Institut für Informationsverarbeitung
Leibniz Universität Hannover
http://www.tnt.uni-hannover.de

Leibniz Universität Hannover

Scientific Computing I
Probeprogrammierprüfung

Datum:	DD.MM.YYYY
Uhrzeit:	HH:MM - HH:MM Uhr
Bearbeitungszeit:	60min
Hilfsmittel:	ein handschriftlich beschriebenes DIN A4 Blatt (beidseitig) die eingebaute Hilfefunktion von MATLAB (*help*, *doc*)
Anzahl der Seiten:	4 (inkl. Deckblatt)

Name: _____

Matrikelnummer: _____

Alle Lösungen müssen in MATLAB implementiert werden.

Legen Sie für jede Aufgabe genau eine eigene Datei an. Nennen Sie die Datei genau so wie die Funktion heißt, die Sie implementieren sollen.

Schreiben Sie ihren Namen, ihre Matrikelnummer und die Aufgabennummer als Kommentar in jede Datei. Dateien ohne diese Angaben werden nicht bewertet, sondern werden genau wie auch Schmierpapier ignoriert.

Sie dürfen **KEINE** Programme außer MATLAB verwenden, insbesondere dürfen Sie **KEINE** Internetverbindung nutzen.

Wenn eine Aufgabe verbietet, dass sie gewisse Funktionen von MATLAB verwenden, dann gilt diese Einschränkung nur für diese eine Aufgabe.

Falls ihre Implementierung bei dem angegebenen Beispielaufruf das richtige Ergebnis liefert, ist das keine Garantie, dass ihre Implementierung fehlerfrei ist.

Die Aufgabenblätter müssen vollständig abgegeben werden.

Aufgabe 1: Determinante [4 Punkte]

Implementieren Sie das Berechnen der Determinante einer Matrix A mit Hilfe des Entwicklungssatzes von Laplace. Verwenden Sie folgenden Funktionskopf:

```
function x = determinante(A)
```

Prüfen Sie, ob die Eingaben korrekt sind:

- A muss quadratisch sein

Sie dürfen die MATLAB-Funktion *det* **NICHT** verwenden!

Sie können ihre Implementierung so in MATLABs Command Window testen:

```
>> determinante([1 2 3; 4 5 6; 7 8 10])
ans =
   -3
```

Aufgabe 2: Nullstelle einer Funktion finden [4 Punkte]

Implementieren Sie das Intervall-Bisektionsverfahren zum Finden einer Nullstelle einer Funktion $f : \mathbb{R} \to \mathbb{R}$ auf dem Intervall $[a, b]$ für verschiedene Toleranzen tol_i. Plotten Sie außerdem die Anzahl der benötigten Iterationen für jede der Toleranzen. Verwenden Sie folgenden Funktionskopf:

```
function x = bisektion(func, a, b, tol)
```

Dabei ist `func` ein Funktionshandle einer Funktion, welche f implementiert. Und `tol` ist ein Vektor von Toleranzen.
Prüfen Sie zunächst, ob die Eingaben korrekt sind:

- es muss $a < b$ gelten
- `tol` muss ein Vektor der Länge 2 oder mehr sein und kann dabei sowohl ein Spalten- als auch ein Zeilenvektor sein

Ihre Implementierung soll für alle tol_i die Nullstelle x_i von f auf dem Startintervall $[a, b]$ suchen. Zählen Sie dabei die Anzahl an Iterationen $iter_i$, d.h. Anpassungen der Intervallgrenzen, die benötigt wurden.
Plotten Sie die benötigten Iterationen für alle Toleranzen, d.h. wenn $iter_i$ die Anzahl der benötigten Iterationen für tol_i ist, dann soll ihr Plot alle Punkte $(i, iter_i)$ enthalten. Nutzen Sie magentafarbene Quadrate zur Darstellung der einzelnen Punkte. Geben Sie ihrem Plot einen Titel und beschriften Sie beide Achsen vernünftig.

Sie dürfen die MATLAB-Funktion *fzero* **NICHT** verwenden!

Sie können ihre Implementierung so in MATLABs Command Window testen:

```
>> bisektion(@(x) x - 0.5, 0, 1, [10^-6 10^-7 10^-8])
ans =
    0.5000    0.5000    0.5000
```

In ihrem Plot sollten nach obigem Aufruf magentafarbene Quadrate an den Positionen (1, 20), (2, 24) und (3, 27) zu sehen sein.

Aufgabe 3: Numerisches Differenzieren [4 Punkte]

Implementieren Sie das numerische Differenzieren einer tabellarisch definierten Funktion. Verwenden Sie folgenden Funktionskopf:

```
function dx = ableiten(dateiname)
```

Der Parameter `dateiname` enthält den Namen einer CSV-Datei, welche eingelesen werden soll. Die Datei hat keine Header oder ähnliches. Sie enthält eine Tabelle mit zwei Spalten und einer variablen Anzahl n an Zeilen ($n \geq 3$). Die erste Spalte enthält Stützstellen x_i (streng monoton steigend) und die zweite Spalte Funktionswerte $f(x_i)$. Der Abstand zwischen zwei beliebigen aufeinanderfolgenden Stützstellen x_i und x_{i+1} ist konstant.
Überprüfen Sie die Korrektheit des Inhalts der CSV-Datei:

- die Tabelle muss mind. drei Zeilen enthalten
- der Abstand zwischen allen Paaren aus aufeinanderfolgenden Stützstellen x_i und x_{i+1} muss konstant sein und zwar mit einem Wert echt größer als 0 (das impliziert auch die geforderte Monotonieeigenschaft)

Ihre Funktion soll die ersten beiden Ableitungen von f mit endlichen Differenzen und zentralem Differenzenquotienten berechnen und zurück geben. Dabei soll `dx` eine Matrix mit drei Zeilen und $n - 2$ Spalten sein:

- 1. Zeile: die Stützstellen $x_2, x_3, \ldots, x_{n-1}$, für welche die Ableitungen bestimmt wurden
- 2. Zeile: die erste Ableitung, so dass gilt: `dx(2,i)` enthält die erste Ableitung von f an Stelle `dx(1,i)`
- 3. Zeile: die zweite Ableitung, so dass gilt: `dx(3,i)` enthält die zweite Ableitung von f an Stelle `dx(1,i)`

Um ihre Implementierung testen zu können, legen Sie zunächst ein neues (MATLAB-)Skript an und nennen es *test.m*. Schreiben Sie folgenden Inhalt in diese Datei:

```
1,3
2,5
3,1
4,2
5,2
6,2
7,3
```

Benennen Sie diese Datei nach dem Speichern in *test.csv* um. Anschließend können Sie ihre Implementierung so in MATLABs Command Window testen:

```
>> ableiten("test.csv")
ans =
    2.0000    3.0000    4.0000    5.0000    6.0000
   -1.0000   -1.5000    0.5000         0    0.5000
   -6.0000    5.0000   -1.0000         0    1.0000
```

Aufgabe 4: Informationsgewinn [4 Punkte]

Implementieren Sie das Aufteilen einer Menge `M` in zwei echte Teilmengen `L` und `R` basierend auf dem Informationsgewinn, so wie es z.B. beim Erzeugen eines Entscheidungsbaums nötig ist. Verwenden Sie folgenden Funktionskopf:

```
function [L, R] = informationsgewinn(M, k)
```

Der Parameter `M` ist dabei eine $2 \times n$-Matrix, welche in der ersten Zeile Werte eines Merkmals enthält und in der zweiten Zeile eine Klassenzugehörigkeit als natürliche Zahl zwischen 1 und k. Beispiel für $k = 2$:

0.9	0.8	0.65	0.5	0.45	0.3	0.15	0.01
1	1	2	2	2	1	1	1

Im Beispiel hat der dritte Eintrag (= die dritte Spalte) eine Merkmalsausprägung von 0.65 und die Klasse 2. Die Sortierung ist Zufall.

Prüfen Sie in ihrer Implementierung, dass `M` in der zweiten Zeile nur Einträge zwischen 1 und k enthält.

Geben Sie das Paar `L` und `R` zurück, welches den maximalen Informationsgewinn IG erzielt. Nutzen Sie dafür die Definition des IG, welche den Gini-Index statt der Entropie nutzt.

Sie können ihre Implementierung so in MATLABs Command Window testen:

```
>> M = [.9 .8 .65 .5 .45 .3 .15 .01; 1 1 2 2 2 1 1 1];
>> [L, R] = informationsgewinn(M, 2)
L =
    0.3000    0.1500    0.0100
    1.0000    1.0000    1.0000
R =
    0.9000    0.8000    0.6500    0.5000    0.4500
    1.0000    1.0000    2.0000    2.0000    2.0000
```

Hinweis: die Reihenfolge der Spalten in der Ausgabe ist beliebig.