Matlab-Befehle

Allgemeines

Befehl	Funktionen
x=…	Zuweisen einer Variable bei einer Rechenoperation
;	Beendet Befehls-Zeilen und unterdrückt Ausgabe
%	Alle Zeichen nach % werden von Matlab ignoriert (Kommentar)
!	Ausführen eines Betriebssystemkommandos
exit *oder* quit	Beenden der Matlab-Sitzung
Crtl+C	Abbruch einer Endlosschleife
who	Ausgabe aller verwendeten Variablen
whos	Ausgabe aller verwendeten Variablen mit Datentyp und Dimension
clear	Löschen (Variable/Arbeitspeicher), *clear (name), clear x*
flops (0)	Setzt den Rechenoperationszähler auf 0
flops	Gibt die Anzahl der Rechenoperationen seit Beginn der Sitzung bzw. von Flops (0) an
format short/format long	Umschalten des Ausgabeformates (short: 4 Stellen nach Komma, long: 14 Stellen), keine Veränderung der Rechengenauigkeit
format short e/format long e	Fließpunktdarstellung
format hex	Hexadezimaldarstellung
format rat	Darstellung in Form von Brüchen
format compact	keine Ausgabe von zusätzlichen Leerzeichen
format loose	zusätzliche Leerzeichen
save(x)	Speichern der Variable auf der Festplatte als mat-File der Form x.mat im aktuellen Verzeichnis
load(x)	Laden der gespeicherten Variablen aus dem File x.mat
diary on/off	Protokoll wird in File namens diary abgelegt; diary on startet die Protokollierung und diary off beendet diese.
disp('…')	Ausgabe des enthaltenen Strings auf den Bildschirm
disp(x)	Ausgabe der Variablen x auf den Bildschirm
x=input(*'Geben sie x ein:'*)	Eingabe der Variablen x mittels Tastatur
pause(n)	Pause von n Sekunden
help	Hilfe-Funktion
help *Befehl*	Hilfe zu dem Befehl (→ helpdesk: Befehlsübersicht)
lookfor *Stichwort*	Stichwortsuche in Hilfetexten
demos	Startet Überblick über Demonstrationen
what	Ausgabe einer Liste der m-Dateien
computer	Ausgabe des Computer-Typs
delete *filename*	Datei löschen
dir /dir *name*	Ausgabe des Verzeichnisses *name*
type *filename*	Anzeige der Datei filename am Bildschirm
pack / pack *filename*	Komprimieren des Speichers→ Variablen werden zwischengespeichert
more on/off	Seitenweise Ausgabe an/aus
fprintf	formatiertes Schreiben

→ In Matlab werden grundsätzlich alle Variablen als Matrizen interpretiert. Das gesamte Kalkül ist matrixorientiert. In den Befehlen der Zusammenstellung wird prinzipiell die einfachste Anwendung dokumentiert.
Für weitere Informationen siehe Hilfe-File.

Formatiertes Schreiben: fprintf(*fid, format, Variablen*)

-formatiert und schreibt den Realteil der Variablen in das Feld fid (s. Befehle fopen, fclose)
-Format: Zeichenkette mit Platzhaltern für die Variablen und Sonderzeichen
-Format wird wiederholt, bis alle Elemente/Variablen abgearbeitet sind
-Platzhalter: %-Zeichen, gefolgt von Steuerzeichen und Ausgabetyp
-Bei fid=1 bzw. fehlendem fid: Ausgabe im Kommandofenster

fprintf-Ausgabetypen

f: Fließkommadarstellung

e,E: Exponentialdarstellung

g,G: Mischung aus f und e je nach Größe der Zahl

c: Einzelnes Zeichen

s: Zeichenkette

d,i: Dezimaldarstellung

o: Oktaldarstellung

x,X: Hexadezimaldarstellung

fprintf-Steuerzeichen

+: Mit Vorzeichen

-: Linksbündig

n.m: Ausgabe mit Mindestbreite von n Zeichen mit m Nachkommastellen; bei den Ausgabetypen: e ,f ,g auf m Stellen mit führenden Nullen aufgefüllt

0: Mit führenden Nullen bis zur Feldbreite aufgefüllt

fprintf-Sonderzeichen

\n: **Zeilenschaltung.** Notwendig, falls die Eingabeaufforderung in einer neuen Zeile erscheinen soll. Beachte: Mit dem Ende der "fprintf"- Anweisung wird nicht automatisch in die nächste Zeile gewechselt.

\r: Wagenrücklauf. Z.B. um während eines Programmablaufs Fortschrittsmeldungen übereinander zu schreiben

\t: Tabulatorschaltung

\b: Vorhergehendes Zeichen löschen

\\: Erzeugt \

%%: Erzeugt %

→"fprintf" kann auch Vektoren und Matrizen ausgeben.
Hierbei wird die entsprechende Formatanweisung für jedes Element der Matrix wiederholt.
Es wird spaltenweise vorgegangen, so dass die gewohnte Reihenfolge (zeilenweise) durch Transponieren der auszugebenden Matrix erzeugt werden muss.(s. letztes Bsp.)

Beispiele:

Eingabe	*Ausgabe*
>> X=1:10; >> fprintf('Elemente von X:'); fprintf(' %d ',X);fprintf('\n');	Elemente von X: 1 2 3 4 5 6 7 8 9 10
>>fprintf('\n')	(Leerzeile)
>> x=3.5; >> fprintf('Inhalt der Variable x: %f\n', x)	Inhalt der Variable x: 3.500000
>> fprintf('Inhalt der Variable x: %+015.2E\n', x)	Inhalt der Variable x: +0000003.50E+00
>> X=[1,2;3,4]; >> fprintf('Die Elemente von X sind: %d %d %d %d\n', X)	Die Elemente von X sind: 1 3 2 4
>> fprintf('Die Elemente von X sind: %d %d\n', X',)	Die Elemente von X sind: 1 2 Die Elemente von X sind: 3 4

Matrizen

-Vektoren und Matrizen

Befehl und Bsp.	Ausgabe	Funktion
x=[… … …] oder x=[…,…,…] Bsp.: x=[1 2 3]	x=1 2 3	Zeilenvektor getrennt durch Leerzeichen oder Kommas
x=…**:**… Bsp: x=1:10	x=1 2 3 4 5 6 7 8 9 10	Zeilenvektor von 1 bis 10 mit Schrittweite 1
x=…**:**…**:**… Bsp.:x=1:0.5:4	x=1 1.5 2 2.5 3 3.5 4	Zeilenvektor von 1 bis 4 mit festgelegter Schrittweise ½
x=linspace(…,...,...) Bsp.: x=linspace(3,4,11)	x=3 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4	Zeilenvektor mit 11 gleichen Abständen von 3 zu 4;Vektor mit linearen Zuwächsen
x=logspace(…,…,…)		Vektor mit logarithmischen Zuwächsen
x=[…;…;…] Bsp.: x=[1;2]	x= 1 2	Spaltenvektor, Trennung der Zeilen durch ;
x=[…,…;…,…] Bsp.: x=[1,2;3,4]	x=9 2 3 4	Matrix
x(i,k)=… Bsp.: x(1,1)=1	x= 1 2 3 4	Elementweiser Zugriff
x(…,**:**) Bsp.: x(2,:)	x=3 4	Zugriff auf bestimmte (Bsp.: 2.) Matrixzeile
x(**:**,…) Bsp.: x(:,2)	x=2 4	Zugriff auf bestimmte Matrixspalte
t='Text'	t= Text	(1,4)-Matrix mit Zeichen als Elemente
z=[x,x]	z= 1 2 1 2 3 4 3 4	Zusammensetzen von Matrizen
rot90(x)	x= 2 4 1 3	Rotation der Elemente in mathematisch pos. Richtung
fliprl (x)	x= 2 1 4 3	links-rechts-Spiegelung
diag (x)	x=1 4	Diagonale der Matrix ausgegeben als Vektor
tril (x)	x= 1 0 3 4	unterer Dreiecksteil
triu (x)	x= 1 2 0 4	oberer Dreiecksteil
reshape(x,m,n) Bsp.: reshape(x,1,4)	x= 1 2 3 4	Neugruppierung in m Zeilen und n Spalten
x(:)	x=1 2 3 4	Umformung einer Matrix in eine einzelne Spalte

-spezielle Matrizen 1

Befehl	Ausgabe	Funktionen
zeros(m,n) *Bsp: zeros(2,2)*	ans= 0 0 0 0	Nullmatrix vom Typ (m,n)
ones(m,n) *Bsp: ones(2,2)*	ans= 1 1 1 1	Matrix vom Typ (m,n), jedes Element 1
eye(m,n) *Bsp: eye(2,2)*	ans= 1 0 0 1	Einheitsmatrix vom Typ(m,n)
rand(m,n)	*Zufall*	Zufallsmatrix vom Typ (m,n) mit reellen Zufallszahlen aus dem Intervall [0;1]

-Rechenoperatoren

(als Matrixoperationen, d.h. Addition und Subtraktion elementweise)

+	*x+y*	Addition
-	*x-y*	Subtraktion
*	*x*y*	Multiplikation
^	*x^y*	Potenz
^-1	x^-1	Inversenbildung
\	*y\x*	y^-1*x
/	*x/y*	x*y^-1
’	*x’*	ergibt x*
.’	*x.’*	ergibt x^T
.* **.^** **./** **.**	(→ gleiche Operation)	Elementweise Ausführung der Operationen. Keine Matrixoperation!

-elementare mathematische Werte/Funktionen (elementare Berechnung)

pi	π
i,j	√-1
inf	∞
NaN	undefiniertes Ergebnis
clock	Uhrzeit
date	Datum
eps	relativer Gleitpunktfehler, 2^(-52)
abs(x)	Betrag von x
sign(x)	Vorzeichen von x
angle(x)	Argument von x , wenn x eine komplexe Zahl ist
rem(x,y)	Divisionsrest
mod(x,y)	Modul nach Division
sqrt(x)	Wurzel
round(x)	Rundung auf nächste ganze Zahl
fix(x)	Rundung zu 0
floor(x)	Rundung nach – ∞(→ immer abrunden)
ceil(x)	Rundung nach ∞(→ immer aufrunden)
real(x)	Realteil von x
imag(x)	Imaginärteil von x
conj(x)	Komplex konjungierte Zahl zu x
exp(x)	Exponentialfunktion e^x
log(x)	Natürlicher Logarithmus ln x
log10(x)	Logarithmus zur Basis 10 (lg x)
sin(x)	Sinus von x
cos(x)	Cosinus von x
tan(x)	Tangens von x
asin(x),acos(x),atan(x)	arcussinus / arcuscosinus / arcustangens von x
any(x) x= 0 2 3 *Bsp .:x=[0, 2, 3 ;0,0,8]* 0 0 8	Abfrage: Ein Elemente einer Spalte einer Matrix ungleich Null? Ergebnis= 0 1 1
all(x) *(Bsp. : siehe any)*	Abfrage: Sind alle Elemente der Spalte =0? Ergebnis= 0 0 1
find(x) x=[1 2 0 2 0 0 5]	Finde Indizes von Elementen eines Vektors für Elemente ungleich 0. Ergebnis= 1 2 4 7
isnam(x)	Abfrage: Ist ein Element undefiniert (NaN)? (→ elementweise Abfrage und Anzeige)
isfinite(x)	Abfrage: Ist ein Element in der Spalte endlich?

→ Anzeige bei Abfragen: wahr: 1, falsch: 0

-Funktionen von Matrizen

sum (x)	Spaltensummen einer Matrix bzw. Summe der Elemente eines Vektors
inv(x) oder x^-1	inverse Matrix zu x
max(x)	ergibt Vektor mit den größten Elementen der einzelnen Spalten von x
min(x)	ergibt Vektor mit den kleinsten Elementen der einzelnen Spalten von x
size(x)	Dimension von x
length(x)	Länge des Vektors x bzw. bei Matrizen max(size(x))
rank(x)	Rang von x
det(x)	Determinante von x
eig(x)	Eigenwerte/Eigenvektoren von x
mean(x)	ergibt Vektor mit den Mittelwerten der Spalten von x
std(x)	ergibt Vektor mit den Standardabweichung der Spalten von x
sort(x)	Sortiert jede Spalte von x in aufsteig. Reihenfolge
prod(x)	Ergibt Vektor mit den Produkten aus den Elementen der Spalten
cumsum(x)	ergibt Vektor mit den kumulierten Summen der Elemente der einzelnen Spalten von x
cumprod(x)	ergibt Vektor mit den kumulierten Produkten der Elementen der einzelnen Spalten von x
hist(x)	Histogramme zu den Spalten von x
diff(x)	ergibt Vektor mit den Differenzen der benachbarten Elemente des Vektors x
corrcoef(x)	ergibt Matrix von Korrelations-Koeffizienten für das Feld x, bei dem jede Zeile eine Beobachtermenge und jede Spalte eine Variable darstellt.
cov(x)	ergibt Matrix von Kovarianzen für das Feld x, bei dem jede Zeile eine Beobachtermenge und jede Spalte eine Variable darstellt.
cplxpair(x)	Sortieren konjungiert-komplexer Paare nach steigendem Realteil
cond(x)	Konditionszahl
rcond(x)	Schätzwert der reziproken Konditionszahl
norm(x,p)	p-Norm der Matrix x
balance(x)	sucht Ähnlichkeitstransformation, Ausgabe von B=T^(-1)AT möglichst nah bei der Diagonaldarstellung
chol(x)	Cholesky-Zerlegung
hess(x)	Hessenberg-Form
lu(x)	Faktoren aus der Gauss'schen Elimination
null(x)	Nullraum
qr(x)	orthogonale Dreieckszerlegung (QR-Algorithmus)

-spezielle Matrizen 2 (→ siehe auch help gallery)

compan(n)	Ähnlichkeitstransformation
hadamard(n)	Hadamard-Matrix
hankel(n,k)	Hankel-Matrix
hilb(n)	Hilbert-Matrix
invhilb(n)	Inverse Hilbertmatrix
magic(n)	Magisches Quadrat
toeplitz(n,k)	Toeplitz-Matrix
vander(n)	Vandermonde-Matrix

Polynome (→ siehe auch polyfun)

poly(x)	Koeffizientenvektor des charakteristischen Polynoms von x
roots(p)	Nullstellen eines Polynoms
polyval(p,x)	Berechnung des Polynoms p, an der Stelle x
polyvalm(p,x)	Berechnung des Matrix-Polynoms p, an der Stelle x
conv(u,v)	Polynommultiplikation
[q,r]=deconv(u,v)	Polynomdivision→ u/v=q+r/v
[R,P,K]=residue(u,v)	Partialbruch-Zerlegung von u/v; R: Vektor der Zähler; P: Vektor der Nennerstelle; K: ganzzahliger rationaler Anteil
polyfit(x,y,n)	Anpassen eines Polynoms des Grades n an Daten (x,y) mittels Methode der kleinsten Quadrate

Lösen nichlinearer Gleichungen und Extremwerte

fminbnd(fun,x1,x2)	Minimum einer Funktion mit einer Variablen in einem bestimmten Interval
fminsearch(fun,x0)	Minimum einer Funktion mit mehreren Variablen
fzero(fun,x0)	Nullstellen einer Funktion einer Variablen

Funktionen für Matrizen (siehe auch help: specfun (spez. Funktionen),elfun (Elementfunktionen), elmat (Matrixerzeugung), datafun (Vektorfunktionen))

expm(x)	Matrix-Exponential-Funktion
logm(x)	Matrix-Logarithmus
sqrtm(x)	Quatratwurzel einer Matrix
funm(x,fun)	Beliebige Matrix-Funktion
g=inline('t^2')	Definition einer Inline-Funktion g(t)=t². Durch Eingabe von g(3) wird g an Stelle 3 ausgewertet. (analog für mehr Parameter)

Programmierung

global x	Definition einer Globalvariablen
exist x	Existenz einer Variablen prüfen
eval *(Bsp.: eval([,y=' fun '(3);'])*	Interpretation von Text als Befehl
feval *((Bsp.: feval(fun,3))→ Funktionswert =3)*	Berechnung einer Funktion, die als Text gegeben ist
etime	Rechenzeit
function *name*	Funktionsdefinition
keyboard	übergibt Steuerung an Tastatur
error	Ausgabe einer Fehlermeldung
echo on/off	Befehlsecho an/aus
startup	M-Datei als Initialisierung von MATLAB wird gerufen
menu	Auswahl aus einem Menü
input(,Zahl')	Zahleneingabe per Tastatur
input(,Name','s')	Stringeingabe
isempty(x)	Abfrage: Ist Matrix x leer?
isstr(x)	Abfrage: Ist Matrix x Textvariable?
strcmp(string1,string2)	Text-Vergleich zwischen string1 und string 2, ergibt 1, wenn string1=string2
nargin	Ausgabe der Anzahl der Eingabe-Elemente einer Funktion
nargout	Ausgabe der Anzahl der Ausgabe-Elemente einer Funktion

-Relationen

<	kleiner
>	größer
<=	kleiner gleich
>=	größer gleich
= =	gleich
~=	ungleich
&	und
|	oder
~	nicht

Programmierung-Schleifen

if c= =3 (→ Bedingung) c=c+1; (→ Anweisung) **end**	Bedingte Anweisung: Falls c=3 erhöhe um 1.
if c= =3 (→ Bedingung) c=c+1; (→ Anweisung) **else** c=c-1; (→ Anweisung) **end**	Bedingte Verzweigung: Falls c=3 wird c um 1 erhöht, sonst wird c um 1 erniedrigt.

if c= =3
c=c+1;
elseif c>0;
c= c-1;
else
c=100;
end
	Bedingte Verzweigung:
	Falls c=3 wird c um 1 erhöht, ansonsten wird folgendermaßen vorgegangen:
	Falls c>0 wird c um 1 erniedrigt, sonst auf 100 gesetzt.

for k=1**:**1**:**10 l(k)=2*k; (→ Anweisung) **end**	For-Schleife: K läuft von 1 bis 10 mit Schrittweite 1, der k. Eintrag von l wird auf das doppelte von k gesetzt. Ergebnis: k=10, l=[2 4 6... 20] (positive k!!!, Schrittweite 1 kann weggelassen werden, oder verändert werden)

n=4;
while n>1 (→ Bedingung)
l(n)=2*n; (→ Anweisung)
n=n-1;
end
	While-Schleife:
	Solange die Bedingung n>1 erfüllt ist,werden die Anweisungen ausgeführt: zu Beginn ist n=4, d.h. l(4)=8,dann wird n um 1 erniedrigt,
	also n=3. N ist immer noch größer als 1, d.h. l(3)=6,
	→ n=2. Im letzten Durchlauf wird l(2)=4, n=1 gesetzt.
	Ergebnis: n=1, l=[0 4 6 8]

x=3; **switch** x (→ Term) **case** Wert1, *z.B. Wert1=0* disp (‘x ist 0’) (→ Befehl) **case** Wert2, *z.B. Wert2=2* disp (’x ist 1’) (→ Befehl) **…** **case** Wert n disp(’x ist n’) (→ Befehl) **otherwise** disp (’Zahl ist unbekannt’) (→ Befehle) **end**	Term wird berechnet und Ergebnis mit Wert1 bis Wertn verglichen und bei Übereinstimmung werden die zugehörigen Befehle ausgeführt, sonst werden die otherwise-Befehle ausgeführt.
break	Bricht (Endlos-)Schleifen ab
pause	Pause bis Taste gedrückt wird

Grafik

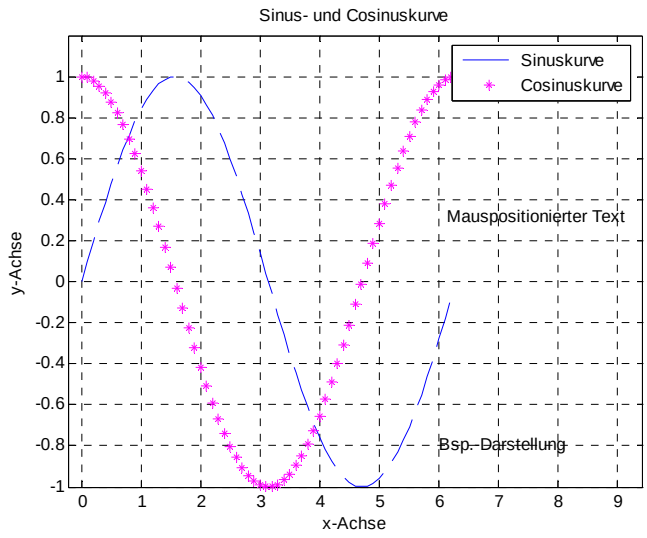
x=0 :0.1 :2*pi ; y=sin(x) ; **plot(x,y)**	Visualisierung der Sinuskurve von 0 bis 2Pi
plot[x1 x2 x3],[y1 y2 y3])	Visualisierung durch die Punkte (x1,y1),(x2,y2),(x3,y3)
hold on/off	Mehrere plots in eine Fenster der grafischen Ausgabe (figure) → on
close/close all	Aktuelles grafisches Fenster schließen, alle schließen
figure/figure(3)	Grafisches Fenster Nummer 3 öffnen oder anzeigen(falls es schon existiert, beziehen sich ab jetzt alle grafischen Befehle auf diese Fenster).
xlabel(‘x-Achse‘)	Beschriftung der x-Achse hinzufügen
ylabel(‘y-Achse‘)	Beschriftung der y-Achse hinzufügen
legend(‘Kurve 1‘,‘Kurve 2‘)	Fügt Legende hinzu, pro gekennzeichnetem Element eine Stringkonstante
axis([xmin xmay ymin ymax])	Ausgabe der Größe der angezeigten Ausschnittes
title(‘Dies ist der Titel der Grafik‘)	Titel der Grafik erstellen
subplot(m,n,p)	Teilt das Grafikfenster in m*n Unterfenster und wählt das p.-te aus. Diese ist dann aktuell, alle grafischen Befehle beziehen sich dann darauf.
colormap(hsv)	Benutzung einer vordefinierten Farbskale ‚hsv’
grid	Gitternetz anzeigen
text(x-Koordinate,y-Koordinate,‘Text‘)	Text in die Grafik einfügen
gtext(’string’)	Maupositionierter Text
ginput(x)	Grafische Eingabe
loglog(x)	Logarithmische Darstellung für x und y
semilogx(y)	Logarithmische Darstellung nur in x
semilogy(x)	Logarithmische Darstellung nur in y
polar(*winkel,radius*)	Plot in polar Koordinaten
mesh(x,y,z)	3D-Grafik: Gitternetzdarstellung
[X,Y]=meshgrid(x,y)	Erzeugt aus der Vektoren x und y ein Gitternetz, das durch die Matrizen X und Y einbeschrieben wird.
contour(x)	Konturdarstellung von x
bar(x)	Balkendiagramm der Spalten von x
stairs(x)	Treppendiagramm der Spalten von x
surf(x,y,z)	3D-Grafik: Flächendarstellung
Grafikbefehl 3 (Bsp.: plot3, contour3,…)	„Grafikbefehl“ auf 3D-Grafik bezogen

Farben und Muster für Graphen

plot(x,y,‘r-’)	Beispiel für Eingabe
‘y‘	gelb
‘m‘	magenta/violett
‘c‘	cyan/hellblau
‘r‘	rot
‘g‘	grün
‘b‘	blau
‘k‘	schwarz
‘w‘	weiß
‘-‘ (durchgezogen); ‘- -‘ (gestrichelt); ‘**-.**‘ (Strich-Punkt); ‘**x**‘(Kreuz); ‘+‘(Kreuz); ‘***‘ (Stern); ‘**:**‘(gepunktet); ‘°‘ (Kreis)	

2D-Grafikbeispiel

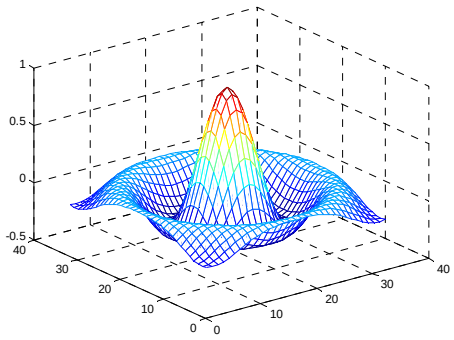
```
>> x=0:0.1:2*pi;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'--b')
>> hold on
>> plot(x,z,'*m')
>> xlabel('x-Achse')
>> ylabel('y-Achse')
>> legend('Sinuskurve','Cosinuskurve')
>> title('Sinus- und Cosinuskurve')
>> grid
>> axis([-0.2,3*pi,-1,1.2])
>> text(6,-0.8,'Bsp.-Darstellung')
>> gtext('Mauspositionierter Text')
```



3D-Grafikbeispiel

```
>> [X,Y] = meshgrid(-8:.5:8);
>>R = sqrt(X.^2 + Y.^2) + eps;
>> Z = sin(R)./R;
>>mesh(Z)
>>grid
```

(Darstellung entlang x- und y-Achse mit Netzindexnummer, nicht mit x- bzw. y-Wert)



```
>> [X,Y] = meshgrid(-8:.5:8);
>>R = sqrt(X.^2 + Y.^2) + eps;
>> Z = sin(R)./R;
>> surf(Z)
>> colormap(hsv)
>>grid
```

(Darstellung entlang x- und y-Achse mit Netzindexnummer, nicht mit x- bzw. y-Wert)