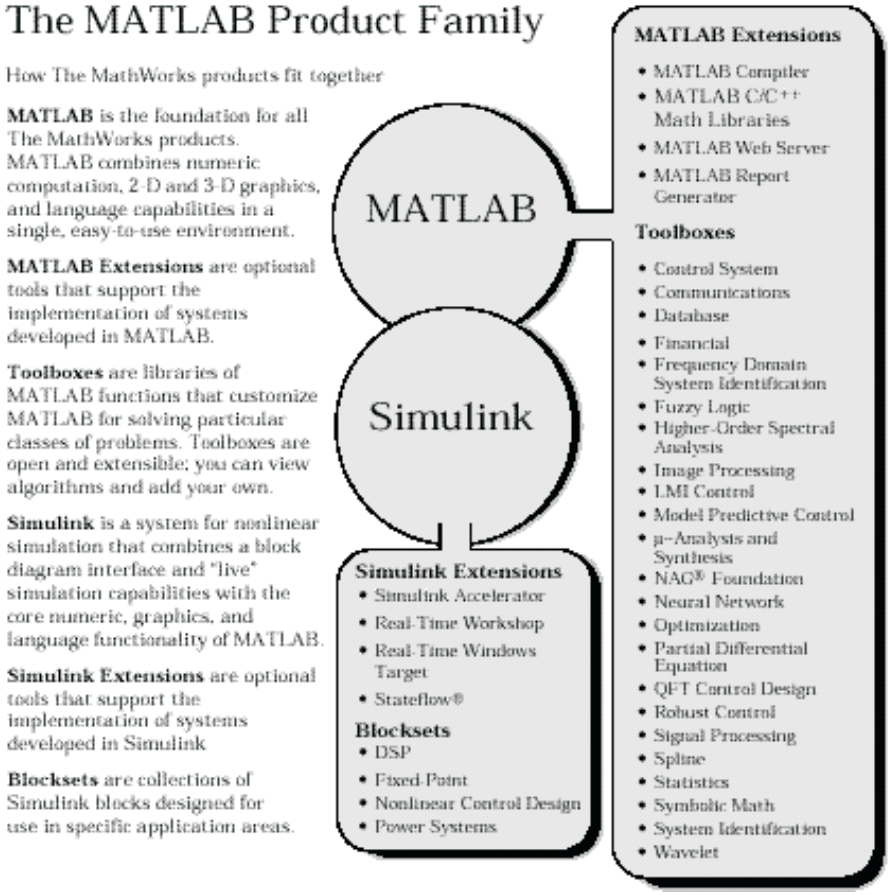
Das Softwaresystem MATLAB

MATLAB (MATrix LABoratory) vereint unter einer einheitlichen Benutzeroberfläche die 3 Hauptkomponenten:

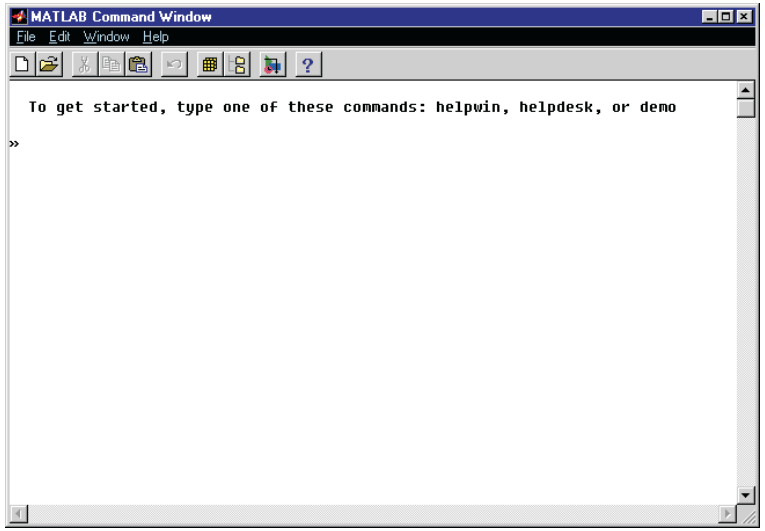
- **BERECHNUNG:** MATLAB stellt eine Sammlung von ausgetesteten Algorithmen bereit.
- **VISUALISIERUNG:** MATLAB verfügt über sehr ausgereifte Techniken zur Visualisierung von Daten.
- **PROGRAMMIERUNG:** MATLAB beinhaltet eine eigene hohe Programmiersprache.

MATLAB und Zusatzsoftware



Interaktives Arbeiten in MATLAB

Verbindung zu MATLAB–Interpreter über das Kommandofenster:



Z.B. Rechenoperationen zwischen Skalaren:

+ Addition; - Subtraktion; * Multiplikation; / Division.

Eingabe und Ausgabe

```
>> 4.5+5.5
ans =
    10
```

MATLAB hat die Berechnung ausgeführt und das Ergebnis unter der *Workspace*–Variable `ans` (von *answer*) gespeichert. Erzeugung eigner *Workspace*–Variablen, z.B. durch

```
>> erg=5*5
erg =
    25
```

- Unterdrückung der Ausgabe durch das Semikolon: ;
- Zeilenübergreifende Ausdrücke: ...

Die Variablen des Workspace

Die im *Workspace* gespeicherten Variablen können mit `who` oder für eine ausführlichere Ausgabe mit `whos` gesichtet werden: Das Kommando `whos` führt nach der bisherigen Variablenzuweisung zu der folgenden Ausgabe:

```
 Name      Size            Bytes  Class
  ans       1x1                8  double array
  erg       1x1                8  double array
Grand total is 2 elements using 16 bytes
```

- `Size`: Feldgröße, hier 1×1, also skalare Werte.
- `Bytes`: Speicherplatz
- `Class`: Datentyp, weitere z.B. `char`, `sparse`, `cell`.

Komplexe Zahlen

Imaginäre Einheit `i` und `j`, d.h. $i = j = \sqrt{-1}$.

`abs`	Absolutwert (Betrag, Modul) $\|z\|$
`real`	Realteil $Re(z)$
`imag`	Imaginärteil $Im(z)$
`angle`	Winkel (Argument) $arg(z)$
`conj`	Konjugiert komplex $\bar{z}$
`abs(z)*exp(i*angle(z))`	$\|z\| \cdot e^{i \cdot arg(z)}$

z.B.

```
>> z=4+5*i
z =
   4.0000+ 5.0000i
>> abs(z)
ans =
    6.4031
```

Weitere vordefinierte Variablen

In MATLAB sind neben `ans`, `i` und `j` die folgenden Variablen vordefiniert:

`eps`	Maschinengenauigkeit
`pi`	Kreiszahl π
`inf`	Unendlich ∞
`NaN`	Not–a–Number
`clock`	Aktuelle Uhrzeit und Datum
`date`	Aktuelles Datum
`flops`	Zählt die Gleitpunktopertionen
`nargin`	Anzahl der Eingabeargumente einer Funktion
`nargout`	Anzahl der Ausgabeargumente einer Funktion
`inputname`	Eingabeargumentname
`computer`	Identifiziert Computer

Wichtig für die Arbeit im interaktiven Modus

`help`	Hilfe
`lookfor`	Informationssuche anhand eines Stichwortes
`clear`	Löscht die Variablen des *workspace*
`clear all`	Löscht alle Variablen (auch globale)
`load`	Laden von Daten
`save`	Speichern von Daten
`print`	Drucken und Speichern von Graphiken
`cd` oder `pwd`	Wechselt ein Verzeichnis
`delete`	Löscht ein file
`dir` oder `ls`	Zeigt den Inhalt eins Verzeichnisses
`exit`	MATLAB beenden

Laufende Operationen können mit `ctrl` und `c` abgebrochen werden.

Operationen mit Matrizen

Matrizen sind die Grundbausteine von MATLAB. Vektoren und Skalare sind Spezialfälle von Matrizen. Es gibt in MATLAB die folgenden Möglichkeiten Matrizen auf den *Workspace* zu bekommen:

- Wir geben die Elemente einer Matrix explizit ein
- Wir erzeugen sie über vordefinierte MATLAB Funktionen
- Wir erhalten sie als Ergebnis eigner Funktionen
- Wir lesen sie von einem Datenträger ein

Explizite Eingabe

```
>> m=[1 2 3;4 5 6;7 8 9]
m =
     1     2     3
     4     5     6
     7     8     9
>> m=[1,2,3;4,5,6;7,8,9]
m =
     1     2     3
     4     5     6
     7     8     9
>> m(2,2)
ans =
     5
```

Extraktion von Zeilen und Spalten

```
>> vz=m(1,:)
vz =
     1     2     3
>> vs=m(:,1)
vs =
     1
     4
     7
>> [Zeilen,Spalten]=size(m)
Zeilen =
     3
Spalten =
     3
```

Extraktion von Untermatrizen

```
>> m=[1 2 3;4 5 6;7 8 9]
m =
     1     2     3
     4     5     6
     7     8     9
>> um=m(1:2,2:3)
um =
     2     3
     5     6
```

Extraktion der Zeilenelemente von 1 bis 2 und die Spaltenelemente von 2 bis 3.

Spezielle Funktionen für Vektoren

- Länge eines Vektors: `length`
- Erzeugen eines Vektors über `v = von : Schrittweite : bis`

```
>> v=1:4
v = 1     2     3     4

>> v=1:0.5:4
v = 1.0000 1.5000 2.0000 2.5000 3.0000 3.5000 4.0000
```

- Erzeugen eines Vektors über `linspace` oder `logspace`

```
a=linspace(0,pi,5)
a = 0    0.7854    1.5708    2.3562    3.1416
```

Spezielle Matrizen

`zeros`	Nullmatrix
`eye`	Einheitsmatrix
`diag`	Diagonal Matrix und die Diagonalen einer Matrix
`hilb`	Hilbert–Matrix
`toeplitz`	Toeplitz–Matrix
`ones`	Matrix deren Elemente alle 1 sind
`rand`	Gleichverteilte Zufallsmatrix
`randn`	Normalverteilte Zufallsmatrix
`[]`	Leere Matrix

```
>> t=toeplitz(1:4)
t =
     1     2     3     4
     2     1     2     3
     3     2     1     2
     4     3     2     1
```

Operationen mit Matrizen

`A+B`	Addition $A + B$
`A-B`	Subtraktion $A - B$
`A*B`	Multiplikation AB
`A/B`	rechte "Division" löst $XA = Y$ nach X
`A\B`	linke "Division" löst $AX = Y$ nach X
`A^p`	Potenzieren A^p
`A'`	Konjugiert Transponieren A_*^T
`A.'`	Transponieren A^T
`kron(A,B)`	Kronecker Tensorprodukt $A \otimes B$
`inv(A)`	Inverse der Matrix A
Elementweise:	
`A.*B`	Multiplikation
`A./B`	rechte Division
`A.\B`	linke Division
`A.^p`	Potenzieren

Beispiele:

```
>> m1=[1,2;3,4];
>> m2=[5,6;7,8];
>> m=m1*m2
m =
    19    22
    43    50
>> m=m1.*m2
m =
     5    12
    21    32
```

Sei `A` eine $n \times n$ Matrix und `b` ein Spaltenvektor der Länge n, dann ist `x=A\b` Lösung von von `Ax=b`.

Sparse–Modus

Die Verwendung des `sparse`-Modus für dünn besetzte Matrizen führt zu den folgenden Vorteilen:

- Es wird weniger Speicher benötigt
- Algorithmen werden schneller

```
>> a=[1,2,0;4,0,0;0,8,0];
>> f=sparse(a)
f =
   (1,1)        1
   (2,1)        4
   (1,2)        2
   (3,2)        8
```

2D–Plots

Die `plot` Funktion ist eine der grundlegenden Graphikfunktionen in MATLAB. Mit ihr können lineare 2D–Darstellungen realisiert werden. Sind `x` und `y` zwei Vektoren von gleicher Länge, dann öffnet der Befehl `plot(x,y)` ein Graphikfenster und stellt die Elemente des Vektors `y` gegen die Elemente des Vektors `x` dar. Die einzelnen Punkte (`x(n)`,`y(n)`), $n = 1, 2, \ldots, l$ werden durch Linen verbunden, so daß ein Polygonzug entsteht. Beispiel:

```
>> x=0:0.1:4*pi;
>> y=sin(x);
>> plot(x,y)
```

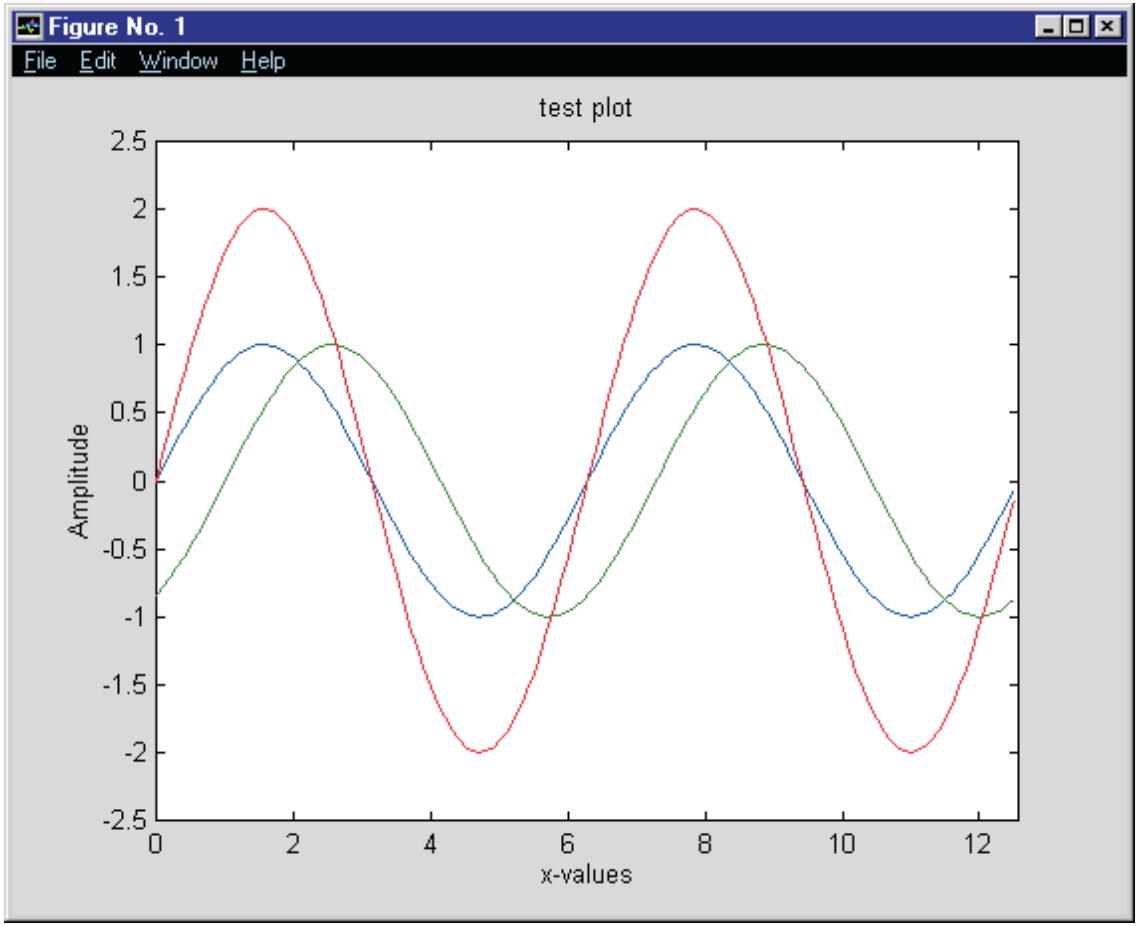
Es können natürlich auch andere Darstellungsformen gewählt werden, z.B.

```
>> plot(x,y,'b.')
```

stellt die Elemente von `y` gegen `x` als blaue Punkte dar.
Mehre Plots in einem einzigen Graph, Skalierung und Achsenbeschriftungen:

```
>> x=0:0.1:4*pi;
>> y1=sin(x);
>> y2=sin(x-1);
>> y3=2*sin(x);
>> plot(x,y1,x,y2,x,y3);axis([0,4*pi,-2.5,2.5]);...
title('test plot');xlabel('x-values');ylabel('Amplitude');
```

Darstellung der 3 Sinus–Funktionen



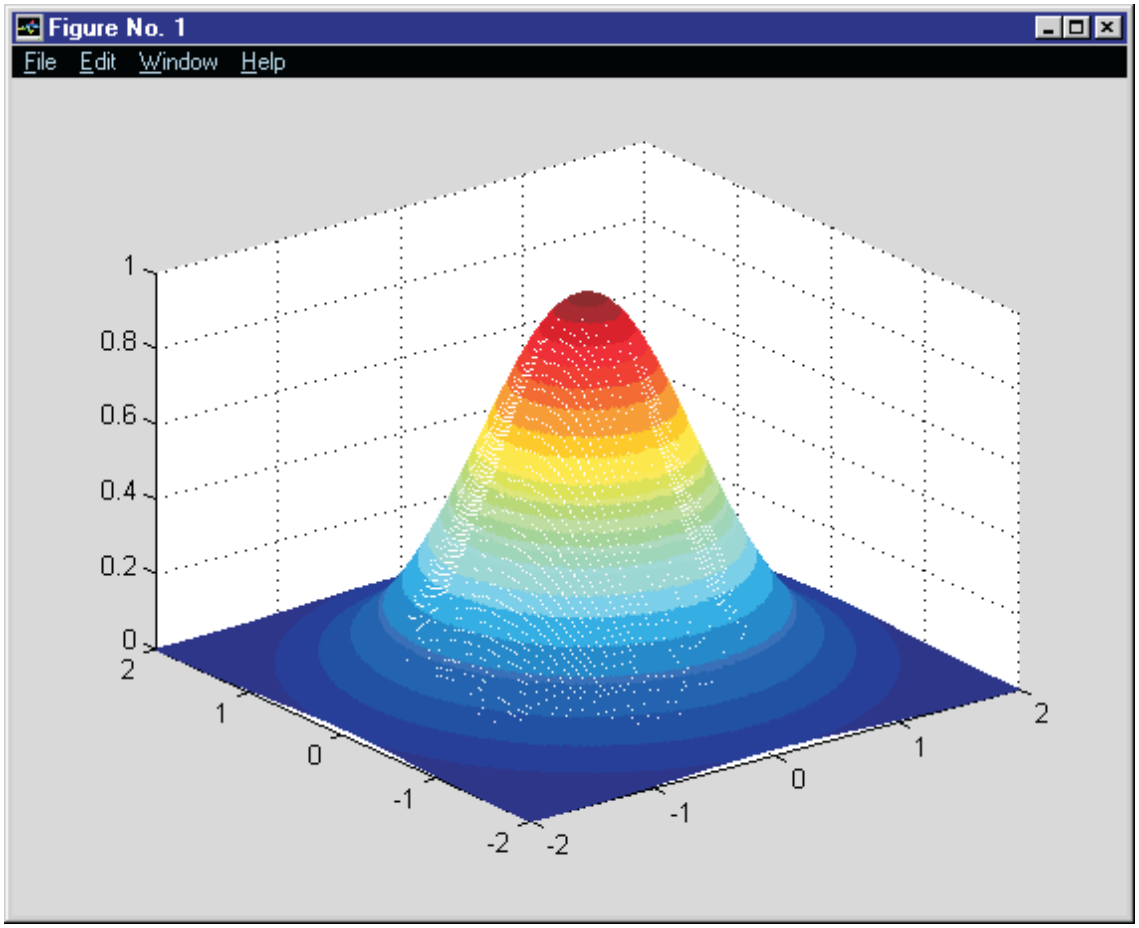
3D–Plots

3D–Plots können z.B. mit der Funktion den Funktionen `mesh` und `surf` realisiert werden. Als Beispiel soll die Funktion $z = f(x,y) = e^{-x^2-y^2}$ über dem Quadrat $[-2,2]\times[-2,2]$ dargestellt werden mit der `mesh` Funktion:

```
>> xx=-2:0.01:2; yy=xx; [x,y]=meshgrid(xx,yy);...
z=exp(-x.^2-y.^2); mesh(z);
```

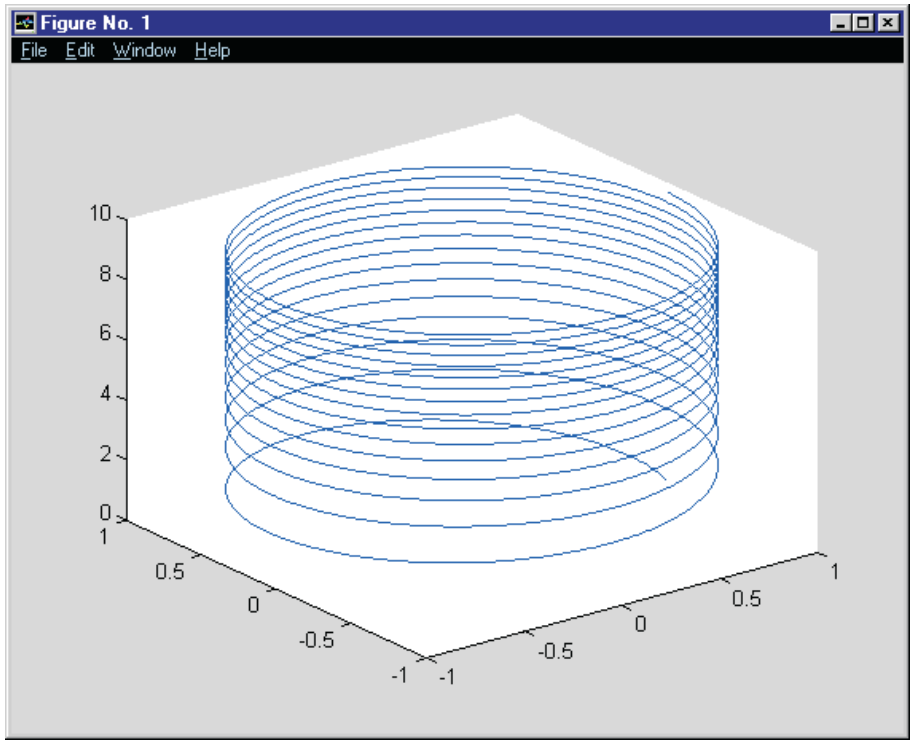
Die Funktion `meshgrid` erzeugt 2D–Gitter in x–y Ebene, was in MATLAB über 2 Gittermatrizen definiert wird. Eine Matrix beinhaltet die x–Koordinaten und die andere die y–Koordinaten.

Darstellung der Funktion $z = f(x, y) = e^{-x^2-y^2}$



Mit `plot3` können 3D–Kurven dargestellt werden:

```
>> t=.01:.01:30*pi;x=cos(t);y=sin(t);...
z=sqrt(t);plot3(x,y,z);
```



Ausgabe von Graphiken

Mit dem Befehl `print` können Graphiken in verschiedenen Formaten in Dateien oder auf dem Drucker ausgegeben werden.

Beispiel: Die folgende Anweisung speichert die aktuelle Graphik (*figure*) unter dem Namen `test.eps` im EPS (*encapsulated postscript*) – Format.

```
>> print -deps test.eps
```

Einlesen von Bildern

Matrizen können als Bilder dargestellt werden.
Beispiel: Das Bild gatlin

```
>> load gatlin
>> whos
  Name           Size          Bytes  Class
  X            480x640       2457600  double array
  caption        2x63            252  char array
  map            64x3           1536  double array
Grand total is 307518 elements using 2459388 bytes
>> caption
caption =
1964 photo from the Gatlinburg Conference on Numerical
Algebra. Wilkinson, Givens, Forsythe, Householder,
Henrici, and Bauer.
```

Das Bild gatlin