Übungen zu Physik auf dem Computer SS 2012

Übungsblatt 10: Lösen von Differentialgleichungen

28. Juni 2012

Allgemeine Hinweise

- Abgabetermin ist **Dienstag, 3.7.2012, 15:45** (ausnahmsweise)!
- Zur Abgabe schickst Du die Lösungsdatei(en) im Anhang einer Email an Deinen Tutor:
  - Florian (`floh@icp.uni-stuttgart.de`; Dienstag, 15:45–17:15)
  - Dominic (`dominic@icp.uni-stuttgart.de`; Dienstag, 15:45–17:15)
  - Olaf (`olenz@icp.uni-stuttgart.de`; Mittwoch, 15:45–17:15)
- Die Übungen werden in Gruppen von jeweils zwei oder drei Leuten bearbeitet. Diese dürfen sich gerne von Blatt zu Blatt unterscheiden. Aus formalen Gründen muss allerdings jeder von Euch eine eigene Lösung abgeben. Schreibt bitte auf die Lösungen, mit wem Ihr zusammengearbeitet habt, um uns das Korrigieren zu erleichtern.
- Die Übungen finden statt im CIP-Pool des Instituts für Computerphysik (ICP) im Pfaffenwaldring 27.

Aufgabe 10.1 (4 Punkte): Eindimensionale Poissongleichung

In dieser Aufgabe gilt es, die eindimensionale Poissongleichung $\frac{\mathrm{d}^2}{\mathrm{d}x^2}\phi(x) = \rho(x)$ numerisch zu lösen.

- 10.1.1 (1 Punkt) Diskretisiere die eindimensionale Poissongleichung mit Hilfe von finiten Differenzen in linearer Ordnung auf die selbe Weise, wie im Skript in Kapitel 6.1.3 die Besselgleichung diskretisiert wurde. Schreibe (am besten in LaTeX) die Gleichung für ϕ_k hin, die der Gleichung (6.15) im Skript entspricht, aus der sich die Matrix A des Gleichungssystems $A\vec{\phi} = \vec{\rho}$ ablesen lässt. Gib die LaTeX- oder PDF-Datei als Lösung ab.
- 10.1.2 (2 Punkte) Implementiere mit Hilfe der Diskretisierung aus der vorigen Aufgabe eine Pythonfunktion `solve_poisson1d_exact(rho,h)`, die die eindimensionale Poissongleichung numerisch annähert. `rho` soll dabei ein eindimensionales NumPy-Array sein, der N Werte der (Ladungs-)Dichte ρ enthält; `h` ist die Schrittweite zwischen den einzelnen Werten von `rho`. Als Randbedingungen sei $\phi = 0$ an den Rändern. Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält.

  Hinweis Zur Lösung des LGS soll dabei die Pythonfunktion `scipy.linalg.solve` verwendet werden. Die Lösung des LGS ist also exakt, dies gilt jedoch nicht für die Lösung der DGL selbst.
- 10.1.3 (1 Punkt) Nähere mit Hilfe der in der vorigen Aufgabe implementierten Funktion die Poissongleichung für $\rho(x) = \sin(\frac{2\pi}{L}x)$ an $N = 10, 50$ Punkten über dem Intervall $[0, 10]$ an. Plotte die numerischen Lösungen für beide Werte von N sowie die analytische Lösung in einen Plot. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.

Aufgabe 10.2 (3 Punkte): Relaxationsverfahren

- 10.2.1 (2 Punkte) Implementiere eine Pythonfunktion `sor(A,b,steps,omega=1.0)`, die das Relaxationsverfahren zur iterativen Lösung des LGS $A\vec{x} = \vec{b}$ implementiert. `steps` sei dabei die Anzahl an Iterationsschritten, `omega` der Parameter ω des Verfahrens. Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält.
- 10.2.2 (1 Punkt) Implementiere mit Hilfe der in der vorigen Aufgabe implementierten Pythonfunktion eine Pythonfunktion `solve_poisson1d_sor(rho,h,steps)`, die die Lösung der eindimensionalen Poissongleichung wie in Aufgabe 10.1.2 annähert, allerdings mit Hilfe von `steps` Schritten des Relaxationsverfahrens (verwende dabei $\omega = 1$, also das Gauß-Seidel-Verfahren). Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält. Verwende die Funktion wie in Aufgabe 10.1.3, um damit die Lösung der eindimensionalen Poissongleichung für $\rho(x) = \sin(\frac{2\pi}{L}x)$ für $N = 10$ Punkte anzunähern. Plotte die errechnete Lösung für 5, 10 und 20 Schritte des Relaxationsverfahrens, sowie die „exakte“ Lösung der Funktion `solve_poisson1d_exact` für 10 Punkte in einen Plot. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.

Aufgabe 10.3 (3 Punkte): Zweidimensionale Poissongleichung

In dieser Aufgabe soll die zweidimensionale Poissongleichung

$$\frac{\partial^2\phi(x,y)}{\partial x^2}\frac{\partial^2\phi(x,y)}{\partial y^2} = \rho(x,y)$$

numerisch gelöst werden. Die Diskretisierung der Gleichung ist im Skript in Kapitel 8 dargestellt.

Die Datei `/share/Courses/PC2012/10/rho.npy` enthält eine zweidimensionale NumPy-Matrix, die eine Ladungsverteilung darstellt, die Du mit Hilfe des folgenden Befehls in Python einlesen kannst:

```
rho = numpy.load('rho.npy')
```

Um eine zweidimensionale Matrix darzustellen eignet sich die Funktion `matplotlib.pyplot.imshow`. So kann mittels `matplotlib.pyplot.imshow(rho,interpolation='nearest')` die Ladungsverteilung ρ geplottet werden.

- 10.3.1 (2 Punkte) Implementiere eine Pythonfunktion `solve_poisson2d(rho,h)`, die entsprechend der Funktion `solve_poisson1d_exact(rho,h)` aus Aufgabe 10.1.2 die zweidimensionale Poissongleichung für die zweidimensionale Ladungsverteilung `rho` löst. Gib ein Pythonskript als Lösung ab, das die Pythonfunktion enthält.

  Hinweis

  - Davor solltest Du den Anfang von Kapitel 8 des Skripts nochmal genau lesen.
  - Der Pythonbefehl `rho_new=rho.reshape(N*N)` wandelt eine $N \times N$-Matrix in einen eindimensionalen Vektor der Länge N^2 um.
- 10.3.2 (1 Punkt) Löse mit Hilfe der Pythonfunktion aus der vorigen Aufgabe die Poissongleichung für die gegebene Ladungsverteilung und plotte die Lösung via `matplotlib.pyplot.imshow`. Gib ein Pythonskript als Lösung ab, das den Plot erzeugt.