Aspekte der Implementierung zeitdiskreter Algorithmen der numerischen Akustik

Frank Ranostaj Arild Lacroix

Institut für Angewandte Physik, Johann Wolfgang Goethe-Universität, Frankfurt am Main

Einführung

Die Berechnung der zeitlichen Entwicklung von Schallfeldern ist im allgemeinen nur numerisch möglich. Der Aufwand für diese Berechnung wächst mit der dritten Potenz der Ausdehnung des Schallfeldes, so daß eine effiziente Implementierung der Algorithmen notwendig ist.

Anhand eines aktuellen Personal Computers (2*Pentium III 450 MHz, BX-Chipsatz) werden Verfahren gezeigt, diesen effizient für numerische Verfahren einzusetzen. Insbesondere wird auf Vektorisierung, die durch eine Parallelisierung auf Befehlsebene realisiert wird, auf die Nutzung schneller Zwischenspeicher (Cache) und auf die Verwendung mehrerer Prozessoren (SMP) eingegangen.

Das hier verwendete numerische Verfahren, finite Differenzen, wurde aufgrund der Repräsentation der zu untersuchenden dreidimensionalen Daten gewählt. Diese resultieren aus einem Abfolge zweidimensionaler Schnittbilder und liegen als in einem kubischen Gitter angeordnete Volumenelemente vor, wobei die untersuchten Strukturen in der gleichen Größenordnung wie die Wellenlänge der Schwingungen des betrachteten Frequenzbereiches sind [1].

Finite Differenzen

Finite Differenzen bieten den Vorzug, daß ihnen eine Raumdiskretisierung in kubische Elemente zugrundeliegt, die unmittelbar den Volumendaten entspricht. In Verbindung mit einer Diskretisierung in Zeitrichtung wird die akustische Wellengleichung

$$\frac{\partial^2 u}{\partial t^2} = c\,\Delta u,$$

dargestellt mit dem Druck u, der Zeit t und der Schallgeschwindigkeit c, in ein System von gekoppelten Differenzengleichungen umgeformt[2]:

$$\begin{aligned} u_{t+1,x,y,z} - 2u_{t,x,y,z} + u_{t-1,x,y,z} \\ = K(u_{t,x+1,y,z} - 2u_{t,x,y,z} + u_{t,x-1,y,z} \\ + u_{t,x,y+1,z} - 2u_{t,x,y,z} + u_{t,x,y-1,z} \\ + u_{t,x,y,z+1} - 2u_{t,x,y,z} + u_{t,x,y,z-1}), \end{aligned}$$

wobei K eine Verbindung zwischen räumlicher und zeitlicher Schrittweite herstellt. Man erhält daraus ein explizites Lösungsverfahren in dem man sie nach $u_{t+1,x,y,z}$ auflöst und die daraus entstandene Beziehung für jeden Zeitschritt über alle Volumenelemente des Hohlraums iteriert. Für die weitere Berechnungen wird $K = \frac{1}{3}$ verwendet, da hierbei der größtmögliche zeitliche Schritt durchgeführt wird, bei dem das Verfahren stabil bleibt [3]. Desweiteren ist hiermit der Vorfaktor des Terms $u_{t,x,y,z}$ gleich Null. Die sich aus dieser Gleichung ergebende Dispersionsrelation ist in Bild 1 dargestellt.

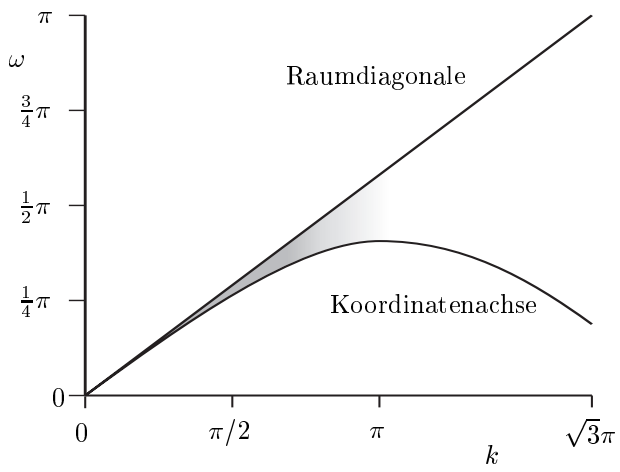

Bild 1: Dispersionsrelation entlang unterschiedlicher Raumrichtungen

Implementierung

Eine Realisierung der Iteration ist in Bild 2 wiedergegeben; dargestellt ist die innere Schleife. Die Variable `u` ist ein dreidimensionales Gleitkomma-Array. Da für die Berechnung eines neuen Wertes die beiden zeitlich vorhergehenden Werte benötigt werden, genügt es die Größe des Arrays in zeitlicher Dimension auf 2 zu beschränken und die Variable `t` in jedem Zeitschritt zwischen 0 und 1 zu alternieren. Desweiteren verläuft die innerste Schleife entlang einer Flächendiagonalen, um Zwischenergebnisse (`s0`, `s1`) des vorhergehenden Durchlaufs wiederzuverwenden und so die Anzahl der Gleitkommaoperationen auf 6 zu reduzieren. (Die Dereferenzierung von `u` bezüglich `t` und `x` wird nicht im Schleifenkern, wie hier der Übersichtlichkeit halber dargestellt, sondern vor Schleifeneintritt vorgenommen.)

```
for( n = n_ini; n < n_max; n += 2*dy+dz ){
  s0 = s1;
  s1 = u[t][x][n+dy] + u[t][x][n+dz];
  u[1-t][x][n] = K * ( s0 + s1
    + u[t][x+1][n] + u[t][x-1][n])
    - u[1-t][x][n];
}
```

Bild 2: Optimierte Schleife in C-Syntax

Vektorisierung

Die auf dem verwendeten System erzielte Rechenleistung liegt für diese Schleife bei 350 MFlops (Millionen Gleitkommaoperation pro Sekunde). Zur weiteren Steigerung kann der sogenannte ISSE-Befehlssatz verwendet werden, der die parallele Verarbeitung von vierelementigen Vektoren erlaubt. Aufgrund der Unterstützung dieser Befehle durch einen C-Compiler des Prozessorherstellers ändert sich der Programmtext des Schleifenkerns nicht, lediglich die Variablendeklaration ist entsprechend angepaßt. Damit wird eine Rechenleistung von 750 MFlops erzielt, entsprechend 1,5 GFlops für zwei Prozessoren.

Speichertransferrate

Geht man davon aus, daß die benötigten Daten einer yz-Ebene in dem Cache gepuffert werden können, so sind pro betrachtetem Volumenelement zwei lesende und ein schreibender Speicherzugriff mit je 4 Bytes notwendig; bei der im vorigen Abschnitt erzielten Leistung ergeben das 3 GByte/s für das 2-Prozessor-System. Dies liegt weit oberhalb der theoretisch möglichen Speichertransferrate von 800 MByte pro Sekunde. Eine günstigere Konfiguration ergibt sich, wenn man den Datensatz in Kuben unterteilt: Puffert man einen Datensatz beispielsweise der Größe $32 * 32 * 32$ Volumenelemente im Cache, kann man auf ihm 30^3-fach den Operator anwenden. Ohne neue Daten in den Cache zu übernehmen können in dem folgenden Zeitschritt 28^3 Operationen durchgeführt werden, daraufhin 26^3 Operationen und so fort. Aufgrund der zeitlichen Symmetrie der Wellengleichung kann dies auch in umgekehrter zeitlicher Richtung erfolgen, die Anzahl der Transfers pro berechnetem Volumenelement ist

$$b_{32} = \frac{32^3 + 2 \cdot 30^3}{30^3 + 2 \cdot (28^3 + 26^3 + ...)} \approx 0{,}54\,, \quad b_n \in O(n^{-1}),$$

so daß sich die Speichertransferrate für das 2-Prozessor-System auf 500 MByte pro Sekunde reduziert. Während des Speichertransfers werden die vierelementigen Vektoren aus entsprechend vier Kuben zusammengesetzt, dies armortisiert sich aufgrund des asymptotischen Aufwands von $O(n^{-1})$ und umgeht deren Plazierungsrestriktionen. Da sich mit zyklisch expandierenden Kuben keine dichte Packung realisieren läßt, werden die Zwischenräume mit Quadern gefüllt, wie in Bild 3 dargestellt.

Multiprocessing

Durch die verwendete Segmentierung der Daten wird die Verteilung der Berechnung auf mehrere Prozessoren und deren Synchronisierung vereinfacht. Lediglich die Reihenfolge

- Kuben (1. Satz),
- angrenzende Quader,
- Kuben (2. Satz),
- angrenzende Quader

muß eingehalten werden.

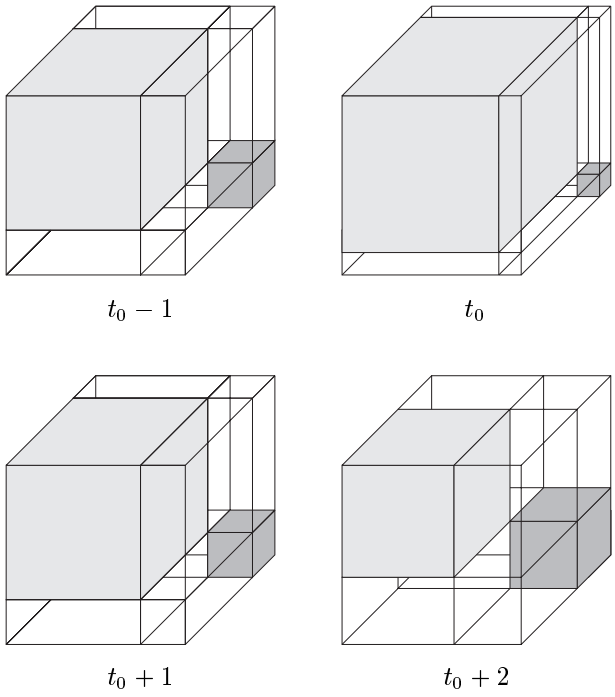

Bild 3: Zeitliche Variation der Unterteilung des Raumes in Kuben und Quader. Das Zentrum der Kuben ist ortsfest.

Zusammenfassung und Ausblick

In diesem Beitrag wird gezeigt, wie für ein numerisches Problem aus der Akustik, dessen Berechnungen ein hohes Maß an Datenlokalität besitzen, unter Ausnutzung der Merkmale moderner Prozessoren eine signifikante Beschleunigung der Berechnung zu erzielen ist.

Diese Techniken lassen sich in zweierlei Richtungen weiterverfolgen: zum einen kann man versuchen, den Operator weiter zu optimieren (gegenwärtig wird ein tetrapodenförmiger Laplaceoperator untersucht, der ein mehrfaches Leistungspotential besitzt), zum anderen kann man die gezeigte Methode der Datenstrukurierung erneut anwenden, um wesentlich größere Datensätze handhabbar zu machen. Dabei wird der Hauptspeicher — ähnlich wie einen Cache — als Zwischenspeicher für die auf einem File-Server liegende Daten verwendet, wobei auf diese Art mehrere Rechner ähnlich dem SMP nahezu verlustfrei parallelisiert werden können; die Größe des verwendeten Datensatzes ist nur durch die Festplattenkapazität beschränkt. Durch die gezeigte Datenstruktur wird die niedrige Bandbreite eines File-Servers vollständig kaschiert.

Literatur

[1] Ranostaj, F., Lacroix, A.:
Analyse der Akustischen Eigenschaften des Nasaltrakts.
Tagungsband der Deutschen Arbeitsgemeinschaft für Akustik DAGA '99, Berlin, 1999.

[2] Schwarz, H. R.: *Numerische Mathematik.*
B. G. Teubner, 3. Auflage, Stuttgart, 1993.

[3] Lacroix, A.: *Digitale Filter.*
R. Oldenbourg, 4. Auflage, München, 1996.