Technische Universität München
M2 - Numerische Mathematik
Dr. Laura Scarabosio

3.2. Übung. Einführung in die Programmierung (MA 8003)

Aufgabe 3.2.1: Plotten Sie mit den Befehlen `contour3`, `mesh`, `meshc`, `surf`, `surfl` und `waterfall` die Funktion

$$f(x,y) := x^2 - y^2 \text{ für } (x,y) \in [-1,1] \times [-1,1].$$

Erzeugen Sie dafür mit `meshgrid` genug Punkte in $[-1,1] \times [-1,1]$.

Drehen und verschieben Sie den Plot mit der Matlab-Oberfläche.

Lösung 3.2.1:

```
>> x = -1:0.01:1; y = -1:0.01:1;
>> [X,Y] = meshgrid(x,y);
>> mesh(X, Y, X.^2 - Y.^2)
>> contour3(X, Y, X.^2 - Y.^2)
...
```

Aufgabe 3.2.2: Wir betrachten folgende iterative Vorschrift mit $c \in \mathbb{C}$ und Startwert $z^1 := c$:

$$z^{k+1} = (z^k)^2 + c \quad \text{für } k = 1, 2, 3, \ldots \tag{1}$$

Die Folge der so erzeugten z^k bleibt entweder beschränkt oder geht dem Betrage nach gegen unendlich.

a) Schreiben Sie eine Funktion `function k = mandel(c, b, n)`, die den Index der Iterationen k zurückgibt, für den zum ersten Mal $|z^k| > b$ gilt. Wenn $|z^k| \leq b$ für alle Indizes von 1 bis `n` gilt, gebe `k`=0 zurück.
Achtung: Wenn `c` ein Vektor/Matrix ist, soll die Iteration für jedes Element `c(i)` durchgeführt werden und die Rückgabe ein Vektor/Matrix von gleicher Größe wie `c` mit Einträgen gleich dem jeweils erreichten Iterationsindex k bzw. `0` sein. Versuchen Sie insgesamt nur eine Schleife zu verwenden.

b) Testen Sie Ihre Funktion an dem Vektor `C = [-.43 + .62i, -.76 + .51i, -0.1 + .44i]` mit $b = 10$ und $n = 100$ Es sollte `k = [13,7,0]` zurückgegeben werden (± 1).

c) Wir wollen nun den Bereich plotten, in dem die Iterierten beschränkt bleiben. Schreiben Sie dazu eine zweite Funktion `plotmandel(x,y)`, welche als Eingabe zwei Vektoren `x` und `y` aus $\mathbb{R}^2$ erhält. Die Funktion soll folgendes machen:

- Erzeugen Sie mit `[X,Y] = meshgrid(...)` ein 300×300-Gitter des Gebiets $[x_1, x_2] \times [y_1, y_2]$.
- Verwenden Sie den Befehl `complex`, um die soeben erzeugten Punktepaare (X_i, Y_i) in einen Vektor `C` in der komplexen Ebene umzuwandeln.
- Rufen Sie nun Ihre Funktion `mandel` auf mit dem Vektor C, $n = 100$ und $b = 10$ und speichern Sie das Ergebnis in `it`.

- Rufen Sie nun `imagesc(X(1,:), Y(:,1), mod(it, 20))` auf.

d) Testen Sie Ihre Funktion an dem Bereich $[-2,1] \times [-1,1]$. Es sollte eine Ihnen bekannte Figur geplottet werden.

e) Suchen Sie sich einen schönen Teilbereich innerhalb $[-2,1] \times [-1,1]$ und verwenden ihn als Argument in `plotmandel`. Gegebenenfalls müssen Sie die Zahl der Iterationen erhöhen.

Lösung 3.2.2:

a)
```
function [ it ] = mandel( c, b, n )
z = c;
l = (abs(c) < b);
it = zeros(size(c));
for k=1:n
    z(l) = z(l).^2 + c(l);
    l2 = (abs(z) >= b);
    it(l2 & it == 0) = k;
    l = ~l2;
end
```

b)
```
function plotmandel( x,y )
[X,Y] = meshgrid(linspace(x(1),x(2), 300), linspace(y(1),y(2), 300));
C = complex(X,Y);
it = mandel(C, 10, 100);
it=reshape(it,300,300);
imagesc(X(1,:), Y(:,1), mod(it, 20))
% alterativ: pcolor(X, Y, mod(it, 20));
end

>> plotmandel([-2,1],[-1,1])
>> plotmandel([-1,-0.5],[0,0.4])
```

Aufgabe 3.2.3: Erzeugen Sie eine 20×20 Einheitsmatrix im dünnbesetzten Format. Wieviel Speicherplatz nimmt die Matrix in diesem Format ein? Visualisieren Sie die Struktur mit dem `spy` Befehl. Addieren Sie eine 20×20 Zufallsmatrix mit Einträgen zwischen 0 und 1, und setzen Sie alle Eintrage < 0.8 gleich 0. Wieviele Nichtnulleinträge hat die so erzeugte Matrix? Konvertieren Sie die Matrix in eine dünnbesetzte Matrix. Invertieren Sie die Matrix und berechnen Sie erneut die Anzahl der Nichtnulleinträge.

Lösung 3.2.3:

```
A = speye(20);
whos A
spy(A)
A = A + rand(20);
A(A<0.8) = 0;
nnz(A)
A = sparse(A);
nnz(inv(A))
```

Aufgabe 3.2.4: Viele wichtige Prozesse in Natur, Wirtschaft und Technik werden durch sogenannte partielle Differentialgleichungen beschrieben. Diese Prozesse haben gemeinsam, dass sie durch lokale Abhängigkeiten beschrieben werden können. Werden diese geeignet diskretisiert, so entstehen dadurch Matrizen, die nur wenige nicht verschwindende Einträge pro Zeile und Spalte haben. Noch dazu ist die Anzahl pro Zeile und Spalte unabhängig von der Dimension der Matrix. Besonders für große Matrizen lohnt sich daher die Speicherung als dünnbesetze Matrix. Ein grundlegendes Beispiel ist der Laplace-Operator, bzw. die 2. Ableitung bei nur einer Variable. Eine einfache Diskretisierung erzeugt (bis auf Konstanten) eine quadratische Matrix, die eine -2 auf der Diagonalen und jeweils eine 1 auf den beiden Nebendiagonalen besitzt. Alle anderen Einträge sind 0.

- Rufen sie die Dokumentation zum Befehl `spdiags` auf und verwenden sie einen geeigneten Aufruf von `spdiags` um solch eine Matrix der Dimension $n \times n$ zu erzeugen. Überprüfen sie das Ergebnis indem sie die Matrix für den Fall $n = 10$ als vollbesetzte Matrix anzeigen.
- Vergleichen Sie die beiden Matrixarten: Wieviele Einträge müssen sie im (nicht unrealistischen Fall) von $n = 10^6$ speichern? Wie wächst die Anzahl der zu speichernden Einträge mit n?
- Überzeugen Sie sich vom unterschiedlichen Speicherbedarf dadurch, dass sie die Matrix mit `spdiags` für $n = 10^6$ anlegen und diese dann (versuchen) in eine vollbesetzte Matrix konvertieren.

Lösung 3.2.4:

```
n = 10; m=10; e = ones(n,1); spdiags([e -2*e e],-1:1,n,m)
```

Aufgabe 3.2.5 (*): Wenn man in der Forschung oder in der Industrie arbeitet, muss man die Fähigkeit haben, Codes anderer Personen zu verstehen und zu bearbeiten.

Laden Sie das Matlab-Skript `zombies_game.m` von der Kurswebseite herunter. Dieses Programm ist eine Simulation eines Zombieangriffs. Die Zombies greifen Personen, die in der Nähe sind, an und werden nutzlos, wenn sie für 100 Minuten keine Person getroffen haben. Schauen Sie das Programm an und modifizieren Sie den Code bzgl. der folgenden Fragen.

- Versuchen Sie die Zombies doppelt so schnell spazieren zu lassen.
- Erlauben Sie eine Transformation von Personen zu Zombies bis zu einer Entfernung von 0.75.
- Lassen Sie die Zombies bis zu 200 Minuten anstatt 100 Minuten laufen.
- Ändern Sie die Hintergrundfarbe des Grafikfensters zu Blau.

Tipp: Um den Code zu verstehen, können die Debug-Werkzeuge nützlich sein.

Lösung 3.2.5:

- Linien 51–52 verändert mit:

```
zombies = zombies + ...
    [2*(0.2*rand(number_of_zombies,2)-.1),ones(number_of_zombies,1)];
```

- Linien 80–81 verändert mit:

```
if sqrt((people(i_people,1)-zombies(i_zombie,1)).^2 + ...
                (people(i_people,2)-zombies(i_zombie,2)).^2) < .75
```

- Linie 104 verändert mit:

```
if zombies(i_rotten,3) > 199
```

- Beide Linien 31 und 132 verändert mit:

```
whitebg('b');
```

Aufgabe 3.2.6 (*): Sehen Sie sich die Demo `echodemo volvec` an, um einen Eindruck zu bekommen, was Matlab an $3D$-Visualisierungen zu bieten hat.