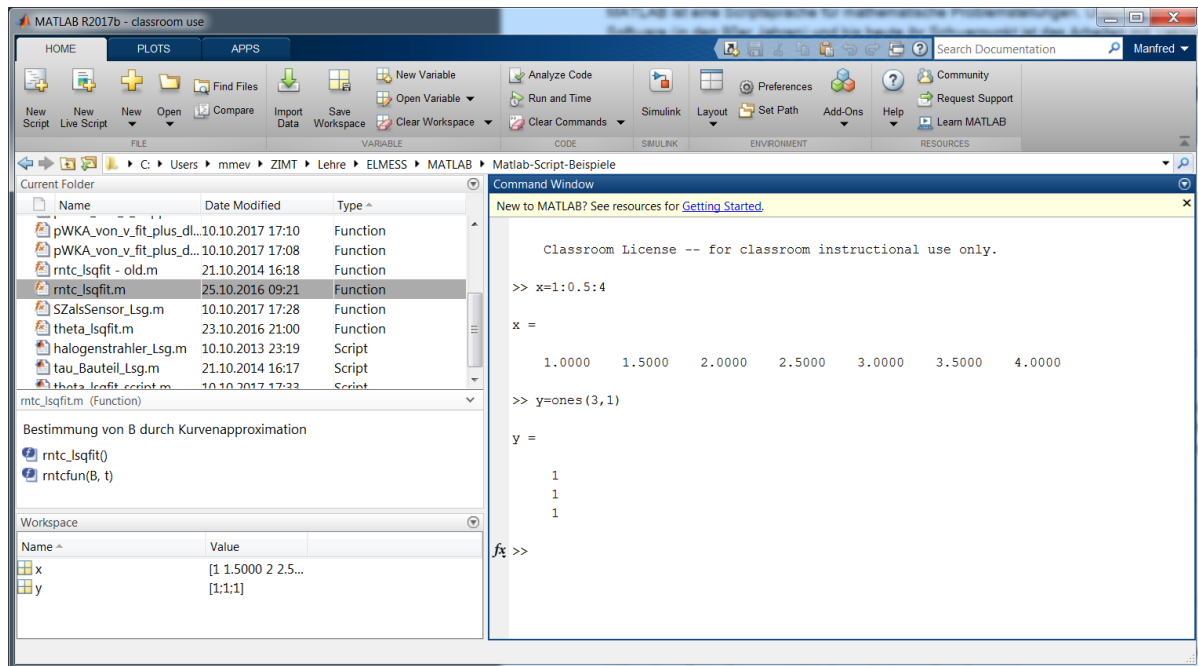


ELMESS - Laborübung "MATLAB-Einführung"

Prof. Dr.-Ing. Manfred Mevenkamp

1 Matlab-Command-Window, Scripts und Functions

MATLAB ist eine Scriptsprache für mathematische Problemstellungen. Ursprung der Software (in den 80er Jahren) und bis heute ihr Schwerpunkt ist das Arbeiten mit Vektoren und Matrizen bei mess- und regelungstechnischen Aufgaben.



Beim Start öffnet sich die Oberfläche mit mehreren Teilfenstern, in der Mitte das "Command Window", in dem Anweisungen in der MATLAB-Scriptsprache eingegeben und ausgeführt werden. Links ein Dateiverzeichnis des aktuellen Ordners mit einem Menu für Verzeichniswechsel etc.

Neben dem direkten Ausführen im Command Window können MATLAB-Scripts auch in Dateien mit der Endung ".m" angelegt werden, entweder

- als M-Scripts, wo die Kommandos so ausgeführt werden, als würden sie im Command Window eingetippt, oder
- als Functions, die ihren eigenen "Workspace" haben und nur über Eingabe- und Ausgabeargumente mit dem aufrufenden Workspace Daten austauschen.


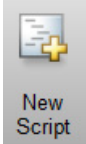
Zum grundlegenden Umgang mit MATLAB gibt es Demonstrationen und Beispiele in der Hilfe, die z. B. im Command Window mit `doc` aufgerufen wird.

Ziel dieser Einführung ist es, speziell mit den Konstrukten vertraut zu werden, die im Zusammenhang mit dem Messtechnik-Labor anfallen. Das sind insbesondere Matrix-Operationen, Cell-Arrays und Grafikroutinen.

In der ELMESS-AULIS-Gruppe finden Sie auch einen Ordner mit etlichen Beispiel-M-Files zur Messdatenanalyse mit MATLAB.

1.1.1 Matlab-Grundlagen

Zu erstellen ist ein M-Script, in dem die unten stehenden Aufgaben abgearbeitet werden.


Benutzen Sie  ("Browse for Folder") um Ihr Arbeitsverzeichnis zu wählen (erstellen Sie dafür z. B. einen neuen Ordner "MATLAB" auf Ihrem Desktop). Nach Klick auf  öffnet sich der MATLAB-Editor.

Tragen Sie in den ersten Zeilen als Kommentar (beginnt mit "%") Name und Zweck des Scripts, Ihren Namen und das Erstelldatum ein. Also etwa

```
% SCRIPTNAME
% Einführungsübung MATLAB
% Klara Fall, dd.mm.yyyy.
```

Speichern Sie die Datei unter dem gewählten Namen (z. B. `myintro.m`).

Beantworten Sie auch die Zwischenfragen weiter unten in Form von Kommentaren!

Führen Sie das M-Script nach jedem Eingabeschritt aus, um die jeweiligen Ergebnisse zu sehen (z. B. mit dem Save&Run-Button ).

A1) Erzeugen Sie einen Spaltenvektor von Einsen mittels `a = ones(3,1)`.

Nutzen Sie die Hilfe (`doc ones`) und finden Sie die einfachste Möglichkeit eine 3x3-Matrix von Einsen in der Variablen `y` zu erzeugen.

A2) Finden Sie dort unter "**See Also**" das Matlab-Kommando zur Erzeugung einer Einheitsmatrix und erstellen Sie die 4x4-Einheitsmatrix in der Variablen `I`.

Woher hat wohl diese Funktion diesen merkwürdigen Namen?

A3) Geben Sie `b = a'` ein und stellen Sie den Unterschied zwischen `b` und `a` fest. Wie nennt man diese Operation in der Mathematik?

A4) Geben Sie ein: 1.) `c = [10, 20]` , 2.) `c = [10, 20]'` und 3.) `c = [10; 20];` und beschreiben Sie die Unterschiede zwischen `,` und `;` innerhalb einer Matrix und am Zeilenende!

Zum Ansehen der aktuell definierten Variablen dienen das Teilfenster "Workspace" und die Kommandos `who` bzw. `whos`.

Löschen von Variablen (Speicherplatz freigeben) geschieht mit `clear`.

A5) Sehen Sie sich das Resultat von `whos`, `clear a`, `whos` an!

1.1.2 Datentypen, Array

A6) Überlegen Sie bei im Folgenden jeweils **VORHER**, was Sie als Ergebnis erwarten! Z. B.: Wird `t` ein Spalten- oder Zeilenvektor und wie viele Elemente wird er haben?

```
t = (0:0.5:10) '
x1 = t + t(end:-1:1)
```

```
x1 = x1 - 8*ones(size(t)) - rand(size(t));
x2 = cumsum(x1/20)
x3 = sin(2*pi*0.2*t);
x4 = t/10 .* x2;      (ggf. → doc punct )
```

A7) Was sind die Unterschiede zwischen 1.) $x4a = t \cdot x2$ 2.) $x4b = t * x2$, 3.) $x4c = t' * x2$ und 4.) $x4d = t * x2'$?

Welche Variante liefert eine Fehlermeldung? Was besagt der Fehlertext?

A8) Untersuchen Sie mit den folgenden Eingaben "NaN" (Wofür steht das?), das Zusammensetzen von Strings, Zahl in einen String einsetzen, Zeilensprung in langen Kommandos ("..."):

```
x5 = sin(t)./t;
ix5 = find(isnan(x5));
infostring = ['NaN in x5 an der Stelle i = ', ...
              num2str(ix5), ' wg. Division durch 0!']
disp(infostring)
```

1.1.3 Zeitverläufe grafisch darstellen: **plot**

Messdatenreihen liegen normalerweise als Zeitreihen in zwei Varianten vor:

- Zeitachse und Verlauf jeder aufgenommenen Größe in einzelnen Vektoren,
- Array, in dem die erste Spalte die Zeitachse ist und die Messgrößen in den weiteren Spalten stehen,

MATLAB erzeugt mit wenigen Kommandos ansprechende Grafiken.

Grafik öffnen, skalieren und positionieren:

```
hf=figure(1);
set(gcf,'units','normalized','position',[0.1,0.05,0.45,0.85])
```

1.) Zeitachse und einzelne zu plottende Vektoren

```
subplot(3,1,1) % bildet eine Grafik mit 3 Diagrammen untereinander
plot(t,x1,t,x2,t,x3)
```

A9) Man kann diesen Aufruf vereinfachen, so dass t nur einmal und nicht mehrmals, also nicht für jede Kurve separat, angegeben werden muss. Wie (-> doc plot)? Ausprobieren!

2.) Array mit Zeitachse

```
array_tx = [t,x2,x3,x4,x5];
subplot(3,1,2), plot(array_tx(:,1), array_tx(:,2:end))
```

Anmerkung: Zur **Übernahme einer Grafik in die Textverarbeitung** speichern Sie die Grafik in einem Format Ihrer Wahl ("Speichern unter") oder nutzen Sie den Menüpunkt Edit → Copy Figure (nur unter Windows).

1.1.4 Indizieren von Arrays

Die besondere Eignung von MATLAB für die Mess- und Simulationsdatenverarbeitung liegt u. a. in der Flexibilität, mit der auf Einzelelemente oder Teilfelder aus größeren Datenarrays zugegriffen werden kann. Schon in **A6** und **A9** wurde damit gearbeitet.

A10) Auch hier: überlegen Sie bei den Eingaben vorher, was Sie als Ergebnis erwarten!

```
t(1:5:end)
length(x5), x5(1) = [], length(x5)
teilX = array_tx(11:14,1:2)
teilX(:)
z = norm(array_tx(:,3)-x3)
```

Bei "norm" geht es um Größe/Betrag/Länge von Vektoren. Wird **sehr** häufig benötigt! Z. B. beim Vergleich, ob Vektoren (Ergebnisse) im Rahmen der Rechnergenauigkeit übereinstimmen oder bei der Berechnung von Optimierungskriterien.

A11) Berechnen Sie z ohne `norm()` mit Hilfe eines Skalarprodukts!

Anmerkung: Wenn es um das Summieren von Elementen geht, braucht man in Matlab **nie for-Schleifen!**

1.1.5 Ablaufstrukturen for, while, if

Von den in MATLAB verfügbaren "Program Control Statements" wird neben `if/else` gelegentlich die `for`-Schleife benötigt.

Als Beispiel einer `for`-Schleife hier eine (nicht empfehlenswerte!!) Alternative zur Berechnung des Effektivwertes einer Spannung $u(t)$, von der ein Vektor von Abtastwerten über eine Periode vorliegt.

```
messdatei = [(0:0.001:0.02)', 2*cos(2*pi*50*(0:0.001:0.02)')];
t = messdatei(:,1); % In der ersten Spalte stehen die Zeitpunkte
u = messdatei(:,2); % In der zweiten Spalte stehen die Messwerte
nu = length(u); ueff = 0;
for i=1:nu
    ueff = ueff + u(i)*u(i);
end
ueff = sqrt(ueff/nu)
disp(['Effektivwert = ', sprintf('%2.3f',ueff)])
```

Aber noch einmal: Wenn Sie meinen, Sie bräuchten für eine Berechnung eine `for`-Schleife - in 90% der Fälle stimmt es nicht! MATLAB bietet andere, wesentlich schnellere und elegantere Alternativen. Hier z. B.

```
ueff = sqrt(u'*u/nu)
```

A12) Beide Varianten programmieren und vergleichen.

Wie berechnen Sie dasselbe mit Hilfe von `norm`?

Welcher Wert war bei dem hier gegebenen Messsignal zu erwarten?