Arbeitsblatt 2

Allgemeine Bemerkungen

- Jeweils 2 Studenten können eine Aufgabe gemeinsam bearbeiten.
- Für den Übungsschein müssen alle 6 Aufgaben gelöst werden.
- Es empfiehlt sich, mit den verschiedenen Parametern der Aufaben zu spielen. Alle Vorgaben sind nur Vorschläge.

Aufgabe 2 - Primfaktorzerlegung

Schreiben Sie ein Programm, das in einer Endlosschleife Zahlen abfragt und für diese die Primfaktorenzerlegung ausgibt, z.B. in der Form 200 = 2 * 2 * 2 * 5 * 5. Bei Eingabe von 0 soll das Programm beendet werden. Das Programm soll einen Fehler melden, wenn die eingegebene Zahl zu groß für den verwendeten `integer`-Datentyp ist. (Hinweis: `iostat`-Parameter der `read`-Funktion verwenden.)

Aufgabe 3 - ein Fraktal

Eine Punktmenge $\{(x_0, y_0), (x_1, y_1), (x_2, y_2), ...\}$ im $\mathbb{R}^2$ werde durch folgenden stochastischen Algorithmus iterativ erzeugt:

1. $x_0 = 0, \;\; y_0 = 0$
2. (x_{n+1}, y_{n+1}) wird aus (x_n, y_n) berechnet:
   a) Mit der Wahrscheinlickeit $p = 0,01$ wird folgende Transformation gewählt:
   $$\begin{aligned} x_{n+1} &= 0 \\ y_{n+1} &= 0,16\; y_n \end{aligned}$$
   b) Mit der Wahrscheinlickeit $p = 0,07$ wird folgende Transformation gewählt:
   $$\begin{aligned} x_{n+1} &= 0,2\; x_n - 0,26\; y_n \\ y_{n+1} &= 0,23\; x_n + 0,22\; y_n + 1,6 \end{aligned}$$
   c) Mit der Wahrscheinlickeit $p = 0,07$ wird folgende Transformation gewählt:
   $$\begin{aligned} x_{n+1} &= -0,15\; x_n + 0,28\; y_n \\ y_{n+1} &= 0,26\; x_n + 0,24\; y_n + 0,44 \end{aligned}$$
   d) Mit der Wahrscheinlickeit $p = 0,85$ wird folgende Transformation gewählt:
   $$\begin{aligned} x_{n+1} &= 0.85\; x_n + 0,04\; y_n \\ y_{n+1} &= -0,04\; x_n + 0,85\; y_n + 1,6 \end{aligned}$$

Verwenden Sie die Vogle-Bibliothek, um ein Bild dieser fraktalen Punktmenge zu erzeugen.

Bemerkungen

- Die erzeugte Punktmenge wird im Bereich $-2.3 < x < 2.7, \;\; 0 \leq y < 10$ liegen.
- Das Programm sollte ca. 100 000 Punkte plotten. Ein Punkt sollte sofort nach seiner Berechnung gemalt werden, so daß man den Fraktal wachsen sehen kann.
- Die Subroutine `random_number(r)` liefert eine zwischen 0 und 1 gleichverteilte `real`-Zufallszahl `r`.
- Zur Übersetzung verwende man z.B.
  `gfortran -O2 -o fraktal fraktal.f90 /usr/local/lib/libvogle.a -lX11`

Die Vogle-Grafikbibliothek

- Die vollständige Dokumentation ist auf der Webseite zur Vorlesung zu finden.
- Beim Kompilieren ist die Bibliothek am Ende des Kommandos mit anzugeben:
  `gfortran ... ... /usr/local/lib/libvogle.a -lX11`
- Einige kommentierte Anweisungen:

```
module vogle
    integer, parameter:: Black = 0, Red = 1, Green = 2, Yellow = 3, &
         Blue = 4, Magenta = 5, Cyan = 6, White = 7
end module vogle
```

  Dieses Modul definiert die 8 Farben, die Vogle kennt. Es sollte am Anfang des Quellcodes (noch vor der ersten `program ...` Anweisung) stehen.

```
use vogle
...
call prefsize(400, 500)
call vinit('X11')
```

  Mit diesen Befehlen wird die Größe des Grafik-Fensters auf 400×500 Bildpunkte (pixel) gesetzt und das Grafiksystem gestartet.

```
call ortho2(-2.5, 3., .0, 10.)
call color(White)
call clear()
call color(Green)
```

  Das Koordinatensystem, welches im Grafikfenster verwendet werden wird, wird so festgelegt, daß der linke Rand $x = -2.5$, der rechte Rand $x = 3$, der obere Rand $y = 10$ und der untere Rand $y = 0$ enstpricht. Anschließend wird der Hintergrund weiß und das Fenster leer gemacht. Für die folgenden Zeichenbefehle wird die Farbe Grün eingestellt.

```
call point2( 0.5, 4.4)
```

  Damit wird bei $(x, y) = (0.5, 4.4)$ ein Punkt gemalt.

```
integer i, getkey
...
i = getkey()
call vexit
```

  Die `getkey()`-Funktion wartet auf einen Tastendruck. Anschließend wird das Grafikfenster geschlossen.

  Beachten Sie, daß die Argumente der Vogle-Routinen wie `point2(), ortho2(),...` im Allgemeinen vom Typ `real(4)` sind!

Aufgabe 4 - Der Wurm

Programmieren Sie mit der Vogle-Bibliothek ein (sehr einfaches) Spiel: Mit Hilfe von 4 Tasten (z.B. "a", "s", "w", "y" für links, rechts, oben, unten) wird ein "Wurm" im Grafikfenster bewegt, welcher bei der Bewegung mit jedem Schritt immer länger wird. Er wächst sozudagen in die Richtung, in die er bewegt wird. Die Welt des Wurms habe die Topologie eines Torus (periodische Randbedingungen). Die Taste "q" beende das Spiel.

Bemerkungen

- Verwenden Sie die Vogle-Funktion `getkey()` zum Einlesen der Tastatureingaben. Die Funktion `ichar()` wandelt `character` in `integer` um, die Funktion `achar()` macht die umgekehrte Umwandlung.
- Vogle kennt unter anderem die Subroutinen
  - `circle(x, y, radius)` zum Malen eines Kreises mit Mittelpunkt (x, y).
  - `rect(x1, y1, x2, y2)` zum Malen eines Rechtecks
  - `polyfill(logical)`: Nach `call polyfill(.true.)` werden die Kreise, Rechtecke etc. mit Farbe gefüllt gemalt.

Aufgabe 5

Eine untere Schranke für 2π, den Umfang des Einheitskreises, erhält man durch die Summe der Seitenlängen eines dem Einheitskreis eingeschriebenen regelmäßigen n-Ecks. Wenn man die Eckenzahl stets verdoppelt, erhält man die Rekursion

$$s_{2n} = \sqrt{2 - \sqrt{4 - s_n^2}} \qquad \text{mit Startwert} \qquad s_4 = \sqrt{2} \tag{1}$$

für die Länge s_n *einer* Seite des eingeschriebenen regelmäßigen n-Ecks. Die Folge $\{n\, s_n\}$ konvergiert monoton von unten gegen den Grenzwert 2π:

$$n\, s_n \uparrow 2\pi \tag{2}$$

Als relativen Fehler kann man definieren

$$\epsilon_n = \left| \frac{n\, s_n - 2\pi}{2\pi} \right| \tag{3}$$

Aufgabenstellung

1. Berechnen Sie s_n und daraus ϵ_n für $n = 2^k, \quad k = 3, ..., 35$ mit den 3 verschiedenen Fließkommatypen `real(4), real(8)` und `real(10)`. Schreiben Sie dazu ein Programm, welches eine Tabelle mit 4 Spalten in eine Datei schreibt. Die erste Spalte soll $k = 3, ..., 35$ enthalten und die folgenden 3 Spalten die mit verschiedener Genauigkeit berechneten Fehler $\epsilon_{2^k}^{(4)}, \epsilon_{2^k}^{(8)}, \epsilon_{2^k}^{(10)}$.

2. Plotten Sie die relativen Fehler $\epsilon_{2^k}^{(4,8,10)}$ für die Fließkommatypen `real(4), real(8)` und `real(10)` als Funktionen von k in einem Plot mit logarithmischer y-Achse. Analysieren Sie das entstehende Bild.

3. Die Gleichung

$$s_{2n} = \frac{s_n}{\sqrt{2 + \sqrt{4 - s_n^2}}} \tag{4}$$

   ist mathematisch äquivalent zu Gleichung (1). (Nachrechnen!) Verwenden Sie in Ihrem Programm anstelle von Gleichung (1) diese Variante und vergleichen Sie die Stabilität der Algorithmen.

Bemerkungen

- Verwenden Sie beim Compilieren des Programms die Compileroption `"-ffloat-store"`. Dies verhindert, daß sich bei den `real(8)`-Berechnungen die höhere Genauigkeit der internen Prozessorregister auswirkt.
- Beachten Sie, das einfache `integer` nur Werte bis ca. 2^{31} darstellen können.
- Es ist nicht einfach möglich, einen `kind`-Parameter als Argument einer Funktion zu übergeben oder ähnliches. Sie müssen den Algorithmus je dreimal für die 3 Fließkommaarten (z.B. in drei Subroutinen) in das Programm einbauen. Parametrisierte Typen, mit denen man das eleganter lösen kann, sind im Fortran-2008 Standard enthalten, aber noch nicht in `gfortran` implementiert.
- Zum Plotten der Fehler kann man das Programm `"gnuplot"` verwenden. Mehr zu `gnuplot` findet man z.B. in der Wikipedia `http://de.wikipedia.org/wiki/Gnuplot` und unter den dort angegebenen Links.

Kurzbeschreibung von gnuplot:

- Das Programm wird mit dem Kommando `gnuplot` auf der Kommandozeile gestartet. Es meldet sich mit der Ausgabe

  ```
  gnuplot>
  ```

  und hier können Kommandos eingegenben werden.

- Einige einfache Kommandos:

`plot sin(x)`	einfacher Plot einer Funktion
`set logscale y`	logarithmische y-Achse
`set xrange [0:20]`	lege Achsenbereich der x-Achse fest
`set yrange [0:1]`	lege Achsenbereich der y-Achse fest
`set key left bottom`	verschiebe Legende nach links unten
`plot "res.dat" using 1:3 with lp`	Plotte die Daten aus der Datei "res.dat", verwende die erste Spalte in der Datei als x-Werte und die 3. Spalte als y-Werte, zeichne durch Linien verbundene Datenpunkte

`set term push;`
`set term postscript enh;`
`set out "bild.ps";`
`replot; set term pop`
	Diese Kommandos erzeugen eine Postscriptdatei "bild.ps" des Plots, die dann ausgedruckt werden kann

- Zum in der Aufgabe gewünschten Plot sind folgende Kommandos nötig (die Ausgabedatei heiße "res.dat"):

  ```
  set logscale y
  set key left bottom
  plot "res.dat" us 1:2 with lp, "" us 1:3 with lp, "" us 1:4 with lp
  ```

- Auch die Gnuplot-Kommandozeile hat eine history-Funktion: Mit den Pfeiltasten kann man vorige Kommandos zurückholen, editieren und erneut eingeben.

Aufgabe 6 - Eindimensionale Wärmeleitung

Wärmeleitung

In einem homogenen Körper M mit der Wärmeleitfähigkeit κ bestehe zur Zeit $t = 0$ eine Temperaturverteilung $T_0(\vec{x})$.

Durch Wärmeleitung wird sich diese Temperaturverteilung $T(\vec{x}, t)$ mit der Zeit ändern. Dies beschreibt die Wärmeleitungsgleichung

$$\frac{\partial T(\vec{x}, t)}{\partial t} = \kappa \, \Delta T(\vec{x}, t)$$

Diese Gleichung hat eine eindeutige Lösung, wenn man eine Anfangsverteilung $T_0(\vec{x})$ (Anfangswert) und den zeitlichen Verlauf der Temperatur am Rand ∂M des Körpers $T_{\partial M}(\vec{x}, t)$ (Randwert) vorgibt.

Im eindimensionalen Fall (Wärmeleitung in einem Stab der Länge L) reduziert sich die Gleichung zu

$$\frac{\partial T(x, t)}{\partial t} = \kappa \, \frac{\partial^2 T(x, t)}{\partial x^2}$$

Die Anfangsbedingung $T(x, 0) = T_0(x)$ beschreibt die Temperaturverteilung längs des Stabes zu einer Zeit $t = 0$. Die Randwertbedingungen $T(0, t) = f_1(t)$ und $T(L, t) = f_2(t)$ beschreiben den zeitlichen Verlauf der von außen vorgegebenen Temperatur an den beiden Enden $x = 0$ und $x = L$.

Wir betrachten im Folgenden die Randbedingungen $T(0, t) = T(L, t) = 0$.

Numerische Lösung

Explizites Schema Numerisch läßt sich dieses Problem durch Diskretisierung lösen. Wir wählen n gleichverteilte Stützstellen zwischen $x_0 = 0$ und $x_{n+1} = L$ auf dem Stab, der Gitterabstand ist $a = \frac{L}{n+1}$. Die Zeitentwicklung wird in Zeitschritten der Größe Δt berechnet. Die einfachste Diskretisierung der 2. Ableitung nach x ist

$$\frac{\partial^2}{\partial x^2} T(x) \Big|_{x=x_i} \approx \frac{1}{a^2} \left(T_{i-1} - 2T_i + T_{i+1} \right)$$

Hier wurde die Abkürzung $T_i := T(x_i)$ verwendet. Analog ist $T_{i,k} := T(x_i, t_k)$.

Es gibt nun verschiedene Möglichkeiten, Diskretisierungen der Zeit- und Ortsableitungen zu kombinieren. Der einfachste Fall ist das sogenannte explizite Schema:

$$\frac{T_{i,k+1} - T_{i,k}}{\Delta t} = \kappa \frac{1}{a^2} \left(T_{i-1,k} - 2T_{i,k} + T_{i+1,k} \right)$$

Dieses Gleichungssystem läßt sich sofort nach den $T_{i,k+1}$ auflösen. Mit $z = \frac{\kappa \Delta t}{a^2}$ erhält man

$$T_{i,k+1} = z(T_{i-1,k} - 2T_{i,k} + T_{i+1,k}) + T_{i,k}$$

Mit Hilfe der Matrix

$$\mathbf{W} = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots 0 & \\ 0 & 1 & -2 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}$$

kann das in Vektorschreibweise auch als

$$\vec{T}_{k+1} = (\mathbf{1} + z\mathbf{W})\vec{T}_k$$

geschrieben werden, wobei $\mathbf{1}$ die $n \times n$ Einheitsmatrix und $\vec{T}_k$ der Vektor der längs des Stabes gemessenen Temperaturen zur Zeit t ist. Damit ist die Berechnung der Temperaturverteilung $\vec{T}_k = \{T(i, k), i = 1 \dots n\}$ auf dem Stab zu einem Zeitpunkt k aus der Verteilung zur Zeit 0 auf eine einfache Reihe von k Matrixmultiplikationen zurückgeführt.

Leider hat dieser einfache Algorithmus, das "explizite Schema", gravierende Nachteile. Die Diskretisierungsfehler sind relativ groß (von der Ordnung Δt) und der Algorithmus ist für $z > \frac{1}{2}$ instabil, d.h. die berechnete Zeitentwicklung entfernt sich immer mehr von der tatsächlichen Lösung.

Crank-Nicholson Schema Im Gegensatz dazu ist das Crank-Nicholson Schema stabil und seine Diskretisierungsfehler sind nur von der Ordnung $(\Delta t)^2$. Hierbei verwendet man auf der rechten Seite der diskreten Wärmeleitungsgleichung den Mittelwert der räumlichen Ableitungen zur Zeit k und zur Zeit $k+1$:

$$\frac{T_{i,k+1} - T_{i,k}}{\Delta t} = \frac{\kappa}{2}\left[\frac{T_{i-1,k+1} - 2T_{i,k+1} + T_{i+1,k+1}}{a^2} + \frac{T_{i-1,k} - 2T_{i,k} + T_{i+1,k}}{a^2}\right]$$

In Vektorschreibweise führt das auf die Gleichung

$$\left(\mathbf{1} - \frac{z}{2}\mathbf{W}\right)\vec{T}_{k+1} = \left(\mathbf{1} + \frac{z}{2}\mathbf{W}\right)\vec{T}_k$$

$(\mathbf{1} - \frac{z}{2}\mathbf{W})$ ist eine Tridiagonal-Matrix, so daß die Lösung dieses linearen Gleichungssystems numerisch nicht sehr aufwendig ist.

Aufgabe

- Schreiben Sie ein Programm, das die eindimensionale Wärmeleitungsgleichung im expliziten Schema löst. Die Randwertbedingung sei einfach $T(0,t) = T(L,t) = 0$. Eingangsdaten seien die Anzahl n der Stützstellen, die Anfangsverteilung der Temperatur $T_i, i = 1 \ldots n$, der Parameter z und die Zahl der Zeitschritte m. Fertigen Sie (z.B. mit Gnuplot) dreidimensionale Plots der Temperaturverteilung $T(x,t)$ an. Probieren Sie verschiedene Anfangsverteilungen aus, z.B. $\sin(\pi x/L)$ (Stabmitte heißer ist als der Rand) oder $\sin^2(2\pi x/L)$ (2 "heiße Flecke" auf dem Stab). Zeigen Sie an Beispielen, daß der Algorithmus für $z > \frac{1}{2}$ instabil wird.

- Schreiben Sie eine Routine, die das lineare Gleichungssystem

  $$\mathbf{A}\vec{x} = \vec{y}$$

  für eine symmetrische Tridiagonalmatrix

  $$\mathbf{A} = \begin{pmatrix} a_1 & b_1 & 0 & \ldots & 0 \\ b_1 & a_2 & b_2 & \ldots 0 & \\ 0 & b_2 & a_3 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & b_{n-1} & a_n \end{pmatrix}$$

  löst, d.h. aus den Inputdaten $\{a_i, b_i, y_i\}$ den Vektor $\{x_i\}_{i=1\ldots n}$ berechnet. Eine Möglichkeit hierzu ist LU-Zerlegung mit nachfolgender Vorwärts- und Rückwärtssubstitution.

- Verwenden Sie diese Routine, um die eindimensionale Wärmeleitungsgleichung im Crank-Nicholson Schema zu lösen. Das Programm verwende dieselben Eingangsdaten wie das Programm für das explizite Schema. Zeigen Sie an Beispielen, daß der Algorithmus auch für $z > \frac{1}{2}$ stabil bleibt und plotten Sie die Ergebnisse.

- Hinweise: Sinnvolle Parameter sind z.B. $n = 50$, Anzahl der Zeitschritte 100...300, $z = 0.3, 0.7, 1, 2$ Das Programm sollte die Tripel $i, k, T(i,k)$ in eine Datei "w.dat" schreiben, ein Datentripel pro Zeile. Dies kann dann mit Gnuplot (s. Arbeitsblatt 2) geplotted werden mit den Kommandos
  `set style data lines; splot "w.dat"`
  Das entstehende Bild kann mit der Maus gedreht werden.