# WiMa-Praktikum 1

Universität Ulm, Sommersemester 2019

# Woche 1

---

## Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Umgang mit der MATLAB-Umgebung
- Matrizen und Vektoren zu erstellen
- Arbeiten mit Matrizen und Vektoren
- Schreiben einfacher Skripte in MATLAB

Am Anfang wollen wir Ihnen eine kurze Einführung in MATLAB geben.

Beantworten Sie danach bitte erst einige Fragen, bevor Sie sich an den Rechner setzen!

---

# Kurze Einführung in Matlab

## Einfache mathematische Operationen

MATLAB ist eine Programmiersprache, deren Programme im Gegensatz zu vielen anderen Programmiersprachen nicht compiliert werden müssen, sondern durch einen Interpreter ausgeführt werden. Daher kann man genauso wie mit einem simplen Taschenrechner auch mit MATLAB einfache mathematische Operationen direkt ausführen, z. B. ergibt die Eingabe

```
≫ 3 + 4
```

die Ausgabe

```
ans =
     7
```

Man beachte, dass MATLAB im Allgemeinen keine Zwischenräume benötigt, um Befehle eindeutig zu verstehen. Alternativ zu dem obigen Beispiel können in MATLAB auch Variablen verwendet werden.

```
≫ a = 3
a =
     3
≫ b = 4
b =
     4
≫ c = a + b
c =
     7
```

MATLAB besitzt folgende einfache arithmetische Operationen:

| Operation | Symbol | Beispiel |
|---|---|---|
| Addition, $a + b$ | $+$ | $5 + 3$ |
| Subtraktion, $a - b$ | $-$ | $23-12$ |
| Multiplikation, $a \cdot b$ | * | 13.3 * 63.13 |
| Division, $a \div b$ | / or \ | $17/4 = 4\backslash 17$ |
| Potenz, $a^b$ | ^ | $3\hat{}4$ |

Die Reihenfolge, in der eine Folge von Operationen abgearbeitet wird, lässt sich wie folgt beschreiben: Ausdrücke werden von links nach rechts ausgeführt, wobei die Potenzierung die höchste Priorität besitzt, gefolgt von Punkt-Operationen, sprich Multiplikation und Division. Die niedrigste Priorität haben Addition und Subtraktion. Mit Hilfe von Klammern kann diese Vorgehensweise geändert werden, wobei innere Klammern vor äußeren Klammern berechnet werden.

Man beachte die besondere Bedeutung von / bzw. \. Der Operator / erzeugt die Rechtsinverse, \ die Linksinverse oder einfacher: „Es wird durch das geteilt, was unter dem Bruchstrich steht“. Für Skalare scheint diese Notation etwas übertrieben zu sein, im Zusammenhang mit Matrizen wird es Ihnen später einleuchten.

## Variablen

Wie in anderen Programmiersprachen hat auch MATLAB Regeln für Variablennamen. Eine Variable repräsentiert ein Datenelement, dessen Wert während der Programmausführung – gegebenenfalls mehrfach – geändert werden kann. Variablen werden anhand ihrer „Namen“ identifiziert. Namen bestehen aus ein bis neunzehn Buchstaben, Ziffern oder Unterstrichen, wobei das erste Zeichen ein Buchstabe sein muss. Man beachte, dass MATLAB Groß- und Kleinschreibung unterscheidet. (Windows ist im Gegensatz zu Linux nicht so restriktiv. Da Programme jedoch betriebssystemunabhängig sein sollten, sollten Windows-Benutzer besondere Aufmerksamkeit walten lassen.)
Einer Variablen ist Speicherplatz zugeordnet. Wenn man eine Variable verwendet, dann meint man damit entweder den zugeordneten Speicherplatz oder den Wert, der dort augenblicklich abgespeichert ist. Einen Überblick über alle aktuell verwendeten Variablen erhält man mit dem Befehl `who` oder `whos`, wobei letzterer die Angabe des benötigten Speicherplatzes beinhaltet.
Zusätzlich zu selbstdefinierten Variablen gibt es in MATLAB verschiedene spezielle Variablen. Diese lauten

| **spezielle Variablen** | **Wert** |
|---|---|
| ans | Standard-Variablenname benutzt für Ergebnisse |
| pi | $3.1415\ldots$ |
| eps | Maschinengenauigkeit |
| inf | steht für Unendlich (eng. infinity). z. B. $1/0$ |
| NaN | eng. Not a Number, z. B. $0/0$ |
| i (und) j | $i = j = \sqrt{-1}$ |

In MATLAB kann der Speicherplatz, der durch Variablen belegt ist, durch den Befehl `clear` wieder freigegeben werden, z. B.

```
≫ clear a b c
```

## Spezielle Funktionen

Eine unvollständige Liste von Funktionen, die MATLAB bereitstellt, ist im Folgenden dargestellt. Die meisten Funktionen sind so definiert, wie man sie üblicherweise benutzt.

```
≫ y = cos(pi)
y =
     -1
```

MATLAB bezieht sich im Zusammenhang mit Winkelfunktionen auf das Bogenmaß.

| Funktion | Bedeutung |
|---|---|
| abs(x) | Absolutbetrag |
| cos(x) | Kosinus |
| exp(x) | Exponentialfunktion: $e^x$ |
| fix(x) | rundet auf die nächste vom Betrag her kleinere ganze Zahl |
| floor(x) | rundet auf die nächste kleinere ganze Zahl |
| gcd(x,y) | größter gemeinsamer Teiler von x und y |
| lcm(x,y) | kleinstes gemeinsames Vielfaches von x und y |
| log(x) | natürlicher Logarithmus |
| rem(x,y) | Modulo (eng. remainder of division), z. B. rem(5,2)=1 |
| sign(x) | Signum Funktion, z. B. sign(2.3) = 1, sign(0) = 0, sign(-.3) = -1 |
| sin(x) | Sinus |
| sqrt(x) | Quadratwurzel |
| tan(x) | Tangens |

## Matrixkonstruktion und Adressierung

| einfache Matrix Konstruktionen | |
|---|---|
| $x$=[1 4 2*pi 4] | erstelle einen Zeilenvektor $x$ mit genannten Einträgen |
| $x$=$anfang$:$ende$ | erstelle einen Zeilenvektor $x$ beginnend mit $anfang$, Inkrement 1 und endend mit $ende$ |
| $x$=$anfang$:$inkrement$:$ende$ | Ähnliches wie oben mit dem Inkrement $inkrement$ |
| $x$=linspace($anfang$,$ende$,$n$) | erzeugt einen Zeilenvektor der Dimension $n$ mit $x(i) = \frac{(n-i)\cdot anfang+(i-1)\cdot ende}{n-1}$ $(i = 1, \ldots, n)$ |

Im Folgenden sind einige charakteristische Beispiele aufgeführt.

```
≫ B = [1 2 3 4; 5 6 7 8]
B =
     1 2 3 4
     5 6 7 8
```

Der Operator ′ liefert für reelle Matrizen die Transponierte.

```
≫ C = B'
C =
     1 5
     2 6
     3 7
     4 8
```

Der Doppelpunkt : in der zweiten Komponente spricht alle vorhandenen Spalten an, d. h. er ist hier ein zu `1:4` äquivalenter Ausdruck.

```
≫ C = B(1,:)
C =
     1 2 3 4
≫ C = B(:,3)'
C =
     3 7
```

Mit `end` kann man auf die jeweilige Dimension einer Matrix zugreifen.

```
≫ B(:,3:end)
ans =
     3 4
     7 8
```

Es lassen sich auch einzelne Komponenten neu definieren.

```
≫ A = [1 2 3; 4 5 6; 7 8 9]
A =
     1 2 3
     4 5 6
     7 8 9
≫ A(1,3) = 9
A =
     1 2 9
     4 5 6
     7 8 9
```

Ist ein Eintrag noch nicht definiert, so verwendet MATLAB die minimale Erweiterung dieser Matrix und setzt undefinierte Einträge zu Null.

```
≫ A(2,5) = 4
A =
     1 2 9 0 0
     4 5 6 0 4
     7 8 9 0 0
```

Im Folgenden werden die Vektoren `(3,2,1)` und `(2,1,3,1,5,2,4)` dazu verwendet, die Matrix `C` zu indizieren, d. h. `C` hat die Struktur

```
A(3,2) A(3,1) A(3,3) A(3,1) A(3,5) A(3,2) A(3,4)
A(2,2) A(2,1) A(2,3) A(2,1) A(2,5) A(2,2) A(2,4)
A(1,2) A(1,1) A(1,3) A(1,1) A(1,5) A(1,2) A(1,4)
```

In MATLAB erhält man nun

```
≫ C = A(3:-1:1,[2 1 3 1 5 2 4])
C =
     8 7 9 7 0 8 0
     5 4 6 4 4 5 0
     2 1 9 1 0 2 0
```

Ein weiteres Beispiel für Indizierung ist

```
≫ C = C(1:2,2:3)
C =
     7 9
     4 6
```

Im nächsten Beispiel wird ein Spaltenvektor dadurch konstruiert, dass alle Elemente aus der Matrix `C` hintereinander gehängt werden. Dabei wird spaltenweise vorgegangen.

```
≫ b=  C(:)'
b =
     7 4 9 6
```

Das Löschen einer ganzen Zeile oder Spalte kann durch das Umdefinieren in eine $0 \times 0$-Matrix geschehen, z. B.

```
≫ C(2,:) = [ ]
C =
     7 9
```

## Skalar-Matrix-Operationen

In MATLAB sind Skalar-Matrix-Operationen in dem Sinne definiert, dass Addition, Subtraktion, Division und Multiplikation mit einem Skalar elementweise durchgeführt werden. Es folgen zwei erklärende Beispiele.

```
≫ B - 1
ans =
     0 1 2 3
     4 5 6 7
≫ 9 + 3 * B
ans =
     12 15 18 21
     24 27 30 33
```

## Matrix-Matrix-Operationen

Die Operationen zwischen Matrizen sind nicht so kanonisch zu definieren wie die zwischen Skalar und Matrix, insbesondere sind Operationen zwischen Matrizen unterschiedlicher Dimension schwer zu definieren. Des Weiteren sind die Operationen $*$ und $.*$, bzw. $/$ und $./$ sowie $\backslash$ und $.\backslash$ zu unterscheiden. In nachfolgender Tabelle sind die Matrixoperationen beschrieben.

| **komponentenweise Matrixoperationen** | |
|---|---|
| Beispieldaten | $a = [a_1, a_2, \ldots, a_n]$, $b = [b_1, b_2, \ldots, b_n]$, $c$ ein Skalar |
| komp. Addition | $a + c = [a_1 + c,\ a_2 + c, \ldots, a_n + c]$ |
| komp. Multiplikation | $a * c = [a_1 \cdot c,\ a_2 \cdot c,\ \ldots,\ a_n \cdot c]$ |
| Matrix–Addition | $a + b = [a_1 + b_1,\ a_2 + b_2,\ \ldots,\ a_n + b_n]$ |
| komp. Matrix-Multiplikationen | $a. * b = [a_1 \cdot b_1,\ a_2 \cdot b_2,\ \ldots,\ a_n \cdot b_n]$ |
| komp. Matrix-Div. von rechts | $a./b = [a_1/b_1,\ a_2/b_2,\ \ldots,\ a_n/b_n]$ |
| komp. Matrix-Div. von links | $a.\backslash b = [b_1/a_1,\ b_2/a_2,\ \ldots,\ b_n/a_n]$ |
| komp. Matrix-Potenz | $a.\hat{}\,c = [a_1^c,\ a_2^c,\ \ldots,\ a_n^c]$ |
| | $c.\hat{}\,a = [c^{a_1},\ c^{a_2},\ \ldots,\ c^{a_n}]$ |
| | $a.\hat{}\,b = [a_1^{b_1},\ a_2^{b_2},\ \ldots,\ a_n^{b_n}]$ |

Es folgen nun einige Beispiele zu Matrixoperationen

```
≫ g = [1 2 3; 4 5 6]; % zwei neue Matrizen
≫ h = [2 2 2; 3 3 3];
≫ g+h   % addiere g und h komponentenweise
ans =
     3  4  5
     7  8  9
≫ ans-g   % subtrahiere g von der vorherigen Antwort
ans =
     2  2  2
     3  3  3
≫ h.*g   % multipliziere g mit h komponentenweise
ans =
      2   4   6
     12  15  18
≫ g*h'   % multipliziere g mit h'
ans =
     12  18
     30  45
```

## Matrix-Operationen und -Funktionen

| **Matrixfunktionen** | |
|---|---|
| reshape(A,m,n) | erzeugt aus den Einträgen der Matrix $A$ eine $m \times n$-Matrix, wobei die Einträge spaltenweise aus $A$ gelesen werden. |
| diag(A) | ergibt die Diagonale von $A$ als Spaltenvektor |
| diag(v) | erzeugt eine Diagonalmatrix mit dem Vektor $v$ in der Diagonalen |
| tril(A) | extrahiert den unteren Dreiecksanteil der Matrix $A$ |
| triu(A) | extrahiert den oberen Dreiecksanteil der Matrix $A$ |

Es folgen einige Beispiele

```
≫ g = linspace(1,9,9)   % ein neuer Zeilenvektor
g =
      1    2    3    4    5    6    7    8    9
≫ reshape(g,3,3)   % macht aus g eine 3 x 3 Matrix
ans =
      1    4    7
      2    5    8
      3    6    9
≫ B = reshape(g,[],3)   % macht ebenso aus g eine 3 x 3 Matrix
B =
      1    4    7
      2    5    8
      3    6    9
≫ tril(B)
ans =
      1    0    0
      2    5    0
      3    6    9
```

| Funktion | Bedeutung |
|---|---|
| R=chol(A) | Choleskyzerlegung, so dass $R^T$*R = A gilt |
| cond(A) | Konditionszahl der Matrix $A$ |
| d=eig(A) | Eigenwerte und -vektoren |
| [V,d]=eig(A) | |
| det(A) | Determinante |
| hess(A) | Hessenbergform |
| inv(A) | Inverse |
| [L,U]=lu(A) | LR-Zerlegung gegeben durch Gauss-Algorithmus |
| norm(A) | euklidische-Norm |
| rank(A) | Rang der Matrix $A$ |

**Bemerkung:** Der \ Operator ist auch für Matrizen definiert und liefert in Kombination mit Vektoren für reguläre Matrizen ihre Inverse, d. h. `A^(-1)x=A\x` .

## Spezielle Matrizen

| | spezielle Matrizen |
|---|---|
| eye(n) | erzeugt eine Einheitsmatrix der Dimension $n \times n$ |
| ones(m,n) | erzeugt eine $m \times n$-Matrix mit den Einträgen $1$ |
| zeros(m,n) | erzeugt eine $m \times n$-Matrix mit den Einträgen $0$ |

## Skript-Dateien

Für einfache Probleme ist es schnell und effizient, die Befehle am MATLAB -Prompt einzugeben. Für größere und umfangreichere Aufgabenstellungen bietet MATLAB die Möglichkeit, sogenannte Skript-Dateien zu verwenden, in denen die Befehle in Form einer Textdatei aufgeschrieben sind und die man am Prompt übergibt. MATLAB öffnet dann diese Dateien und führt die Befehle

so aus, als hätte man sie am Prompt eingegeben. Die Datei nennt man Skript-Datei oder M-Datei, wobei der Ausdruck M-Datei daher rührt, dass diese Dateien das Suffix `.m` haben, z. B. `newton.m`. Um eine M-Datei zu erstellen, ruft man einen Editor auf und speichert die Datei in dem Verzeichnis, von dem aus man MATLAB gestartet hat oder starten wird. Die Datei, z. B. `newton.m`, wird in MATLAB dann durch Eingabe von `newton` am Prompt aufgerufen, sofern diese Datei im aktuellen Verzeichnis liegt.
Für die Benutzung von Skript-Dateien hat MATLAB unter anderem folgende hilfreichen Befehle

| **M-Datei-Funktionen** | |
|---|---|
| disp(ans) | zeigt den Wert der Variablen `ans`, ohne ihren Namen auszugeben |
| input | erwartet vom Benutzer eine Eingabe |
| keyboard | übergibt zeitweise die Kontrolle an die Tastatur |
| pause | hält das Programm an, bis eine Taste betätigt wird |

Die folgende Skript-Datei `beispiel1.m`

```
% beispiel1.m
% Beispiel fuer eine Skript-Datei
tmp = input(' Geben Sie bitte eine Zahl an >' );
3 * tmp
```

führt zu der Ausgabe

```
≫ beispiel1
Geben Sie bitte eine Zahl an > 6
ans =
     18
```

Wie soeben festgehalten, lässt sich eine Abfolge von MATLAB-Befehlen auch in einer Datei speichern. Diese kann man dann am Befehl-Fenster aufrufen und die gespeicherte Folge von Befehlen wird ausgeführt. Alle o. g. Befehle lassen sich in einer Datei z. B. mit dem Namen `abc.m` zusammenstellen. Für die Wahl des Dateinamens gelten dabei die folgenden Regeln:

- das erste Zeichen ist ein Buchstabe und
- die Datei hat die Endung **.m**

Um ein solches M-File zu erstellen, ruft man den Editor (z. B. mit `edit` von der Kommandozeile aus) auf, der es erlaubt, Text einzugeben und diesen als Datei zu speichern. Wenn man den Editor gestartet hat, gebe man die folgenden Befehle ein und speichere die Datei unter dem Namen `abc.m`

```
a = 1;
b = 3;
c = -5;
x1 = (-b + sqrt(b^ 2 - 4*a*c))/(2*a)
x2 = (-b - sqrt(b^ 2 - 4*a*c))/(2*a)
```

Um nun die Befehle aus der Datei `abc.m` auszuführen, gibt man im MATLAB-Befehlsfenster

```
≫ abc
```

ein. MATLAB sucht im aktuellen Pfad nach der Datei `abc.m` und führt die darin enthaltenen Befehle aus. Das aktuelle Verzeichnis wird angezeigt, wenn man

```
≫ pwd
```

eingibt (pwd, engl. print working directory).

## Hilfe

**Online-Hilfe:** Da sich nicht jeder Benutzer alle MATLAB-Befehle merken kann oder auch von einigen auch nur die Syntax unklar ist, bietet MATLAB die Möglichkeit der Online-Hilfe. Dabei gibt es prinzipiell mehrere Möglichkeiten.
Die einfachste Möglichkeit, sich Hilfe zu verschaffen, besteht darin, das Helpdesk aufzurufen. Wenn Sie `helpdesk` oder `doc` am Prompt eingeben, öffnet sich die Hilfe-Umgebung von MATLAB . Wenn man beispielsweise `doc sqrt` eingibt, erhält man die Hilfe-Umgebung mit einer Dokumentation zu dem Befehl `sqrt`.
Ist einem ein Befehl bekannt und man sucht Informationen über die Syntax, so gibt es alternativ auch den Befehl `help`.

```
≫ help sqrt
  SQRT   Square root.
         SQRT(X) is the square root of the elements of X. Complex
         results are produced if X is not positive.

         See also SQRTM.
```

Als Beispiel haben wir uns hier die Hilfe zu dem Befehl `sqrt` ausgeben lassen.

# Offline 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. | `1:5` | a. | Liefert das aktuelle Verzeichnis. |
| _____ | 2. | MATLAB | b. | Ist eine Interpretersprache. |
| _____ | 3. | `sqrt(3)` | c. | Ist äquivalent zu `[1,2,3,4,5]`. |
| _____ | 4. | `ans` | d. | Programme müssen vor der Ausführung compiliert werden. |
| _____ | 5. | `help cos` | e. | Standardvariablenname für Ergebnisse. |
| _____ | 6. | `pwd` | f. | Liefert die aktuell verwendeten Variablennamen. |
| _____ | 7. | `det` | g. | Erzeugt die Einheitsmatrix der Dimension $7$. |
| _____ | 8. | `eye(7)` | h. | Liefert die Determinante einer Matrix. |
| _____ | 9. | `who` | i. | Berechnet die Wurzel von $3$. |
| _____ | 10. | C/C++ | j. | Liefert die online-Hilfe zu $\cos x$. |

Ihre Antwort:

## Füllen Sie die Lücken aus

Ergänzen Sie die folgenden Sätze.

11. Das Kommandofenster dient der interaktiven Eingabe von MATLAB- _____ .
12. Durch die Entwicklung eigener MATLAB- _____ ist es möglich, die Basisfunktionalität von MATLAB zu erweitern.
13. Mit dem MATLAB-Befehl _____ kann man die MATLAB-Dokumentation aufrufen .
14. Die Eingabe von `edit ode45` am Prompt öffnet die _____ im Texteditor.
15. Im KIZ gibt es für 20 Euro eine _____-Version von MATLAB zu kaufen.
16. Zugriff auf die MATLAB-Dokumentation liefern die Befehle _____ .
17. Variablennamen müssen mit einem _____ beginnen .
18. Der Befehl `length(vek)` liefert die _____ des Vektors `vek` .
19. Vektoren gleicher Dimension können _____ und _____ werden.
20. Matrizen mit der gleichen Dimension können mit dem Befehl `.*` _____ werden.

Ihre Antwort:

**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.

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

    Ihre Antwort:

22. Bei welchen Operatoren kann der Punkt . vorangestellt werden?

    Ihre Antwort:

23. 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 erhält man die Fehlermeldung „`??? Undefined command/function 'peter'`“, wenn peter(5) im Kommando-Fenster eingeben wird?

Ihre Antwort:

## Programmausgaben

Lesen Sie für jedes der folgenden Programmsegmente zuerst die Zeilen und schreiben Sie danach die Ausgabe an die dafür vorgesehene Stelle.

24. Wie lautet die Ausgabe des folgenden Skripts?

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

Ihre Antwort:

25. 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:

26. 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:

27. 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 Semantik- oder Syntaxfehler handelt. Schreiben Sie die korrigierten Anweisungen jeweils in jeden dafür vorgesehenen Bereich unter der Problemstellung. Bemerkung: Es kann sein, dass ein Programm mehrere oder keine Fehler enthält.

28. 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,1)
6           C(i,j) = C(i,j) + A(i,k) * B(k,i);
7        end
8      end
9    end
10 end
```

Ihre Antwort:

29. Das folgende Skript sollte die Funktion $\arctan(x^2)$ auf dem Bildschirm ausgeben:

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

Ihre Antwort:

## Praktikumsaufgabe - Einfacher Funktionsaufruf

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:** Rufen Sie die folgende Funktion im Kommando-Fenster von MATLAB auf.

```
1 % ex01x01.m
2 % simple program using transpose and array power
3
4 function A = ex01x01
5 k = [1,2,3,4];
6 k = k';
7 A = [k.^1,k.^2,k.^3,k.^4];
```

Sie sollten auf dem Bildschirm eine $4 \times 4$-Matrix ausgegeben bekommen mit Einträgen $a_{jk} = j^k$.

Erstellen Sie die Datei `ex01x02.m` und schreiben Sie eine Funktion `ex01x02(n)` mit den folgenden Eigenschaften:

- Die Funktion bekommt die Dimension $n$ als Parameter übergeben.
- Als Rückgabewert erhält man eine $n \times n$-Matrix mit mit Einträgen $a_{jk} = j^k$.

**Vorlage:** Die Datei `ex01x01.m` ist eine lauffähige MATLAB-Funktion, welche in Moodle zum Download zur Verfügung steht.