Einführung in GNU Octave

Egmont Schreiter, eschreiter@hs-zigr.de, https://www.hszg.de/?id=5364

11. Oktober 2016

Beispieldateien sind **in** der PDF-Datei als Anhang gespeichert!

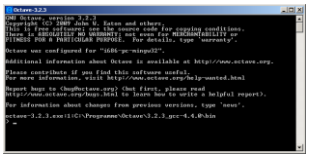
Inhalt

Installation

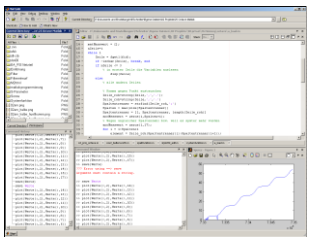
Quelle für Download:
https://www.gnu.org/software/octave/
(http://wiki.octave.org/Octave_for_Windows Portable-Version, dort im /bin Verzeichnis octave bzw. octave-gui starten)

Screenshot

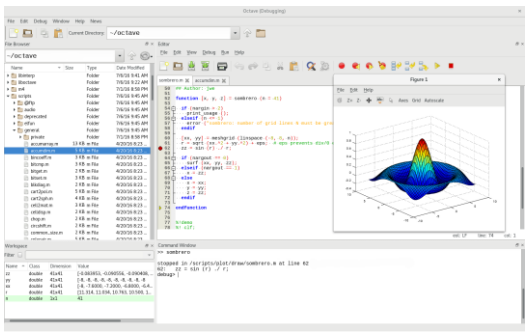
Octave Konsole:



Matlab:



GNU Octave GUI



Tutorials

http://www.amm.mw.tum.de/fileadmin/w00bkc/www/Image_Archive/Lehre/Prakt_MKS/Tutorial.pdf
http://www.inf.tu-dresden.de/content/institutes/iai/tis-neu/lehre/files/Praktika/Einfuehrung_MATLAB_Simulink.pdf
http://tait.e-technik.uni-ulm.de/downloads/books/signal_und_systemtheorie_matlab.pdf
http://www.pervasive.jku.at/Teaching/_2009SS/EmbeddedSystems/Begleitmaterial/02.%C3%9Cbungen/03%20Uebungsstunde%20Octave%20Signal%20Processing.pdf
http://www.christianherta.de/octaveMatlabTutorial.html
https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.050/vorlesungen/sose15/err/matlab_einfuhrung.pdf
…

Vergleich Matlab – Octave: Fehlende Funktionen, Unterschiede

http://www.gnu.org/software/octave/missing.html
…

Los geht's

Variablennamen unterscheiden sich nach Groß und Kleinschreibung,
Dezimalzeichen ist der PUNKT!

```
a=3.9
A=6.1
A+a
ans = 10
```

Die **Ausgabe wird durch ein Semikolon am Ende unterdrückt**, sehr wichtig bei Verarbeitung großer Datenmengen, bei denen Darstellung am Bildschirm wesentlich länger als Berechnung dauert

```
A=6.1;
```

Bei **langen Ausgaben** kann mit den Tasten ‚f', ‚b' gescrollt und mit ‚q' die Ausgabe quittiert werden (nur Octave).

Kommentare werden mit % eingeleitet

Mit eckigen Klammern werden Arrays oder Matrizen erstellt, Leerzeichen oder Komma trennt Spalten, Semikolon trennt die Zeilen

> VarZeile = [1 2 3 4 5]
> VarZeile2 = [1, 2, 3, 4,5]
> VarSpalte = [5; 6; 7; 8; 9]

Mit Matrizen kann direkt gearbeitet werden:
> VarZeile + VarZeile2
ans = 2 4 6 8 10

Zugriff auf Elemente (die als Index bei 1 anfangen)

> VarZeile(2)
Ans = 3

ändern

VarZeile(2) = 3+5i

Transponieren und konjugiert komplex
> VarSpalt = VarZeile'
Nur transponiert
> VarSpalt = VarZeile.'

Es gibt Operationen die können elementeweise durchgeführt werden z.B.
SkalarMultiplikation 2 * [1 2 3]

> Mit Variablen aber unsicher welcher Typ vorliegt:
> a * [1 2 3]
> ist a = 2, ist das Ergebnis [2 4 6], war a z.B. [4; 5; 6].' ergibt es
> 4 8 12
> 5 10 15
> 6 12 18

Was aber wenn ich [1 2 3] element-weise mit [2 4 6] multiplizieren will, also a(1)*b(1) und a(2)*b(2), …?
[1 2 3] * [2 4 6] ergibt Fehlermeldung
[1 2 3] .* [2 4 6] erzwingt element-weise Ausführung

Laufvariabeln erzeugen
n = 1:10

Schrittweite festlegen mit drittem Element
Phi = 0:pi/10:2*pi

Zugriff auf Teilbereiche in Matrix
> MatrGross = rand(10,10); % Zufallswerte erzeugen, 10 Zeilen, 10 Spalten
> MatrKlein = MatrGross(3,8) % dritte Zeile, 8. Spalte

> MatrKlein = MatrGross(3;:) % dritte Zeile, alle Spalten
> MatrKlein = MatrGross(3;4:7) % dritte Zeile, Spalten 4 bis 7
> MatrKlein = MatrGross(3;8:end) % dritte Zeile, Spalten 4 bis Ende
> MatrKlein = MatrGross(4:7;1:4)

> size(MatrKlein)
ans = 10 2

Oder
> [Anz_Zeilen, Anz_Spalten] = size(MatrKlein)
Anz_Zeilen = 10
Anz_Spalten = 2

Zeile oder Bereich einer Matrix löschen:
MatrGross(3,:) = [] % dritte Zeile löschen

OPERATOREN							
Matrix		**Array**		**relational**		**logisch**	
+	Addition	+	Addition	<	kleiner	&	UND
-	Subtraktion	-	Subtraktion	<=	kleiner gleich	|	ODER
*	Multiplikation	.*	Multiplikation	>	größer	~	NICHT
/	rechtsseitige Divisi-on	./	rechtsseitige Divi-sion	>=	größer gleich		
\	linksseitige Division	.\	linksseitige Divisi-on	==	gleich		
^	Potenz	.^	Potenz	~=	ungleich		
‘	transponiert						

str_Vorname = „Hans“
str_Name = „Glueck“
angehangen = [str_Vorname, str_Name]
mehrzeilig = [str_Vorname; str_Name]

Umgebung

Variablen sind gespeichert, können abgefragt werden mit
> whos
Der befehl „> clear a“ löscht die Variabe a, „> clear“ allein löscht **alle** Variablen,
Variable „ans“ enthält letztes Ergebnis

Pfeiltasten erlauben Zugriff auf vorherigen Befehl, Tab-Taste vervollständigt Befehle

Hilfe findet man mit z.b.
> help cos
Das ist wichtig und hilfreich bei Zweifel über Syntax!
Ausführlicher
> doc cos
Handbuchseiten, Navigation mit Pfeil, Bildlauftasten, Strg-C und Q, …

Übung: Was gibt size zurück? Was gibt length zurück? (Hilfe verwenden?)
size(MatrKlein)
length(MatrKlein)

Wechsel von Verzeichnis mit DOS / Unix Kommandos, (‚cd', ‚dir', ‚ls')
cd e:
cd 'Eigene Dateien'
vor abschließendem Hochkomma kann auch mit Tabulatortaste zur Vervollständigung durchgeführt werden

Mathematische Funktionen

Es sollten alle bekannten mathematischen Funktionen (z.B. sin, cos, exp, log, atan, abs) hinterlegt sein, man muss „nur" die korrekte Schreibweise finden
y1 = sin(Phi)
y2 = cos(Phi)

sqrt(2) % Quadratwurzel, squareroot
ans = 1.414
Längere Ausgabe: format long|short|…

Grafisch Funktion zeichnen

Befehle: hold on|off|all, plot, xlabel, ylabel, legend, title, subplot, grid on|off, …

plot(y1)
X- und Y-Werte übergeben
plot(Phi, y1)

mehrere Funktionen anzeigen?

plot(Phi, y2) löscht vorherige Ansicht
entweder
plot(Phi, y1)
hold on
plot(Phi,y2)
hold off

oder z.B. plot(Phi, [y1;y2])

title('sin und cos')
xlabel('\Phi \rightarrow') % Es können griechische Buchstaben zur Beschriftung genutzt werden, ("T_EX" Syntax)
legend('sin','cos')
grid on % schaltet Gitterlinien an

help plot

mehrere Plots in ein Fenster
> subplot(3,1,1)
> plot(Phi,y1)
> subplot(3,1,2)
> plot(Phi,y2)

Neues Fenster für Grafik öffnen, bzw. umschalten
> figure(2)

Übung/Demo:
Zufällige Werte erzeugen, darstellen, rechts daneben ein Histogramm (Häufigkeit der Werte (über y-Achse als Bereich), y-Achse gleich skalieren, … (Titel, Beschriftung, Farbe etc.)

Speichern

Grafik speichern/ drucken
> print('-dpng', 'plot.png')

Einzelne Variablen oder Messwerte als Datei speichern und einlesen
> help save
> help load

save(‚-ascii', ‚test.txt','Phi') % Achtung: Verlust der Genauigkeit bei ASCII Dateien!

alle Variablen im Speicher als Datei sichern
> save(‚test.mat') % kein Genauigkeitsverlust

später wieder die Umgebung herstellen
> load(‚test.mat')

Programmsteuerung

Zustand = 3
If Zustand == 2
  Plot(y1)
End
If Zustand == 3
  Plot(y2)
End

```
for i=0:-2:-10
  printf("%d\n",i);
end

while a < A
  a=a+1;
end


if a==0
  error( "a ist 0!" )
else
  b=c/a;
end



switch pnorm
```

```
  case 1;
    sum(abs(v))
  case inf;
    max(abs(v))
  otherwise
    sqrt(v *v)
end
```

for, if else end, while, switch

Achtung: oft gibt es “schnellere” Möglichkeiten als Schleifen über die einzelnen Elemente zu programmieren (siehe Zeitmessung):

```
for i=1:1000
  x(i)=sin(i/pi) % hier wird jedesmal eine Varialbe ergänzt, d.h neuer
Speicherplatz angefordert
end

besser:
n=1000;
x=zeros(1,n); % eine Variable mit benötigter Größe erzeugen = einmal Speicher
belegen
for i=1:1000
  x(i)=sin(i/pi)
end

schneller und octave-typisch:
i=1:1000;
x=sin(i/pi);
```

Unterprogramme

Sammlung von Befehlen in Textdatei mit Endung *.m gespeichert, kann über Eingabe des Dateinamens ausgeführt werden, so also ob Befehle direkt eingegeben wurden, erzeugte oder geänderte Variablen sind danach im Speicher, auf Variablen die im Speicher sind, kann zugegriffen werden

Funktionen

Sammlung von Befehlen in Textdatei mit Endung *.m gespeichert kann über Eingabe des Dateinamens ausgeführt werden
Werte mit denen die Funktion arbeiten soll, werden als **Parameter übergeben** (oder als globale Variable definiert)
Rückgabewerte werden als Antwortwert der Variable übergeben
Ergebnis = eigene_funktion(a,b,c)
Die *.m Datei unterscheidet sich lediglich durch die erste(n) Zeilen
function x = eigene_funktion(a,b,c)
oder bei **mehrerern Rückgabewerten**
function [x, y, z] = eigene_funktion(a,b,c)

Wieviele Parameter übergeben wurden oder zurückgegeben werden, kann abgefragt werde
If nargin == 0; a = 1;, b = 2; c = 3 end
If nargin == 1; b = 2; c = 3 end
If nargin == 2; c = 3 end

If nargout == 3
X = a+b
Y = a-b
Z = a*b
End

If nargout == 1
Y = a-b
Z = a*b

X = [a+b, y, z]
End

Innerhalb von Funktionen können neue Variablen definiert werden, die nur dort und nur zur Laufzeit bekannt sind, falls im aufrufenden Programm diese Variable schon existiert, sind es unterschiedliche Variablen!

Zeitmessung

‚tic' ‚tac' ermittelt verstrichene Zeit zwischen den beiden Befehlen

Darstellung

Es können mehrere Ausgabefenster erzeugt werden in denen unterschiedliche Darstellungen zu sehen sind.

- figure (2)
  - erzeugt neue Grafische Fenster bzw. wählt sie aus
- close all
  - Schließt alle offenen grafischen Fenster

Plot

Mit den folgenden Befehlen wird ein Diagramm gezeichnet.

- plot(x), plot(x,y)
  - Zeichnet ein Diagramm der Variable ‚x'
  - plot(x,y,'r') Zeichnet die Kurve rot
  - plot(x,y,'r', 'linewidth',3) Zeichnet die Kurve drei punkte breit
- semilogx, semilogy
  - Zeichnet ein halblogarithmisches Diagramm
- Loglog
  - Zeichnet ein doppelt logarithmisches Diagramm
- hold on|off
  - mehrere Plotbefehle in das aktuelle Diagramm überlagern

Subplot

In einem grafischen Fenster können mehrere Diagramme dargestellt werden.

- Subplot(2,3,1)
  - Diagramme mit in Spalten und drei Zeilen anordnen, aktuell wird in Diagramm 1 d.h. oben links geschrieben; Subplot(2,3,6) aktiviert unten rechts
- Subplot(2,2,[1 3])
  - platziert ein Diagramm links über 2 Diagrammplatzhalter

Beschriftung

- title(‚Diagrammtitel')
  - Der Titel wird über das aktuelle Diagramm dargestellt
- xlabel('Zeit in s')
  - Beschriftung der X-Achse
- ylabel('Amplitude in Volt')
  - Beschriftung der Y-Achse

Messergebnis speichern

- Save dateiname a b c
  - Speichert die Variablen „a“ „b“ und „c“ in der Datei „dateiname.mat“
- save('datei.txt', 'Messung', '-ascii')
  - speichert die Variable “Messung” in “datei.txt” als ASCII Daten
- Open dateiname x y z
  - Liest Variablen aus der Datei dateiname.mat und macht sie in „x“, „y“, „z“ verfügbar
- `Messung = load('-ASCII', '../HP/HP1.txt');`
  - Liest die Daten aus HP1.txt in die Variable “Messung” ein
- Print('-dpng', 'dateiname.png')
  - speichert das aktuelle Grafische Fenster als Dateiname.png (evtl. mit figure auswählen)

Weiteres

andere Plotfunktionen
plot3D
stem
…

- function, help, plot, semilogx, semilogy loglog, axis, axes, hold, legend, subplot, print, save, title, xlabel, ylabel, while, for, if, end

Beispiele

Signalverarbeitung

Beispiel:
bsp_dsv.m

Audio

Befehle:
Audio = Record(seconds, f_abtast)

sound(audio,f_abtast)
loadaudio
playaudio
saveaudio

X = fft(x); % DFT von x
pwr = X.*conj(X)/N; % Leistung des Signals
frs = (b-1)/N*fs; % Frequenzen
plot(frs,pwr);

Filter

Befehle: fir1 fir2 filter frequz stem impz, …
Beispiele:
bsp_filter.m
bsp_filter2.m
bsp_filter3.m

DLL

Kommt im Praktikum → DSV00

Interpolation

Beispiel
bsp_interpolieren.m

Korrelation

Versuch Signale und Systeme 2&3

Polynome, polyval

Regression (Gleichungsparameter an gegebene Werte anpassen)

Bildzugriff, Pixel lesen, schreiben …

Datei- öffnen/schreiben, Werte, Grafik, binär

Sound-Dateien öffnen und schreiben, als generische Form von Signalverläufen

Webzugriff

Version

23. Mai 2012 erster Entwurf, Eschreiter
29. Mai 2012 kleine Ergänzungen, Fehlerkorrekturen
30. Mai 2012 Formatierungen, Hyperlinks, Ergänzungen Screenshots
31. Mai 2012 Befehle grafische Ausgabe ergänzt
15. März 2013 Zeichenketten
27. Mai 2014 Kleinigkeiten
27. April 2015 Octave 3.8.2, Filter ergänzt
11. Oktober 2016 Octave 4.0.1 GUI Bild, plot erweitert, Übung 1 Histogramm