Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. M. Ohlberger, Dr. B. Haasdonk

18.10.2007

Aufgaben zum Praktikum

Wissenschaftliches Rechnen

WS 2007/2008 — Blatt 1

Aufgabe 1 (Kennenlernen der Werkzeuge)
Für die Folgenden Aufgaben wird als Referenz auf die Beschreibung und Parameterbeschreibung der Hilfsprogramme auf der Kursseite verwiesen.

(a) Legen Sie ein Praktikumsverzeichnis in ihrem Home-Directory an (`mkdir`) und Laden Sie das Programm `ugly_pi.cc` von der Kursseite, welches Berechnung von π durch die Reihendarstellung

$$\pi = 4 \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

ermöglicht. Öffnen Sie die Datei in ihrem Editor (`emacs` o.ä.) in einem C++ Modus, so dass die Syntax des Programms farblich sichtbar ist, und führen Sie automatische Zeileneinrückung durch, um das Programm lesbar zu machen und speichern Sie es als `pi.cc`

(b) Compilieren Sie das Programm per Hand in der Shell (`g++`) ohne Optimierung aber mit Debugging-Informationen. Mit `./pi 10000` kann das Programm mit einem Kommandozeilenparameter gestartet werden, der die Anzahl der Reihenglieder in der Berechnung von π angibt. Messen Sie die Laufzeit des Programmes mit dem `time` Kommando, dessen Verwendung Sie mit `man time` einsehen können

(c) Starten Sie den Debugger GDB im Emacs via `M-x gdb` (die Meta-Taste `M` steht für die Escape-Taste) oder in der Unix-shell durch die Oberfläche `ddd`. Laden Sie Ihr Programm in den GDB durch ein geeignetes Kommando und starten Sie es mit dem Parameter 1000. Setzen Sie einen Breakpoint an den Anfang der main-Routine und starten Sie Ihr Programm erneut. Welchen Wert hat `argc` an dieser Stelle? Lassen Sie sich im Debugger den Wert von `argv[0]` und `argv[1]` ausgeben. Setzen Sie den Wert von `argc` auf 3, lassen Sie das Programm weiterlaufen und erläutern Sie das Ergebnis.

(d) Compilieren Sie das Programm erneut nun mit Optimierung und mit Debugging-Informationen. Messen Sie erneut die Zeit und vergleichen Sie mit den vorigen Messwerten. Starten Sie das Programm im Debugger und verfolgen Sie den Programmablauf, indem Sie das Programm zeilenweise ausführen lassen. Was beobachten Sie?

Aufgabe 2 (2D Punkttrajektorien)
Sei $v : \mathbb{R}^2 \to \mathbb{R}^2$ ein stetiges Geschwindigkeitsfeld. Die Bewegungsgleichung für ein Teilchen an Position $x(t)$ für Zeit $t \in [0, T]$ lautet $\frac{d}{dt}x(t) = v(x(t))$. Mit Anfangsposition $x(0) = x_0$ ist dies ein Anfangswertproblem einer gewöhnlichen Differentialgleichung. Es soll das Euler-Verfahren zur Diskretisierung implementiert werden.

(a) Implementieren Sie eine Funktion `velocity` mit geeigneter Parameterliste, die die Punkt-Auswertung des Geschwindigkeitsfeldes $v(x) = (-x_2, x_1)^T$ in einem Punkt $x = (x_1, x_2)^T \in \mathbb{R}^2$ realisiert.

(b) Schreiben Sie ein Haupt-Programm, das als Kommandozeilen-Parameter die Anfangsposition x_0, die Zeitschrittweite Δt, die Endzeit T und den Dateinamen einer Ausgabedatei übergeben bekommt. Das Programm soll das Euler-Verfahren zu dem vorgegebenen Geschwindigkeitsfeld durchführen. Das Programm soll eine Textdatei erzeugen, die zeilenweise die Position zu den jeweiligen Zeitpunkten enthält.

(c) Visualisieren Sie die Trajektorie des Punktes mit `gnuplot`, indem Sie die erzeugte Text-Datei dort einladen. Veranschaulichen Sie sich den Effekt von größeren Endzeiten, Variation des Startpunktes und Verringern der Zeitschrittweite, indem Sie diese mit dem Programm verändern und die Ergebnisse wiederholt darstellen.

(d) Überlegen Sie sich zu obigem Geschwindigkeitsfeld, wie die exakte analytische Lösung $x(t)$ aussieht. Implementieren Sie eine Funktion, die zu gegebenem Anfangswert x_0 und Zeit t diese exakte analytische Position $x(t)$ berechnet.

(e) Erweitern Sie ihr Programm so, dass Sie den *experimental order of convergence (EOC)* des Verfahrens für die Endposition $x(T)$ bestimmen können. Was erhalten Sie asymptotisch für kleiner werdende Δt?