_{jk}(\underline{a}) \tag{27}$$

where A is the needed operator which is defined for any angle θ and simplified for $\theta = 0$ which corresponds to a rotation about the z axis.

$$A^{lm}_{jk}(\underline{b}) = O_{l-j,m-k}(\underline{b}) \overset{\vartheta=0}{=} \frac{b^{l-j}}{(l-j+|m-k|)!}\delta_{m-k,0} \tag{28}$$

The transformed multipole moments can be written as

$$\omega_{lm}(\underline{a}+\underline{b}) = \sum_{j=0}^{l}\frac{b^{l-j}}{(l-j)!}\omega_{jm}(\underline{a}) \tag{29}$$

Operator B

Furthermore we need an operator which translates a given multipole expansion centered at the origin into a local Taylor expansion about another center shifted by $\underline{b}$. On the basis of the relation

$$\sum_{l=0}^{\infty}\sum_{m=-l}^{l} M_{lm}(\underline{r})O_{lm}(\underline{a}+\underline{b}) = \frac{1}{|\underline{r}-\underline{a}+\underline{b}|} = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} O_{lm}(\underline{a})\left[\sum_{j=0}^{\infty}\sum_{k=-j}^{j} M_{j+l,k+m}(\underline{r})O_{jk}(\underline{b})\right] \tag{30}$$

we obtain

$$M_{lm}(\underline{a}-\underline{b}) = \sum_{j=l}^{\infty}\sum_{k=-j}^{j} B_{jk}^{lm}(\underline{b})O_{jk}(\underline{a}) \tag{31}$$

where B is the transformation operator.

$$B_{jk}^{lm} = M_{j+l,k+m}(\underline{b}) \overset{\vartheta=0}{=} \frac{(l-j+|m-k|)!}{b^{l+j+1}}\delta_{m+k,0} \tag{32}$$

A transformation a the z axis is defined by

$$\mu_{lm}(\underline{a}-\underline{b}) = \sum_{j=0}^{\infty}\frac{(j+l)!}{b^{j+l+1}}\omega_{j,-m}(\underline{a}) \tag{33}$$

The truncation of the expansion causes an additional error within the FMM approach. White and Head-Gordon [2] have shown that the error has approximately the magnitude compared to the error caused by truncation of the multipole expansion.

Operator C

Finally we need an operator which translates a given Taylor expansion into another one about a different origin. Similar to the determination of operator B we can obtain an expression for the operator C

$$\frac{1}{|\underline{r}-\underline{a}+\underline{b}|} = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} O_{jk}(\underline{a})\left[\sum_{l=j}^{\infty}\sum_{m=-l}^{l} O_{l-j,m-k}(\underline{b})M_{lm}(\underline{r})\right] \tag{34}$$

$$M_{lm}(\underline{r}-\underline{b}) = \sum_{j=0}^{\infty}\sum_{k=-j}^{j} C_{jk}^{lm}(\underline{b})M_{jk}(\underline{r}) \tag{35}$$

C is defined as follows

$$C_{jk}^{lm}(\underline{b}) = O_{l-j,m-k}(\underline{b}) \overset{\vartheta=0}{=} \frac{b^{l-j}}{(l-j+|m-k|)!}\delta_{m-k,0} \tag{36}$$

The Taylor coefficients can be translated about the z axis as follows

$$\mu_{lm}(\underline{r}-\underline{b}) = \sum_{j=l}^{\infty}\frac{b^{j-l}}{(j-l)!}\mu_{jm}(\underline{r}) \tag{37}$$

The truncation of the expansion does not cause an error of higher magnitude in comparison to the operator B.

The algorithm

The FMM is one of the methods which uses a hierarchical separation of space.
Fist all particles are enclosed by a box with coordinate ranges [0,1]x[0,1]x[0,1]. The parent box is divided in half along the Cartesian axis to yield a set of 8 smaller child boxes illustrated in fig. 5. The parent box which contains all particles indicates the first level of the FMM tree ($level = 1$). The 8 child boxes of the parent box form the second level of the tree. On the level $s+1$ the number of child boxes is equal to 8^s . The number of subdivisions is chosen to keep the number of particles in every box on the lowest level approximately independent on the total number of particles. The numeration of boxes in all the tree levels is shown in fig. 6.

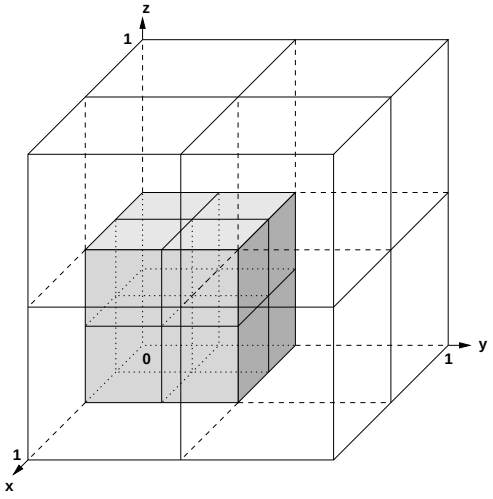


Figure 5: The particle space is devided in child boxes along the Cartesian axis

The length of the multipole expansion L and the depth of the multipole tree are parameters of the Fast Multipole Method. The third parameter ws denotes the number of boxes separating two boxes which can interact via multipoles. In our FMM approach ws is set to 1. $ws > 1$ would decrease the FMM error and increase the work in the far field part. To take advantage of the multipole approximation $ws = 1$ is always preferred.

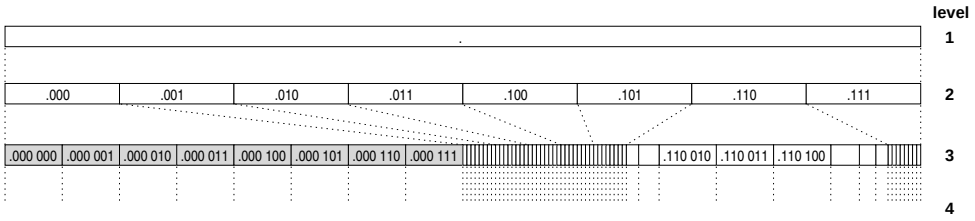


Figure 6: Boxes are numbered by an integer consisting of triples of bits. Every triple contains the information about the box location on the current level. The number of the parant box can be computed by a logical right shift.

The FMM consists of several steps. In pass1 the charges contained within each lowest level box are expanded in multipoles about the center of the box. In pass 2 the multipole expansions are transformed into Taylor expansions. The two boxes must be separated by at least one box on the current level of the tree, but only provided that parents of the two boxes are not separated on the next higher level on the tree. Pass 2 is by far the most time-consuming step of the FMM. In pass 3 the parent's Taylor expansions are translated to the centers of the parent's children. At the end of pass 3 each lowest level box contains a Taylor expansion of all far field interactions. In pass 4 for each lowest level box the multipole expansion and the Taylor expansion are multiplied. The sum over all lowest level boxes gives the far field energy. Finally, in pass 5 the remaining near field energy is computed by the direct method.

Pass 1

The particles are sorted according to their box number. The multipole moments of all particles for each lowest level box are calculated with respect to the box center. Multipole moments about the same origin can of course be summed. At the end of pass 1 each lowest level box of the FMM tree contains a multipole expansion for the particles.

The multipole moments of all child boxes are shifted to the centers of the parent boxes until level 3 is reached. The multipole moments of the child boxes can be summed about the common origin of the parent box. In fig. 7 pass 1 is illustrated in 3 dimensions and for 3 levels.

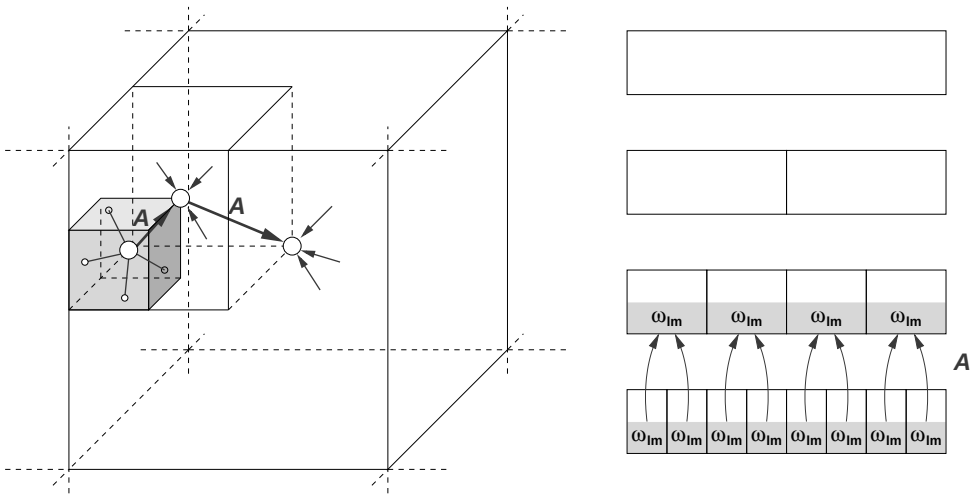


Figure 7: Pass 1: Calculation and shifting of multipole moments.

Since the number of particles per lowest level box is approximatly constant the number of boxes grows linearly with N and the pass 1 requires $O(N)$ work.

Pass 2

In pass 2 the operator B is used to convert the multipole expansions created in pass 1 into Taylor expansions. These expansions can be summed in every box. The two boxes must be separated by at least one box. The parent boxes must not be separated on the next higher level of the FMM tree.
The pass 2 is illustrated in fig. 8. Pass 2 scales linearly with regard to the number of particles.

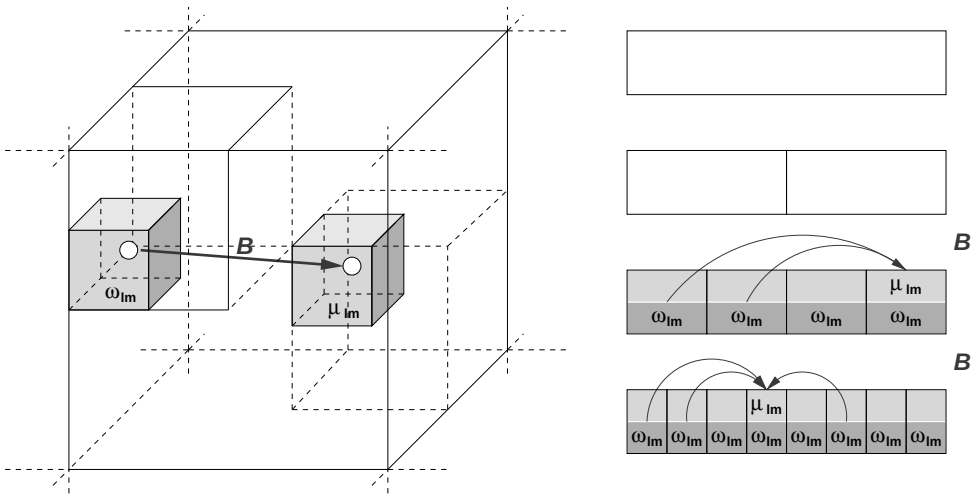


Figure 8: In pass 2 the multipole expansions are transformed into Taylor expansions. The two boxes must be separated. The paraents of the two boxes must not be separated on the next higher level.

Pass 3

In Pass 3 we use the operator C to shift the Taylor expansion of every box created in pass 2 to the centers of their child boxes as illustrated in fig. 9. The pass 3 is the complement to pass 1. In pass 1 the

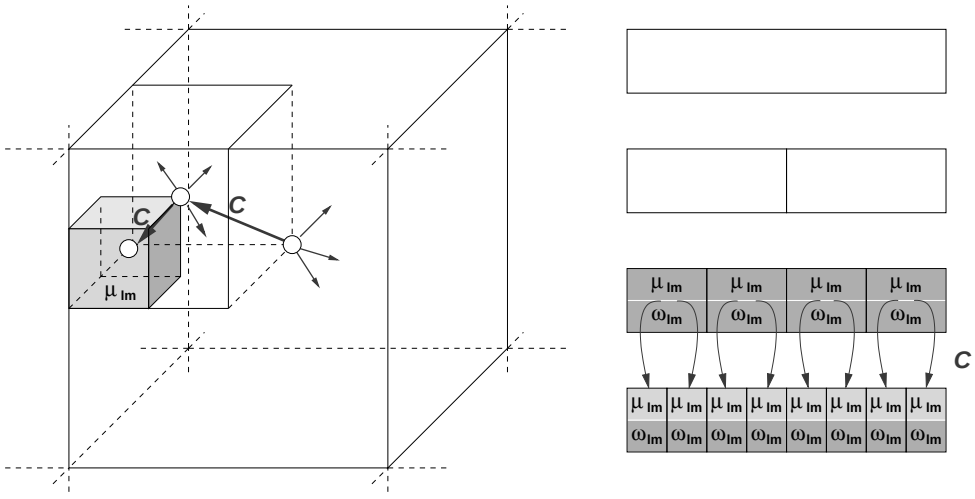


Figure 9: The Taylor expansions are shifted to the centers of the child boxes.

expansions are moved upwards, in pass 3 downwards. Consequently pass 3 scales lineraly regarding the number of particles.
At the end of pass 3 each lowest level box contains a Taylor expansion of all far field interactions for all particles in this box.

Pass 4

In pass 4 the far field energy is computed as the product of the multipole expansion and the Taylor expansion. The product is computed for each lowest level box. The sum over all lowest level boxes gives the total far field energy of the system. Pass 4 requires $O(N)$ work.

Pass 5

The remaining near field energy is computed in pass 5 by the direct method. It encloses the interaction of the particles within a lowest level box and the interaction of particles in two not separated boxes on the lowest level of the FMM tree. Because the number of particles is approximately constant in each lowest level box pass 5 shows linear scaling.

Sorting

The need to sort

An effective implementation of FMM requires an efficient sorting algorithm. There are four two dimensionally arrays for storing the ω_{lm}'s and μ_{lm}'s

```
romegatree(1:(L+1)(L+2)/2, 1:nboxes)
iomegatree(      ...     ,   ...   )
rmutree   (      ...     ,   ...   )
imutree   (      ...     ,   ...   )
```

The first dimension is used to store all the moments of a multipole or Taylor expansion. We need $\sum_{l=0}^{L}\sum_{m=0}^{l} 1 = \frac{(L+1)(L+2)}{2}$ entries. The second dimension is determined by the number of all boxes on all levels. Because of the symmetry relations $\omega_{l,-m} = \omega^*_{l,m}$ and $\mu_{l,-m} = \mu^*_{l,m}$only moments for $m \geq 0$ need to be stored. Nevertheless the memory requirement is enormous.
For a realistic system of point charges most of the boxes are empty. All empty boxes should be truncated from the FMM tree. The second dimension should be determined by the number of occupied boxes. For each tree level an integer array called `ibox()` is used which contains the numbers of all occupied boxes. The position of a certain box number is associated to the second dimension parameter. In case the array `ibox()` is not sorted the position search for a given box number scales lineraly regarding the number of boxes. The sorted array allows a logarithmic search which is much faster.
In pass 2 all possible destination boxes are computed for a given box by logical bit operations. An ef-

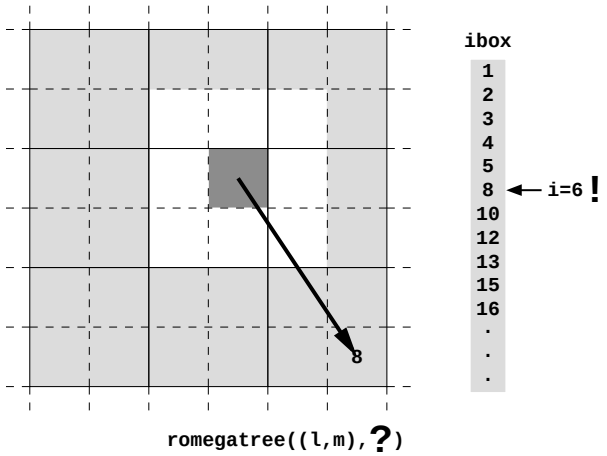


Figure 10: Pass 2 needs to know not only the names of the destination boxes for the transformation but also at which position their ω_{lm}'s and μ_{lm}'s in the corresponding arrays are stored.

ficient implementation of pass 2 requires fast access to the multipole moments and Taylor coefficients. The logarithmic search algorithm provides a possibility of fast access. On the other hand the time for sorting must not become significant. Because the FMM scales linearly a sort algorithm should be used which scales linearly too. The scaling of the sort algorithm is important as long as the computation time for sorting increases the total computation time by less than 1%.

Radix exchange sort

To sort the array `ibox()` a radix exchange sort algorithm is used in the given implementation. We want to explain the serial version of the radix exchange sort first to get a better understanding for possible parallelization approaches.
In general sorting algorithms assume that the keys to be sorted can be directly compared and be ordered in sequence in a finite time. The direct comparision of keys requires $O(N \log(N))$ scaling and algorithms such as quicksort or heapsort break down in efficiency in case the number of keys is in a range of several millions. Due to the FMM algorithm the integer array `ibox()` contains many identical entries because a box contains in general hundreds of point charges. Only the radix exchange sort algorithm can take advantage of identical entries. The algorithm is illustrated at an example in fig. 11.
To explain the radix sort algorithm we assume the keys are words made of letters out of a set called alphabet. In our case words are integer values in binary representation consisting of only two letters 0 or 1 respectivly.
First we look at the most significant bit which is the most left one. Two pointers i and j are used to scan

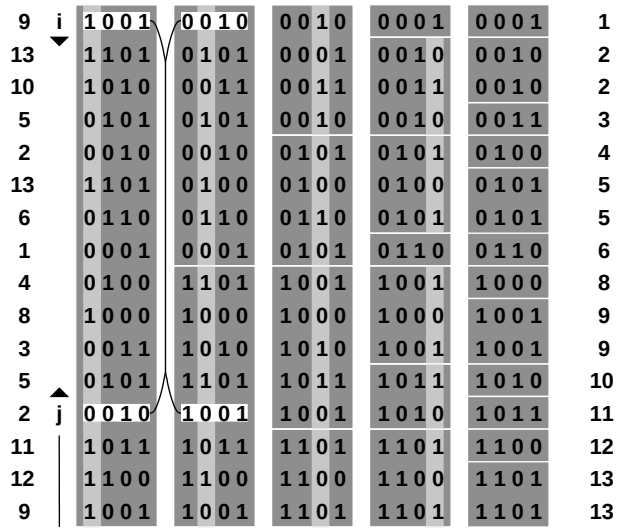


Figure 11: The radix exchange sort algorithm. Two pointers start from the opposite ends of the array and scan the bits starting at the most significant bit position. Pointer i stops if a 1 is found, j stops if a 0 is found. Both keys must be exchanged.

the array. i starts at the first key in the array and goes downwards, j starts at the end and walks upwards in key position. Both indices always test the bit of the keys at the recent bit position. In case a 1 is found at i's position, i stops and waits until j has gone until a 0 is found. Both keys at position i and j have to be exchanged now. After the exchange i continues searching for 1's, the same does j for 0's and if both have stopped the keys have to be exchanged again. The procedure is continued until i and j have met. The result is a separation of the array into two classes, one class contains keys having a 0 at the recent bit position, the other class contains keys having a 1 at the same bit position.
All keys in class 0 are smaller than those in class 1. This allows us to repeat the described procedure seperatly on both classes looking at the next bit position. A recursive implementation is possible.
When the last bit position has reached, we have created 2^{b-1} classes in the right order containing more similar keys. b is the number of bit positions which had to be examined. Keys which are similar were not changed in sequence. This is called a stabil sorting algorithm.
Due to the fact that at every bit position a key is touched only once and the number of examined bit positions does not depend on the total number of keys the radix sort scales linearly with regard to the total number of keys.

The static idea of a parallel radix sort

The efficient parallelization of a sorting algorithm is always difficult because sorting algorithms are based on comparision between all keys. In case the information is distributed over more than one process a large ammount of data has to be exchanged.
The communication itself consists not only of sending data. The network needs some time to set up the communication. It is not a good approach to send a single integer.
Suppose the array `ibox()` is stored in a global array and we have `nproc` available processes. Every participating process copies a part of size `length(ibox())/nprocs` into a local buffer. After it every process begins to sort its portion using the radix sort algorithm starting at the most significant key $bpos_0$ and stops sorting at a bit position $bpos_1$ which is not the last one. After it every process holds $2^{bpos_0 - bpos_1}$ independend classes. But these classes are not complete classes. Every process holds only a part of the global class. The class has to be reassembled to be coharent.
Every process should hold a complete class in his local buffer. To reach this goal, either the processes

put the small parts of a class to the global array or a global operation is necessary at the end. In our implementation a global array is used.
Due to the need to have information about the length of all classes to determine the right destination location in the global array a slightly different version of the serial radix sort is used in the presorting part. It puts out class lengths without spending time on counting 0's or 1's.
After the first part of creating classes and some communication due to coordinate reassembling every process holds a complete class in its local buffer such that it can be sorted until $bpos = 0$ without any restriction or communication. After all the ordered sequence is written to the global array with respect to the order of the processes implied by the order of the classes they hold. The whole method is illustrated at an example in fig 12.

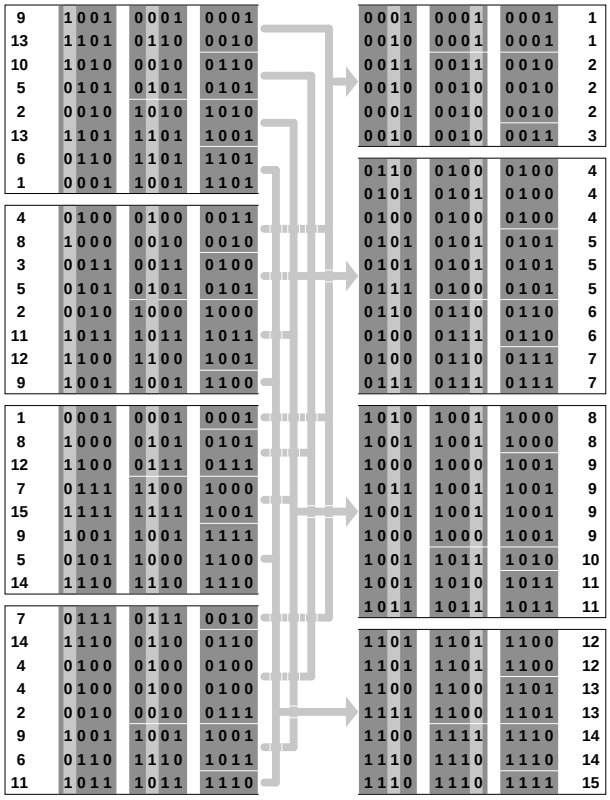


Figure 12: The static version of parallel radix sort. After the first part of presorting until there are as many classes as processes, the reassembled classes can be sorted until the end independendly.

This version has some disadvantages. As the first drawback we must mention that due to the fact that with every sorted bit position the number of classes is multiplied by 2 and the method expects that there are exactly the same number of processes as the number of classes at $bpos_1$. An implementation would only use the number of processes which is a power of 2.
Another disadvantage is that every process in the second part of the algorithm has to sort its class with no help from any other processes. This means if there is a class which is almost as big as the entire global array, one processor has to make almost the whole work and there is no speedup. For a flat distribution of keys this method works fine as one can see fig. 14. But especially in more interesting cases of charge distribution calculated by the FMM it happens that charges are more concentrated around one or more points. In this case the distribution of box numbers in the array `ibox()` follows approximatly the distribution of charges. So we have to think about load balancing.

The dynamic idea of a parallel radix sort

In the first part of the algorithm there is a homogeneous distribution of work to the processes but as one can see there is the need for more flexibility of distributing work for the second part.
But dynamic load balancing isn't applicable here. It would be a good idea having a lot of classes and some processes taking a task and in case it is done looking for tasks again. But this would force a process to wait until another has finished with its work to send the information about which class he takes as next job because all processes need to have the whole information about available tasks.
So it is another idea which we have implemented.
The work a class requires grows linearly with the length of the class. It is possible to use the information about class length we have after part one to have some information about the work which is still to do. Of course every process needs to know the length of the classes of all other processes. Some communication is needed. Through summing the length of the corresponding class parts of every process, every process can determine the whole length of every class and succeeding choosing the optimal number of processes which should work on it. With working through the list of classes starting with the most length class the following order of events take place.
The entire method is made such that it can be used recursivly. So if the ideal number of processes for

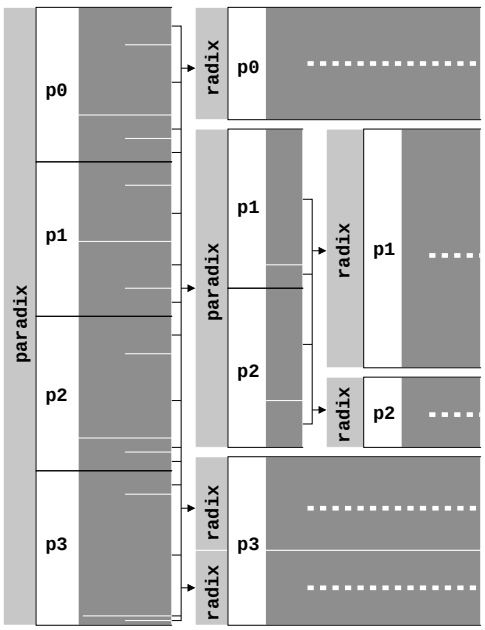


Figure 13: The dynamic version of parallel radix sort. It is the same principle as in the static version but processes can work together on one large class (p1 and p2) by calling paradix again together on the large class or a process can work on more than one smaller class. It is a more abstract representation but both cases of class distributing can be seen. The lighter grey areas represent the data in the Global Array, the darker areas data in the local buffer.

a class is greater than 1, this number of processes call again the whole sorting subroutine but with the length of their class and the bit position which was reached after the presorting part one. This limited number of processes do the same as all did together after the first call of the parallel sorting routine. They presort their class again till enough subclasses are available for independend processing or call the sorting routine again for a big subclass.
Classes which are not big enough for collective processing must be sorted by a single process. This has not to be done by a recursive call. The serial version of radix sort for its class is called. Also it could be that some classes are such small that one process can work on more than one of them.
The entire procedure contains a lot of weighing of the numbers of processes per class or classes per

process. Some parameters are used to influence the behaviour.
If the length of a class is shorter than a minimum class length the processor takes the next smaller class too. As the third parameter we use the maximum possible recursion steps and as the last parameter we have to define how many classes should by the result of presorting depending on the number of available processes.

Results

As one can see in fig. 15 the speedup for the parallel radix version with load balancing is not as good as the speedup for the version without load balancing in the case of a flat distribution. But it can be used with different distributions and changes its speedup behaviour not that much except if there would be a very thin peak in the distribution where all keys are in only one class even for very high recursive dividing.
The four parameters of the algorithm were estimated coarsely for now and should be optimized. This requires more testing with different distributions.

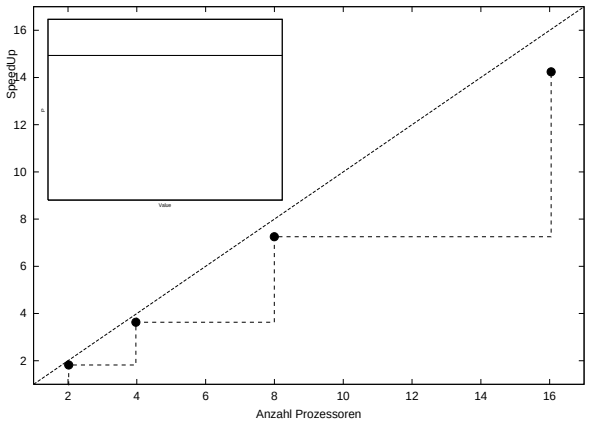


Figure 14: The speedup for the static version of parallel radix sort is almost perfect in the case of a flat distribution. The dashed line shows that the static version works only with a number of processes which is a power of 2. All other processes would have to be excluded. Measurements are done on a CrayT3E with a maximum number of 2^{23} keys.

To realize a recursive parallel program its necessary to use a barrier where only chosen processes wait for the others of their group. Neither in the GA package a call is included for this nor in the TCGMSG, which is the message passing libriary and foundation of Global Arrays. So it was used a construct made of `snd` and `rcv` which works as a selective barrier but it needs a lot more time. This can be improved.

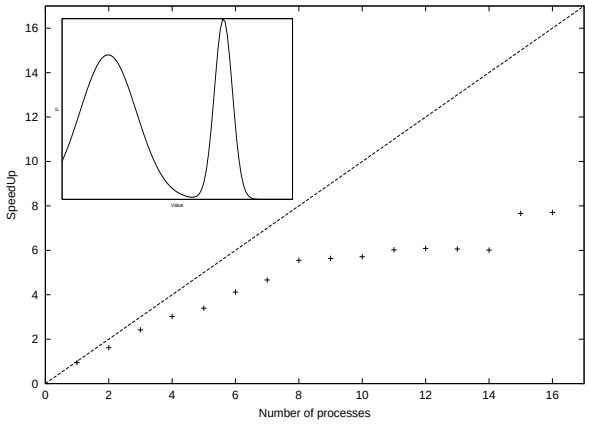


Figure 15: The speedup for the dynamic version of parallel radix sort. It is not that good in comparision to fig. 14 for the static version with a flat distribution but the static version wouldn't produce any good speedup with distributions like this one. According to interesting cases of charge distribution in the FMM, this distribution is composed of two Gaussian distributions. Measurements are done on a CrayT3E with a maximum number of 2^{23} keys.

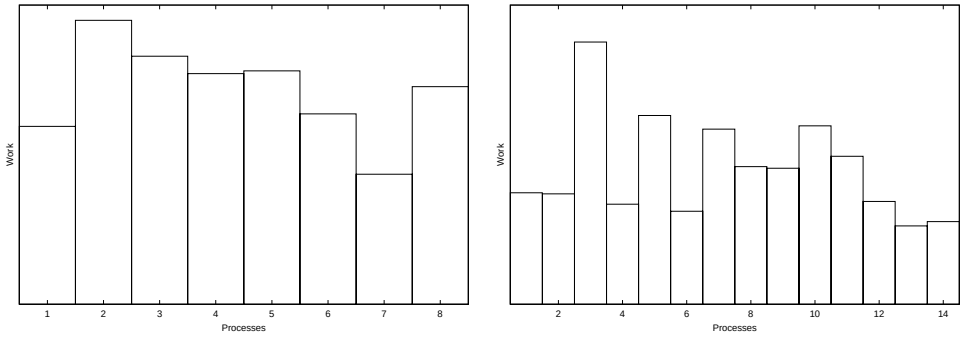


Figure 16: This shows which amount of work every single process had within the whole dynamic parallel radix sort. The load balancing works better for 8 processes than for 14 processes in this case. The 4 parameters of the dynamic radix sort have to be adjusted to avoid that one process needs a lot more time than the others. The data used here is out of the same measurements as for fig. 15.

Acknowledgements

I would like to thank anyone who make the gueststudents program possible. Special thanks of course to Dr. Holger Dachsel for patience in describing details of the FMM.

References

1. L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems,MIT,Cambridge,1988
2. C.A. White and M. Head-Gordon, Derivation and effi cient implementation of the fast multipole method, J.Chem.Phys.101, 6593 (1994)
3. C.A. White and M. Head-Gordon, Rotating around the quartic angular momentum barrier in fast multipole method calculation,J.Chem.Phys 105, 5061 (1996)
4. W. Jahnke, Pseudo Random Numbers: Generation and Quality Checks, published in "Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms", NIC Series Vol. 10, pp.447-458, 2002
5. R. K. Beatson and L. Greengard, A short course on fast multipole methods, in Wavelets, Multilevel Methods and Elliptic PDEs (M. Ainsworth, J. Levesley, W. Light, and M. Marletta, eds.), pp. 1-37, Oxford University Press, 1997
6. J.D. Jackson, Klassische Elektrodynamik, de Gruyter, Berlin (1982)

Robuste Verfahren zur Berechnung von Korrelationsmatrizen sowie zur Hauptkomponentenanalyse

Natali Zint

Ernst-Moritz-Arndt-Universität Greifswald, Institut für Mathematik und Informatik
Jahnstr. 15a, 17487 Greifswald

E-Mail: nz000211@mail.uni-greifswald.de

Zusammenfassung:
Im Rahmen des GALA-Projektes (Grünenthal Applied Life Science Analysis) wurden alternativ zu den bisher entwickelten und implementierten Verfahren zur automatisierten Datenauswertung Module zur robusten Berechnung von Korrelationsmatrizen und zur robusten Hauptkomponentenanalyse implementiert. Die Erläuterung dieser robusten Verfahren sowie ein kurzes Vorstellen von Ergebnissen ist Ziel des vorliegenden Berichtes.

Einleitung

Bei der Analyse von Daten, die aus Experimenten gewonnen werden, stellt sich das Problem der Ausreißer, d.h. es gibt Daten, die vom Gros der Daten stark abweichen. Dabei muss man zwischen zwei Arten von Ausreißern unterscheiden:
Auf der einen Seite kann man davon ausgehen, dass etwa $0.1 - 10\%$ der Daten auf Grund von fehlerhaften Messungen oder Fehlern bei der Übertragung in Datenbanken falsch sind [5]. Solche Ausreißer sind unerwünscht.
Auf der anderen Seite können aber auch extreme Werte erwünscht sein. So werden in der pharmazeutischen Forschung in manchen Versuchen Substanzen gesucht, die herausragende Eigenschaften haben und deshalb bei der Entwicklung von Arzneistoffen von Interesse sein könnten.

Aus diesem Grund werden Verfahren gesucht, die Ausreißer erkennen. Außerdem möchte man Korrelationsberechnungen sowie darauf aufbauende Analysen durchführen, ohne dass die Ergebnisse von den Ausreißern abhängen. Denn wenn aus Daten, in denen Ausreißer vorhanden sind, mittels der herkömmlichen Verfahren Korrelationen berechnet werden, können bestehende Abhängigkeiten verdeckt, aber auch nicht bestehende Abhängigkeiten vorgetäuscht werden [1].

Eine mögliche Vorgehensweise dabei ist, die Ausreißer zunächst zu identifizieren, diese zu eliminieren und die gewünschten Berechnungen dann mit den verbleibenden Daten durchzuführen. Dieses Verfahren wird in [1] erläutert. Dort werden Mahalanobisdistanzen verwendet, um Ausreißer zu identifizieren. Dabei müssen allerdings oft mehrere Iterationen durchgeführt werden, damit alle Ausreißer erkannt werden (vgl. Abb. 1). Es können aber auch Fälle auftreten, in denen das Verfahren versagt und manche Ausreißer nicht identifiziert werden.

Alternativ dazu können robuste Verfahren verwendet werden, in denen den Beobachtungen abhängig von den Daten unterschiedliche Gewichte gegeben werden. Solche Verfahren wurden im Rahmen dieser Arbeit in der Programmiersprache C implementiert, und sie werden im Folgenden vorgestellt. Im zweiten

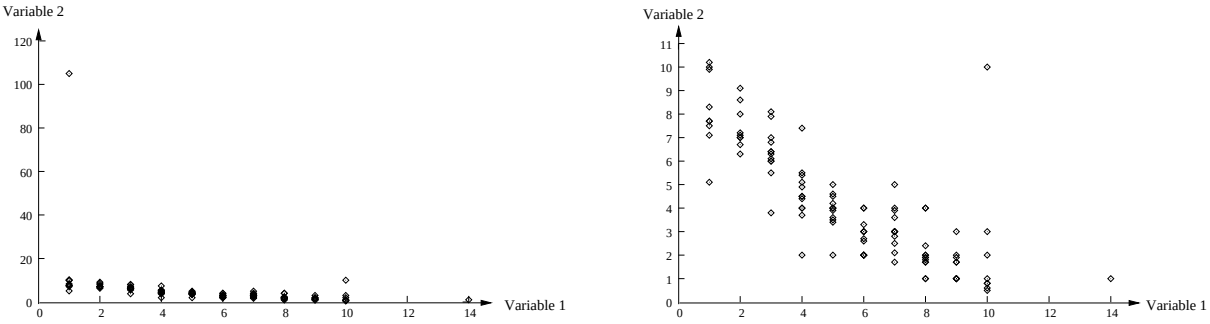


Abbildung 1: Auf der linken Seite Plot von 109 Beobachtungen zweier Variabler, auf der rechten Seite Plot der Daten nach Eliminierung des Ausreißers $(1, 105)$. Man kann erkennen, dass nach der ersten Iteration zur Eliminierung von Ausreißern ein weiterer Ausreißer identifiziert werden kann. In anderen Fällen können noch sehr viel mehr Iterationen nötig sein.

Abschnitt werden robuste Schätzer von Lage- und Streuungsparametern sowie das darauf basierende Verfahren zur robusten Berechnung von Korrelationsmatrizen erläutert. Anschließend wird im dritten Abschnitt eine robuste Hauptkomponentenanalyse vorgestellt.

Robuste Berechnung von Korrelationsmatrizen

Eine Möglichkeit zur robusten Berechnung einer Korrelationsmatrix ist die robuste Schätzung jedes einzelnen Korrelationskoeffizienten für sich, unabhängig von den anderen. Ein solches Verfahren betrachten wir im Folgenden. Es beruht auf der robusten Schätzung von Streuungsparametern, die in engem Zusammenhang zur robusten Schätzung von Lageparametern stehen.

Robuste Lageparameterschätzer

Gewöhnlich wird als Lageparameterschätzer der Stichprobenmittelwert

$$T_n = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{1}$$

wobei x_i die Beobachtungen und n deren Anzahl sind, verwendet. Dieser ist für die Normalverteilung der beste Schätzer, er ist jedoch nicht robust, da allen Beobachtungen das gleiche Gewicht gegeben wird. Im Folgenden werden einige robuste Lageparameterschätzer [3] vorgestellt, die in zwei Klassen eingeteilt werden können.

L-Schätzer

Seien $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$ die Ordnungsstatistiken einer Stichprobe der Größe n und $a_1, a_2, \ldots, a_n$ reelle Zahlen, $0 \leq a_i \leq 1$, $i = 1, \ldots, n$, so dass $\sum_{i=1}^{n} a_i = 1$. Ein L-Schätzer T mit den Gewichten $a_1, a_2, \ldots, a_n$ ist

$$T = \sum_{i=1}^{n} a_i X_{(i)}. \tag{2}$$

In dem implementierten Programm wurden folgende L-Schätzer verwendet:

- α-getrimmter Mittelwert:

$$T = \frac{1}{n(1-2\alpha)} \left((1-f)X_{(r+1)} + \sum_{i=r+2}^{n-r-1} X_{(i)} + (1-f)X_{(n-r)} \right), \tag{3}$$

- α-winsorisierter Mittelwert:

$$T = \frac{1}{n} \left(rX_{(r+1)} + \sum_{i=r+1}^{n-r} X_{(i)} + rX_{(n-r)} \right), \tag{4}$$

wobei $\alpha n = r + f$, r eine ganze Zahl und $0 \leq f \leq 1$.

L-Schätzer sind zwar leicht zu berechnen, je nach Wahl des Parameters α können jedoch Ausreißer ein zu großes Gewicht, oder aber richtige Beobachtungen am Rand ein zu kleines bekommen.

Eine Alternative sind

M-Schätzer

Ein M-Schätzer $T_n(x_1, \ldots, x_n)$ für eine Funktion ρ und die Stichprobe $x_1, \ldots, x_n$ ist der Wert von t, der die Funktion $\sum_{i=1}^{n} \rho(x_i; t)$ minimiert. Wenn man die Ableitung ψ von ρ kennt, muss der M-Schätzer der Gleichung

$$\sum_{i=1}^{n} \psi(x_i; t) = 0 \tag{5}$$

genügen.

Der bekannteste M-Schätzer ist der Stichprobenmittelwert. Dabei wird die Funktion $\sum_{i=1}^{n}(x_i - t)^2$ minimiert. Damit ist $\psi(u) = u$, wobei $u = x - t$. Dieser Schätzer ist allerdings nicht robust, da extreme Werte stärker in die Berechnung eingehen als alle anderen. Im Graphen der Funktion ψ (vgl. Abb. 2) erkennt man dies daran, dass die Funktion ψ unbeschränkt ist und konstante Steigung 1 hat. Eine robuste Alternative ist der Median. Dabei wird die Funktion $\sum_{i=1}^{n} |x_i - t|$ minimiert und damit ist $\psi(u) = sgn(u)$. Dieser Schätzer ist zwar robust, da die Funktion ψ beschränkt ist, das Problem bei dieser Funktion ist jedoch der Sprung im Punkt $(0, 0)$ (vgl. Abb. 2), der dazu führt, dass der Schätzer sehr stark von den mittleren Werten abhängt.

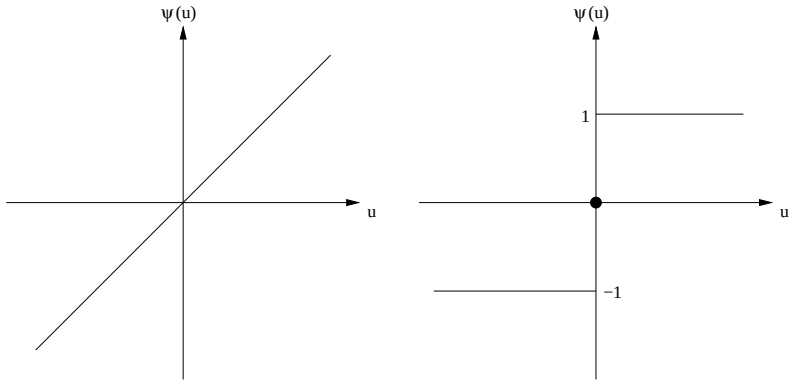


Abbildung 2: Graphen der ψ-Funktionen für die M-Schätzer Stichprobenmittelwert und Median

In dem implementierten Programm wurden drei robuste M-Schätzer verwendet, die allgemeinere Funktionen minimieren:

1. Huber (vgl. Abb. 3):

$$\psi(u) = \begin{cases} u, & \text{falls } |u| \leq 1; \\ 1, & \text{falls } u > 1; \\ -1, & \text{falls } u < -1. \end{cases} \tag{6}$$

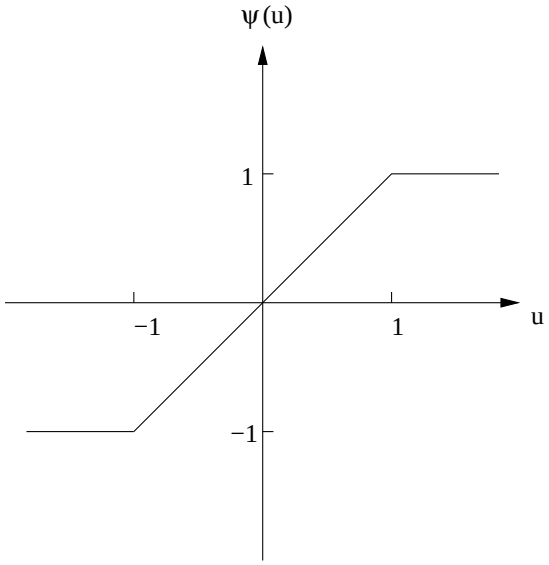


Abbildung 3: Graph der Huber-ψ-Funktion

2. Tukey's biweight (vgl. Abb. 4):

$$\psi(u) = \begin{cases} u(1-u^2)^2, & \text{falls } |u| \leq 1; \\ 0, & \text{falls } |u| > 1. \end{cases} \tag{7}$$

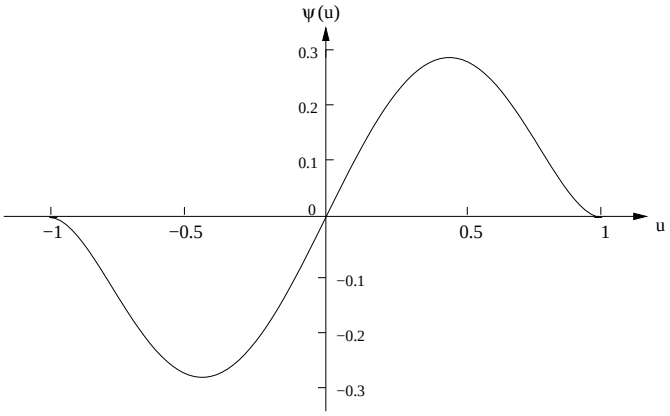


Abbildung 4: Graph der biweight-ψ-Funktion

3. Andrew's wave (vgl. Abb. 5):

$$\psi(u) = \begin{cases} \frac{1}{\pi} sin(\pi u), & \text{falls } |u| \leq 1; \\ 0, & \text{falls } |u| > 1. \end{cases} \tag{8}$$

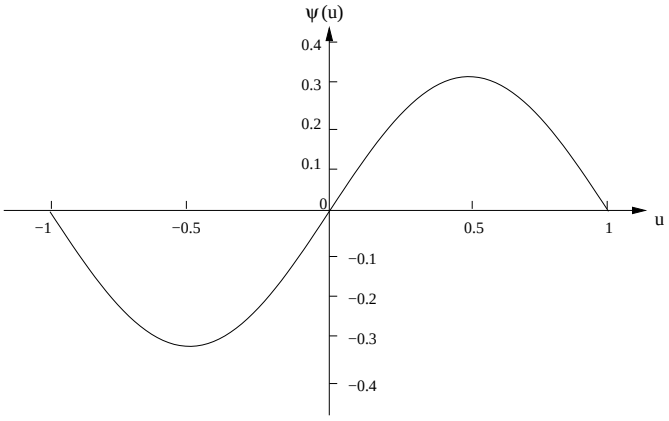


Abbildung 5: Graph der wave-ψ-Funktion

Dabei setzt sich die Huber-Funktion, um die beiden Nachteile im Falle von Median und Stichprobenmittelwert auszugleichen, aus den beiden entsprechenden ψ-Funktionen zusammen. Damit werden betragsmäßig großen Werten kleinere Gewichte gegeben, die Beobachtungen in der breiten Mitte erhalten aber alle das Gewicht 1. Nachteile bei dieser Funktion sind zum einen, dass die Funktion nicht glatt ist, und zum anderen, dass Ausreißer immer noch ein Gewicht > 0 erhalten. Diese Nachteile wurden im Falle der biweight- und wave-Funktion ausgeglichen.

Die M-Schätzer können mit Hilfe des Newtonverfahrens berechnet werden:

$$T_n^{(k+1)} = T_n^{(k)} - \frac{h\left(T_n^{(k)}\right)}{h'\left(T_n^{(k)}\right)}, \tag{9}$$

wobei

$$h = \sum_{i=1}^{n} \psi(u_i^{(k)}). \tag{10}$$

Damit erhält man einen 1-Schritt-M-Schätzer durch:

$$T_n = T_n^{(0)} + cS_n \frac{\sum_{i=1}^{n} \psi(u_i)}{\sum_{i=1}^{n} \psi'(u_i)}, \tag{11}$$

wobei c ein Tuning-Parameter, $T_n^{(0)}$ ein erster Lageparameterschätzer und S_n ein erster Streuungsschätzer sind. c und S_n erscheinen in der Gleichung, da

$$u_i = \frac{x_i - T_n^{(0)}}{cS_n}. \tag{12}$$

Als erster Lageparameterschätzer kann der Median und als Streuungsparameter die MAD (median absolute deviation) verwendet werden.

Robuste Streuungsparameterschätzer

Der gewöhnliche Streuungsparameterschätzer ist die Standardabweichung

$$S_n = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - T_n)^2}, \tag{13}$$

wobei x_i die Beobachtungen, n die Anzahl der Beobachtungen und T_n der Stichprobenmittelwert sind. Dieser ist jedoch nicht robust, da allen Beobachtungen das gleiche Gewicht gegeben wird. Robuste Alternativen dazu sind

- durch Trimmen bzw. Winsorisieren erhaltene Schätzer sowie
- A-Schätzer.

Die A-Schätzer

$$S_T = \frac{cMAD\sqrt{n}\sqrt{\sum_{i=1}^{n}\psi(u_i)^2}}{|\sum_{i=1}^{n}\psi'(u_i)|} \tag{14}$$

erhält man aus der asymptotischen Varianz der M-Lageparameterschätzer.

Robuste Berechnung von Korrelationskoeffizienten [4]

Sei S_X ein robuster Schätzer der Standardabweichung von X. Wir bilden die Summe Y_1 und Differenz Y_2 der skalierten Variablen:

$$Y_1 = \frac{X_1}{S_{X_1}} + \frac{X_2}{S_{X_2}} \text{ und } Y_2 = \frac{X_1}{S_{X_1}} - \frac{X_2}{S_{X_2}}. \tag{15}$$

Die robuste Berechnung des entsprechenden Korrelationskoeffizienten erfolgt dann mittels der Formel

$$r_{X_1X_2} = \frac{S_{Y_1}^2 - S_{Y_2}^2}{S_{Y_1}^2 + S_{Y_2}^2}. \tag{16}$$

Man erhält diese Formel über die Identität

$$Cov(X_1X_2) = \frac{1}{4}\{Var(X_1 + X_2) - Var(X_1 - X_2)\} \tag{17}$$

und einen Korrekturterm, da robuste Schätzer verwendet werden.

Ergebnisse im Beispieldatensatz (vgl. Abb. 1)

Verfahren	Korrelation	Ausreißer	Tuning-Parameter
einfache Korrelation	-0.35	keiner	
	-0.81	(1,105)	
	-0.88	(1,105), (10,10)	
	-0.89	(1,105), (10,10), (14,1)	
0.01-getrimmt	-0.81	(1,105), (10,10)	
0.05-getrimmt	-0.90	(1,105), (10,10), (14,1)	
0.01-winsorisiert	-0.75	(1,105), (10,10)	
0.05-winsorisiert	-0.90	(1,105), (10,10), (14,1)	
biweight	-0.90	(1,105), (10,10), (14,1)	9
	-0.91	(1,105), (10,10), (14,1)	6
wave	-0.90	(1,105), (10,10), (14,1)	6.6
Huber	-0.91	(1,105), (10,10), (14,1)	1.5

Die Korrelation der Daten bei Berücksichtigung der Ausreißer ist so gering, dass kein linearer Zusammenhang vermutet wird. Durch Eliminierung der Ausreißer und anschließende Berechnung [1] erhöht sich der Wert auf mehr als das Doppelte und es wird ein linearer Zusammenhang der Variablen festgestellt. Wie man sieht, liefern die robusten Verfahren in etwa die gleiche Korrelation wie nach Beseitigung von drei Ausreißern. Beim Trimmen bzw. Winsorisieren kommt es sehr auf den Parameter an, ansonsten liefern die verschiedenen Verfahren sehr ähnliche Ergebnisse.

Robuste Hauptkomponentenanalyse

Hauptkomponentenanalyse [4]

Die grundlegende Idee einer Hauptkomponentenanalyse ist die Dimensionsreduktion eines Datenraumes, bei der ein möglichst großer Teil der Information erhalten bleiben soll. Dazu ermittelt man orthogonale Linearkombinationen, so dass die Streuung der dadurch entstehenden Zufallsvariablen maximal ist. Diese neuen Zufallsvariablen sind die Hauptkomponenten.

Durchführung

Um die erste Hauptkomponente zu erhalten benötigt man den Koeffizientenvektor $\mathbf{a}^\top = (a_1, \ldots, a_p)$, so dass die Linearkombination $\mathbf{a}^\top \mathbf{x}$ in der Klasse aller Linearkombinationen mit $\mathbf{a}^\top \mathbf{a} = 1$ maximale Varianz besitzt.

Mit $\mathbf{S}$ als Kovarianzmatrix der Daten ist $\mathbf{a}^\top \mathbf{S} \mathbf{a}$ die Varianz von $\mathbf{a}^\top \mathbf{x}$.

Um $\mathbf{a}$ zu bestimmen ist deshalb der Vektor (ungleich dem Nullvektor) gesucht, der das Verhältnis

$$\frac{\mathbf{a}^\top \mathbf{S} \mathbf{a}}{\mathbf{a}^\top \mathbf{a}} \tag{18}$$

maximiert.

Der maximale Wert dieses Verhältnisses ist der größte Eigenwert λ_1 von $\mathbf{S}$ und die gesuchte Lösung für $\mathbf{a}$ der zu λ_1 gehörende Eigenvektor.

Damit kann eine Hauptkomponentenanalyse durchgeführt werden, indem die größten Eigenwerte und die zugehörigen Eigenvektoren der Kovarianzmatrix ermittelt werden. Dabei sollten soviele Hauptkomponenten berechnet werden, dass der Anteil der durch diese erklärten Streuung, d.h. der Anteil der Information, möglichst groß ist.

Alternativ zur Kovarianzmatrix kann auch die Korrelationsmatrix verwendet werden, wenn die Maßstäbe einheitlich sein sollen.

In Abb. 6 sind für die Daten aus obigem Beispiel (vgl. Abb. 1) die Hauptkomponentenachsen, die man wie beschrieben mittels der Kovarianzmatrix erhalten hat, eingezeichnet. Man kann erkennen, dass diese nicht das Gros der Daten widerspiegeln. Die Richtung der ersten Hauptkomponente entspricht genau der Richtung des Ausreißers $(1, 105)$, da dieser dafür sorgt, dass die Streuung in dieser Richtung am größten ist. Damit Ausreißer bei der Berechnung vernachlässigt werden, eliminiert man diese zunächst, oder verwendet robuste Verfahren, die im Folgenden vorgestellt werden.

Robuste Verfahren

Es gibt verschiedene Möglichkeiten, eine robuste Hauptkomponentenanalyse durchzuführen [6]. Die Verfahren unterscheiden sich darin, dass in den unter 1 genannten eine robuste Korrelations- bzw. Kovarianzmatrix berechnet und anschließend eine gewöhnliche Hauptkomponentenanalyse durchgeführt wird. Im Gegensatz dazu werden in dem unter 2 genannten Verfahren die Hauptkomponenten direkt robust geschätzt. Implementiert wurden das erste unter 1 und das unter 2 genannte Verfahren.

1.
   - Die Korrelationsmatrix wird elementweise wie oben beschrieben robust berechnet. Ein Nachteil dabei ist, dass die Korrelationsmatrix nicht positiv semidefinit sein muss.

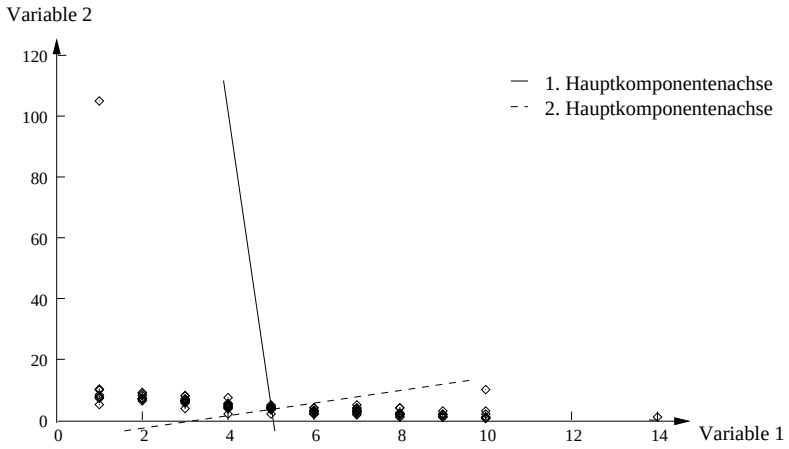


Abbildung 6: Plot der 109 Beobachtungen zweier Variabler mit eingezeichneten Hauptkomponentenachsen (berechnet mittels Kovarianzmatrix)

- Um den Nachteil des im letzten Punkt genannten Verfahrens auszugleichen, führt man ein shrinking durch [4]:

$$g(r_{jk}) = \begin{cases} z^{-1}\left(z(r_{jk}) + \Delta\right), & \text{falls } r_{jk} < -z^{-1}(\Delta); \\ 0 & \text{falls } |r_{jk}| \leq z^{-1}(\Delta); \\ z^{-1}\left(z(r_{jk}) - \Delta\right), & \text{falls } r_{jk} > z^{-1}(\Delta), \end{cases} \tag{19}$$

  wobei r_{jk} die Korrelationskoeffizienten sind, $z = \tanh^{-1}(r_{jk})$ und $\Delta = 0.25/\sqrt{n}$ mit $n =$ Anzahl der Beobachtungen. Diese Transformation wird so lange durchgeführt, bis die Matrix positiv definit ist. Der Nachteil bei diesem Verfahren ist, dass kleine Eigenwerte zu groß werden.

- Die Daten werden multivariat getrimmt [5]. Dazu werden für alle Beobachtungen die Mahalanobisdistanzen

$$d_m = \sqrt{(\mathbf{x}_m - \bar{\mathbf{x}})^\top \mathbf{V}^{-1} (\mathbf{x}_m - \bar{\mathbf{x}})} \tag{20}$$

  berechnet. Dann wird ein fester Anteil der Beobachtungen mit den größten Mahalanobisdistanzen beiseitegesetzt und $\bar{\mathbf{x}}$ sowie $\mathbf{V}$ neu geschätzt. Dieser Prozess wird abgebrochen, wenn sich $z(r_{jk})$ zwischen aufeinanderfolgenden Iterationen um nicht mehr ändert als 10^{-3} oder nach 25 Iterationen.

2. Die Hauptkomponenten werden mittels der Methode nach Campbell [2] robust geschätzt. Dieses Verfahren wird im Folgenden vorgestellt.

Robuste Hauptkomponentenanalyse nach Campbell [2]

M-Schätzung von Mittelwertsvektoren und Kovarianzmatrizen:

$$\begin{aligned} \bar{\mathbf{x}} &= \frac{\sum_{m=1}^{n} w_m \mathbf{x}_m}{\sum_{m=1}^{n} w_m} \\ \mathbf{V} &= \frac{\sum_{m=1}^{n} w_m^2 (\mathbf{x}_m - \bar{\mathbf{x}})(\mathbf{x}_m - \bar{\mathbf{x}})^\top}{\sum_{m=1}^{n} w_m^2 - 1} \end{aligned} \tag{21}$$

Berechnung der Gewichte:

$$w_m = \begin{cases} 1, & \text{falls } d_m \leq d_0; \\ \frac{d_0}{d_m} \exp \frac{-(d_m - d_0)^2}{2 \cdot (1.25)^2}, & \text{falls } d_m > d_0, \end{cases} \tag{22}$$

wobei $d_0 = \sqrt{v} + \sqrt{2}$ mit $v =$ Anzahl der Variablen und d_m wie in Glg. 20 definiert.

Algorithmus

1. Berechnung der Kovarianzmatrix mit den Gewichten 1 und Bestimmung des zum größten Eigenwert gehörigen Eigenvektors $\mathbf{u}_1$.

2. Berechnung der Hauptkomponentenscores gemäß $y_m = \mathbf{u}_1^\top \mathbf{x}_m$.

3. M-Schätzung von Mittelwert und Varianz der Scores sowie der zugehörigen Gewichte mittels der Formeln 21 und 22. Dabei werden der Median und $(0.74 \text{ Interquartilsabstand})^2$ als Anfangsschätzer benutzt.

4. Neue Berechnung der Kovarianzmatrix mittels der berechneten Gewichte nach Formel 21 und erneute Bestimmung des zum größten Eigenwert gehörigen Eigenvektors.

5. Wiederholung der Schritte 2 - 4 bis aufeinanderfolgende Schätzungen von Eigenwerten genügend nahe beieinander liegen.

6. Projektion der Daten auf den Raum, der orthogonal zu dem von den bisher berechneten Eigenvektoren aufgespannten ist, gemäß

$$\mathbf{x}_{im} = (\mathbf{I} - \mathbf{U}_{i-1}\mathbf{U}_{i-1}^\top)\mathbf{x}_m, \tag{23}$$

   wobei $\mathbf{U}_{i-1} = (\mathbf{u}_1, \ldots, \mathbf{u}_{i-1})$.

7. Wiederholung der Schritte 2 - 5 mit den $\mathbf{x}_{im}$ anstatt der $\mathbf{x}_m$, um den ersten Eigenvektor $\mathbf{u}$ zu berechnen.

8. Die Hauptkomponentenscores erhält man mittels

$$\mathbf{u}^\top \mathbf{x}_{im} = \mathbf{u}^\top (\mathbf{I} - \mathbf{U}_{i-1}\mathbf{U}_{i-1}^\top)\mathbf{x}_m, \tag{24}$$

   und somit ist

$$\mathbf{u}_i = (\mathbf{I} - \mathbf{U}_{i-1}\mathbf{U}_{i-1}^\top)\mathbf{u}. \tag{25}$$

9. Wiederholung der Schritte 6 - 8 bis alle Hauptkomponenten berechnet sind oder eine genügend große Gesamtstreuung erreicht ist.

Ergebnis

Für die Daten aus obigem Beispiel (vgl. Abb. 1) wurde eine robuste Hauptkomponentenanalyse durchgeführt. Die Hauptkomponentenachsen sind in Abb. 7 dargestellt. Man kann erkennen, dass diese nun das Gros der Daten widerspiegeln.

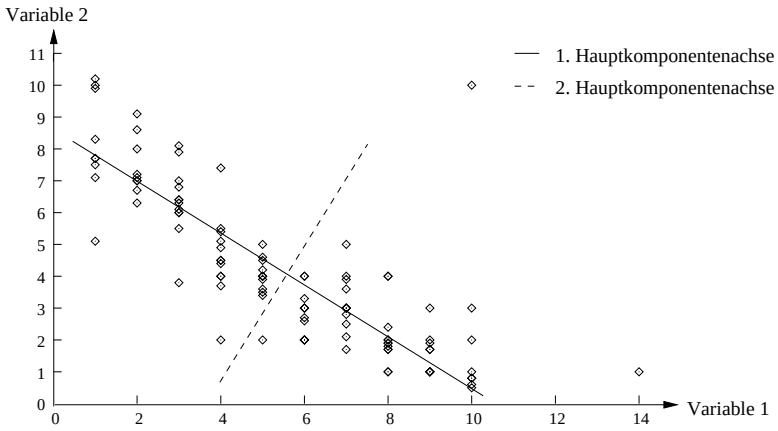


Abbildung 7: Plot der 108 Beobachtungen (außer dem Ausreißer) zweier Variabler mit eingezeichneten Hauptkomponentenachsen (robust berechnet mittels Kovarianzmatrix)

Literatur

1. Dickhaus, T. (2003). Statistische Verfahren für das Data Mining in einem Industrieprojekt. Interner Bericht FZJ-ZAM-IB-2003-08, ZAM Forschungszentrum Jülich.
2. Campbell, N. A. (1980). Robust Procedures in Multivariate Analysis I: Robust Covariance Estimation. Appl. Stat., 29, 231-237.
3. Hoaglin, D. C., Mosteller F., Tukey, J. W. (1983). Understanding Robust and Exploratory Data Analysis. John Wiley and Sons, Inc., New York.
4. Gnanadesikan, R. (1997). Methods for Statistical Data Analysis of Multivariate Observations, 2. Auflage. John Wiley and Sons, Inc., New York.
5. Seber, G.A.F. (1984). Multivariate Observations. John Wiley and Sons, Inc., New York.
6. Jackson, J.E. (1991). A User's Guide To Principal Components. John Wiley and Sons, Inc., New York.