Tag 1: Einführung in Programmierung und Benutzung von Matlab

Stand: 21.08.2016

Für dieses Skript gibt es drei Versionen, die sich inhaltlich nicht unterscheiden. Die html-Version ist insofern praktisch, als sie links z.B. für den Download von Dateien enthält. Die pdf-Version ist am besten zum Ausdrucken geeignet, in die Word-Version können Sie Notizen direkt einfügen und sich so ein persönliches Skript erstellen. Es ist Geschmackssache, welche Version Sie bevorzugen.

0) Vorbereitung: Anmeldung beim cloudstorage System der Uni Oldenburg

Wir werden uns in diesem Kurs bemühen, Ihnen täglich zu mindestens einer von Ihnen programmierten Aufgabe Rückmeldung zu geben. Um uns Ihre Lösungen zur Verfügung zu stellen, möchte ich Sie darum bitten, sich für diesen Kurs beim **Cloudstorage-System der Uni** anzumelden. Dieser Dienst ist erreichbar unter

https://cloudstorage.uni-oldenburg.de/

weitere Informationen zum Uni-Cloudsystem gibt es unter

http://www.uni-oldenburg.de/itdienste/services/datenhaltung/cloudstorage/

Sie müssen sich dort einmal mit Ihrem Uni-account und Passwort anmelden, damit Ihre email-Adresse für mich verfügbar ist, um meine Ordner mit Ihnen zu teilen.

Sobald ich die Ordner eingerichtet habe, sollten Sie auf Ihrer Cloudseite **zwei neue Ordner** finden:

- **Matlab2016** Der allgemeine Kursordner, den alle KursteilnehmerInnen lesen können. Hier finden Sie für jeden Kurstag Dateien zum Download, sowie jeweils am nächsten Tag Beispiellösungen zu den Übungsaufgaben. In diesem Ordner haben nur die drei Tutorinnen und ich Schreibrecht.
- **Nachname_Vorname_Matlab2016** (z.B. Mueller_Lieschen_Matlab2016) Ihr persönlicher Ordner, auf den NUR Sie, die beiden Tutoren und ich Zugriff haben. Bitte kopieren Sie dorthin Ihre Lösungen. Wir werden die von uns kommentierten Programme ebenfalls dorthin speichern.

Hinweise zur Nutzung des Cloudsystems:

- Sobald Sie sich beim Uni-Cloudsystem anmelden, ist Ihre email-Adresse (wie in Stud.IP) auch für die anderen NutzerInnen des Uni-Cloudsystem einsehbar. Sie können dann z.B. mit anderen Studierenden Ordner teilen und so Daten austauschen. Wenn Sie nicht dauerhaft möchten, dass Ihre email-Adresse sichtbar ist, können Sie sich nach Ende des Kurses wieder von dem Dienst abmelden.
- Meine Tutoren und ich teilen NUR diese beiden Ordner mit Ihnen. Wir können NICHT einsehen, was Sie in anderen Ordnern auf dem Cloudstorage speichern.
- Die Kurs-Ordner werde ich einige Zeit nach Ende des Kurses löschen. Bitte benutzen Sie diese also nicht zur dauerhaften Speicherung von Daten.
- Wenn Sie möchten, können Sie sich auf Ihrem privaten Computer einen client einrichten, um das Cloudsystem zur Datenspeicherung zu nutzen. Für diesen Kurs ist es aber völlig ausreichend, wenn Sie das web-Interface https://cloudstorage.uni-oldenburg.de/ benutzen, um dorthin jeweils Ihre fertigen, vorher lokal gespeicherten Programme zu kopieren.

Bitte:

- Benennen Sie Ihre auf dem cloudstorage System gespeicherten Dateien für uns eindeutig, indem Sie den Dateinamen aus der Aufgabennummer und Ihrem Namen zusammensetzen, z.B. T1H1_Sonja.m (Wir suchen die Lösungen zu einer Aufgabe automatisch mit einem Skript heraus. Anders benannte Lösungen finden wir nicht.)
- Korrigierte / kommentierte Lösungen speichern wir unter geändertem Namen wieder in Ihren persönlichen Ordner.

A) DIE MATLAB-OBERFLÄCHE UND MATLAB ALS TASCHENRECHNER:

Matlab ist eine Programmierumgebung, die im wissenschaftlichen Bereich häufig genutzt wird. Im Rahmen dieses Kurses werden wir nur einen kleinen Teil der Möglichkeiten von Matlab ausschöpfen. Im ersten Schritt wollen wir uns erst einmal mit der Oberfläche von Matlab vertraut machen. (Die folgende Beschreibung bezieht sich auf Matlab2016a auf einem Mac, andere Versionen können geringfügig anders aussehen und man kann sich die Oberfläche nach den eigenen Vorstellungen individuell gestalten.)

Oberfläche:

Das wichtigste Fenster ist das große "Command window". Dort wird durch den Cursor >> angezeigt, dass Matlab bereit ist, Befehle entgegenzunehmen, die hier eingetippt werden können. Wenn Matlab gerade mit der Abarbeitung eines Programms beschäftigt ist, wird kein Cursor dargestellt und in der Leiste ganz unten steht "busy". Das command window hat einige praktische Eigenschaften, um einem das Tippen leichter zu machen:

- Um Befehlszeilen zu wiederholen, die man bereits benutzt hat, kann man im Command window die "Pfeil nach oben"-Taste benutzen. Diese zeigt bei mehrfachem Drücken nacheinander die letzten Zeilen an.
- Tippt man einen zuvor verwendeten Zeilenanfang an und drückt die "Pfeil nach oben"-Taste, werden die letzten mit diesem Anfang aufgetretenen Zeilen ergänzt.
- Das command window kann auch Befehle ergänzen, die vorher in einer Matlab Sitzung noch nicht benutzt wurden. Tippt man einen Wortanfang und drückt dann die "Tab"-Taste, erscheint eine Liste möglicher Ergänzungen.

Unten links ist ein kleineres Fenster mit dem Namen "Workspace". Der Workspace ist die Gesamtheit aller zurzeit definierten Variablen. Diese werden als Liste in dem Fenster angezeigt. Eine Variable ist ein Name, der einen bestimmten Inhalt bezeichnet. Z.B. lässt sich nach der Definition *a=1* mit a weiterarbeiten. Sein Inhalt lässt sich für Berechnungen benutzen (z.B. *b=3*a*) oder ändern (z.B. *a=a+1*).

- Doppelklick auf eine Variable im Workspace Fenster öffnet automatisch den Variable Editor, der den Inhalt der Variable als Tabelle darstellt.

Darüber befindet sich das Fenster "Current Folder"; es zeigt alle Dateien des aktuellen Verzeichnisses an. Dieses kann man mit dem kleinen pull-down-Menü ganz oben ändern. (Das Thema Speichern und Laden behandeln wir morgen.)

Außerdem findet man oben noch zwei Leisten. Sämtliche Funktionalitäten zu erklären würde über diesen Kurs hinausgehen, aber beide Leisten sind stückweise selbsterklärend. Zwei besonders wichtige Funktionen sind

- Das Hilfefenster, durch das Fragezeichen symbolisiert.
- Der Texteditor zum Schreiben und Speichern von Programmen, durch das leere Blatt ganz links symbolisiert.

Hilfe:

Bei der Hilfe handelt es sich um einen html basierten Browser zum Suchen und Darstellen von Matlabbefehlen. Wenn Sie Toolboxen installiert haben (was bei der Uni-Lizenz grundsätzlich der Fall ist), sind diese automatisch ebenfalls in der Hilfe erklärt. Neben Erklärungen zu den einzelnen Funktionen (die man über die Stichwortsuche findet) gibt es auch Tutorials (zum eigenständigen Lernen von Matlab und / oder speziellen Konzepten), Beispiele, Filme, Release notes (Hinweise zu verschiedenen Matlab Versionen) und vieles mehr – durchklicken lohnt sich!

Syntax Grundlagen:

- Variable: Eine Variable wird mit der Syntax *Name=Wert* definiert. Das Gleichheitszeichen heißt also "gesetzt gleich" und die Reihenfolge ist entscheidend (Name links, Wert rechts).
- Dezimalzahlen: Matlab ist ein Amerikanisches Programm. Dezimalstellen werden mit . und nicht wie im Deutschen mit , abgetrennt. z.B. *u=8.765* (NICHT u=8,765).
- Bildschirmausgaben: Normalerweise wird das Ergebnis jeden Rechenschritts auf dem Bildschirm dargestellt. Wenn man diese Bildschirmausgabe unterdrücken will, schreibt man ein Semikolon hinter den Befehl, z.B. *a=75;*
- Lange Zeilen: Wenn eine Programmzeile sehr lang ist, so dass man sie nicht mehr vernünftig lesen kann, dann kann man den Befehl mit *...* unterbrechen und in der nächsten Zeile fortsetzen.

Aufgaben:

T1A1) Starten Sie Matlab und machen Sie sich mit den verschiedenen Fenstern vertraut. Starten Sie die Hilfe. Welche Möglichkeiten bietet sie?

T1A2) Benutzen Sie das Matlab Command Window als Taschenrechner.

- Probieren Sie die Grundrechenarten aus z.B. *7.32*674.986* oder *100086+654.965-83.7* (jeweils mit Return abgeschlossen).
- Wie unterscheidet sich die Antwort auf die Eingabe von *5*10+2* von *5*(10+2)* und *(5*10)+2*?
- Achtung: an die amerikanischen Schreibweise mit Punkt statt Komma denken!

T1A3) Wenn Sie einige Rechnungen durchgeführt haben, tippen Sie *ans* ein.

- Wie reagiert Matlab? Warum?

- Wie reagiert Matlab auf *ans*2*?
- Was passiert, wenn Sie mehrfach hintereinander *ans*2* wiederholen? (Dazu können Sie die Pfeil-nach-oben-Taste verwenden.)

T1A4) Führen Sie mit *a=4* eine Variable a ein.

- Beobachten Sie die Reaktionen in den verschiedenen Fenstern.
- Jetzt tippen Sie *b=10;* ein. Wie unterscheiden sich die Ausgaben?
- Wie reagiert Matlab auf die Eingabe *a+b*?
- Wie unterscheiden sich davon die Reaktionen auf *c=a+b;* ?

T1A5) Variablen können vom Benutzer beliebig eingeführt und mit Werten belegt werden. Es gibt aber auch Konstanten, die in Matlab vordefiniert sind, beispielsweise *pi*. Lassen Sie sich den Wert ausgeben.

T1A6) Außer den Grundrechenarten kennt Matlab natürlich auch komplexere mathematische Operationen. Testen Sie *2^3* und *b^4*. Was bedeutet dieser Haken?

T1A7) Ein weiteres Beispiel ist die Quadratwurzel - da es für diese aber kein Symbol auf der Tastatur gibt, ruft man sie in Form der Funktion *sqrt* auf. Dabei muss das Argument, die Zahl, auf die die Funktion angewandt werden soll, in Klammern hinter den Funktionsnamen geschrieben werden:*sqrt(16)* oder *d=sqrt(a)*. Weitere Funktionen sind z.B. *sin, cos, exp, log, abs*. Was könnten diese Namen bedeuten? Probieren Sie Ihre Vermutung mit Beispielen aus.

T1A8) Um nachzusehen, ob Sie richtig geraten haben, schauen Sie die Funktionsnamen in der Hilfe nach.

- Eine ganz kurze Beschreibung bekommen Sie, wenn Sie z.B. *help sin* eintippen.
- Sehr viel mehr Information gibt es im Hilfefenster: suchen Sie nach *sin* und den anderen Funktionsnamen.

T1A9) Manche mathematischen Operationen brauchen mehrere Argumente. Z.B. bei *b/a* kommt uns das ganz natürlich vor. Wenn es nicht wie bei den Grundrechenarten eine abkürzende Schreibweise gibt, werden Matlab-Funktionen alle Argumente durch Komma getrennt in der Klammer übergeben, die dem Funktionsnamen folgt. Z.B. wird der bei der ganzzahligen Division übrigbleibende Rest von der Funktion*rem* (remainder) berechnet, die zwei Argumente übergeben bekommt, z.B. *rem(104, 10)*. Dabei ist die Reihenfolge der Argumente wichtig: was passiert, wenn man *rem(10, 104)* eingibt?

T1A10) Vielleicht haben Sie sich während der Übung bereits irgendwann vertippt. Was ist dann passiert? Falls Ihnen noch nichts dergleichen passiert ist, tippen Sie einmal ein *e1=sin(2*pi)*, dann*e2=sine(2*pi)* und schließlich *e3=sin(2pi)*. Wie reagiert Matlab darauf jeweils? Wie unterscheidet sich die Reaktion, wenn Sie *f=10/0* eintippen?

T1A11) Zwar sind heute Ihre Programme noch sehr kurz, das wird sich aber bald ändern. Deshalb beginnen wir heute schon einmal damit, Programme (also Abfolgen von Befehlen) abzuspeichern. (Den theoretischen Hintergrund dazu besprechen wir morgen):

- Bevor Sie anfangen, Dateien abzuspeichern, definieren Sie einen Ort, wo Sie im Weiteren alle Dateien des Kurses hinsortieren. Erzeugen Sie zunächst in Ihrem "Eigene Dateien" Verzeichnis einen neuen Ordner "matlabkurs" (oder welcher Name Ihnen gefällt). Wechseln Sie in Matlab in dieses Verzeichnis, indem Sie auf das icon ... neben der Anzeige "*Current Folder*" klicken.
- Achten Sie in Zukunft jedes Mal, wenn Sie Matlab neu starten, darauf, dass Sie sich im richtigen Verzeichnis befinden.
- Öffnen Sie den Matlab-Editor, indem Sie auf das Symbol für ein neues Dokument klicken. Und tippen Sie dort ein:
- *a=5.5*
  b=0.75
  *c=a*b*
- Speichern Sie dieses Programm als *test1.m* ab (unter *File->save as*). Achten Sie darauf, dass sie im richtigen Verzeichnis sind. Rufen Sie im Command Window Ihr erstes Programm auf indem Sie *test1* eintippen.
- Wenn Sie möchten, können Sie schon heute Ihre Programme als Gedankenstütze abspeichern. Dabei muss die Datei jeweils die Endung *.m* haben, zum Aufruf des Programms im Kommandofenster wird diese Endung weggelassen und nur der restliche Dateiname verwendet. Ab den heutigen Hausaufgaben sollten Sie ALLE Aufgaben mit Hilfe abgespeicherter Skripte und Funktionen lösen.

B) VEKTOREN UND MATRIZEN:

Vektoren und Matrizen sind das wichtigste Konzept der Programmierung in Matlab. Nahezu alle Daten werden in Matrizen organisiert und verarbeitet. Wenn Sie nicht sowieso mit Matrizenrechnung vertraut sind, lesen Sie bitte vor dem Kurs die kurze Einführung aus dem Buch "Matlab und Mathematik kompetent Einsetzen" von S. Adam (Download in Stud.IP und im Cloudstorage).
Matrizen sind Verbundvariablen, die zusammengehörige Daten in einem Objekt vereinigen, indem sie diese in einem Rechteckschema aus Zeilen und Spalten anordnen. Vektoren sind spezielle Matrizen, die entweder nur aus einer Zeile oder nur aus einer Spalte bestehen (Zeilenvektor bzw Spaltenvektor). Die einzelnen Einträge in Vektoren und Matrizen bezeichnet man als Elemente der Matrix oder des Vektors.

Einige Rechenregeln zum Umgang mit Vektoren und Matrizen:

- Addition und Subtraktion funktioniert grundsätzlich nur für Vektoren und Matrizen gleicher Größe (also mit gleich vielen Zeilen und Spalten). Man addiert bzw subtrahiert jeweils die Elemente mit gleichen Indizes.
- Bei der Multiplikation eines skalaren Werts mit einer Matrix (bzw einem Vektor) wird jedes Element einzeln mit dem Skalar multipliziert.

- Die Multiplikation von Matrizen ist mathematisch etwas komplizierter definiert (bitte schauen Sie sich dazu den Text zur Matrizenrechnung in Stud.IP an), wird aber in der Biologie nicht sehr oft gebraucht. Eine größere Rolle spielt die punktweise Multiplikation, bei der jeweils die Elemente von Matrizen (oder Vektoren) gleicher Größe miteinander multipliziert werden, die gleiche Indizes besitzen.
- Beim Transponieren werden Zeilenindizes und Spaltenindizes miteinander vertauscht. Dadurch wird aus einem Spaltenvektor ein Zeilenvektor und umgekehrt. Eine Matrix wird dadurch an der Hauptdiagonalen gespiegelt.
- Der Euklidische Abstand von Punkten A und B im Raum ist definiert als:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \quad \text{wobei} \quad A = (a_1, \ldots, a_n) \in \mathbb{R}^n \quad \text{und} \quad B = (b_1, \ldots, b_n) \in \mathbb{R}^n$$

z.B. im dreidimensionalen Raum für A=(a_1,a_2,a_3) und B=(b_1,b_2,b_3):

$$d_{AB} = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2} \quad \text{im } \mathbb{R}^3$$

Matlab Syntax:

- Die Werte der Elemente von Vektoren oder Matrizen werden in Matlab mit eckigen Klammern umschlossen angegeben, z.B. *v=[1 2 3]*.
- Innerhalb der eckigen Klammern werden Elemente getrennt:
  - innerhalb einer Zeile werden Elemente durch Leerzeichen oder Komma voneinander getrennt, z.B. *v=[1 2 3]* ist gleichbedeutend mit *v=[1, 2, 3]*.
  - Spalten werden durch Semikolon getrennt, z.B. *s=[7;0.4;-3]*. Eine Matrix ist definiert z.B. durch *M=[1 0; -3 7]*.
- Indizes werden in runden Klammern angegeben, z.B.
  - *v(2)* ist das zweite Element des Vektors *v*.
  - *s(1)=5* weist dem ersten Element von *s* den Wert *5* zu.
  - *k=s(1)* weist der Variable *k* den Wert des ersten Elements von *s* zu.
- Für Matrizen werden die Indizes innerhalb der runden Klammer durch Komma getrennt, z.B.
  - *M(2,1)=77* weist dem Element in der zweiten Zeile und ersten Spalte der Matrix *M* den Wert *77*zu.
  - *neu=M(2,2)* überträgt den Wert des Elementes in der zweiten Zeile und zweiten Spalte der Matrix *M* auf die Variable *neu*.
- Der Doppelpunkt dient in Matlab dazu, Zahlenreihen zu erzeugen. Das ist insbesondere beim Zählen und beim Indizieren von großer Bedeutung.

Aufgaben:

T1B1) Rechnen Sie zunächst von Hand: *a=[1; 2]* (Spaltenvektor), *b=[1 2]* (Zeilenvektor). Was ist*c=a*b*? Multiplizieren Sie das Ergebnis *c* mit der Matrix *d=[0 1; 0.1 10]*. Überprüfen Sie Ihre Ergebnisse anschließend mit Matlab.

T1B2) Definieren Sie folgende Vektoren: *a=[1 2 3]; b=[0.1 0.2 0.3]; c=[10; 20; 30]; d=[100; 200]; e=[-1; -2]* und *f=7.5*. Welche Vektoren sind Spalten- und welche Zeilenvektoren?
Überlegen Sie und probieren Sie aus: Welche der Vektoren kann man addieren? Subtrahieren? Multiplizieren? Punktweise multiplizieren?
In welchen Fällen ist die Reihenfolge der Vektoren wichtig, in welchen nicht?
T1B3) Wenn Sie sich in den Kopf setzen, dass Sie gerne einen Vektor *g* definieren wollen, dessen Werte die Summe aus den Elementen von *a* und *c* sind, müssen Sie etwas tricksen: Einer der Vektoren muss transponiert werden. Probieren Sie aus: *g1=a+c'* und *g2=a'+c*. Was ist der Unterschied?
T1B4) Für den Umgang mit Vektoren werden von Matlab ein Paar sehr praktische Funktionen bereitgehalten. Probieren Sie an einigen Beispielen die Arbeitsweise folgender Befehle aus und sehen Sie im Zweifelsfall in der Hilfe nach:
sum(vektor), mean(vektor)
*) Probieren Sie aus, welche der in A7) eingeführten Funktionen auch für Vektoren funktionieren.
T1B5) Definieren Sie zwei Matrizen: *M1=[1 1 1; 5 6 7]* und *M2=[0 -1; -2 0.5; 0.1 1]* und schauen Sie im Array Editor an. Welche Indizes hat der Eintrag *6* in *M1*? Greifen Sie einen Eintrag aus *M2* heraus, z.B. *M2(1,2)*.
T1B6) Achtung! Matlab kennt zwei Arten, eine Matrixkomponente zu indizieren:

- entweder, man benutzt die Schreibweise *M1(1,2)* mit *(zeile, spalte)* in der Klammer,
- oder man zählt die Matrix als einen langen Vektor durch, wobei die Spalten zu einem langen Spaltenvektor aneinandergehängt werden, z.B. *M1(2)*.
- Welcher Einzelindex entspricht *M1(1,2)*?

T1B7) Berechnen Sie *M1*M2* und *M2*M1*. Transponieren Sie *M2* und berechnen Sie die Summe mit *M1*.
* T1B8) Welche der Vektoren aus Aufgabe B2 kann man mit *M1* oder *M2* multiplizieren? Kann man sie auch addieren?
T1B9) Zum Glück braucht man in Matlab große Matrizen nicht komplett einzutippen. Probieren Sie aus, was passiert, wenn man *o=ones(1,8)* und *z=zeros(3,3)* eingibt.
T1B10) Eckige Klammern umschließen jeweils alle Komponenten einer Matrix oder eines Vektors. Man kann sie deshalb auch nutzen, um Vektoren oder Matrizen zu kombinieren. Überlegen Sie, was folgende Ausdrücke basierend auf den oben definierten Vektoren und Matrizen liefern sollten, und probieren sie Ihre Hypothesen aus:
t1=[a b], t2=[a;b], t3=[a b f], t4=[d e], t5=[d;e], t6=[a c],
t7=[M1; a], t8=[M1; M2'], t9=[M1' M2]; t10=[[pi pi/2];[1 2];e']

T1B11) Der Doppelpunktoperator spielt in Matlab eine wichtige Rolle. In seiner einfachsten Form benutzt man ihn zum zählen: *zahlen=1:10* erzeugt einen Vektor mit den Zahlen von 1 bis 10.

T1B12) "Zählen" kann man nicht nur in Schritten der Länge 1. Die Schrittweite wird zwischen Anfangs und Endwert in Doppelpunkte eingeschlossen angegeben.

- Probieren Sie aus: *zahlen2= 10:10:100, zahlen3=0:0.1:0.5*.
- Erzeugen Sie einen Vektor, der vom Startwert 5 mit Schrittlänge von 2.5 bis zum Endwert 20 läuft.

T1B13) Besonders wichtig ist der Doppelpunktoperator beim Indizieren, also um auf definierte Elemente von Vektoren oder Matrizen zuzugreifen.

- Welche Werte ergibt *zahlen(2:4)*?
- Was könnte man alternativ schreiben, um an diese Werte heranzukommen?
- Greifen sie das 4. bis 7. Element aus dem Vektor *zahlen2* heraus.

T1B14) Wenn der Doppelpunktoperator beim Indizieren alleine steht, z.B. *zahlen(:)*, bedeutet es, dass der ganze Vektor genommen werden soll. Für Vektoren ist das recht langweilig, denn in diesem Fall könnte man auch einfach *zahlen* schreiben.
Anders ist das bei Matrizen.

- Definieren Sie *M3=[1 1 1 1; 5 6 7 8; 10 20 30 40]*.
- Probieren Sie aus: *M3(1,:), M3(:,3)*.
- Was bedeutet *M3(:)* im Gegensatz zu *M3* und zu *M3(:,:)*?

T1B15) Und jetzt kombinieren wir das Gelernte: Greifen Sie aus *M3* den dritten und vierten Eintrag der zweiten Zeile heraus.

T1B16) Eine sehr praktische Vereinfachung beim Indizieren ist der Begriff *end*. Beispielsweise der Ausdruck *a(7:end)* bezeichnet alle vom siebten bis zum letzten Eintrag des Vektors *a*. Dadurch braucht man die Länge des Vektors nicht zu kennen. Greifen sie die letzten drei Elemente aus dem Vektor*zahlen* heraus, indem Sie *end* benutzen.

T1B17) Noch sind Ihre Variablen so klein, dass Sie diese im Workspace-Fenster komplett angezeigt bekommen. Das wird sich aber bald ändern. Um sich die Werte von Variablen ansehen zu können, gibt es den Variable Editor. Er öffnet sich automatisch, wenn man auf eine der Variablen im Workspace Fenster doppelt klickt. Dieser Editor zeigt einem die Werte der Variable in einer Tabelle an. Man kann sie hier auch durch Eintippen verändern - damit sollte man allerdings sehr vorsichtig umgehen, denn diese Aktionen sind später nicht mehr nachvollziehbar und insbesondere nicht automatisch zu wiederholen.

T1B18) Für viele Anwendungen muss man die Größe von in Variablen gespeicherten Matrizen oder Vektoren kennen.

- Für einen Vektor *v* ist *s=length(v)* die Länge des Vektors.
- Bei Matrizen bietet es sich an, den Befehl *size* zu verwenden:
- *zeilen=size(M3,1);* % speichert die Anzahl der Zeilen (also die Länge der 1. Dimension) in der Variable *zeilen*

- *spalten=size(M3,2);* % speichert die Anzahl der Spalten (also die Länge der 2. Dimension) in der Variable *spalten*
- *groesse=size(M3);* % erzeugt einen Vektor *groesse*, dessen erster Wert die Zeilen- und dessen zweiter Wert die Spaltenanzahl ist.

C) GRAFISCHE DARSTELLUNG VON VEKTOREN:

Die grafische Darstellung von Daten ist für viele Anwendungen höchst wichtig, gerade im wissenschaftlichen Bereich. Daten, die man einmal "gesehen" hat, kann man sich besser vorstellen, als wenn sie in Form von Vektoren vorliegen. Deshalb werden z.B. in Praktikumsprotokollen, Abschlussarbeiten etc. grundsätzlich grafische Darstellungen der Auswertungsergebnisse gefordert. Mit Matlab hat man die Möglichkeit, sich in Vektoren oder Matrizen angeordnete Daten sehr einfach grafisch darstellen zu lassen. Der Grundbefehl dafür lautet *plot*.

T1C1) Jetzt stellen wir einen Vektor grafisch dar: Definieren Sie den Vektor *v=[1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 0 0 0 0 0 1.5 1.5 1.5 1.5 1.5]*. Lassen Sie sich diese Daten mit *plot(v)* grafisch darstellen. Was ist auf der x- und der y-Achse dargestellt?

T1C2) Erzeugen Sie mit Hilfe des Doppelpunktoperators einen Vektor mit einer Zahlenreihe, z.B.*v=0.01:0.01:10;* und stellen diesen mit *plot(v)* grafisch dar. Probieren Sie verschiedene Zahlenreihen aus. Was passiert mit den Achsen?

*T1C3) Erzeugen Sie eine Matrix *M*, die aus mehreren verschiedenen Zahlenreihen besteht und stellen diese mit *plot(M)* dar.

- Was ist zu sehen?
- Wie kann man eine einzelne Zeile der Matrix grafisch darstellen?
- Wie eine Spalte?

T1C4) Bisher wurden die Werte jeweils gegen den index der Vektoren bzw Matrix aufgetragen. Das ist nicht immer sinnvoll.

- Generieren Sie zwei Vektoren *x=-1:0.1:1;* und *y=x.^2;*
- mit *plot(x,y)* werden die Werte gegeneinander aufgetragen.
- Was passiert bei *plot(y,x)*?
- Erzeugen Sie Parabeln für verschiedene Vektoren *x*.

T1C5) Wie trägt man einen einzelnen Punkt in eine Graphik ein? Plotten Sie den Punkt *P=[4.5 3]*.

T1C6) Um sich nur einen Ausschnitt der vorhandenen Daten anzusehen oder um Daten in Abbildungen zu skalieren (z.B. damit die Darstellung eines einzelnen Punktes sinnvoll wird) gibt es den Befehl *axis([xmin xmax ymin ymax])*.

- Probieren Sie verschiedene Achsenskalierungen für Ihre Kurven aus.
- Erzeugen Sie außerdem eine Parabel mit einem größeren Definitionsbereich, z.B. von -10 bis 10, plotten Sie diese und sehen sie sich mit verschiedenen Skalierungen an.

- Wie bekommt man einen plot der genauso aussieht wie der ursprüngliche in T1C4?

T1C7) Es gibt eine Reihe von Optionen, um Daten möglichst schön oder praktisch aufzutragen.

- Probieren Sie *plot(y,x,'r')* aus.
- Was bewirken die folgenden Optionen: *r, b, g, k, c, m, *, ^, s, d, :, -, --*
- Notieren Sie sich die jeweilige Bedeutung und probieren Sie sinnvolle Kombinationen aus, z.B.*plot(y,x,'r^:')*

T1C8) Es gibt mehrere Möglichkeiten, wie man mehr als einen Vektor gleichzeitig graphisch auftragen kann. Definieren Sie zwei Vektoren *v1=-3:6* und *v2=[5 5 5 5 0 0 0 4 4 4]* und einen x-Vektor*x=1:10*, gegen den Sie die Vektoren auftragen. Probieren Sie aus (sie können die Zeilen abtippen bzw in das Command window kopieren, oder sie kopieren alles zusammen in den Editor und speichern es als Programm ab):

```
% Darstellung jeweils eines Vektors nacheinander:
plot(x,v1)
hold on
plot(x,v2)
hold off
% Oeffnen eines neuen Graphikfensters
figure
% Darstellung beider Vektoren gleichzeitig:
plot(x,v1,x,v2)
```

Gibt es Unterschiede zwischen den beiden Abbildungen?

D) HAUSAUFGABEN:

Die Aufgaben mit den roten Nummern werden jeweils am nächsten Tag besprochen bzw. ab dem 3. Tag zum Punktesammeln verwendet. Wir kommentieren Lösungen zu Aufgaben mit roten Nummern, Aufgaben mit * und Aufgaben mit **.

- **Bitte** speichern Sie mir Ihre Lösung in Ihren Ordner auf dem cloudstorage System.
- **Bitte** wählen Sie einen Dateinamen, der zunächst die Aufgabennummer und dann Ihren eigenen Namen beinhaltet (z.B. T1H1_Sonja.m). Wir suchen die Lösungen automatisch mit einem Skript – das findet Ihre Lösung nur, wenn die Aufgabennummer im Namen enthalten ist. Ihr eigener Name muss darin stehen, damit wir Ihnen nach der Korrektur das File wieder zuordnen können. Ansonsten würde es auch passieren, dass verschiedene Leute gleiche Namen wählen und wir die Dateien beim Download gegenseitig überschreiben.
- **Bitte** sortieren Sie Ihre Dateien in Ihrem Cloud-Ordner sinnvoll, gerne nach Kurstagen, damit wir die jeweiligen Aufgaben mit möglichst wenig Aufwand finden.

T1H1) Der Mathematiker Karl Friedrich Gauss sollte als Grundschüler damit beschäftigt werden, alle Zahlen von 1 bis 100 zusammenzuzählen. Er war innerhalb weniger Minuten damit fertig, da er das Prinzip erkannte, dass jede Summe einer Zahl aus der oberen Hälfe der Reihe mit einer passenden Zahl aus der unteren Hälfte den Wert 101 ergab (z.B. 1+100=101, 2+99=101 etc). Da es 50 solcher Paare gibt, ist die Lösung 5050=101*50.

- Die allgemeine Formel für die Summe einer Reihe natürlicher Zahlen von a bis b lautet s=1/2*(a+b)*(b-a+1).
- Testen Sie diese Formel, indem sie verschiedene Reihen mit dem Befehl r=a:b erzeugen und deren Summe mit *sum(r)*, sowie mit der obigen Formel berechnen.

T1H2) Erzeugen Sie eine Matrix mit einer Einmaleins-Tabelle für das kleine Einmaleins, also

1	2	3	...						10
2	4	6	...						
3	6	9	...						
4	8	12	...						
..	...	...							
10	20	30							100

(natürlich vollständig ausgefüllt).

T1H3) Verwenden Sie das Prinzip, Matrizen als Bestandteile von Matrizen zu definieren, um ein 8*8 Schachbrett aus 0 und 1 zu erzeugen.

- Sehen Sie sich Ihr Resultat im Array Editor an.
- Man kann sich den Inhalt einer Matrix auch grafisch anzeigen lassen. Wenn Ihre Matrix-Variable "schach" heißt, tippen Sie einfach ein *imagesc(schach)*.

T1H4) Durch welche Abfolge von Matlab-Befehlen erzeugt man den Inhalt (ohne Beschriftung) der folgenden Tabelle der Höhendifferenzen:

km	Duforspitze	Matterhorn	Gornergrat	Zermatt	Brig
Duforspitze	0	-156	-814	-3014	-3956
Matterhorn	156	0	-658	-2858	-3800
Gornergrat	814	658	0	-2200	-3142
Zermatt	3014	2858	2200	0	-942
Brig	3956	3800	3142	942	0

aus dem Vektor der Höhenwerte *v=[4634; 4478; 3820; 1620; 678]* ?

*T1H5) Gegeben sind die beiden Punkte *A=[1,5]* und *B=[3,1]*. Stellen Sie diese in einem Koordinatensystem, bei dem beide Achsen den Bereich von 0 bis 10 umfassen, mit verschiedenen Symbolen dar. Zeichnen Sie außerdem die Strecke zwischen den beiden Punkten mit einer anderen Farbe ein.

*T1H6) Wer Grafik noch schöner haben möchte als es mit den in T1_C7 erwähnten Grafikoptionen möglich ist, kann dafür sogenannte "line properties" nutzen. Diese werden der Plotfunktion jeweils als Schlüsselbegriff und Wert übergeben, z.B. *plot(x,y,'LineWidth',5)*.

- Weitere line properties sind *'Color', 'MarkerFaceColor', 'MarkerEdgeColor'*, und*'MarkerSize'*. Was bedeuten diese?
- Stellen Sie Ihre Kurve mit einer dicken roten Linie und blauen Dreiecken als Datenpunkten dar.

*T1H7) Spielen Sie ein wenig mit den interaktiven Möglichkeiten, Ihre Grafik zu verändern, indem Sie die Klick-Tools in der Leiste oben verwenden: zoomen sie auf einen gewissen Bereich, ändern Sie die Liniendicke, geben Sie der Grafik einen Titel etc. Welche Möglichkeiten gibt es, die Grafik interaktiv zu verschönern?

*T1H8) Um sich weiter mit dem Umgang mit Vektoren und Matrizen in Matlab vertraut zu machen, schauen Sie sich die folgenden beiden (englisch kommentierten) Skripte aus einem anderen Kurs an:

vectors_and_matrices.m

matrix_mathematics.m