Allgemeine Bemerkungen

- Von den 5 Aufgaben müssen 4 gelöst werden.
- Mindestens 2 Aufgaben müssen in Fortran gelöst werden. (Es können auch alle Aufgaben in Fortran gelöst werden.)
- Jeweils zwei Studenten können eine Aufgabe gemeinsam bearbeiten.
- Verwendung von Fortran mit den Lapack-Bibliotheken im Pool:

```
. ifort-env
ifort meinprog.f90 -o meinprog -llapack -lblas -lg2c
```

Aufgabe 1

Aufgabe Schreiben Sie ein Programm, welches in einer Endlosschleife Zahlen zwischen 1 und 999 einliest und sie als römische Zahlen wieder ausgibt. Bei Eingabe einer Zahl außerhalb dieses Bereichs soll sich das Programm beenden.

Hinweise Für die römischen Zahlen sind die römischen Ziffern

Zeichen	I	V	X	L	C	D	M
Wert	1	5	10	50	100	500	1000

zu verwenden. Die bekannte Subtraktionsregel:

- Wenn eine niedrigerwertige römische Ziffer links von einer höherwertigen steht, ist sie zu subtrahieren (Beispiel: IV = 4, CM = 900).

soll nur in ihrer minimalen, klassischen Form angewendet werden, d.h. es gelten die Einschränkungen:

- Von V und X darf nur I subtrahiert werden; von L und C darf nur X subtrahiert werden; von D und M darf nur C subtrahiert werden.
- Mehrfache Subtraktionen von einer römischen Ziffer sind unzulässig.

Damit wird z.B. 39 zu XXXIX und nicht zu IXL; 99 zu XCIX und nicht zu IC; 8 zu VIII und nicht zu IIX; 450 zu CDL und nicht zu LD.
Die längste Zahl im angegebenen Bereich $1 \leq n \leq 999$ ist damit 888 = DCCCLXXXVIII (12 Zeichen).

Aufgabe 2 - Iterationen und Chaos: die "logistische Abbildung"

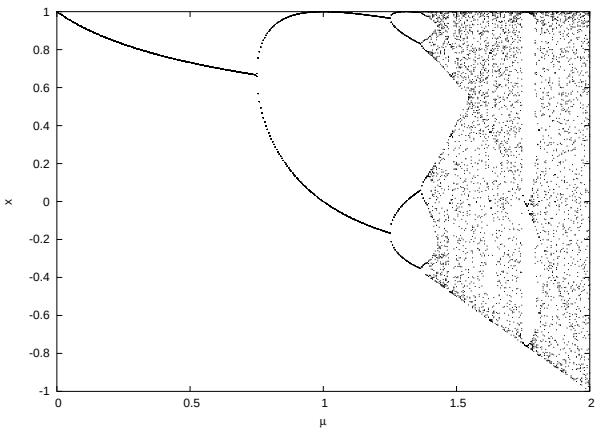
Einführung

- Was geschieht, wenn man, beginnend mit einem beliebigen Anfangswert $x \in [-1, 1]$, die Abbildung $\boxed{f : x \to 1 - 0.5x^2}$ immer wieder anwendet und $x, f(x), f(f(x)), f(f(f(x))), f(f(f(f(x)))), \ldots$ berechnet? Die Folge konvergiert, unabhängig vom Startwert, gegen den Fixpunkt $\sqrt{3}-1 \approx .7320$, für den $x = 1 - 0.5x^2$ gilt.*

- Wie ist das Iterationsverhalten von $\boxed{f_1 : x \to 1 - x^2}$? Eine erste Vermutung wäre, daß ebenfalls alle Folgen mit dem Anfangswert aus einem gewissen Intervall gegen den Fixpunkt $x_u = 1 - x_u^2$, $x_u = \frac{\sqrt{5}-1}{2} \approx .6180$ laufen. Ein kleines Computerexperiment zeigt, daß diese Annahme falsch ist: Alle Startwerte $x \in [-1, 1]$ außer x_u selbst führen zu einem Zyklus $0, 1, 0, 1 \ldots$ mit der Periode 2, z.B.:
$0.9, 0.19, 0.9639, 0.07089, 0.9949, 0.01002, 0.9998, 0.0002, 0.9999, 8{*}10^{-8}, 0.999999, 10^{-14}, 1, 0, 1, 0, 1, 0, \ldots$
1 und 0 sind, neben $\frac{-1\pm\sqrt{5}}{2}$, die Lösungen der Fixpunktgleichung 4. Grades $x = f_1(f_1(x))$. In der Sprache der Renormierungsgruppentheorie sind 1 und 0 attraktive, stabile Fixpunkte dieser Iteration und x_u ist ein repulsiver, instabiler Fixpunkt. Jede Abweichung von x_u führt dazu, daß der Iterationsprozeß von diesem Fixpunkt wegläuft.

- Die Verallgemeinerung führt auf folgendes Problem: *Untersuche das Verhalten der Iteration der "logistischen Abbildung"* $\boxed{f_\mu : x \to 1 - \mu x^2}$ *in Abhängigkeit vom Parameter*† $\mu \in [0, 2]$.

- Das folgende Bild zeigt die stabile Fixpunktmenge (den "Attraktor") als Funktion von μ.



Für $\mu < \mu_1 = \frac{3}{4}$ gibt es einen stabilen Fixpunkt. Bei $\mu_1 = \frac{3}{4}$ kommt es zur *Bifurkation*, jetzt ist der stabile Endzustand der Iteration ein periodischer Zyklus der Länge 2 (für $\mu = 1$ besteht der Attraktor aus $\{0, 1\}$). Bei $\mu_2 = \frac{5}{4}$ erfolgt eine weitere Bifurkation, nun beobachtet man einen stabilen Zyklus der Länge 4. Im Weiteren kommt es zu einer ganzen Kaskade von Periodenverdoppelungen $2^{k-1} \to 2^k$ für bestimmte Parameterwerte μ_k. Die Folge μ_k konvergiert gegen einen kritischen Wert $\mu_\infty = 1.401155189\ldots$, den *Feigenbaum-Attraktor*. Der Feigenbaum-Attraktor markiert die Schwelle zwischen Ordnung und Chaos: Für noch größere μ verhalten sich die Iterationsfolgen chaotisch, mit der Ausnahme gewisser "Fenster". Z.B. wird bei $\mu = 1.75$ ein stabiler Zyklus der Periode 3 beobachtet, der für ansteigende μ-Werte wieder eine Kaskade von Periodendopplungen aufweist.

*Die andere Wurzel der quadratischen Fixpunktgleichung, $-\sqrt{3} - 1$, ist von einem Startwert aus $[-1, 1]$ offensichtlich nicht erreichbar.

†Der Wertebereich von μ garantiert, daß das Intervall $[-1, 1]$ nicht verlassen wird.

Aufgabe

- Teil 1: Erzeugen Sie (mit C oder Fortran) die Daten für das Bild der vorhergehenden Seite. Dazu wurden für Startwerte von x_0 zwischen -1 und 1 in Schritten von 0.05 und für μ-Werte in Schritten von 0.005 zwischen 0 und 2 jeweils 15 000 Iterationen gemacht und der erreichte Wert ausgegeben in eine Datei "feigenbaum.dat", die in der erste Spalte μ und in der 2. Spalte x_{15000} enthält:

```
 0 1
0.005 0.995049
0.01 0.990195
0.015 0.985434
...
```

  Zum Plotten wurde `gnuplot` verwendet. Nach dem Aufruf `"gnuplot"` auf der Kommandozeile gibt man

```
gnuplot> plot "feigenbaum.dat" notitle
```

  ein und bekommt den Plot auf den Bildschirm. Die Kommandos

```
gnuplot> set linestyle 1 pointtype 7 pointsize .2
gnuplot> set term postscript
gnuplot> set out "feigenbaum.ps"
gnuplot> plot "feigenbaum.dat" notitle linestyle 1
gnuplot> set term x11
```

  erzeugen das Bild als druckbare Postscript-Datei `feigenbaum.ps`. Mehr zu `gnuplot` findet man z.B. auf `http://www.we.fh-osnabrueck.de/fbwe/vorlesung/edv2/gplot/gplot.html` oder zum Ausdrucken unter `http://www.we.fh-osnabrueck.de/fbwe/vorlesung/edv2/gplot/gplot.ps`.

- Teil 2: Schreiben Sie ein Programm, welches μ_k für $k = 3, 4, 5$ berechnet.

  Man kann z.B. für ein beliebiges x_0 und für einen μ-Wert zwischen $\mu_2 = \frac{5}{4}$ und $\mu_\infty = 1.4011$ einige zehntausend Iterationen machen und dann testen, welche Periodizität der erreichte Zyklus hat. Da Gleitkommaoperationen stets nur eine endliche Genauigkeit haben, sollte man statt `if(a==b)` besser etwas wie `if(fabs(a-b)<1.e-15)` als Test auf Gleichheit verwenden. Wenn man einen Wert μ_1 mit der Periode N und einen Wert μ_2 mit der Periode $2N$ gefunden hat, teste man $\frac{\mu_1+\mu_2}{2}$ und schachtele so den Bifurkationspunkt ein. Wenn man für einen μ-Wert auch nach sehr vielen Iterationen keinen periodischen Zyklus findet, modifiziere man ihn (oder x_0) leicht. Natürlich sind auch andere (und bessere) Algorithmen denkbar.

Aufgabe 3: Verwendung von LAPACK

Das `LAPACK` *(Linear Algebra Package)* ist ein Open-Source Softwarepaket von Fortran77-Unterprogrammen zur Lösung von Standardproblemen der linearen Algebra. Es enthält Programme zur Lösung linearer Gleichungssyteme, zur Bestimmung von Eigenwerten und Eigenvektoren, zur Faktorisierung von Matrizen und anderes. `LAPACK` selbst setzt auf die Routinen der `BLAS`-Bibliothek *(Basic Linear Algebra Subroutines)* auf, welche Vektor- und Matrizenadditionen und -multiplikationen in für die jeweilige CPU optimierter Form implementieren.

Die Namen der `LAPACK`-Routinen folgen der Konvention, daß der erste Buchstabe den verwendeten Zahlentyp (`s` single precision, `d` double precision, `c` complex, `z` double complex), die nächsten 2 Buchstaben den Matrizentyp (z.B. `he` hermitesch, `or` orthogonal, `sy` symmetrisch, `tr` Dreiecksmatrix, `st` symmetrisch tridiagonal u.a.) und die letzten 2-3 Buchstaben die Art der Berechnung (`ev` eigenvalues, `sv` solve linear system u.v.a.) bezeichnet.

- Informationen über `LAPACK` finden Sie auf `http://www.netlib.org/lapack`.
- Informieren sie sich über die Routine zur Eigenwertberechnung `dsyev`. Sie berechnet die Eigenwerte doppelt genauer symmetrischer Matrizen.
- Schreiben Sie ein Fortran-Programm, welches eine Zahl N einliest und die Eigenwerte der $N \times N$-Matrix $a_{ij} = i * j$ durch Aufruf von `dsyev` berechnet und ausgibt.
- Zusatzaufgabe: Mathematischer Hintergrund.
  Die Matrix hat genau einen Eigenwert ungleich Null, und eine allgemeine Formel für diesen Eigenwert als Funktion von N läßt sich durch einige Beispiele mit kleinem N leicht erraten. Man kann die Formel auch herleiten aus der Tatsache, daß die Matrix die Form eines Tensorprodukts $\vec{v} \otimes \vec{v}$ hat.

Aufgabe 4: Schrödingergleichung in einer Dimension

Ein quantenmechanisches Teilchen sei in einer Dimension (auf einer Linie) durch ein äußeres Feld zwischen $x = -1$ und $x = 1$ eingeschlossen. Die stationären Zustände des Teilchens werden durch Lösungen der Schrödingergleichung

$$-\frac{\hbar}{2m}\frac{d^2}{dx^2}\Psi(x) = E\Psi(x)$$

beschrieben. Diese Gleichung hat ein diskretes Spektrum von Lösungen $\Psi_n(x)$ mit Energien E_n. Berechnen Sie näherungsweise durch Diskretisierung nach der Methode der finiten Differenzen die niedrigsten 3 Eigenwerte E_n und erzeugen Sie einen Plot der dazugehörigen Eigenfunktionen:

- Wir wählen Einheiten, in denen $\frac{\hbar}{2m} = 1$.
- Wir wählen ein Gitter mit N Punkten[‡] zwischen $x_0 = -1$ und $x_{N+1} = 1$, Gitterabstand $a = \frac{2}{N+1}$. Die Randbedingungen sind $\Psi(x_0) = \Psi(x_{N+1}) = 0$ und wir verwenden die Abkürzung $\Psi_i = \Psi(x_i)$.
- Finite-Differenzen-Methode: die einfachste Diskretisierung der 2. Ableitung ist

  $$\left.\frac{d^2}{dx^2}\Psi(x)\right|_{x=x_i} \approx \frac{1}{a^2}\left(\Psi_{i-1} - 2\Psi_i + \Psi_{i+1}\right)$$

  Für jeden Gitterpunkt $i = 1, \ldots, N$ gilt nun die diskretisierte Schrödingergleichung:

  $$(\Psi_{i-1} - 2\Psi_i + \Psi_{i+1}) = -a^2 E\Psi_i$$

  Zusammen mit den Randbedingungen kann man das als $N \times N$-Matrixgleichung schreiben:

  $$\begin{pmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots 0 & \\ 0 & -1 & 2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \vdots \\ \Psi_N \end{pmatrix} = a^2 E \begin{pmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \vdots \\ \Psi_N \end{pmatrix}$$

  Das heißt, wir suchen Eigenwerte $\lambda_i = a^2 E_i$ und Eigenvektoren $\vec{\Psi} = \{\Psi_i\}$ dieser symmetrischen tridiagonalen Matrix.
- Hierfür kann man die `LAPACK`-Routine `dstev` verwenden. Neben der Ausgabe der drei niedrigsten Eigenwerte schreibe man die drei dazugehörigen Eigenvektoren in eine Datei `e.dat` in folgendem Format:

  ```
  do  i=1,N
     write(2, *) -1.+i*a, z(i,1), z(i,2), z(i,3)
  end do
  ```

  Dabei ist `z` die Matrix, die `dstev` als 5. Argument übergeben bekommen hat und a die Gitterkonstante, `-1+i*a` also der Wert der x-Koordinate.
- Zum Plotten der Eigenfunktionen kann man wieder `gnuplot` verwenden:

  ```
  gnuplot> plot "e.dat" using 1:2, "e.dat" using 1:3, "e.dat" using 1:4
  ```

  ein und bekommt den Plot auf den Bildschirm. Die Kommandos

  ```
  gnuplot> set term postscript
  gnuplot> set out "e.ps"
  gnuplot> replot
  gnuplot> set term x11
  ```

  erzeugen das Bild als Postscript-Datei `e.ps`.

[‡] N zwischen 100 und 1000, probieren Sie verschiedene Werte aus

Aufgabe 5: Wärmeleitungsgleichung in einer Dimension

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

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

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 dies 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 $x = L$.

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 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 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 zurückführen.

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

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

Zum Glück ist $(\mathbf{1} - \frac{z}{2}\mathbf{W})$ 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

  $$\mathbf{A}\vec{u} = \vec{v}$$

  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 $\{a_i, b_i, v_i\}_{i=1\ldots n}$ den Vektor $\{u_i\}_{i=1\ldots n}$ berechnet. Eine Möglichkeit hierzu ist LU-Zerlegung mit nachfolgender Vorwärts- und Rückwärtssubstitution. Wenn Sie Fortran verwenden, können sie natürlich die entsprechende LAPACK-Routine **dstev** einsetzen.
- 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.