Übungsaufgaben 3
(Funktionen, Rekursion,Numpy, Scipy, Matplotlib)

Hausaufgabe: Lösen Sie mindestens 6 der folgenden Aufgaben (manche Aufgaben zählen doppelt, d.h so viel wie zwei gelöste Aufgaben.) Die Abgabe erfolgt im ISIS-Kurs, Termin wird dort bekannt gegeben.

Aufgaben aus dem Labor

1. **Karten mischen**
   Bestimmen Sie durch Simulation, mit welcher Wahrscheinlichkeit beim Mischen eines Kartenstapels aus 52 Karten mindestens eine Karte an der alten Stelle zu liegen kommt. Schreiben Sie dazu eine Funktion `ziehe(l)`, die ein zufälliges Element der Liste l zurückgibt und dieses Element aus der Liste entfernt, eine Funktion `mische(n)`, die eine zufällige Anordnung der Zahlen $0,\ldots,n-1$ durch sukzessives Ziehen erzeugt und als Liste zurückgibt, eine Funktion `pruefe(l)`, die True zurückgibt, falls die übergebene Anornungs-Liste eine Karte an ihrer alten Stelle hat (d.h. falls es ein i gibt, so dass $l[i] == i$) und ein Hauptprogramm, das mit Hilfe der Funktionen `mische` und `pruefe` das gegebene Problem löst. Für die Funktion `ziehe` können Sie `numpy.random.randint` benutzen.
   Zusatzfrage: Können Sie die Antwort auch auf dem Papier bestimmen?

2. **Drachenkurve**
   Schreiben Sie ein Programm, das (mit Hilfe von Rekursion) die Drachenkurve zeichnet. Sie können dazu turtle-Graphik verwenden. Hinweise dazu auch auf ISIS.

3. **Kantenbild**
   Erzeugen Sie aus einem Bild ein Schwarz-Weiß-Bild, das nur die Kanten des Bilds zeigt, und machen Sie das möglichst gut. Sie können dabei `scipy.ndimage` und `scipy.misc` benutzen.

4. **Das Spiel** * (zählt doppelt)
   Schreiben Sie eine Funktion, die für das Spiel (Erklärung auf der ISIS-Seite) mit der Ausgangsposition aus (1,5,9) Steinen bestimmt, ob eine Position eine Siegposition ist, also von der aus sich durch das richtige Verhalten ein Sieg garantieren lässt. Entsprechend schreiben Sie eine Funktion, die ermittelt, ob eine Position eine Verlustposition ist, also ob von dort jeder Zug zu einer Siegposition des Gegners führt. Außerdem ist die Position (0,0,0) auf jeden Fall eine Verlustposition, denn diese bedeutet, dass der letzte Zug vom anderen Spieler gemacht wurde. Wie erklärt, ergibt das eine rekursive Struktur. Verwenden Sie diese Funktionen, um ein Programm zu schrei-

ben, dass für eine Anfangskonfiguration einen Zug empfiehlt, der zum Sieg führt, wenn es einen solchen gibt.

Nützlich für die Programmierung wird es sein, noch eine weitere Funktion zu schreiben, die für eine gegebene Position die Liste der möglichen Züge, bzw. genauer der möglichen Positionen nach einem erlaubten Zug ausgibt.

5. **Fortsetzung der Wetterdatenaufgabe** ** (zählt doppelt doppelt)
In der Aufgabe zu den Wetterdaten vom letzten Blatt sollten nur Eigenschaften der Temperaturdaten durch geeignete Mittelwertbildungen (graphisch) erkennbar werden.

   Eine systematischere Art, aus den Daten Informationen zu ziehen, ist ein *Regressionsmodell*. Die vorliegenden Daten bilden eine Messreihe. In jeder Messung i der Reihe wird (untere anderem) der Wert der Durchschnittstemperatur `av_temps[i]` am Tag `dates[i]` ermittelt. Wir suchen ein Modell, dass aus dem Datum `dates[i]` die mittlere Temperatur `av_temps[i]` voraussagt. Das geht natürlich nicht genau, aber wir können das Modell suchen, das den kleinsten Vorhersagefehler macht. Bei der „klassischen“ linearen Regression werden Modelle betrachtet, bei denen die Voraussage mit Hilfe linearer Funktionen gemacht und der Fehler durch die Summe der Quadrate der einzelnen Fehler gemessen wird.

   Für unser Beispiel wollen wir ein lineares Regressionsmodell bauen, das einen Trend der Temperaturentwicklung unter den jahreszeitlichen Schwankungen erkennt.

   Wir bezeichen die Zielgröße `av_temps[i]` im weiteren als `y[i]` und in das in Tage ab einem gewissen Stichdatum umgerechnete Datum `matplotlib.dates.date2num(dates[i])` als `x_0[i]`. Die (tropische) Jahreslänge beträgt $d = 365.24219052$ Tage. Wir führen Größen `x_k[i]` ($k = 1,\ldots,366$) ein, wobei `x_k[i]` den Wert 1 hat, wenn `dates[i]` auf den k. Tag des Jahres fällt, sonst den Wert 0. (Durch `np.ceil(dates%d)` lässt sich das jeweilige k ermitteln.)

   Es sind nun Koeffizienten `b_0`, …, `b_366` so zu bestimmen, dass der quadratische Fehler

$$\sum_i \left(\left(\sum_{k=0}^{366} \texttt{b\_k}\ \texttt{x\_k[i]} - \texttt{y[i]} \right) \right)^2$$

   minimal wird. b_0 ist als der langfristige Trend interpretierbar, die übrigen modellieren die jahreszeitliche Schwankung.

   Wenn Sie in Linearer Algebra das kleinste-Quadrate-Problem schon behandelt haben, können Sie loslegen. Wenn nicht, ist folgende Information nützlich: Sei X die Matrix, deren Spalten die `x_0`, …, `x_366` sind und b der (Spalten-) Vektor (`b_0`, …, `b_366`), so ist dasjenige b, das die Fehlerquadrate minimiert, durch

$$b = (X^\top X)^{-1} X^\top y$$

Verwenden Sie `numpy` und speziell `numpy.linalg`. Welcher Trend ist aus dem Ergebnis abzulesen (Temperaturveränderung/Jahr)?

Das können Sie natürlich auch für die Maximaltemperatur und Minimaltemperatur machen.

Plotten sie außerdem die Koeffizienten $b[1], \ldots, b[366]$ und interpretieren Sie den Plot.

6. **Mandelbrot-Fraktal** * (zählt doppelt)
   Die Mandelbrot-Menge ist ein vergleichsweise einfach zu generierendes Fraktal. Benannt ist es nach Bernoit Mandelbrot, der mit seiner Veröffentlichung *Les objets fractals, forme, hasard et dimension* (1975) den Begriff des Fraktals geprägt hat.
   Die Menge lebt in der Komplexen Zahlenebene und wird durch folgende Vorschrift beschrieben:

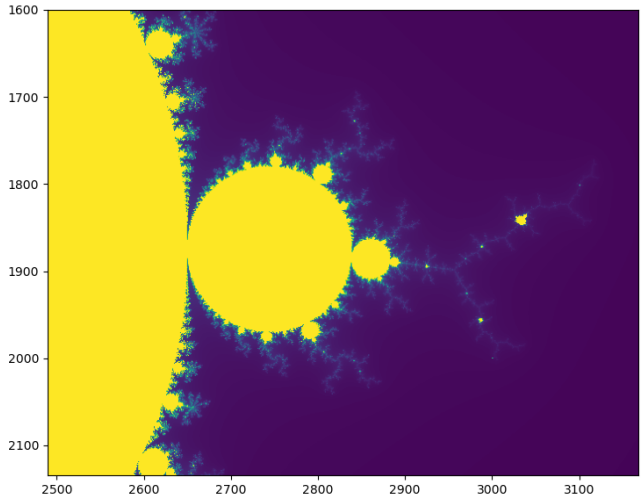
$$z_0 = 0$$
$$z_{n+1} = z_n^2 + c$$

   Hier bei entspricht c einem Punkt der komplexen Zahlenebene. Divergiert z für c nicht, gehört c zur Menge. Für die Simulation lässt sich das vereinfachend durch einen fest gesetzen Maximalwert überprüfen. Numpy Arrays dürfen auch Komplexe Zahlen enthalten, wenn das Array entsprechend initialisiert wird (np.zeros((xres, yres), dtype=np.complex_)

   Empfindliche Parameter sind die Pixeldichte (startet mit 200x200) und die Interationstiefe (erst mal begrenzen auf 200). Oft wird das Fraktal in Farbverläufen dargestellt, indem man die Iterationstiefe auswertet die benötigt wird um den Maximalwert zu erreichen. Eine einfache Variante eine Matrix darzustellen ist imshow(B). Will man eine Funktion generieren, die auf jedes Element eines Numpy-Arrays wirkt, empfiehlt es sich mit np.vectorize zu arbeiten.

   Schafft ihr es, die Achenbeschriftung so anzupassen, dass sie zu den Werten der Komplexen Zahlen passt (anders als unten im Bild...) ?

   In jedem Fall empfehlenswert: Auf Youtube nach “Mandelbrot Zoom” suchen.



Andere Aufgaben

7. **ggT**
   Schreiben Sie eine Funktion, die den größten gemeinsamen Teiler (ggT) zweier Zahlen rekursiv berechnet. Schlagen Sie dazu, wenn nötig, den "euklidischen Algorithmus" nach.

8. **Minimum suchen**
   Schreiben Sie eine Funktion `finde_minimum(f,x0,x1,feinheit=0.01)`. Dabei ist f eine Funktion (die ein reelles Argument hat und eine reelle Zahl liefert), x0 und x1 sind die Ränder des Intervals, in dem wie ein Minimum suchen, `feinheit` die Feinheit der Unterteilung des Intervalls, die wir bei der Suche verwenden. (Eine Aussage über die Genauigkeit des so gefundenen Minimums können Sie nur treffen, wenn Sie noch mehr über die Funktion wissen. Können Sie einen Satz formulieren und beweisen, der Bedingungen formuliert dafür, dass ein so gefundene Minimalstelle einen Fehler von höchstens ε hat?) Testen Sie ihre Funktion an der Kosinus-Funktion im Intervall $[2,4]$.

9. **Zeichnen**
   Schreiben Sie ein Programm, das mit Turtle-Graphik zeichnet. Wenn Sie mit der Maus an eine Stelle gehen und kicken, soll sich die Turtle zur neuen Position begeben und dabei zeichnen. Verwenden Sie dazu

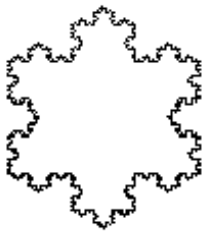
   ```
   screen=turtle.Screen()
   screen.onclick(...)
   turtle.mainloop()
   ```

   Lesen Sie die Verwendung von 'onclick' in der Dokumentation nach. Sie können sich auch noch alle möglichen Erweiterungen ausdenken.

10. **Kochkurve**
    Die Kochkurve ist eine fraktale Kurve, die 1904 vom Schwedischen Mathematiker Helge von Koch eingeführt wurde. Bei jedem Rekursionsschritt wird jede Strecke in drei Teile geteilt und die mittlere Teilstrecke durch zwei Strecken ersetzt, die im Winkel von 60° anliegen.

Links: Rekursionstiefen 0 bis 5 der Kochkurve. Rechts: Rekursionsschritt

Schreiben Sie ein Programm, das eine Rekursionstiefe n einliest (und eventuell noch eine Streckenlänge für den letzten Rekursionsschritt) und die sich ergebende Kochkurve mit der Turtlegrafik malt.



„Schneeflockenkurve“: Aus Kochkurven kann man eine Schneeflocke zusammensetzen

11. **Damenproblem** * (zählt doppelt)
    Beim 8-Damenproblem geht es darum, 8 Damen auf einem Schachbrett ($8*8$ Felder) so zu setzen, dass keine Dame von einer anderen geschlagen werden kann (d. h. keine zwei Damen sind in einer Zeile, Spalte oder Diagonalen). Allgemeiner formuliert setzt man beim n-Damenproblem n Damen auf einem Brett mit n*n Feldern.

    Das Problem kann man mit *Backtracking* lösen. Dabei setzt man rekursiv eine Dame nach der anderen. Wenn eine neu gesetzte Dame eine der bereits gesetzten Damen schlagen könnte, so wird der Zug zurückgenommen und die nächste mögliche Position für die Dame wird ausprobiert. Ebenso wird, wenn es für die rekursiv nachfolgenden Damen keine Lösung gibt, die Dame zurückgenommen und die folgende Position versucht. Aufgrund des Zurücksetzens wird der Ansatz als Backtracking bezeichnet.

    Man beachte hier, dass man für jede Zeile (oder Spalte) von vorne herein nur eine Dame platzieren muss.

Aufgaben zu Numpy, Scipy, Matplotlib

11. **Winkel berechnen**
    Sehen Sie sich die Dokumentation des Pakets `numpy.linalg` an. Schreiben Sie eine Funktion, die zwei Vektoren im $\mathbb{R}^n$ als Argumente akzeptiert und den von den beiden Vektoren eingeschlossenen Winkel zurückgibt.

12. **Gleichungssystem lösen, I**
    Bestimmen Sie die Matrix der linearen Abbildung $\mathbb{R}^3 \to \mathbb{R}^3$, die die Punkte $\begin{pmatrix}1\\2\\3\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\-1\end{pmatrix}$ auf die Punkte $\begin{pmatrix}2\\1\\1\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\-1\\-1\end{pmatrix}$ abbildet.

13. **Gleichungssystem lösen, II**
    Erstellen Sie für $n = 2,3,4,10$ die Matrix $M = (m_{ij})_{1\le i,j\le n}$, $m_{i,j} = 1/(i+j-1)$. (Beachten Sie,

dass numpy-Arrays die Indizierung bei Null beginnen lassen, während man in der Mathematik mit 1 anfängt.) Lösen Sie das Gleichungssystem $Mx = \begin{pmatrix} 1 \\ 1 \\ \dots \\ 1 \end{pmatrix}$. Machen Sie jeweils die Probe, um festzustellen, ob die Lösung tatsächlich eine ist. Was fällt Ihnen auf für $n = 10$?

14. **Schwerpunkt, Trägheitstensor** * (zählt doppelt)
Schreiben Sie eine Funktion, die eine Liste von Vektoren im $\mathbb{R}^3$ akzeptiert und daraus den Schwerpunkt und den Trägheitstensor, sowie die Eigenvektoren des Trägheitstensors berechnet. Zusatz*: Visualisieren Sie das.

15. **Mittelwert und Varianz**
Schreiben Sie eine Funktion, der eine Liste von Zahlen übergeben wird, und die daraus Mittelwert und Standardabweichung berechnet.

16. **π durch Zufallsexperiment** * (zählt doppelt)
Ziehen Sie N ($N = 1000, 10000, 100000$) Punkte (x_i, y_i) aus dem Quadrat $[-1,1] \times [-1,1]$ (d.h. beide Koordinaten werden aus einer Gleichverteilung über das Intervall $[-1,1]$ gezogen, siehe `numpy.random`). Zeigen Sie die gezogenen Punkte in einer Graphik, wobei die Punkte innerhalb des Einheitskreises eine andere Farbe haben sollen. Ermittelns Sie die Anzahl der Punkte N_E, die innerhalb des Einheitskreises liegen und geben Sie $4\frac{N_E}{N}$ aus. (Welchen Wert erwarten Sie? Sie können auch versuchen, sich zu überlegen, wie wahrscheinlich eine Abweichung von mehr als 1 Prozent von dem erwarteten Wert ist.)

17. **Mikrozensus, I**
Verwenden Sie die Mikrozensusdaten in der Datei `algebuei.csv` und lesen Sie sie mit Hilfe von `numpy.genfromtxt` in ein Array. Erstellen Sie ein Histogramm aller Einkommen, der Einkommen der Männer, der Einkommen der Frauen. Versuchen Sie, beide Histogramme in einer Graphik mit 'gestapelten Balken' anzuzeigen.

18. **Mikrozensus, II**
Bestimmen Sie den Mittelwert und den Median des Einkommens für jedes Bundesland. Stellen Sie die Ergebnisse in einer Graphik dar.

19. **Iris-Daten** * (zählt doppelt)
Lesen Sie die Daten in `iris.csv` in ein Array. Sie stellen gemessene Längen und Breiten von Blütenblättern drei verschiedener Schwertlilienarten dar. Stellen Sie die Daten als 3d-Punktwolke bezüglich drei der vier Parameter dar, wobei die Farbe der Punkte die Art bezeichnen soll.