Universität Stuttgart – Institut für Computerphysik

# Klausur

## Physik auf dem Computer SS 2013

JP Dr. Axel Arnold  Dr. Olaf Lenz  Tobias Richter
Elena Minina

7 August 2013

| | |
|---|---|
| **Name** | |
| **Vorname** | |
| **Matrikelnummer** | |

### Hinweise

- In der Regel gibt der verfügbare freie Platz einen Hinweis darauf, welchen Umfang die Lösung haben sollte.
- Die Antworten sind, soweit möglich, ganzzahlig oder Brüche mit kleinen Nennern.
- Lies Dir *alle* Fragen am Anfang durch, bevor Du anfängst, sie zu beantworten.
- Falls der Platz nicht ausreichen sollte, verwende zusätzliche Blätter. Beschrifte diese unbedingt mit Deinem Namen und Matrikelnummer!
- Die Maximalpunktzahl ist 100.
- Zum Bestehen der Klausur sind 50 Punkte notwendig.

# Viel Erfolg!

# 1 Lineare Algebra I (10 Punkte)

**Aufgabe 1:** (1 Punkt)

Welches numerische Verfahren kannst Du zum exakten Lösen eines linearen Gleichungssystems verwenden? Nenne zwei Varianten.

**Antwort:**

- Gausselimination (kanonisch, Total-, Spaltenpivotwahl)
- LU-Zerlegung
- Cholesky

**Aufgabe 2:** (4 Punkte)

Löse das folgende lineare Gleichungssystem mit Hilfe der Gaußelimination mit kanonischer Pivotwahl und Rücksubstitution.

$$\begin{pmatrix} 2 & 4 & 0 \\ 1 & 3 & 2 \\ 2 & 3 & -1 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 46 \\ 57 \\ 25 \end{pmatrix}$$

**Antwort:**

$$\left(\begin{array}{ccc|c} 2 & 4 & 0 & 46 \\ 1 & 3 & 2 & 57 \\ 2 & 3 & -1 & 25 \end{array}\right) \quad \text{(Eliminiere Zeilen 2 und 3)}$$

$$\Rightarrow \left(\begin{array}{ccc|c} 2 & 4 & 0 & 46 \\ 0 & 1 & 2 & 34 \\ 0 & -1 & -1 & -21 \end{array}\right) \quad \text{(Eliminiere Zeile 3)}$$

$$\Rightarrow \left(\begin{array}{ccc|c} 2 & 4 & 0 & 46 \\ 0 & 1 & 2 & 34 \\ 0 & 0 & 1 & 13 \end{array}\right) \quad \text{(Rücksubstitution)}$$

$$\Rightarrow \mathbf{x} = \begin{pmatrix} 7 \\ 8 \\ 13 \end{pmatrix}$$

**Aufgabe 3:** (3 Punkte)

Invertiere die folgende Matrix mit Hilfe der Gausselimination mit kanonischer Pivotwahl.

$$\begin{pmatrix} 1 & 3 \\ 2 & 7 \end{pmatrix}$$

**Antwort:**

$$\left(\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 2 & 7 & 0 & 1 \end{array}\right)$$ Eliminieren der zweiten Zeile

$$\left(\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & 1 & -2 & 1 \end{array}\right)$$

$$\left(\begin{array}{cc|cc} 1 & 0 & 7 & -3 \\ 0 & 1 & -2 & 1 \end{array}\right)$$

$$A^{-1} = \begin{pmatrix} 7 & -3 \\ -2 & 1 \end{pmatrix}$$

**Aufgabe 4:** (2 Punkte)

Was macht folgende Pythonfunktion? Was enthält `a` nach dem Ausführen des Algorithmus?

```
def doit(a):
    N, N = a.shape
    for k in range(0,N-1):
        for i in range(k+1,N):
            lam = a[i,k]/a[k,k]
            a[i,k+1:N] = a[i,k+1:N] - lam*a[k,k+1:N]
            a[i,k] = lam
    return a
```

**Antwort:**

LU-Zerlegung. `a` enthält anschließend oberhalb der Diagonale die resultierende Matrix U, unterhalb der Diagonalen die Matrix L, wobei die Diagonalelemente mit Wert 1 ergänzt werden müssen.

# 2 Darstellung von Funktionen (10 Punkte)

**Aufgabe 5:** (2 Punkte)
Berechne das Taylorpolynom dritten Grades der Funktion $f(x) = \ln(x)$ um den Punkt $x_0 = 1$.

**Antwort:**

$$\begin{aligned} P(x) &= \ln(1) + \frac{1}{1}(x-1) - \frac{1}{2!}(x-1)^2 + \frac{2}{3!}(x-1)^3 + \mathcal{O}(x^4) \\ &= x - 1 - \frac{1}{2}(x-1)^2 + \frac{1}{3}(x-1)^3 + \mathcal{O}(x^4) \end{aligned}$$

**Aufgabe 6:** (3 Punkte)
Berechne die Koeffizienten des Hornerschemas für ein Polynom 3. Grades mit den Nullstellen $-1$, 2, 5. Dabei sei der führende Koeffizient $c_3 = 1$.

**Antwort:**

$$\begin{aligned} P(x) & \\ &= (x+1)(x-2)(x-5) \\ &= x^3 - 6x^2 + 3x + 10 \\ &\Rightarrow \; c_0 = 10; c_1 = 3; c_2 = -6; c_3 = 1 \end{aligned}$$

**Aufgabe 7:** (2 Punkte)
Die Polynominterpolation mit Chebyshev-Stützstellen konvergiert garantiert. Warum benutzt man trotzdem manchmal die Interpolation an äquidistanten Stützstellen?

**Antwort:**

Das hängt von den Daten ab, Simulationsdaten, Messdaten sind üblicherweise äquidistant oder ungleichmäßig verteilt.

**Aufgabe 8:** (3 Punkte)
Berechne den Wert des interpolierenden Polynoms 2. Grades mit den Stützstellen $(x, y) \in \{(-1, 52), (0, 17), (2, 127)\}$ an der Stelle $x = 1$ mit Hilfe des Neville-Aitken-Schemas.

**Antwort:**

$$P_{0,1} = 52, \quad x_0 = -1, \quad P_{1,1} = 17, \quad x_1 = 0, \quad P_{2,1} = 127, \quad x_2 = 2$$

$$P_{0,2}(x=1) = \frac{P_{0,1}(x_1 - x) + P_{1,1}(x - x_0)}{x_1 - x_0} = -18$$

$$P_{1,2}(x=1) = \frac{P_{1,1}(x_2 - x) + P_{2,1}(1 - x_1)}{x_2 - x_1} = 72$$

$$P_{0,3}(x=1) = \frac{P_{0,2}(x=1)(x_2 - x) + P_{1,2}(x=1)(1 - x_0)}{x_2 - x_0} = 42$$

# 3 Signalverarbeitung (13 Punkte)

**Aufgabe 9:** (4 Punkte)
Schreibe eine Pythonfunktion `stats(x)`, die den Mittelwert und die Varianz einer Datenreihe berechnet und zurückgibt, ohne dabei die entsprechenden NumPy-Funktionen zu benutzen. Dabei sei $x$ ein eindimensionales Numpy-Array.

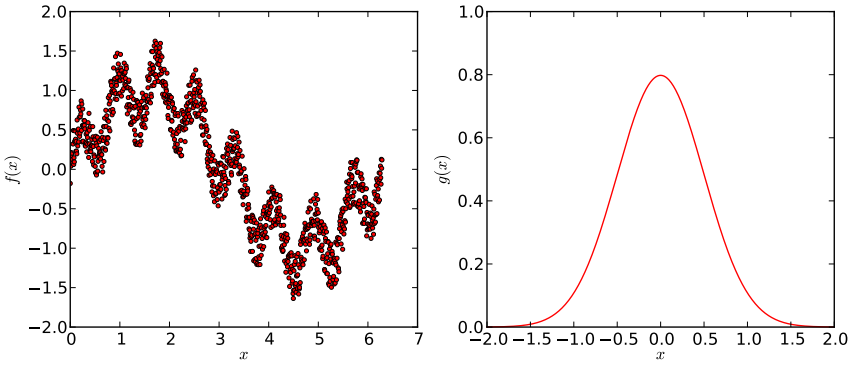
**Antwort:**

```
def stats(x):
  N = float(len(x))

  mean = x.sum()/N
  mean2 = (x**2).sum()/N

  var = N/(N-1.0)*(mean2 - mean**2)

  return mean, var
```

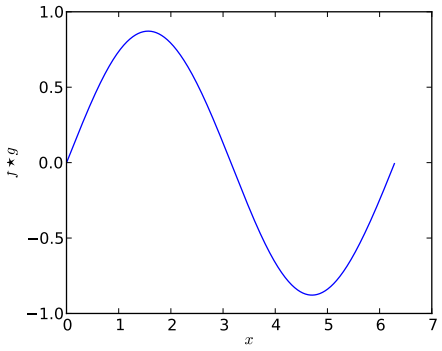
Bemerkung: `float` ist nötig, falls x ein array von Integertypen ist.

**Aufgabe 10:** (3 Punkte)

Skizziere die Faltung $f \star g$ der beiden in der folgenden Abbildung skizzierten Funktionen $f$ (links) und $g$ (rechts).

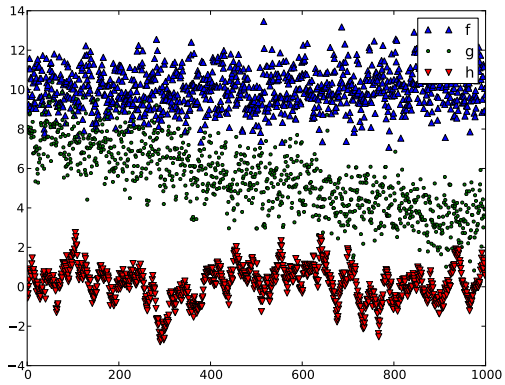


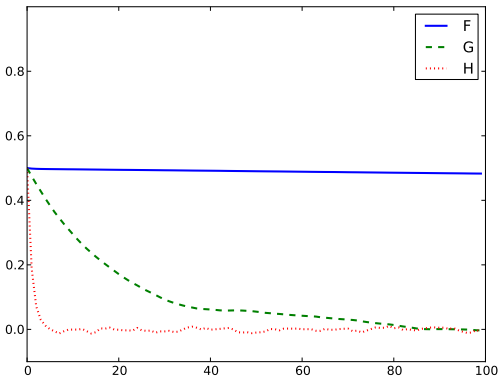

**Antwort:**



**Aufgabe 11:** (3 Punkte)

Ordne die in der linken Abbildung geplotteten Datenreihen $f$, $g$ und $h$ den in der rechten Abbildung geplotteten Autokorrelationsfunktionen $F$, $G$ und $H$ zu.




**Antwort:**

$$F = \mathrm{acf}(g)$$
$$G = \mathrm{acf}(h)$$
$$H = \mathrm{acf}(f)$$

**Aufgabe 12:** (1 Punkt)

Das JPG-Format verwendet die *diskrete Kosinustransformation*, eine enge Verwandte der Fouriertransformation. Warum ist das Format gut geeignet zum Speichern von Fotos, aber nicht zum Speichern von mathematischen Plots?

**Antwort:**

Fouriertransformierte sind schlecht dazu geeignet, um Funktionen mit großen Sprüngen darzustellen. Mathematische Plots haben viele scharfe Sprünge, wohingegen Fotos typischerweise eher langsame Farbverläufe aufweisen.

**Aufgabe 13:** (2 Punkte)
In einer (equilibrierten) Computersimulation misst Du alle 200 Schritte den Druck $P$ des Systems, und erhältst so $N = 50.000$ Messungen $P_i$, $i = 1(1)N$. Der mittlere Druck aller Messungen ist $\bar{P} = 2.0$, dessen Standardabweichung ist $\sigma = 2.0$, die integrierte Autokorrelationszeit ist $\tau_{\text{int}} = 100$ Messungen. Berechne den Fehler der Druckmessung (mit Konfidenzniveau $1\sigma$).

**Antwort:**

Der erwartete quadratische Fehler ist

$$\begin{aligned}\epsilon^2 &= (1 + 2\tau_{\text{int}})\frac{\sigma^2}{N} \\ &= (1 + 200)\frac{2^2}{50000} \\ &\approx 0.016\end{aligned}$$

Der erwartete Fehler ist daher $\sqrt{0{,}016} = 0{,}126$.

# 4 Nichtlineare Gleichungssysteme (9 Punkte)

**Aufgabe 14:** (1 Punkt)
Warum gibt es keine Bisektionsmethode zur Berechnung einer Nullstelle einer Funktion in mehr als einer Dimension?

**Antwort:**

Die Bisektionsmethode nutzt im eindimensionalen aus, dass es eine Nullstelle innerhalb eines Intervalls $[a, b]$ geben muss, wenn die Vorzeichen der Funktionswerte bei $a$ und $b$ unterschiedlich sind. Im Mehrdimensionalen wären die Grenzen $a$ und $b$ jedoch keine Punkte, sondern Linien bzw. Hyperflächen, denen nicht eindeutig ein Vorzeichen zugeordnet werden kann.

**Aufgabe 15:** (3 Punkte)

Bestimme die Nullstelle der Funktion $f(x) = x^2 - 3$ im Intervall $[0, 2]$ mit Hilfe des Bisektionsverfahrens und des Taschenrechners auf $\pm 0{,}1$.

**Antwort:**

$f(0) = -3, f(2) = 1$

$\Rightarrow I_0 = [0; 2], \quad m_0 = 1, \quad f(m_0) = -2$

$\Rightarrow I_1 = [1; 2], \quad m_1 = 1{,}5, \quad f(m_1) = -0{,}75$

$\Rightarrow I_2 = [1{,}5; 2], \quad m_2 = 1{,}75, \quad f(m_2) = 0{,}0625$

$\Rightarrow I_3 = [1{,}5; 1{,}75], \quad m_3 = 1{,}625, \quad f(m_3) = -0{,}359$

$\Rightarrow I_4 = [1{,}625; 1{,}75]$

Abbruch, da Intervallgrenzen innerhalb der gefordeten Genauigkeit.

**Aufgabe 16:** (5 Punkte)

Es sei eine Pythonfunktion `newton(f, fp, x0)` zur Nullstellensuche (dabei sei `f` die Funktion, `fp` deren Ableitung und `x0` der Startwert der Suche) und `exp(x)` zur Berechnung von $e^x$ gegeben. Schreibe mit deren Hilfe eine Pythonfunktion `log(b,x)` die $\log_b(x)$ berechnet. Die Funktion `newton` muss hier *nicht* definiert werden, sondern wird als gegeben vorausgesetzt.

**Antwort:**

```
def log(b, x):
    def fx(s): return exp(s) - x
    def fb(s): return exp(s) - b
    def fp(s): return exp(s)
    lnx = newton(fx, fp, 1.0)
    lnb = newton(fb, fp, 1.0)
    return lnx/lnb
```

# 5 Numerisches Differenzieren und Integrieren (14 Punkte)

**Aufgabe 17:** (1 Punkt)

Die sogenannte Zustandssumme eines Zweiteilchensystems in einer Box sei

$$\int_0^L \int_0^L \int_0^L \int_0^L \int_0^L \int_0^L e^{-\beta \frac{1}{2}\left[(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2\right]} \,\mathrm{d}x_1 \,\mathrm{d}y_1 \,\mathrm{d}z_1 \,\mathrm{d}x_2 \,\mathrm{d}y_2 \,\mathrm{d}z_2.$$

Welche Methode zur numerischen Integration würdest Du zur Berechnung dieses Integrals benutzen, und warum?

**Antwort:**

Da hochdimensional: Monte-Carlo oder Quasi-Monte-Carlo.

**Aufgabe 18:** (4 Punkte)

Schreibe eine Pythonfunktion `midpoint(f,a,b,N)`, die die zusammengesetzte Mittelpunktsregel implementiert und dadurch das Integral $\int_a^b f(x)\,\mathrm{d}x$ an $N$ äquidistanten Stützstellen berechnet.

**Antwort:**

```
def midpoint(f, a, b, N=100):
    h = abs( (b-a)/N )
    integral = 0.0
    for i in range(N):
        integral += f(a+(i+0.5)*h)
    return integral*h
```

**Aufgabe 19:** (3 Punkte)

Skizziere (graphisch oder verbal) zwei Methoden, wie man die Kreiszahl $\pi$ numerisch annähern kann.

**Antwort:**

1. Monte-Carlo-Integration: Zufällig Punkte $(x, y) \in [0,1]^2$ ziehen, und zählen, wie viele davon $x^2 + y^2 \leq 1$ erfüllen. Dann gilt

$$\frac{N_{\text{disk}}}{N_{\text{total}}} \approx \frac{\text{Fläche des Viertelkreises}}{\text{Gesamtfläche des Quadrats}} = \frac{\pi}{4}$$

2. Ausmessen der Fläche des Einheitskreises durch numerische Integration von

$$2 \int_{-1}^{1} \sqrt{1 - x^2}\, \mathrm{d}x = \pi$$

zum Beispiel mittels Trapezregel.

3. Numerische Integration von

$$\int_0^1 \frac{1}{1 + x^2}\, \mathrm{d}x = \arctan(1) = \frac{\pi}{4}$$

zum Beispiel mittels Trapezregel.

**Aufgabe 20:** (6 Punkte)

Wir betrachten einen getriebenen, gedämpften harmonischen Oszillator, der der Differentialgleichung

$$\ddot{x}(t) = -kx(t) - \gamma\dot{x}(t) + g(t)$$

unterliegt, wobei $g(x)$ periodisch mit Periodenlänge $T$ sei. Wir wollen nun entsprechend die periodische Lösung dieser Differentialgleichung mit Hilfe eines finiten Differenzenschemas numerisch annähern. Benutze bei Schrittweite $h$ für die zweite Ableitung die Dreipunktnäherung

$$\ddot{f}(t) \approx \frac{1}{h^2}\left[f(t-h) - 2f(t) + f(t+h)\right]$$

und für die Ableitung die zentrale Differenz

$$\dot{f}(t) \approx \frac{1}{2h}\left[f(t+h) - f(t-h)\right].$$

Beschreibe alle notwendigen Schritte, um eine Näherung für die Lösung der Differentialgleichung auf dem Intervall $[0, T[$ mit Schrittweite $h$ und Startwert $x(0) = 0$ zu berechnen. Verfahren zur Lösung linearer Gleichungssysteme können vorausgesetzt werden.

**Hinweis** Beachte, dass Dein Gleichungssystem lösbar sein sollte, also quadratisch! Du musst also eine der Gleichungen, die Du aus der kanonischen Diskretisierung erhältst, durch die Randbedingung $x(0) = 0$ ersetzen.

**Antwort:**

Setze $x_i = x(t_i)$ und $g_i = g(t_i)$, $i = 0(1)N-1$, mit $N = T/h$. ganzzahlig. Die diskretisierte DGL lautet:

$$\frac{1}{h^2}\left[x_{i-1} - 2x_i + x_{i+1}\right] + kx_i + \frac{\gamma}{2h}\left[x_{i+1} - x_{i-1}\right] = g_i,$$

wobei an den Rändern geeignet umgefaltet wird. Daraus ergibt sich das lineare Gleichungssystem

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \lambda & \mu & \nu & 0 & \ddots & 0 \\ 0 & \lambda & \mu & \nu & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \lambda & \mu & \nu \\ \nu & 0 & 0 & 0 & \lambda & \mu \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} 0 \\ g_1 \\ \vdots \\ g_{N-1} \end{pmatrix}$$

mit $\lambda = \frac{1}{h^2} - \frac{\gamma}{2h}$, $\mu = -\frac{2}{h^2} + k$ und $\nu = \frac{1}{h^2} + \frac{\gamma}{2h}$, das etwa mit Hilfe des Gaußverfahrens gelöst werden kann.

Anmerkungen: Die erste Zeile stellt die Randbedingung $x(0) = x_0 = 0$ dar, und ist nötig, weil die Zeilen der rein periodischen Matrix linear abhängig sind. Das $\nu$ am linken Rand der letzten Zeile schließt die periodischen Randbedingungen. Da die DGL nur dann eine periodische Lösung hat, wenn $g$ im Mittel Null ist, kann $g_0$ aus den restlichen Werten ermittelt werden, und geht daher nicht ein.

# 6 Zufallszahlen (8 Punkte)

**Aufgabe 21:** (1 Punkt)

Nenne einen Vor- und einen Nachteil von Pseudozufallszahlen gegenüber "echten" Zufallszahlen.

**Antwort:**

Vorteile: Reproduzierbarkeit, u.U. Geschwindigkeit
Nachteile: kein „echter" Zufall, d.h. im Prinzip vorhersagbar; schlecht für z.B. kryptographische Anwendungen

**Aufgabe 22:** (2 Punkte)

Eine Methode zur Bestimmung der Qualität von Zufallszahlen ist, die Mittelwerte $X_i$ von jeweils $k$ Werten einer Reihe $x_i$ von vermeintlichen Zufallszahlen zu bestimmen. Berechne die Standardabweichung $\sigma(X_i)$, die die Verteilung der Mittelwerte aufweisen sollte, damit die Reihe als Reihe von Standardzufallszahlen, also unabhängig und gleichverteilt in $[0, 1]$, gelten kann?

**Antwort:**

$$\sigma(X_i) = \frac{1}{\sqrt{k}}\sigma(x) = \frac{1}{\sqrt{12k}}$$

**Aufgabe 23:** (5 Punkte)

Schreibe eine Pythonfunktion `myrand()`, die eine Zufallszahl im Intervall $[0, 4]$ zurückliefert, die der Verteilung $P(x) = xe^{-x}$ genügt. Dabei soll sie die Pythonfunktion `random.random()` benutzen, die eine gleichverteilte Zufallszahl zwischen 0 und 1 erzeugt.

**Antwort:**

Es gilt $\frac{dP}{dx}(x) = (1 - x)e^{-x}$ und $P(x) > 0$, daher ist $\max_{x\in[0,4]} P(x) = 1/e$. Wir benutzen die Verwerfungsmethode mit entsprechender Normierung:

```
def P(x): return x*exp(-x)
def myrand():
    while True:
        r = 4.0*random.random()
        if random.random() < P(r)*e:
            return r
```

# 7 Lineare Algebra II (10 Punkte)

**Aufgabe 24:** (1 Punkt)

Welches numerische Verfahren kannst Du zum approximativen Lösen eines linearen Gleichungssystems verwenden? Nenne zwei Varianten.

**Antwort:**

- SOR
- Gauss-Seidel
- Jacobi

**Aufgabe 25:** (1 Punkt)

Erstelle eine L+D+U-Zerlegung der Matrix

$$\begin{pmatrix} 0 & 3 & 1 & 7 \\ 1 & 0 & 4 & 1 \\ 4 & 1 & 3 & 0 \\ 3 & 5 & 1 & 1 \end{pmatrix}$$

**Antwort:**

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 3 & 5 & 1 & 0 \end{pmatrix} D = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} U = \begin{pmatrix} 0 & 3 & 1 & 7 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

**Aufgabe 26:** (4 Punkte)

Die folgende Pythonfunktion soll das Jacobi-Verfahren implementieren, ist aber fehlerhaft. Korrigiere die drei Fehler (gerne auch direkt im Code). Diese sind sowohl logischer als auch syntaktischer Natur.

```
from numpy import *
def Jacobi(A, b, x0, eps)
    N = len(x0)
    err = eps
    while (err >= eps):
        x = empty_like(x0)
        for i in range(N):
            x[i] = b[i]
            for j in range(N):
                if (j != i): x[i] += A[i,j]*x0[j]
            x /= A[j,j]
            err = amax(abs(x0-x))
            x0 = x
       return x
```

**Antwort:**

```
from numpy import *

def Jacobi(A, b, x0, eps):
    N = len(x0)
    err = 2*eps
    while (err >= eps):
        x = empty_like(x0)
        for i in range(N):
            x[i] = b[i]
            for j in range(N):
                if (j != i): x[i] -= A[i,j]*x0[j]
            x[i] /= A[i,i]
        err = amax(abs(x0-x)) ←
        x0 = x ←
    return x ←
```

**Aufgabe 27:** (4 Punkte)
Wende die Gram-Schmid-Methode zur Orthogonalisierung der folgenden Matrix an.

$$A = \begin{pmatrix} 1 & 3 & 6 \\ -2 & 4 & -7 \\ -2 & 5 & 1 \end{pmatrix}$$

**Antwort:**

Setze

$$\begin{aligned} a_1 &= (1,-2,-2)^T, \\ a_2 &= (3,4,5)^T, \\ a_3 &= (6,-7,1)^T \end{aligned}$$

1. Norm $r_{11} = \|a_1\| = \sqrt{1+4+4} = 3$
   $\Longrightarrow\ q_1 = a_1/r_{11} = (1/3,-2/3,-2/3)^T$
2. Projektion $r_{12} = (a_2,q_1) = \frac{-15}{3} = -5\ \Longrightarrow$

$$b_2 = a_2 - (a_2,q_1)q_1 = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} + 5 \begin{pmatrix} 1/3 \\ -2/3 \\ -2/3 \end{pmatrix} = \begin{pmatrix} 14/3 \\ 2/3 \\ 5/3 \end{pmatrix}$$

   Normierung $r_{22} = \|b_2\| = \sqrt{\frac{196}{9} + \frac{4}{9} + \frac{25}{9}} = \sqrt{\frac{225}{9}} = \sqrt{25} = 5$
   $\Longrightarrow\ q_2 = (14/15, 2/15, 1/3)^T$
3. Projektionen $r_{13} = (a_3,q_1) = \frac{18}{3} = 6$ und $r_{23} = (a_3,q_2) = \frac{25}{5} = 5$
   $\Longrightarrow$

$$b_3 = a_3 - (a_3,q_1)q_1 - (a_3,q_2)q_2 = \begin{pmatrix} 6 \\ -7 \\ 1 \end{pmatrix} - 6 \begin{pmatrix} 1/3 \\ -2/3 \\ -2/3 \end{pmatrix} - \begin{pmatrix} 14/15 \\ 2/15 \\ 1/3 \end{pmatrix} = \begin{pmatrix} -2/3 \\ -11/3 \\ 10/3 \end{pmatrix}$$

   Normierung $r_{33} = \|b_3\| = \sqrt{\frac{4}{9} + \frac{121}{9} + \frac{100}{9}} = \sqrt{\frac{225}{9}} = \sqrt{25} = 5$
   $\Longrightarrow\ q_3 = (-2/15, -11/15, 2/3)^T$.

Die QR-Zerlegung ist daher

$$A = QR = \begin{pmatrix} 1/3 & 14/15 & -2/15 \\ -2/3 & 2/15 & -11/15 \\ -2/3 & 1/3 & 2/3 \end{pmatrix} \begin{pmatrix} 3 & -5 & 6 \\ 0 & 5 & 5 \\ 0 & 0 & 5 \end{pmatrix}$$

# 8 Optimierung (7 Punkte)

**Aufgabe 28:** (1 Punkt)

Was ist der Sinn einer Schrittweitensteuerung bei lokalen Minimierungsalgorithmen?

**Antwort:**

Ohne Schrittweitensteuerung kann eine mehrdimensionale leicht über das Minimum hinausschießen und es dadurch sogar komplett verfehlen. Die Schrittweitensteuerung stellt sicher, dass das Verfahren absteigt.

**Aufgabe 29:** (3 Punkte)

Gesucht ist ein lokales Minimum der Funktion $f(x,y) = x^2 + y^2 + (x-2)^2 + (y-1)^2$. Führe (mit Hilfe des Taschenrechners) vom Ausgangspunkt $x = y = 0$ drei Schritte des Verfahrens des steilsten Abstiegs mit Schrittweite $\lambda = 0{,}2$ aus.

**Antwort:**

Schritt ist

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \lambda \nabla f(x_i, y_i)$$

mit

$$\nabla f(x,y) = \begin{pmatrix} 2x + 2(x-2) \\ 2y + 2(y-1) \end{pmatrix} = \begin{pmatrix} 4x - 4 \\ 4y - 2 \end{pmatrix}$$

$\Longrightarrow$

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \lambda \begin{pmatrix} -4 \\ -2 \end{pmatrix} = \begin{pmatrix} 0{,}8 \\ 0{,}4 \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \lambda \begin{pmatrix} -0{,}8 \\ -0{,}4 \end{pmatrix} = \begin{pmatrix} 0{,}96 \\ 0{,}48 \end{pmatrix}$$

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \lambda \begin{pmatrix} -0{,}16 \\ -0{,}08 \end{pmatrix} = \begin{pmatrix} 0{,}992 \\ 0{,}496 \end{pmatrix}$$

**Aufgabe 30:** (3 Punkte)
Bringe das Problem

$$\min_{x,y\in\mathbb{R}^2} (\pi, 1)^T (x, y) \quad \text{unter den Nebenbedingungen} \quad y \geq 0,\, y \leq x \text{ und } y \leq 1 - x$$

auf Simplex-Normalform. Begründe Deine Umformungen!

**Antwort:**

Es gilt offenbar $x \geq y \geq 0$, daher muss x nicht in positiven und negativen Teil zerlegt werden.
Wir fügen zwei Zusatzvariablen für die beiden Ungleichungen ein: $z_1 := x - y \geq 0$ und $z_2 := 1 - x - y \geq 0$. Das ergibt die Gleichungsbedingungen $x - y - z_1 = 0$ und $x + y + z_2 = 1$ und damit das erweiterte Problem

$$\min_{w=(x,y,z_1,z_2)\in\mathbb{R}^4} c^T w \quad \text{unter der Nebenbedingung} \quad w \geq 0, Aw = b$$

mit

$$A = \begin{pmatrix} 1 & -1 & -1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{und} \quad c = \begin{pmatrix} \pi, 1, 0, 0 \end{pmatrix}^T,$$

das Simplex-Normalform hat.

# 9 Differentialgleichungen (12 Punkte)

**Aufgabe 31:** (1 Punkt)
Was ist der Unterschied in der Implementierung zwischen einer impliziten und einer expliziten Runge-Kutta-Methode?

**Antwort:**

Für das implizite Verfahren muss in jedem Iterationsschritt eine nichtlineare Gleichung gelöst werden, da $y_{n+1}$ zur Berechnung von sich selbst benötigt wird bzw. zur Näherung benötigte Koeffzienten implizit voneinander abhängen. Explizite Verfahren können im Gegensatz dazu trivial berechnet werden.

**Aufgabe 32:** (1 Punkt)

Es sei die Differentialgleichung erster Ordnung

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= \frac{\gamma}{x} y + \mathrm{e}^x\end{aligned}$$

gegeben. Formuliere diese Differentialgleichung als Gleichung zweiter Ordnung.

**Antwort:**

$$\ddot{x} = \frac{\gamma}{x}\dot{x} + \mathrm{e}^x$$

**Aufgabe 33:** (4 Punkte)

Gegeben sei folgendes Butcher-Tableau für die *Heun*-Methode:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Führe mit Hilfe des Taschenrechners zwei Schritte der Methode für die Differentialgleichung $\dot{y} = f(t, y) = y\,\mathrm{e}^t$ mit Schrittweite $h = 0{,}1$ und Anfangsbedingung $y(t = 0) = 1$ durch.

**Hinweis** Die analytische Lösung der DGL ist $y(t) = \exp(\mathrm{e}^t - 1)$.

**Antwort:**

$$\begin{aligned}
y_{n+1} &= y_n + h\sum_i b_i k_i \quad \text{mit } k_1 = f(t_n, y_n) \text{ und } k_2 = f(t_n + h, y_n + k_1) \\
&= y_n + \frac{h}{2}\Big[\underbrace{f(t_n, y_n)}_{=k_1} + \underbrace{f(t_n + h, y_n + hf(t_n, y_n))}_{=k_2}\Big] \\
y_0 &= 1 \\
y_1 &= 1 + h/2(\underbrace{f(0,1)}_{=k_1=1} + \underbrace{f(h, 1+h)}_{=k_2\approx 1.2157}) \approx 1.111 \\
y_2 &= 1.111 + h/2(\underbrace{f(h, 1.111)}_{=k_1\approx 1.228} + \underbrace{f(2h, 1.111 + h\cdot 1.1228)}_{=k_2\approx 1.5063}) \approx 1.2477 \\
y(0{,}1) &= 1.1109 \\
y(0{,}2) &= 1.2478
\end{aligned}$$

**Aufgabe 34:** (6 Punkte)

Leite die sogenannten Verlet-Methode zum numerischen Annähern der Bewegungsgleichung her. Entwickle dazu die Position $r(t)$ an der Stelle $t$ und werte das Taylorpolynom an den Stellen $t+h$ und $t-h$ bis zur vierten Ordnung aus. Gib daraus eine Gleichung zur numerischen Berechung der Position zum Zeitpunkt $t+h$ an. Worin unterscheidet sich diese Methode grundlegend von einer Runge-Kutta-Methode? Beschreibe, wie mit Hilfe der Gleichung die Bewegungsgleichung angenähert werden kann. Welche Fehlerordnung hat der Algorithmus?

**Antwort:**

$$
\begin{aligned}
&I && r(t+h) = r(t) + h\dot{r}(t) + h^2/2\ddot{r}(t) + h^3/6\dddot{r}(t) + \mathcal{O}(t^4)\\
&II && r(t-h) = r(t) - h\dot{r}(t) + h^2/2\ddot{r}(t) - h^3/6\dddot{r}(t) + \mathcal{O}(h^4)\\
&I+II: && r(t+h) + r(t-h) = 2r(t) + h^2\ddot{r}(t) + \mathcal{O}(h^4)\\
&\rightarrow && r(t+h) = 2r(t) - r(t-h) + h^2\ddot{r}(t) + \mathcal{O}(h^4)
\end{aligned}
$$

Fehlerordnung: $\mathcal{O}(h^4)$
Unterschied Runge Kutta: die letzte Position $r(t-h)$ wird für den neuen Iterationsschritt benötigt.

# 10 Programmieren (7 Punkte)

**Aufgabe 35:** (6 Punkte)

Schreibe zwei Versionen einer Pythonfunktion, die die Summe $\sum_{i=0}^{N-k-1} x_i x_{i+k}$ einer Datenreihe $x_i$ berechnet, die im NumPy-Array `x` gespeichert ist. `sum_python(x)` soll dabei eine Schleife in Python verwenden, `sum_numpy(x)` einen einzigen NumPy-Ausdruck.

**Antwort:**

```
def sum_python(x):
    sum = 0.0
    for i in range(len(x)-k):
        sum += x[i]*x[i+k]
    return sum

def sum_numpy(x):
    return (x[:-k]*x[k:]).sum()
```

**Aufgabe 36:** (1 Punkt)

Wenn Du zwei Pythonfunktionen hast, die beide dasselbe berechnen, aber einmal mit Hilfe einer Schleife in Python und einmal in einem NumPy-Ausdruck, welche der beiden Funktionen ist typischerweise schneller, und warum?

**Antwort:**

Der NumPy-Ausdruck ist normalerweise schneller, weil die Schleife im kompilierten NumPy-Code und nicht im interpretierten Python-Code ausgeführt wird.