Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Umgang mit der Matlab-Umgebung
- Schreiben einfacher Skripte und Funktionen in Matlab
- Berechnung der Komplexität von Algorithmen
- Umgang mit Zahldarstellungen zur Basis 256
- Realisierung einer Routine zum Thema Rundung, insbesondere „round2even“

Am Anfang wollen wir Ihre Kenntnisse über das Lösen von linearen Gleichungssystemen und die Programmierung in Matlab etwas auffrischen. Dazu stellen wir Ihnen einige Fragen bzw. einige off-line Aufgaben.

Machen Sie bitte zuerst die folgenden off-line Übungen bevor Sie sich einloggen!

Off-line Aktivitäten

Übereinstimmen

Schreiben Sie vor jeden Begriff auf der linken Seite den passenden Buchstaben der Beschreibung, die am besten mit der aus der rechten Spalte übereinstimmt.

____	1.	$O(n^3)$ Operat.	a.	Aufwand zur Berechnung von `Rx` (`R` Dreiecksmatrix).
____	2.	`i = i + 4`	b.	Zeichenkette
____	3.	RAM	c.	Berechnet den größten gemeinsamen Teiler.
____	4.	`gcd(6,12)`	d.	Skriptsprache
____	5.	n^2 Operationen	e.	Inkrement $\neq 1$
____	6.	$\approx 2n^3$ Operat.	f.	Anweisung
____	7.	`lcm(14,8)`	g.	Berechnet das kleinste gemeinsame Vielfache.
____	8.	'23'	h.	Aufwand des Gauß-Verfahrens
____	9.	1 : 3 : 5	i.	Random-access memory
____	10.	Matlab	k.	Aufwand zur Berechnung von `AB` (`A,B` aus $R^{n \times n}$).

Füllen Sie die Lücken aus

Ergänzen Sie die folgenden Sätze.

11. The default variable name used for results is _____.
12. Variablenames in Matlab must start with a letter, followed by any number of underscores, digits, or _____.
13. Flops is an abbreviation for floating point _____ per _____.
14. **`r = size(A,2)`** returns the number of _____ in **`A`** in the variable **`r`**.
15. The statement _____ extracts upper triangular part of the matrix **`A`**.
16. Comments in a Matlab program will be made by using _____.

Kurz und knapp

Geben Sie bitte eine kurze Antwort zu jeder der folgenden Fragen. Ihre Antwort sollte so kurz und präzise wie möglich sein; versuchen Sie es mit zwei bis drei Sätzen.

17. Wieso lautet das Ergebnis der Anweisung **`y = floor(1.16 * 100)`**
nicht **`116`** sondern **`115`**?
Ihre Antwort:

18. Bei welchen Operatoren kann der Punkt . überall auftreten?
Ihre Antwort:

19. Die folgenden Funktionen sind gemeinsam in einer Datei `paul.m` gespeichert?

```
1        function value = peter(A)
2        value = multipliziere(A);
3
4        function B = multipliziere(A)
5        B = 3*A;
6        B = B-1;
```

Wieso erhalte ich die Fehlermeldung
`??? Undefined command/function 'peter',`
wenn ich `peter(5)` im Kommando-Fenster eingebe?

Ihre Antwort:

Programmausgaben

Für jedes der folgenden Programmsegmente, lesen Sie zuerst die Zeilen und schreiben Sie die Ausgabe an die dafür vorgesehene Stelle.

20. Wie lautet die Ausgabe des folgenden Skripts?

```
1        a = [1;2;3;4];
2        b = a + i*a
```

Ihre Antwort:

21. Wie lautet die Ausgabe des folgenden Skripts?

```
1        a = [1;2;3;4];
2        b = a + i*a;
3        c = b'
4        d = b.'
```

Ihre Antwort:

22. Wie lauten die Werte von x und y nach Ausführen des folgenden Skripts?

```
1        B = [5 -3; 2 -4];
2        x = abs(B)>2;
3        y = B(abs(B)>2);
```

Ihre Antwort:

23. Welche Dimension haben **`i, r, c`** nach dem die Zeilen 1-3 ausgeführt wurden?

```
1        A = [1 2 3; 4 5 6; 7 8 9];
2        i = find(A>5);
3        [r,c] = find(A>5);
```

Ihre Antwort:

Korrigieren Sie den Code

Für jedes der folgenden Codesegmente sollen Sie feststellen, ob ein Fehler enthalten ist. Falls ein Fehler vorliegt, markieren Sie diesen und spezifizieren Sie, ob es sich dabei um einen logischen oder Syntaxfehler handelt. Schreiben Sie die korrigierten Anweisungen jeweils in jeden dafür vorgesehenen Bereich unter der Problemstellung. Falls das Segment keinen Fehler enthält, schreiben Sie einfach „kein Fehler“. [Bemerkung: Es kann sein, dass ein Programm mehrere Fehler enthält.]

24. Die folgende Matlab-Funktion soll das Produkt der beiden Matrizen A und B zurückgeben.

```
1        function C = MatrixProdukt(A,B);
2        C = zeros(size(A,1), size(B,2));
3        for i = 1 : size(A,1)
4          for j = 1 : size(B,2)
5            for k = 1 : size(A,2)
6              C(i,j) = C(i,j) + A(i,k) * B(k,i)
7            end
8          end
9        end
```

Ihre Antwort:

25. Das folgende Skript sollte die Funktion **`y(x) = arctan(x^2)`** auf dem Bildschirm ausgeben:

```
1        fplot('arctan(x^2)',[-10 10]);
```

Ihre Antwort:

Kurz und knapp II

Geben Sie bitte eine kurze Antwort zu jeder der folgenden Fragen. Ihre Antwort sollte so kurz und präzise wie möglich sein; versuchen Sie es mit zwei bis drei Sätzen.

1. Welche Zahltypen sind schwieriger zu vergleichen **`int32`** oder **`double`**?

   Wieso?

   Ihre Antwort:

2. Wieso lautet das Ergebnis der Anweisung **`y = char(0+48)`** nicht **`48`** sondern **`0`**?

   Ihre Antwort:

3. Wieso lautet das Ergebnis der Anweisungen **`a = '0'; b=a+3;`** nicht **`b = 3`** sondern **`b = 51`**?

   Ihre Antwort:

4. Worin unterscheiden sich die Routinen **`fix`**, **`floor`**, **`ceil`** und **`round`**?

   Ihre Antwort:

5. Durch welchen Trick kann ich die oben genannten Routinen dazu bringen die Rundungen bzgl. der ersten, zweiten oder dritten Stelle nach dem Komma zu betreiben?

   Ihre Antwort:

Praktikumsaufgabe 5 – „round to even“

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

• **Problembeschreibung**

Schreiben Sie eine Routine **`round2even.m`** welche zu einem skalaren Argument, das „round2even“-gerundete Ergebnis liefert.

Zur Erinnerung: sind die Nachkommastellen kleiner oder größer ½ so wird „normal“ auf- bzw. abgerundet, z.B. mit der Routine **`round`**. Ist der Wert der Nachkommastellen gerade ½ , so wird zur nächsten geraden Zahl gerundet. Die Routine soll für positive und negative Zahlen funktionieren.

Praktikumsaufgabe 6 – Darstellung von Zahlen zur Basis 256

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

• **Problembeschreibung**

Mit dem Datentyp **`uint8`** (unsigned integer 8 Bit) lassen sich Zahlen von 0 bis 255 darstellen. Identifizieren wir in einem gegebenen Vektor c=(c_0,... c_n) einen Eintrag c_j mit dem Koeffizienten c_j in der Darstellung c_0 * 256^k + c_1 * 256^(k-1) + ... +c_n * 256^(k-n), so entsprechen die c_j's den Ziffern in einer Zahldarstellung zur Basis 256.

Man schaue sich die Funktionsweise der Routine **`mpShortDiv`** an und programmiere eine Routine **`mp2DezInt,`** die eine ganzzahlige Darstellung zur Basis 256 in eine Darstellung zur Basis 10 umrechnet und den Wert als String zurückgibt. Dabei seien beide Zahlen in der normalen Richtung zu lesen, von links nach rechts.
Beispiel: Zur Eingabe **`c =[2,3,1]`** soll die Routine den String **`'131841'`** zurückgeben. Man beachte 131841 = 2 * 256^2 + 3 * 256 + 1.

```
function s = mp2DezInt(a)
% Converts a radix 256 integer a(1:n) (radix point after a(n)) to a
% decimal integer represented as an ascii string.
s = [];
while ~isempty(a) && ~(length(a)==1 & a(1)==0)
  %
  % hier soll die Routine mpShortDiv verwendet werden,
  % die a durch 10 teilt, das Ergebnis sollte dann wieder
  % in a gespeichert werden und der Rest in rem.
  % Wieso funktioniert das?
  %
  if a(1) == 0
    a = a(2:end);
  end
  s = [num2str(rem,1),s];
end
if isempty(s)
  s = num2str(0,1);
end
```

• **Vorlage**
Die Dateien `mpShortDiv.m` und `Mp2DezInt.m` können Sie von der Vorlesungs-Homepage downloaden.

Praktikumsaufgabe 7 – π mit mehrfacher Präzision

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

• **Problembeschreibung**
Man mache sich klar, dass die Routine **`mpAdd`** die übliche Addition zur Basis 256 durchführt, so wie Sie sie zur Basis 10 kennen. Addition von rechts Ziffer für Ziffer; überschreitet die Summe zweier Ziffern, den darstellbaren Zahlbereich, so bleibt ein Übertrag im Register („einen im Sinn“). Diese Aufteilung entspricht gerade den Routinen loByte und hiByte.

Vervollständigen Sie die Routine **`mpSub`** so, dass sie als Ergebnis die Differenz zweier Zahlen **`u`** und **`v`** (**`u`** und **`v`** als Vektoren gleichlang) liefert und zusätzlich einen Flag, welcher -1 sein soll, falls u<v ist und 0 sonst.

Vervollständigen Sie die folgende Routine zur Subtraktion.

```
function [w,flag] = mpSub(u,v)
% Subtracts the unsigned radix 256 integer v(1:n) from u(1:n)
% yielding the unsigned integer w(1:n).
% If the result is negative (wraps around),
% flag is returned as -1; otherwise it is returned as 0.
n = length(u);
if n ~= length(v)
  error('Input vectors must have the same length!')
end
w = zeros(1,n,'uint8');
ireg = 256;
for j=n:-1:1
  %
  % verknüpfen Sie hier die Ausdrücke 255, double(u(j)),
  % double(v(j)),double(hiByte(ireg)) sinnvoll und weisen Sie
  % das Ergebnis ireg zu
  %
  w(j)=loByte(ireg);
end
flag=hiByte(ireg)-1;
```

Testen Sie das ganze z.B. An

```
a = [1,2,3]; b = [1,2,2];
[c ,flag] = mpSub(a,b);
mp2DezInt(a)
mp2DezInt(b)
mp2DezInt(c)
flag
```

und, stimmt es?

Haben Sie alle Routinen richtig realisiert, so ist **`mpPi.m`** die Realisierung des AGM-Verfahrens, mit dem sich π mit vorgegebener Genauigkeit berechnen läßt.

Arithmetic Geometric Mean Method (AGM) to Compute π

Initiate

$$X_o = \sqrt{2}$$
$$\pi_0 = 2 + \sqrt{2}$$
$$Y_0 = \sqrt[4]{2}$$

and then, for i=0,1,2,… repeat the iteration

$$X_{i+1} = \frac{1}{2}\left(\sqrt{X_i} + \frac{1}{\sqrt{X_i}}\right)$$
$$\pi_{i+1} = \pi_i \left(\frac{1 + X_{i+1}}{1 + Y_i}\right)$$
$$Y_{i+1} = \frac{Y_i \sqrt{X_{i+1}} + \frac{1}{\sqrt{X_{i+1}}}}{1 + Y_i}$$

Wenn Sie `mpPi(70)` eingeben, sollten Sie auf dem Bildschirm nach 1-2 Minuten (und 9 kleinen Punkten) das Ergebnis

```
pi=
3.14159265358979323846264338327950288419716939937510582097494459230781640628
62089986280348253421170679821480865132823066470938446095505822317253594081
284811174502841016
```

erhalten.

Aufgabe*: Man zeige, dass die Zahl 2^512+1 keine Primzahl ist, denn sie hat Teiler 7,455,602,825,647,884,208,337,395,736,200,454,918,783,366,342,657 und 2,424,833.

Exercise*: As an exercise, you might enjoy checking the first hundred digits given above against the first 12 terms of Ramanujan's celebrated identity

$$\frac{1}{\pi} = \frac{\sqrt{8}}{9801} \sum_{n=0}^{\infty} \frac{(4n)!(1103 + 26390n)}{(n!396^n)^4}$$

using the above routines.