D-ITET, D-MATL
Dr. R. Käppeli
T. Gauksson

Serie 13

Diese Ferien-Serie behandelt Aufgaben zum Kapitel 6 der Vorlesung. Bitte beachten Sie die Bemerkung in den Vorlesungsnotizen (erste Seite oben von Kap. 6) bez. Prüfungsstoff. Nichtsdestotrotz kann es von Interesse sein ;-).

1. *Mehrkörpersimulation*

Lösen Sie mithilfe ihres RKF45-Lösers aus Serie 9[1] das folgende Problem:

Wir betrachten im Weltraum p Körper, die in einer Ebene liegen, und Koordinaten $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_p$, mit $\boldsymbol{r}_p \in \mathbb{R}^2$, und Masse $m_1, \ldots, m_p$ haben. Laut Newton's Gravitationsgesetz wirkt Körper k auf den Körper j mit der Kraft

$$Gm_j m_k \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{|\boldsymbol{r}_k - \boldsymbol{r}_j|^3}.$$

Die Kraft auf Körper $\boldsymbol{r}_j$ zeigt in die Richtung von $\boldsymbol{r}_k$ mit Stärke proportional zu $|\boldsymbol{r}_k - \boldsymbol{r}_j|^{-2}$. Wir wählen jetzt die Zeitskala so, dass die universale Gravitationskonstant gleich 1 ist. Die Bewegung der Körper, die über Gravitation aufeinander einwirken, ist gegeben durch

$$\ddot{\boldsymbol{r}}_j = A_j(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_p) = \sum_{k \neq j} m_k \frac{\boldsymbol{r}_k - \boldsymbol{r}_j}{|\boldsymbol{r}_k - \boldsymbol{r}_j|^3}.$$

Mit Hilfe von Anfangswerten $\boldsymbol{r}_j(0)$ und $\dot{\boldsymbol{r}}_j(0)$, kann man das Problem auf ein Anfangswertproblem für Systeme erster Ordnung zurückführen:

$$\begin{aligned}
x_{4(j-1)+1} &= r_{x,j} \quad \text{(die } x\text{-Komponente von } \boldsymbol{r}_j) \\
x_{4(j-1)+2} &= r_{y,j} \quad \text{(die } y\text{-Komponente von } \boldsymbol{r}_j) \\
x_{4(j-1)+3} &= \dot{r}_{x,j} \quad \text{(die } x\text{-Komponente von } \dot{\boldsymbol{r}}_j) \\
x_{4(j-1)+4} &= \dot{r}_{y,j} \quad \text{(die } y\text{-Komponente von } \dot{\boldsymbol{r}}_j)
\end{aligned}$$

[1] Sie finden auch eine Version in den Templates.

Bitte wenden!

Auf diesem Wege erhalten wir $x_1 = r_{x,1}$ und $x_8 = \dot{r}_{y,2}$. Das ODE System für $\boldsymbol{x}(t)$ ist dann

$$\frac{d}{dt}\begin{pmatrix} x_{4(j-1)+1} \\ x_{4(j-1)+2} \end{pmatrix} = \begin{pmatrix} x_{4(j-1)+3} \\ x_{4(j-1)+4} \end{pmatrix}$$
$$\frac{d}{dt}\begin{pmatrix} x_{4(j-1)+3} \\ x_{4(j-1)+4} \end{pmatrix} = \begin{pmatrix} A_{x,j}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_p) \\ A_{y,j}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_p) \end{pmatrix}$$

Dieses Problem ist in der Datei `body_problem.m` für 2 Körper und eine Wahl von Anfangsdaten vorimplementiert. Schreiben Sie eine Funktion `twobodyrhs.m` und rufen Sie das Template `body_problem.m` auf, um die Simulation zu sehen. Wie verhalten sich die adaptiven Schrittweiten?

2. *Molekular-Dynamik mit dem Verlet-Algorithmus*

In dieser Aufgabe betrachten wir eine Anwendung von sog. Strukturerhaltenden Verfahren auf Molekulardynamik (MD) Simulationen. Als Verfahren soll der berühmte *Verlet-Algorithmus* verwendet werden, welcher oft in praktischen MD-Simulationen verwendet wird wegen seinen "guten" Eigenschaften[2].

Als System betrachten wir einen (kleinen) gefrorenen Argon-Kristall bestehend aus $N = 7$ Argon Atomen. Sechs der Atome sind auf einem regelmässigen Hexagon um ein zentrales Atom verteilt. Um die Wechselwirkung zwischen den Atomen zu beschreiben benutzen wir ein Lennard-Jones Potential (welches oft ein elementarer Bestandteil einer verfeinerten Behandlung in praktischen MD-Simulationen ist). Das Potential der Wechselwirkung eines Atoms α mit einem Atom β ist dann gegeben durch

$$U_{\alpha\beta}(\boldsymbol{r}_\alpha, \boldsymbol{r}_\beta) = 4\varepsilon \left(\left(\frac{\sigma}{r_{\alpha\beta}} \right)^{12} - \left(\frac{\sigma}{r_{\alpha\beta}} \right)^{6} \right).$$

Hier sind ε und σ Konstanten, $\boldsymbol{r}_{\alpha\beta} = \boldsymbol{r}_\alpha - \boldsymbol{r}_\beta$ der Vektor von der Position von Atom β zum Atom α und $r_{\alpha\beta} = |\boldsymbol{r}_{\alpha\beta}|$. Das totale Potential des Systems ist dann gegeben durch die Summe

$$U_{\text{tot}}(\boldsymbol{r}_1, ..., \boldsymbol{r}_N) = \sum_{\substack{\beta=1 \\ \beta\neq\alpha}}^{N} U_{\alpha\beta}$$

und die Bewegunsgleichungn des Systems lauten dann

$$m\ddot{\boldsymbol{r}}_\alpha = -\frac{\partial U_{\text{tot}}}{\partial \boldsymbol{r}_\alpha}(\boldsymbol{r}_1, ..., \boldsymbol{r}_N) \quad (\alpha = 1, ..., N)$$

[2]Für mehr Informationen zu MD-Simulationen und dem Verlet-Algorithmus verweisen wir z.B. auf M. P. Allen, D. J. Tildesley, "Computer Simulation of Liquids", Oxford University Press, 1989.

Siehe nächstes Blatt!

wobei m die Masse eines Atoms ist.

Gegeben die Anfangs-Positionen und -Geschwindigkeiten der Atome haben wir also ein Anfangswert-Problem (AWP). Die Idee von MD-Simulationen ist nun dieses AWP zu lösen um dadurch Informationen über (z.B.) thermodynamische Grössen des Systems zu gewinnen. Hier wollen wir nicht weiter in die Kunst von MD-Simulationen eingehen, sondern einfach dieses "Spielzeug" Problem mit dem berühmten Verlet-Algorithmus lösen. Dieses Verfahren hat Konsistenz-Ordnung $p = 2$ und sehr gute Erhaltungs-Eigenschaften von gewissen Kerngrössen.

Der Verlet-Algorithmus berechnet die Position und Geschwindigkeit des α-sten Atom zum nächsten Zeitschritt $(\boldsymbol{r}_\alpha^{j+1}, \boldsymbol{v}_\alpha^{j+1})$ aus den derzeitigen $(\boldsymbol{r}_\alpha^{j}, \boldsymbol{v}_\alpha^{j})$ mit folgender Rekursion

$$\begin{aligned}
\boldsymbol{v}_\alpha^{j+1/2} &= \boldsymbol{v}_\alpha^{j} + \frac{1}{m_\alpha}\boldsymbol{F}_\alpha^{j}\frac{\Delta t}{2} \\
\boldsymbol{r}_\alpha^{j+1} &= \boldsymbol{r}_\alpha^{j} + \boldsymbol{v}_\alpha^{j+1/2}\Delta t \\
\boldsymbol{v}_\alpha^{j+1} &= \boldsymbol{v}_\alpha^{j+1/2} + \frac{1}{m_\alpha}\boldsymbol{F}_\alpha^{j+1}\frac{\Delta t}{2}.
\end{aligned}$$

Hier bezeichnet $\boldsymbol{F}_\alpha^j$ die totale Kraft zur Zeit t^j auf das α-ste Atom. Diese hängt von den Positionen aller Atome ab und ist hier gegeben durch

$$\boldsymbol{F}_\alpha^j = -\frac{\partial U_{\text{tot}}}{\partial \boldsymbol{r}_\alpha}(\boldsymbol{r}_1^j, ..., \boldsymbol{r}_N^j) = \sum_{\substack{\beta=1 \\ \beta\neq\alpha}}^{N} \boldsymbol{f}_{\alpha\beta}^j$$

wobei

$$\boldsymbol{f}_{\alpha\beta}^j = \frac{48\varepsilon}{\sigma^2}\left[\left(\frac{\sigma}{r_{\alpha\beta}^j}\right)^{14} + \frac{1}{2}\left(\frac{\sigma}{r_{\alpha\beta}^j}\right)^{8}\right]\boldsymbol{r}_{\alpha\beta}^j$$

die Kraft des Atom β auf Atom α zur Zeit t^j ist.

Lassen Sie das Matlab-Programm `argon.m` laufen. Dieses simuliert die Bewegung der 7 Argon Atome für 0.5 Nanosekunden, einmal mit dem Matlab Löser `ode45` und einmal dem obigen Verlet-Algorithmus, und plottet die zeitliche Entwicklung der totalen Energie

$$E_{\text{tot}} = \sum_{\alpha=1}^{N} \frac{m}{2}v_\alpha^2 + U_{\text{tot}}.$$

Was beobachten Sie?

3. *Lineare Advektions-Gleichung*

Bitte wenden!

Die lineare Advektions-Gleichung beschreibt die Advektion (von lat. *advectare* = heranbewegen) einer grösse $u(x,t)$ mit der (Advektions-) Geschwindigkeit a and ist gegeben durch:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0. \tag{1}$$

Betrachten wir das Gebiet $-\infty < x < \infty$ und die Anfangsbedingung

$$u(x,0) = u_0(x),$$

so ist die Lösung der Advektions-Gleichung gegeben durch

$$u(x,t) = u_0(x - at).$$

Hierbei ist $u_0(x)$ eine beliebige Funktion. Die Anfangsbedingung wird also mit der Geschwindigkeit a nach rechts (links) für $a > 0$ ($a < 0$) advektiert/transportiert.

Wir wollen nun die Gleichung im periodischen[3] Gebiet $x \in [0,1]$ numerisch lösen. Wie in der Vorlesung besprochen benutzen wir hierzu die sog. Linien-Methode. In der ersten Etappe diskretisieren wir den Raum $[0,1]$ (bzw. das Intervall hier) in ein $(M+1)$-Gitter durch einführen der Gitter-Knoten

$$x_i = i\Delta x$$

wobei $i = 0, \ldots, M-1$ und $\Delta x = \frac{1}{M}$ die Gitterkonstante ist. Die Approximationen der Lösung an den Gitter-Knoten bezeichnen wir mit $u_i(t) \approx u(x_i, t)$ und verwenden verschiedene finite Differenzen um die räumliche Ableitung in Gl. (1) zu approximieren. In der zweiten Etappe diskretisieren wir das Zeit-Intervall $[0,T]$ in eine gewisse Anzahl Zeitpunkte t^n wobei der obere Index n den Zeitschritt kennzeichnet

$$t^n = n\Delta t \quad (n = 0, 1, 2, \ldots)$$

und Δt die (Zeit)-Schrittweite. Die Approximationen der Lösung an den Gitter-Knoten zur Zeit t^n bezeichnen wir mit $u_i^n \approx u(x_i, t^n)$ und diskretisieren die Zeit-Ableitung in Gl. (1) mit dem expliziten Euler Verfahren. Wir betrachten nun folgenden Verfahren:

(i) Zentrale finite Differenzen im Raum und expliziter Euler in der Zeit:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x}$$

(ii) Linke finite Differenzen im Raum und expliziter Euler in der Zeit: (das sog. *first-order upwind scheme* für $a > 0$)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -a\frac{u_i^n - u_{i-1}^n}{\Delta x}$$

[3] D.h. es gilt $u(0,t) = u(1,t)$ und wir vermeiden das behandeln von komplizierteren Randbedingung.

Siehe nächstes Blatt!

(iii) Rechte finite Differenzen im Raum und expliziter Euler in der Zeit: (das sog. *first-order upwind scheme* für $a < 0$)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -a \frac{u_{i+1}^n - u_i^n}{\Delta x}$$

Alle Verfahren sind bereits im Matlab-Skript `linadv.m` implementiert.

a) Führen Sie das Skript `linadv.m` aus und variieren Sie dabei den Parameter `scheme` (welche zwischen den Verfahren wechselt) und alternieren Sie das Vorzeichen der Advektion-Geschwindigkeit `a`.
Wie in der Vorlesung besprochen, sollten Sie folgendes beobachten:

- Verfahren (i) ist instabil unabhängig vom Vorzeichen von a.
- Verfahren (ii) ist stabil für $a > 0$ und $0 < a\frac{\Delta t}{\Delta x} < 1$. Für $a < 0$ ist es instabil.
- Verfahren (iii) ist stabil für $a < 0$ und $-1 < a\frac{\Delta t}{\Delta x} < 0$. Für $a > 0$ ist es instabil.

b) Verwenden Sie die von Neumann Stabilitätsalyse um die Beobachtungen in **a)** zu erklären.
Hinweis: Studieren Sie die Beispiele 6 und 7 in Kapitel 6 der Vorlesungsnotizen.

Abgabe: Keine.