Übungen zur Einführung in die Computerphysik

Klessen / Spurzem

Sommersemester 2009

Blatt 12 (Abgabe bis spätestens 3. Juli 2009)

Präsenzübung: Darstellung von Diffusionsprozessen

Die Diffusionsgleichung in einer Dimension reduziert sich auf

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) \tag{1}$$

mit dem Diffusionskoeffizienten D.

Explizites Verfahren
Diskretisiert man die Gleichung auf einem äquidistanten Gitter

$$x_i = i \cdot L/N, \qquad i \in 0 \ldots N \tag{2}$$

mit einer gesamten Gebietslänge L und N Stützstellen, so lässt sich die Gleichung als Differenzengleichung umschreiben. In dem sogenannten *central differencing* sieht die Gleichung dann folgendermaßen aus:

$$\frac{1}{\Delta t}\left(u_i^{n+1} - u_i^n\right) = \frac{D}{(\Delta x)^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right). \tag{3}$$

Diese Gleichung ist erster Ordnung in der Zeit und explizit, da die Berechnung des Wertes für den Zeitschritt $(n+1)$ nur von Werten vergangener Zeitschritte n abhängt.

Stabilität
Das Verfahren ist stabil für Zeitschritte Δt, die kleiner sind als die charakteristische Zeit des Diffusionsproblems τ,

$$\Delta t \leq \frac{(\Delta x)^2}{2D} \equiv \tau\ . \tag{4}$$

Dieser Zeitschritt wird für kleine Δx oder große D sehr sehr klein, was das Verfahren in der Praxis schnell unpraktikabel macht.

Implizite Verfahren
Bezieht man in der Berechnung der Werte für t_{n+1} an der Stelle i die Werte des Zeitschritts

t_{n+1} mit ein, so nimmt die Gleichung folgende Gestalt an:

$$\frac{1}{\Delta t}\left(u_i^{n+1} - u_i^n\right) = \frac{D}{(\Delta x)^2}\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right). \tag{5}$$

Unter Verwendung der Abkürzung $a \equiv D\Delta t/(\Delta x)^2$ kann man für jeden räumlichen Punkt, der kein Randpunkt ist ($i \in 1 \ldots N-1$), die Gleichung schreiben als

$$-au_{i-1}^{n+1} + (1+2a)u_i^{n+1} - au_{i+1}^{n+1} = u_i^n \tag{6}$$

mit gegebenen Randwerten u_0 und u_N. Dieses System von Gleichungen lässt sich nun schreiben als

$$A \cdot \mathbf{u}^{n+1} = \mathbf{u}^n \tag{7}$$

mit der Matrix

$$A \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -a & 1+2a & -a & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -a & 1+2a & -a & \cdots & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & -a & 1+2a & -a & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -a & 1+2a & -a \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

Dises Verfahren ist stabil für *alle* Zeitschrittweiten, wobei ab einer gewissen Größe der Zeitschritte die Genauigkeit abnimmt.

Crank-Nicholson-Verfahren

Neben den rein impliziten und rein expliziten Methoden, die beide erster Ordnung in der Zeit sind, lassen sich diese beiden Verfahren kombinieren zu einem semi-impliziten Schema. Das Crank-Nicholson-Verfahren ist eine direkte Kombination der beiden oben vorgestellten Methoden mit folgender Differenzengleichung:

$$\frac{1}{\Delta t}\left(u_i^{n+1} - u_i^n\right) = \frac{D}{2(\Delta x)^2}\left[\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right)\right]. \tag{9}$$

Defieniert man sich ein b analog zum impliziten Verfahren als $b \equiv D\Delta t/2(\Delta x)^2$, so lassen sich die Funktionswerte wieder als Matrixgleichung formulieren

$$B_1 \cdot \mathbf{u}^{n+1} = B_2 \cdot \mathbf{u}^n \tag{10}$$

mit den Matrizen

$$B_1 \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -b & 1+2b & -b & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -b & 1+2b & -b & \cdots & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & -b & 1+2b & -b & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -b & 1+2b & -b \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

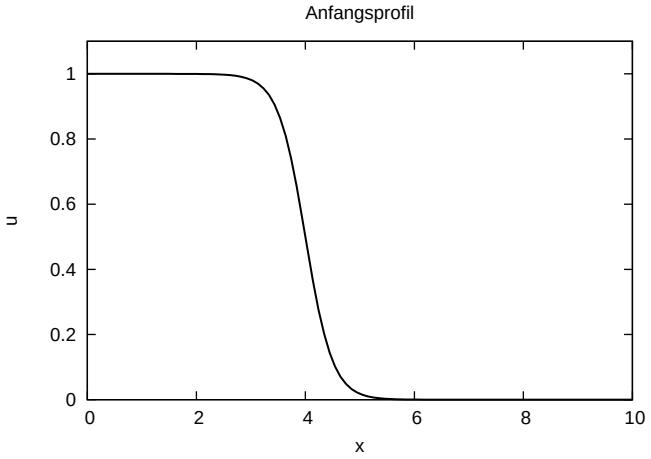
$$B_2 \equiv \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ b & 1-2b & b & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & b & 1-2b & b & \cdots & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & 0 & \cdots & b & 1-2b & b & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & b & 1-2b & b \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

Hausaufgabe **(20 Punkte)**

Verschiedene Methoden zur Berechnung des Diffusionsprozesses

Untersuchen Sie ein Gebiet der Länge $L = 10$ mit ca. 500 Stützstellen und einem Anfangsprofil der Form

$$u_j^0 = \frac{1}{2}\left[1 - \tanh\left(\frac{x_i - 4}{w \cdot \Delta x}\right)\right], \qquad w = 8. \tag{13}$$



Dieses Profil wird häufig anstelle einer Stufenfunktion verwendet, je kleiner w, desto kleiner die Breite der Stufe.

Teil 1

Implementieren Sie das explizite Schema und wenden Sie es auf das Anfangsprofil an. Betrachten Sie dazu die Änderung des Profils für 3 verschiedene Zeitschritte um den kritischen Stabilitätswert (0.9τ,1.1τ,1.2τ), und überzeugen Sie sich von der Stabilität des ersten Wertes. Plotten Sie das Profil nach 300 Zeitschritten und untersuchen Sie für die beiden Werte größer als τ das Verhalten des Profils bis nach ca. 200 Zeitschritten. Ab welchem Zeitschritt wird die Instabilität für die zwei Faktoren sichtbar? An welcher Stelle des Profils tritt die Instabilität auf?

Ändern Sie nun das Anfangsprofil auf $w = 4$ und wiederholen Sie die Berechnungen für 1.1τ und 1.2τ. Wann und wo treten die Instabilitäten jetzt auf?

Teil 2

Implementieren Sie nun das implizite Verfahren mit der gegebenen Matrix A und wenden Sie es auf das Anfangsprofil an. Die Matrix kann mit Hilfe der Numerical Recipes Funktionen `ludcmp` und `lubksb` gelöst werden (siehe Anhang). Überzeugen Sie sich von der Stabilität des Verfahrens, indem Sie Zeitschritte verwenden, die um einen Faktor 10, 100, 1000 über der Stabilitätsgrenze τ des expliziten Verfahrens liegen. Wie viele Schritte benötigt man bei den drei Faktoren ungefähr, um das Endprofil für u zu erhalten.

Wählen Sie einen Zeitpunkt t_1 aus, an dem das Profil noch Struktur hat. Vergleichen Sie für diesen Zeitpunkt die Profile für die drei unterschiedlichen Zeitschrittweiten.

Teil 3

Implementieren Sie das Crank-Nicholson-Verfahren und vergleichen Sie es mit dem impliziten Verfahren. Wählen Sie dazu $dt = 100\,\tau, 1000\,\tau, 10000\,\tau$ und plotten Sie je die Lösung beider Verfahren nach ca. 4 Integrationsschritten. Was fällt auf?

Numerical Recipes

Hier ein paar Hinweise zur Verwendung der Numerical Recipes Funktionen. In den Code muss lediglich die Datei `nr.h` in das Hauptprogramm eingebunden werden. Die verwendeten Matrizen werden mit der Klasse `Mat_xx A(dim,dim)` erstellt, wobei `xx` durch `DP` für `double`-Werte bzw. durch `INT` für `integer`-Werte zu ersetzen ist. Analoge Klassen gibt es für Vektoren: `Vec_DP x(dim)` bzw. `Vec_INT indx(dim)`. Auf die Elemente der Matrix kann wie gewohnt mit dem Operator `[][]`, also `A[i][j]` zugegriffen werden, bei den Vektoren ist dies analog `v[i]`.

Das Lösen der Matrixgleichung erfolgt hier mit zwei separaten Funktionen, der LU-Zerlegung und der Rücksubstitution. Die Aufteilung ist hier besonders von Vorteil, da sich die Matrix während der Berechnung nicht ändert. D wird zu Beginn einmal festgelegt, ebenso dx, d.h. die LU-Zerlegung braucht nur einmal zu Beginn der Simulation durchgeführt werden.

NR-Funktion `ludcmp` Diese Funktion zerlegt eine Matrix in eine obere rechte und eine untere linke normierte Dreiecksmatrix. Funktionsdefinition:

```
ludcmp(Mat_DP A, Vec_INT indx, double d)
```

A ist die gegebene Matrix, die nach Aufruf die LU-zerlegte A' enthält, `indx` speichert die Vertauschungen und `d` ob die Anzahl der Vertauschungen gerade oder ungerade ist. Die Funktion wird dann aufgerufen mit

```
NR::ludcmp(A,indx,d);
```

NR-Funktion `lubksb` Die Funktion `lubksb` löst das Gleichungssystem mit Hilfe der zuvor durchgeführten LU-Zerlegung. Funktionsdefinition:

```
ludcmp(Mat_DP A, Vec_INT indx, Vec_DP x)
```

A ist hier die von `lubksb` zerlegte Matrix, `indx` der Vertauschungsvektor und `d` enthält nach Aufruf der Funktion die Lösung des Gleichungssystems. Die Funktion wird dann aufgerufen mit

```
NR::lubksb(A,indx,x);
```