Universität Tübingen
Mathematisches Institut
Dominik Edelmann

Tübingen, den 11.10.2021

Übungsaufgaben zum Matlab–Vorkurs

Aufgabe 1 Welche der folgenden Variablennamen sind in Matlab zulässig?

(a) `2nach1`

(b) `die erste`

(c) `dieZweite`

(d) `die_dritte`

(e) `variable-nummer-1`

(f) `variable-nummer.2`

(g) `zins_in_%`

Aufgabe 2 Welchen Wert hat die Variable `n` nach Zeile 4 und nach Zeile 8? Versuchen Sie zunächst, die Frage zu beantworten, ohne die Befehle in Matlab auszuführen.

```
1:  n = 3;
2:  n = n+1;
3:  n = n/2;
4:  n = n^3;
5:  n = n-4;
6:  n = n/4;
7:  n = 5;
8:  n = n+1;
```

Aufgabe 3 Schreiben Sie ein Skript `aufgabe3.m`, in dem Sie die 8 Zeilen aus Aufgabe 2 eingeben und führen Sie dieses aus. Ändern Sie nun den Wert von `n` in der ersten Zeile und führen Sie das Skript erneut aus. Was fällt Ihnen auf? Können Sie erklären, was passiert?

Aufgabe 4 Schreiben Sie ein Skript `division.m`, in dem Sie zwei Variablen `a` und `b` deklarieren und sich den Quotienten `c=a/b` ausgeben lassen. Führen Sie dieses Skript für unterschiedliche Werte von `a` und `b` aus.

Aufgabe 5 Schreiben Sie ein Skript `aufgabe5.m`, in dem Sie zwei Variablen `x` und `y` deklarieren. Das Skript soll dann mithilfe einer `if`-Abfrage die kleinere Zahl von der größeren abziehen.

Aufgabe 6 Schreiben Sie ein Skript `aufgabe6.m`, in dem Sie eine natürliche Zahl `n` deklarieren. Das Skript soll mithilfe einer `if`-Abfrage ausgeben, ob die Zahl `n` dreistellig ist.

Aufgabe 7 Schreiben Sie ein Skript, welches für eine gegebene ganze Zahl x die Anzahl ihrer Dezimalstellen ausgibt. Das Programm soll eine Fehlermeldung ausgeben, falls $x < 0$ ist, die exakte Anzahl der Stellen, falls $0 \leq x < 1000$ ist und *mehr als drei Stellen*, falls $x \geq 1000$ ist.

Was gibt das Skript aus, wenn Sie es mit einer reellen Zahl $x \in (0, 1000)$ aufrufen, die keine ganze Zahl ist?

Aufgabe 8 (a) Schreiben Sie ein Skript, welches die Zahlen von 0 bis 10 ausgibt.

(b) Ändern Sie ihr Programm so, dass es die Zahlen $k = 2i + 1$ für $i = 0, \ldots, 10$ ausgibt (also die ungeraden Zahlen von 1 bis 21).

Aufgabe 9 Schreiben Sie ein Skript, welches die Summe

$$S = \sum_{k=1}^{n} \frac{1}{k^2}$$

für gegebenes n berechnet.

Anmerkung: Der Grenzwert für $n \to \infty$ dieser Reihe ist $\frac{\pi^2}{6}$.

Aufgabe 10 Schreiben Sie ein Skript, welches die Summe

$$S = \sum_{k=0}^{n} q^k$$

für gegebenes q und n berechnet.

Welche Bedeutung hat diese Formel? Was ist der Grenzwert für $n \to \infty$ (und für welche q)?

Aufgabe 11 Schreiben Sie ein Skript, in dem Sie eine positive Zahl `x` und eine Toleranz `tol` deklarieren. Schreiben Sie dann eine `while`-Schleife, in der die Zahl `x` so lange halbiert wird, bis sie kleiner als `tol` ist.

Bauen Sie eine Variable `count` ein, die zählt, wie oft `x` halbiert wurde.

Aufgabe 12 Man kann zeigen, dass $\lim_{n\to\infty} \sum_{k=0}^{n} (-1)^k/(2k+1) = \pi/4$ gilt. Man kann also die Zahl π approximieren, indem man $4S_n$ für großes n berechnet, wobei S_n die n-te Partialsumme bezeichne.

Schreiben Sie mithilfe einer while-Schleife ein Programm, welches das kleinste n bestimmt, so dass $|4S_n - \pi| < 10^{-8}$ gilt. Sie dürfen die in Matlab gespeicherte Zahl `pi` in der Abbruchbedingung verwenden.

Warum ist ihre Abbruchbedingung eigentlich unsinnig, wenn Sie die Zahl π approximieren möchten?

Überlegen Sie sich, ob obige Formel (aus numerischer Sicht) sinnvoll ist, um einen Näherungswert für π zu bestimmen.

Aufgabe 13 Temperaturen in Grad Celsius können mit der Formel $t_{Fahr} = \frac{9}{5} t_{Cel} + 32$ in Grad Fahrenheit umgerechnet werden. Schreiben Sie dafür eine Matlab-Funktion `function [tFahr] = celinfahr(tCel)`.

Aufgabe 14 Schreiben Sie eine Funktion `function S = geomSum(q,N)`, welche die N-te Partialsumme der geometrischen Reihe berechnet:

$$S = \sum_{k=0}^{N} q^i \,.$$

Rufen Sie die Funktion für unterschiedliche Werte von `q` und `N` auf.

Machen Sie sich noch einmal den Unterschied zwischen einem Skript und einer Funktion klar, in dem Sie diese Funktion mit dem Skript aus Aufgabe 10 vergleichen.

Aufgabe 15 Gegeben seien zwei Vektoren $a, b \in \mathbb{R}^n$. Welche Arten der Multiplikation von Vektoren kennen Sie? Welche Dimensionen haben die jeweiligen Resultate? Schreiben Sie eine Funktion `function [y] = innerProduct(a,b)`, welche das Skalarprodukt zweier Vektoren berechnet.

Aufgabe 16 Schreiben Sie eine Matlab-Funktion `function [y] = mean(v)`, die das arithmetische Mittel der Elemente von `v` berechnet.

Aufgabe 17 Schreiben Sie eine Matlab-Funktion `function [n1,n2,nInf] = norms(v)`, die die $\|\cdot\|_1$-, $\|\cdot\|_2$- sowie die $\|\cdot\|_\infty$-Norm des Vektors `v` berechnen.

Aufgabe 18 Schreiben Sie eine Matlab-Funktion `function [sumElements] = addElements(A)`, die sämtliche Einträge einer Matrix `A` addiert.

Aufgabe 19 Schreiben Sie eine Matlab-Funktion, die für eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Vektor $v \in \mathbb{R}^n$ das Matrix-Vektor-Produkt $w = Av$ berechnet.

Aufgabe 20 Schreiben Sie eine Funktion `function [C] = matMult(A,B)`, welche zwei Matrizen $A \in \mathbb{R}^{n \times m}$ und $B \in \mathbb{R}^{m \times k}$ multipliziert.
Hinweise: Die Elemente von C sind

$$c_{ij} = \sum_{l=1}^{m} a_{il} b_{lj}, \quad i = 1, \ldots, n \quad j = 1, \ldots, k$$

Verwenden Sie die `size`-Funktion zur Ermittlung von `m, n` und `k`. Achten Sie auf die richtige Initialisierung von `C`.

Aufgabe 21 Überlegen Sie sich, ob die in Aufgabe 20 programmierte Funktion auch das Matrix-Vektorprodukt wie in Aufgabe 19 berechnen kann.

Aufgabe 22 Plotten Sie die Funktion

$$f : [-2, 2] \to \mathbb{R}, \quad x \mapsto \begin{cases} -x^2, & x < 0 \\ x, & 0 \leq x \leq 1 \\ x^2 + 1, & x > 1 \end{cases}$$

Schreiben Sie dafür eine Funktion `func_f`, welche obige Funktion realisiert, sowie ein Skript `plot_f`, in der die obige Funktion ausgewertet und geplottet wird.

Achten Sie auf eine sinnvolle Beschriftung des Plots.

Wie unterscheidet sich das Verhalten des Plots vom Verhalten der Funktion f im Punkt $x = 1$? Erklärung?

Aufgabe 23 Schreiben Sie eine Funktion `y = fschar(x,b)`, welche die Parameter x und b erwartet und den Wert $y(x) = bx^2$ zurück gibt. Schreiben Sie ein Skript `plot_fschar` welches diese Funktion für $x \in [-1, 1]$ und $b \in \{1/2, 1, 3/2, 2\}$ in ein Schaubild plottet. Achten Sie auf eine sinnvolle Beschriftung.

Aufgabe 24 Schreiben Sie ein Skript `schaubenplot.m`, welches die Funktion

$$f : [0, 10] \to \mathbb{R}^3, \quad t \mapsto (\cos(t), \sin(t), t)$$

plottet. Wie lässt sich die Anzahl der *Windungen* erhöhen?

Hinweis: Die eingebauten Matlab-Funktionen wirken komponentenweise, wenn man Vektoren übergibt.

Aufgabe 25 Schreiben Sie ein Skript `graphenplot.m`, welches die Funktion

$$f : [-3, 3] \times [-3, 3] \to \mathbb{R}, \quad (x, y) \mapsto \exp(x) \sin(y^2)$$

graphisch darstellt.

Hinweis: Die eingebauten Matlab-Funktionen wirken komponentenweise, wenn man Vektoren übergibt. Vektoren können mit `.*` komponentenweise multipliziert werden.

Aufgabe 26 Betrachten Sie ihre Funktion zur Berechnung der geometrischen Partialsummen aus Aufgabe 14. Ändern Sie diese so ab, dass nur noch Addition und Multiplikation verwendet werden, d. h. die Potenz q^k soll mittels einer Schleife als k-faches Produkt der Zahl q mit sich selbst berechnet werden.

Ermitteln Sie mit den Matlab-Befehlen `tic` und `toc` die Laufzeit für verschiedene Werte von N. Können Sie einen Zusammenhang zwischen der Anzahl der Summanden und der Laufzeit feststellen?

Kopieren Sie nun Ihr ineffizientes Programm und versuchen Sie, q^k ohne die vorhin eingebaute Schleife, aber auch ohne die Matlab-Potenzfunktion zu berechnen. Hinweis: $q^{k+1} = q \cdot q^k$.

Ermitteln Sie nun erneut die Laufzeit für verschiedene Werte von N. Können Sie wieder einen Zusammenhang zwischen der Laufzeit und der Anzahl der Summanden feststellen?

Wie erklären Sie sich dieses Phänomen?

Aufgabe 27 Für eine Funktion $f : [a, b] \to \mathbb{R}$ kann die Ableitung $f'(x_0)$ in einem Punkt x_0 approximiert werden, indem man den Differenzenquotienten

$$\frac{f(x_0 + h) - f(x_0)}{h}$$

für ein kleines h berechnet.

Erstellen Sie ein Programm, indem Sie ein Intervall $[a, b]$, eine Funktion $f : [a, b] \to \mathbb{R}$ und eine Anzahl von (äquidistanten) Stützstellen im Intervall $[a, b]$, sowie einen Parameter h angeben. Das Programm soll nun $f(x_i)$ in den Stützstellen berechnen und die Ableitung $f'(x_i)$ in den Stützstellen x_i mithilfe des Differenzenquotienten approximieren. Als Approximation für $f'(b)$ verwenden Sie den linksseitigen Differenzenquotienten $(f(b) - f(b - h))/h$. Plotten Sie anschließend f und f' in einem Schaubild.

Machen Sie sich klar, welche Teile des Programms zum Preprocessing, zum Processing und zum Postprocessing gehören.

Aufgabe 28 Schreiben Sie ein Skript, in dem Sie die Sinusfunktion mithilfe des Differenzenquotienten näherungsweise in $x_0 = 0$ differenzieren. Dazu brauchen Sie einen Vektor mit verschiedenen Werten für $h > 0$, z. B.können Sie mit `h=0.01:0.01:1` einen Vektor mit den Einträgen 0.01, 0.02, ..., 1 erzeugen.

Berechnen Sie für jeden Wert von h den Differenzenquotienten

$$\frac{f(h) - f(0)}{h}$$

sowie die den Betrag der Differenz zum exakten Wert der Ableitung $f'(0)$. Plotten Sie die Fehler in einem doppelt logarithmischen Schaubild. Plotten Sie im selben Schaubild die Funktionen $h \mapsto h$ und $h \mapsto h^2$. Mit welcher Konvergenzordnung konvergiert der Differenzenquotient?

Wiederholen Sie dasselbe mit der Exponentialfunktion. Mit welcher Konvergenzordnung konvergiert der Differenzenquotient in diesem Fall?

Bonusfrage (schwierig): Können Sie erklären, woran das liegt?

Aufgabe 29 Schreiben Sie eine nicht-rekursive Funktion zur Berechnung der N-ten Zahl der Fibonacci-Folge

$$f_{n+2} = f_{n+1} + f_n, \qquad f_2 = f_1 = 1\,.$$

Hinweis: Initialisieren Sie einen geeigneten Vektor der Länge N und verwenden Sie eine Schleife.

Vergleichen Sie die Laufzeit dieser (iterativen) Funktion mit der rekursiven Funktion für verschiedene Werte von N.

Bonusfrage: Wie viele Operationen sind (ungefähr) beim rekursiven Algorithmus erforderlich, um die n-te Fibonacci-Zahl zu berechnen? Wie wächst die Anzahl der Operationen in n?

Bonusaufgabe In dieser Aufgabe implementieren Sie ein Programm zur Visualisierung der Mandelbrot-Menge.

Die Mandelbrotmenge $\mathcal{M}$ ist definiert als die Menge aller komplexen Zahlen c, für die die rekursive Folge

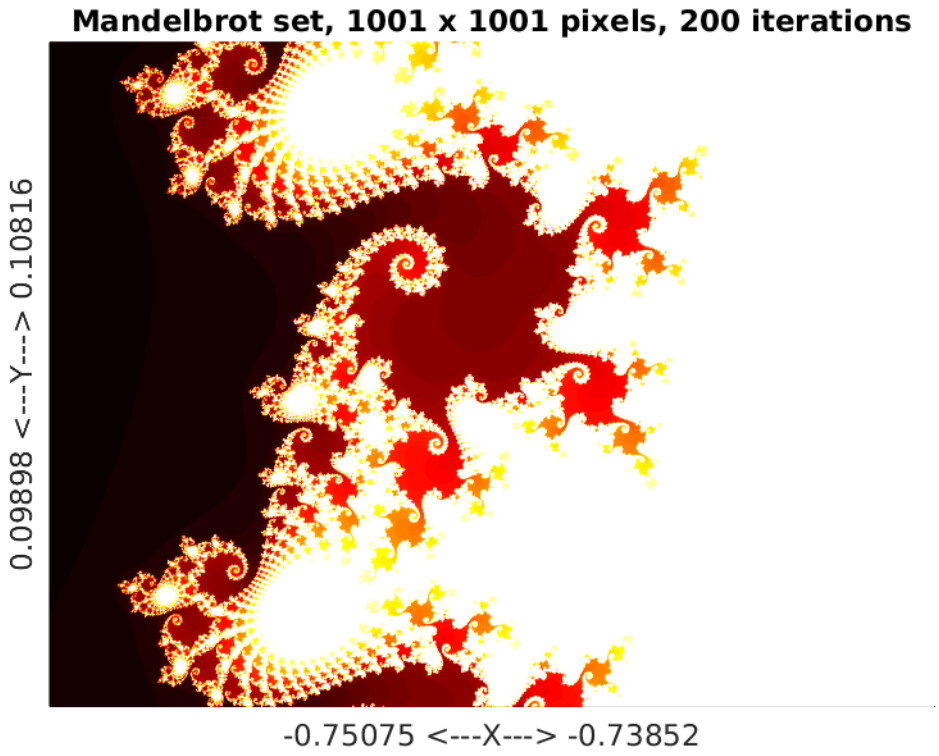
$$z_{n+1} = z_n^2 + c \quad \text{mit Startwert} \quad z_0 = 0$$

für $n \to \infty$ beschränkt ist. Da nicht unendlich viele Iterationen möglich sind, um zu bestimmen, ob $c \in \mathcal{M}$, behilft man sich folgendermaßen: Es ist bekannt, dass sobald ein z_n mit $|z_n| > 2$ auftritt (komplexer Betrag), dass die Folge dann unbeschränkt ist. Insbesondere ist also $\mathcal{M}$ in einem Kreis mit Radius 2 um den Ursprung enthalten. Man gibt sich also ein $n_{\max}$ vor und berechnet $z_1, z_2, \ldots$ solange bis man ein z_n findet mit $|z_n| > 2$, oder die maximale Anzahl an Iterationen erreicht ist, also $z_{n_{\max}}$ berechnet wurde. Im ersten Fall, oder falls $|z_{n_{\max}}| > 2$, entscheidet man, dass $c \notin \mathcal{M}$, im zweiten Fall $c \in \mathcal{M}$.

Um die gesamte Menge $\mathcal{M}$ oder Teilmengen davon grafisch darzustellen, implementiert man folgenden in Umgangssprache formulierten Algorithmus:

- Wähle ein Rechteck $R = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.
- Wähle endlich viele Punkte aus R, die auf Zugehörigkeit in $\mathcal{M}$ überprüft werden sollen. Die einfachste Methode ist, sich je eine Anzahl an Gitterpunkten N_x in x-Richtung und N_y in y-Richtung vorzugeben und dann äquidistante Punkte x_i, $i = 1, \ldots, N_x$ von $x_{\min}$ bis $x_{\max}$ bzw. y_j, $j = 1, \ldots, N_y$ von $y_{\min}$ bis $y_{\max}$ auszuwählen. Der Matlab-Befehl `linspace` kann hier hilfreich sein.
- Wähle eine obere Schranke $n_{\max}$, wie viele Folgenglieder maximal berechnet werden sollen.
- Initialisiere eine $(N_x \times N_y)$-Nullmatrix M.
- Führen Sie für jeden Punkt $c_{ij} = (x_i, y_j)$ folgende Schritte durch
  - Berechnen Sie die zugehörige komplexe Zahl $c_{ij} = x_i + i \cdot y_j$ (wobei das kleine i für den Index, das andere für die komplexe Einheit steht).
  - Berechnen Sie für diese Zahl die Folge (z_n) (initialisieren nicht vergessen!).
  - Prüfe nach jedem neu berechneten z_n, ob $|z_n| > 2$. In diesem Fall kann abgebrochen werden.
  - Falls die maximale Anzahl $n_{\max}$ von Iterationen erreicht ist und $|z_{n_{\max}}| < 2$ ist, setze $M(i,j) = 1$. Dies *markiert*, dass $c_{ij} \in \mathcal{M}$.
  - Bonus: Setzen Sie für diejenigen Punkte c_{ij}, bei der die Berechnung abgebrochen wurde $M(i,j) = \frac{\ell}{n_{\max}}$, wobei ℓ der Index war, bei dem die Berechnung abgebrochen wurde. Damit können Sie auch den Punkten, die nicht zu $\mathcal{M}$ gehören, einen Wert zuweisen, der angibt, wie schnell die Folge die kritische Schwelle von 2 überschreitet: je höher der Wert, desto länger hat es gedauert.
- Die Matrix M, die Sie als Ergebnis erhalten, können Sie als eine Funktion von $\mathbb{R}^2 \to \mathbb{R}$ betrachten, wobei $M(i,j)$ angibt, wie lange die obige Folge mit $c_{ij} = x_i + i \times y_j$ beschränkt bleibt. Funktionen von $\mathbb{R}^2 \to \mathbb{R}$ können mithilfe einer `colormap` visualisiert werden, wobei gleiche Farben für gleiche Funktionswerte stehen (vgl. Niveaulinien). Dies ist in der Vorlage bereits implementiert.

Laden Sie das Skript `mandelbrot_vorlage.m` herunter und realisieren Sie den beschriebenen Algorithmus an der in der Vorlage markierten Stelle. Testen Sie ihren Algorithmus mit verschiedenen Parametern, die Sie aus `mandelbrot_parameter.m` entnehmen können oder mit eigenen Einstellungen. Wenn Sie alles richtig implementiert haben, sollten Sie schöne Bilder wie das folgende erhalten.


Mandelbrot set, 1001 x 1001 pixels, 200 iterations
0.09898 <---Y---> 0.10816
-0.75075 <---X---> -0.73852