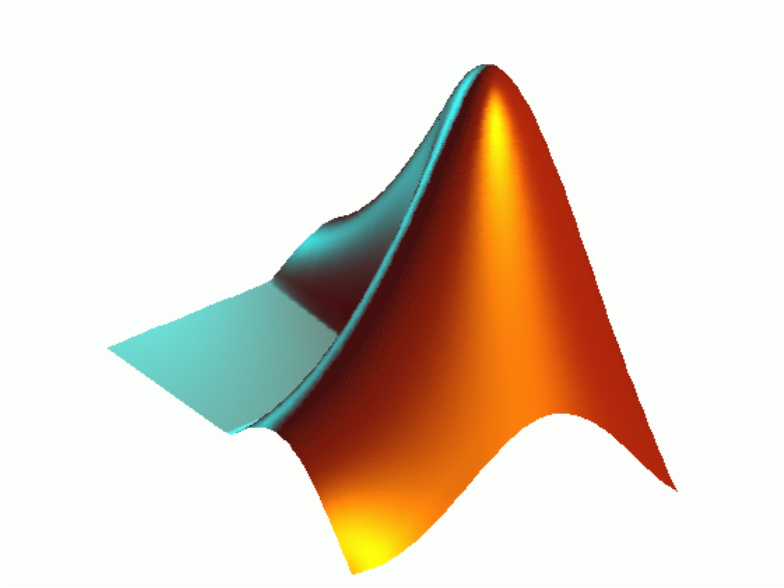
ulm university universität ulm



Wima Praktikum I

Matlab Praktikum - Tag 4

Prof. Dr. Stefan Funken,
Andreas Rupp

Institut für Numerische Mathematik

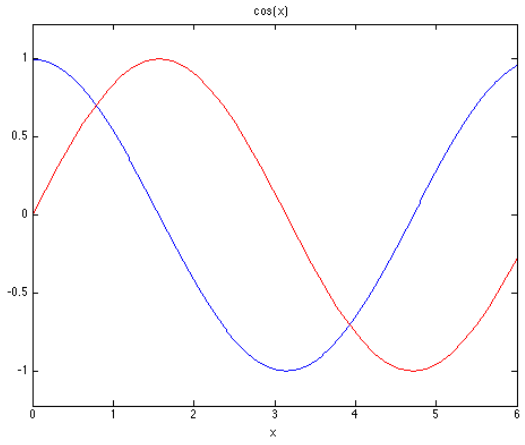
Sommersemester 2012

2D Plots

Es gibt mehrere Möglichkeiten in Matlab zweidimensionale Graphen von Funktionen zu plotten:

- Erzeugen von Punktemengen $(x, f(x))$ und Plotten mit der Funktion `plot`;
- Plotten einer Funktion mit der Funktion `ezplot`.

```
>> ezplot('cos(x)', [0, 2*pi]);
>> hold on
>> x=0:pi/20:2*pi;
>> plot(x, sin(x), 'r')
>> hold off
```



Zur Berechnung eines Graphen kann ein Gitter des Urbildbereichs erzeugt werden.

- `linspace` oder der Operator `a:b:c` erzeugt eine äquidistante Unterteilung eines Intervalls,
- `logspace` erzeugt eine logarithmische Unterteilung eines Intervalls.

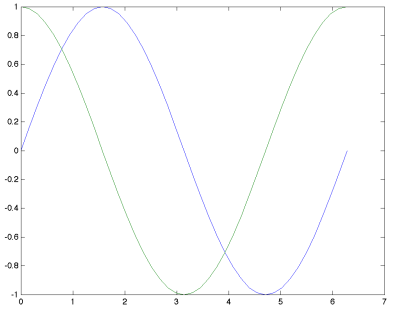
```
>> linspace(0,1,5)
ans =
         0    0.2500    0.5000    0.7500    1.0000
>> logspace(0,1,5)
ans =
    1.0000    1.7783    3.1623    5.6234   10.0000
```

2D Plots meherer Graphen

Soll mehr als ein Graph in ein Koordiantensystem geplottet werden, so kann dies auf verschiedene Arten erreicht werden:

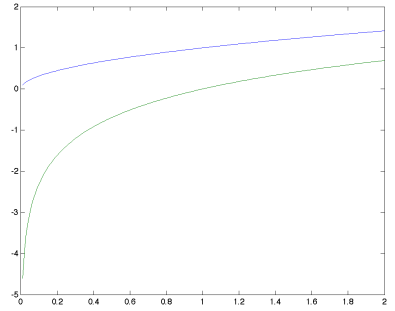
- Man erstellt einen Spaltenvektor von x-Werten, erzeugt eine Matrix mit den Funktionswerten und plottet die Spalten der Matrix:

```
>> x=[0:pi/20:2*pi]';
>> plot(x, [sin(x) cos(x)]);
```



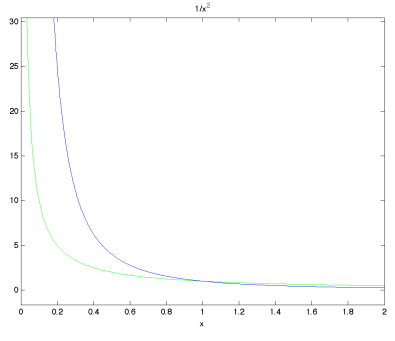
- Die Graphen werden einzeln aufgelistet:

```
>> x=[0+0.01:.01:2];
>> y=[0+0.01:.02:2];
>> plot(x, sqrt(x), y, log(y))
```



- Mit `hold` werden Graphen in einem Figure nicht durch neue Plotbefehle überschrieben. Neue Graphen werden ins Koordinatensystem eingefügt:

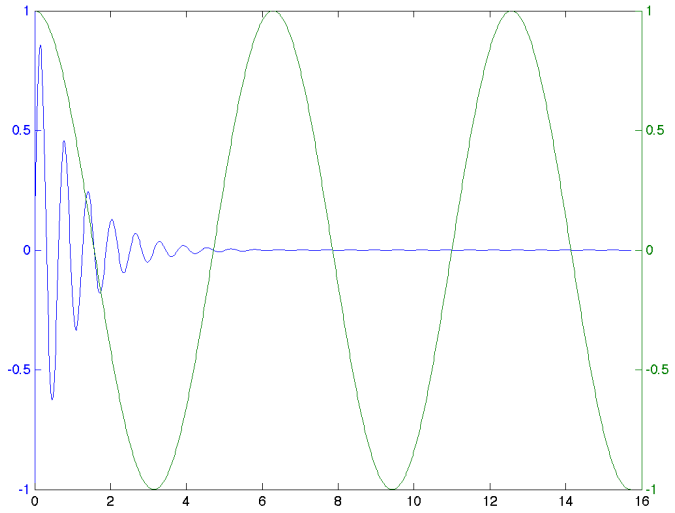
```
>> x=[0+0.01:.01:2];
>> plot(x, 1./x, 'g');
>> hold on
>> ezplot('1./x.^2', [0, 2]);
>> hold off
```



2D Plots meherer Graphen

- Sollen mehrere Graphen mit unterschiedlicher Skalierung der y-Achsen geplottet werden, so kann der Befehl `plotyy` verwendet werden:

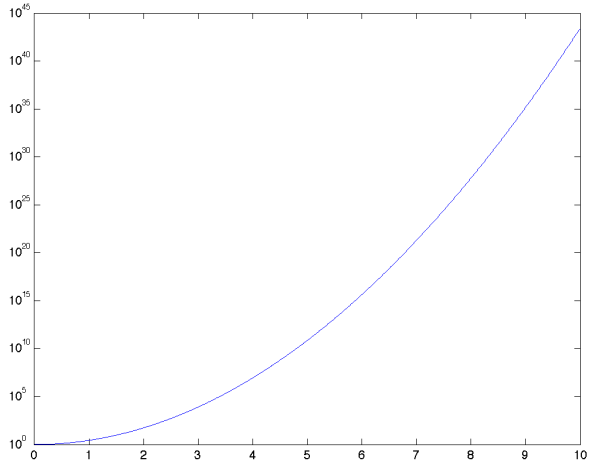
```
>> t=linspace(0, 5*pi, 1000);
>> x=exp(-t).*sin(10.*t);
>> y=cos(t);
>> plotyy(t,x,t,y);
```



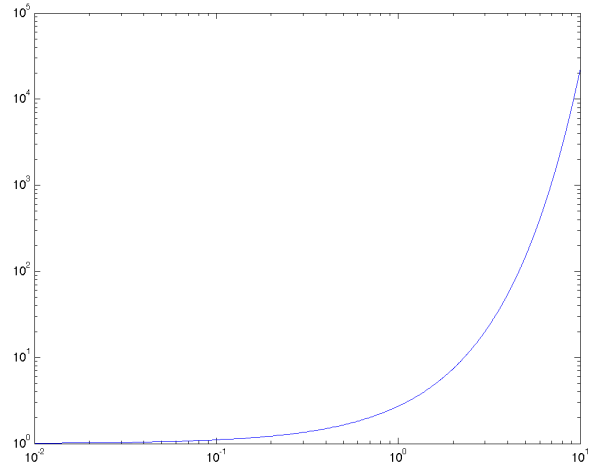
Graphen mit logarithmischer Skalierung

Oft ist es sinnvoll Graphen mit logarithmisch skalierten Achsen zu verwenden. Diese können mit `semilogx`, `semilogy` bzw. `loglog` für eine logarithmisch skalierte x-Achse, y-Achse bzw. logarithmische Skalierung beider Achsen erzeugt werden.

```
>> x=0:.01:10;
>> y=exp(x.^2);
>> semilogy(x, y)
```



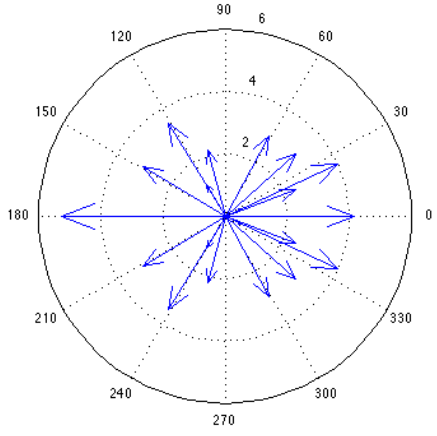

```
>> x=0:.01:10;
>> y=exp(x);
>> loglog(x,y)
```



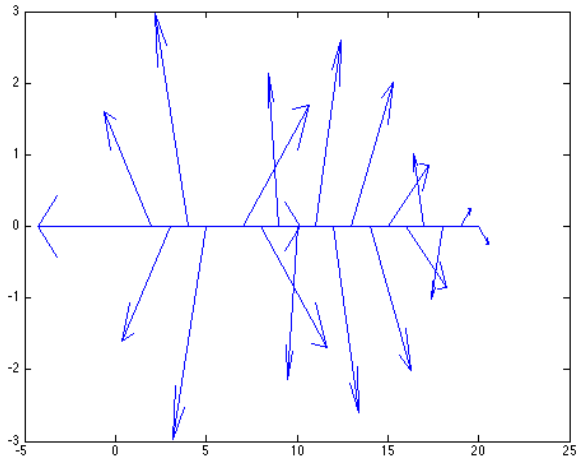
Plots komplexer Zahlen

Komplexe Zahlen lassen sich in 2D Plots veranschaulichen.

- `compass` stellt komplexe Zahlen in Polarkoordinaten dar
- `feather` stellt zweidimensionale Vektoren bzw. komplexe Zahlen als Vektoren entlang einer Geraden dar.

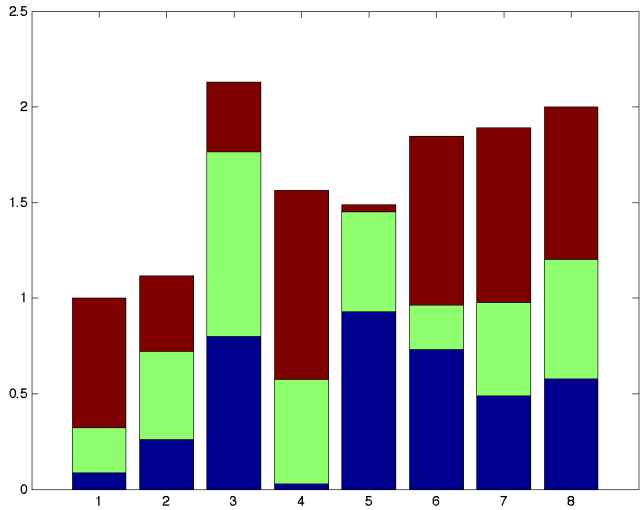


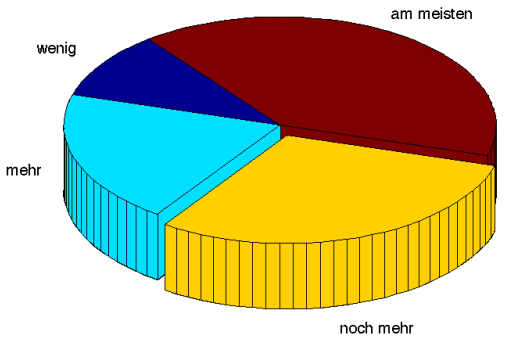

```
>> Z=eig(randn(20));
>> compass(Z)
>> feather(Z)
```



Datenanalyse

```
>> bar(rand(8,3), 'stacked')
>> Zahlen=[1 2 3 4];
>> Gewichtung={'wenig', 'mehr', 'noch mehr', 'am meisten'};
>> pie3(Zahlen, [0 0 1 0], Gewichtung)
```



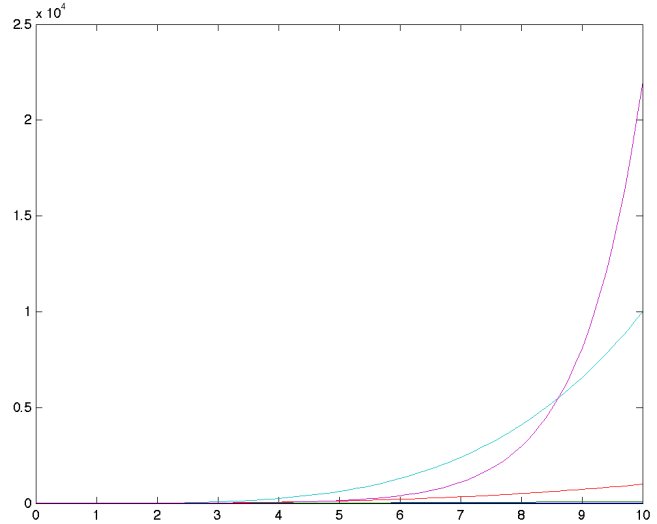


Achsengestaltung

Zur Gestaltung können bei Plots die verschiedenen Elemente der Grafik wie Achsen, Beschriftungen, Legende, Farben und Ansichtswinkel (bei 3d Plots) angepasst werden.

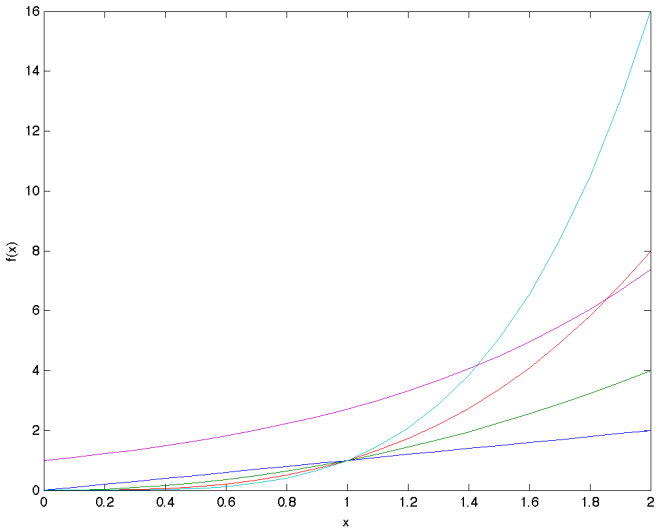
Als Beispiel nehmen wir einen Plot von verschiedenen Potenzfunktionen und der Exponentialfunktion.

```
>> x=[0:.1:10]';
>> Y=[x x.^2 x.^3 x.^4 exp(x)];
>> plot(x,Y);
```



Mit `axis` kann die Skalierung der Achsen angepasst werden. Die Beschriftung der Achsen kann mit `xlabel` und `ylabel` geändert werden.

```
>> axis([0 2 0 16])
>> xlabel('x')
>> ylabel('f(x)')
```

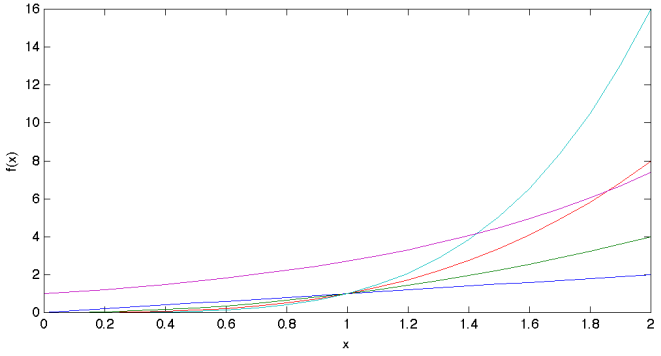


Achsengestaltung (cont'd)

Mit `pbaspect` können die Längenverhältnisse der Achsen festgelgt werden:

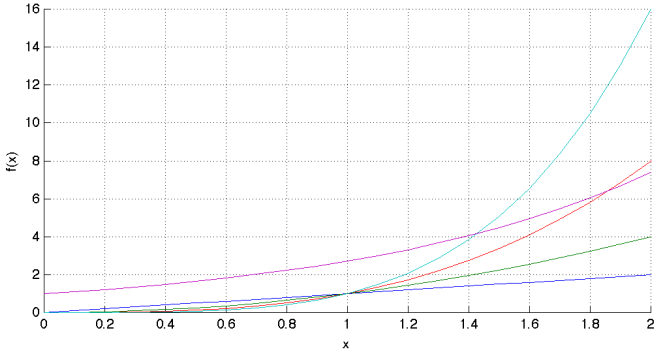
```
>> pbaspect([2 1 1])
```

Mit dem Kommando `zoom` kann desweiteren in das Bild hereingezoomt werden.



Mit `box` kann die Umrahmung des Bildes an und ausgeschaltet werden, ebenso kann mit `axis` die Achsen angezeigt oder unterdrückt werden. `grid` stellt ein Gitter hinter dem Graphen dar.

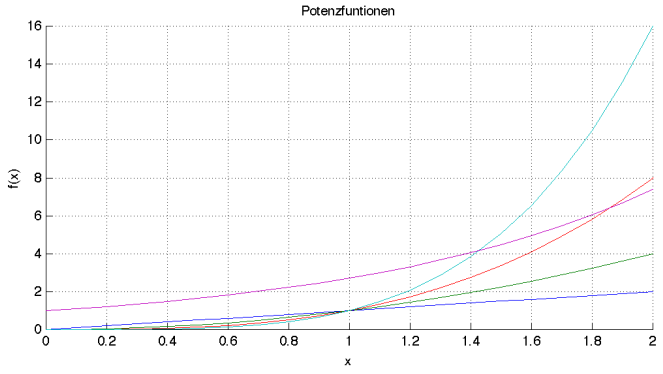
```
>> box off
>> grid on
```



Beschriftungen

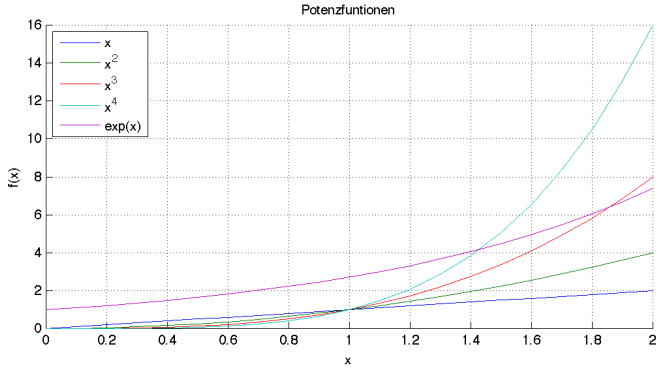
Zur schnellen Auffassung der Aussage eines Graphen sind Beschriftungen oft sehr hilfreich. Eine Überschrift zu einem Graphen kann mit `title` festgelegt werden.

```
>> title('Potenzfuntionen')
```



Hilfreich ist es auch bei Plots mit mehreren Graphen eine Legende zu haben. Diese lässt sich mit `legend` hinzufügen.

```
>> legend('x','x^2','x^3␣','x^4', ...
          'exp(x)','Location','NW')
```

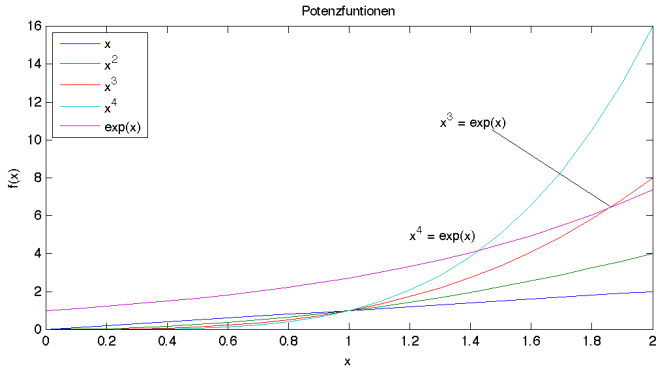


Textmarken

Besondere Stellen im Graphen können mit `text` mit einem Textfeld markiert werden.

Mit `annotation` können auch Linien, Pfeile und andere Graphikelemente eingefügt werden.

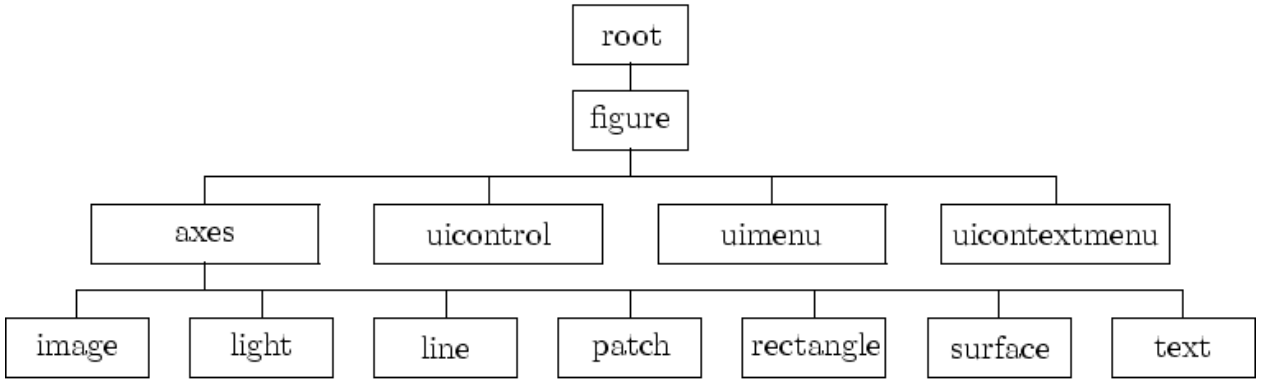
```
>> text(1.2, 5, 'x^4 = exp(x)')
>> ann=annotation('line', ...
          [0.85 0.7],[0.47 0.6]);
>> t2=text(1.3, 11, 'x^3 = exp(x)')
```



Bei der Beschriftung können im eingeschränkten Umfang LaTeXKommandos verwendet werden, die entsprechend zu formatiertem Text umgesetzt werden.

Aufbau eines Grafikfensters

Ein Grafikfenster ist hierarchisch aufgebaut.

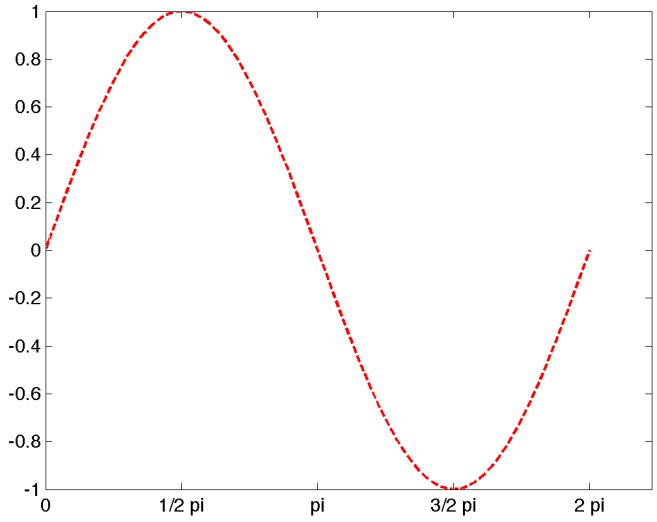


- Optionen der Grafikobjekte können mit `get` ausgelesen und mit `set` gesetzt werden. Dabei gibt `get` eine Struktur mit Objekteigenschaften zurück.
- Mit `reset` können Objekteigenschaften zurückgesetzt werden;
- Grafikhandles können mit
  - `gco` für das aktive Objekt,
  - `gcf` für das aktive Grafikfenster,
  - `gca` für das aktive Achsensystem und
  - `findobj` für ein Objekt mit vorgegebenen Objekteigenschaften

  abgerufen werden.
- Objekte können mit `delete` gelöscht werden;

Grafikeigenschaften (Beispiel)

```
>> t=linspace(0,2*pi,100);
>> h=plot(t,sin(t));

>> set(h,'LineStyle','--', ...
       'Color','r', ...
       'LineWidth',2);
>> set(gca,'FontSize',14, ...
       'XTick', ...
       [0 pi/2 pi 3/2*pi 2*pi], ...
       'XTickLabel', ...
       {'0' 'pi/2' 'pi' '3pi/2' '2pi'});
```

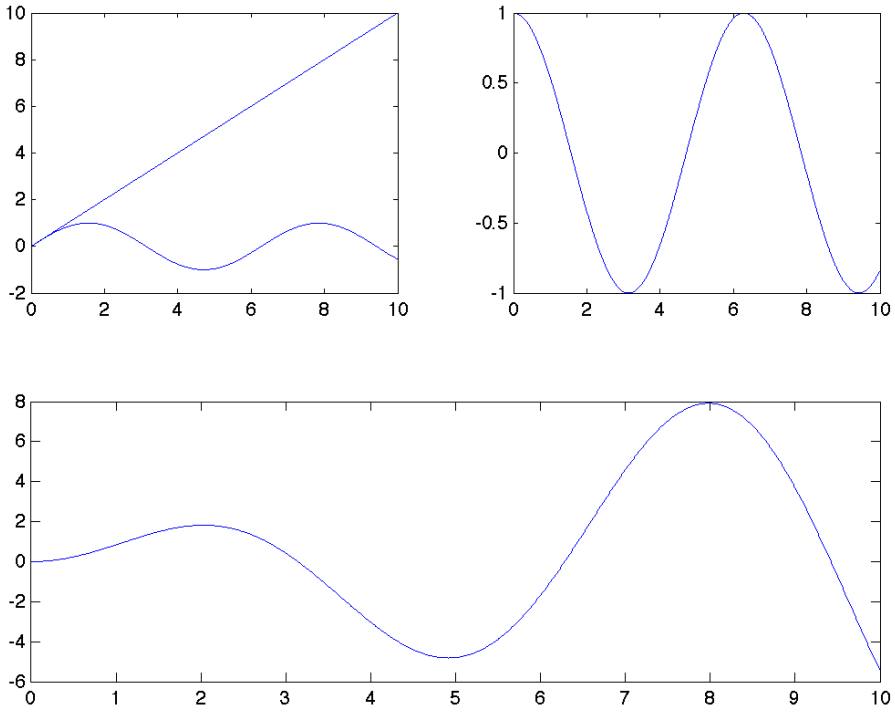


Eine weitere Möglichkeit die Grafikeigenschaften eines Objektes zu untersuchen/verändern bietet der Befehl `inspect`.

Verschiedene Graphen in einer Figure

In einem Figure Objekt können auch mehrere verschiedene Plots dargestellt werden. Dazu kann mit `subplot` der Bildbereich des Plotfensters rechteckig unterteilt werden.
Dabei kann beliebig zwischen den Teilfenstern hin und her gesprungen werden. Änderungen auf Graphiken wirken sich jeweils nur auf das aktive Teilfenster aus.

```
>> x =0:.1:10;

>> subplot(2,2,1)
>> plot(x,sin(x))

>> subplot(2,2,2)
>> plot(x,cos(x))

>> subplot(2,2,3:4)
>> plot(x,x.*sin(x))

>> subplot(2,2,1)
>> hold on
>> plot(x,x)

>> print -dpng subplot.png
```



Speichern und Drucken von Graphiken

Zum Speichern und Drucken einer Graphik steht die Funktion

- `print` ohne weitere Argumente zum drucken des aktuellen Graphikfensters mit dem Standarddrucker oder mit
- `print -d<Format> <Dateiname>` zur Ausgabe in eine Graphikdatei zur Verfügung. Gebräuchliche Grafikformate sind dabei `bmp, epsc, jpeg, pdf, png, tiff`.

Weitere nützliche Tools sind

- `orient` zum Ändern der Papierorientierung;
- `prindlg` zum Öffnen eines Dialogs mit Druckeinstellungen;
- `pagesetupdlg` zum Öffnen eines Dialogs mit Seiteneinstellungen;
- `printpreview` zum Öffnen einer Druckvorschau.

3D Plots

Dreidimensionale Objekte können auf verschiedene Arten mathematisch ausgedrückt werden, z.B.

- als Graph einer reelwertigen Funktion von zwei Variablen,
- als parametrisierte vektorwertige Funktion,
- implizit über eine Menge die gewissen Bedingungen genügt.

Eine Möglichkeit ist – wie im zweidimensionalen – vektorwertige Funktionen mit `ezplot3` darzustellen.

Andere Möglichkeiten bestehen darin 3D Punktmengen zu Plotten. Hierzu gibt es die Funktionen

- `plot3`: Plottet räumliche Punkte und zeichnet Verbindungslinien zwischen den Punkten
- `surf`: Zeichnet eine Oberfläche durch eine Menge von 3D Punkten mit und ohne Höhenlinien.
- `mesh`: Zeichnet Punkte und verbindet die Punkte durch ein Gitter mit und ohne Höhenlinien
- `waterfall`: Zeichnet Punkte und verbindet die Punkte in einer Koordinatenrichtung.

Die Funktionen `mesh` und `surf` gibt es jeweils noch als Befehle mit angehängtem `c` bzw. `z`, bei denen zusätzlich Höhenlinien oder vertikale Linien zu den Punkten gezeichnet werden.

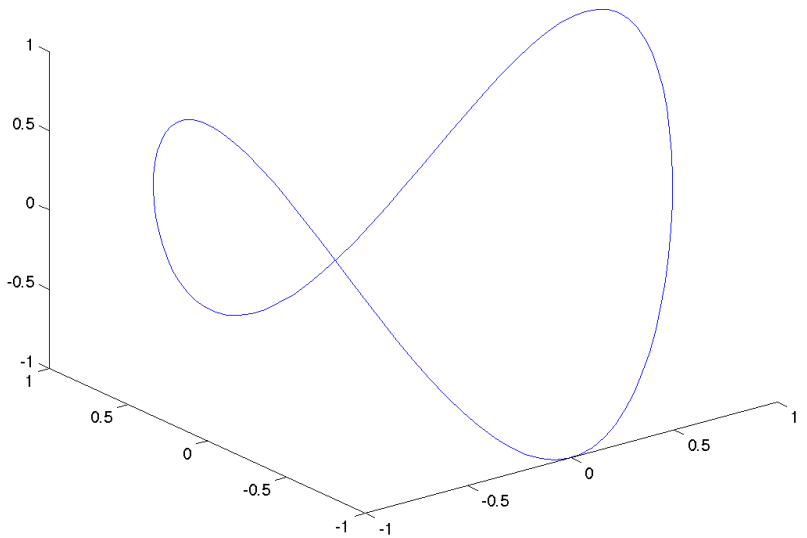
3D Plot Beispiel: Parametrisierte Kurve

Zum Plotten von räumlichen Kurven $\gamma : \boldsymbol{R} \longrightarrow \boldsymbol{R}^3$ wird eine Unterteilung für den Kurvenparameter berechnet, die Koordinaten zu den Bogenparametern berechnet und die Werte mit der Funktion `plot3` dargestellt.

```
>> t=linspace(0,2*pi);
>> plot3(cos(t),sin(t),sin(2*t));
```

oder alternativ:

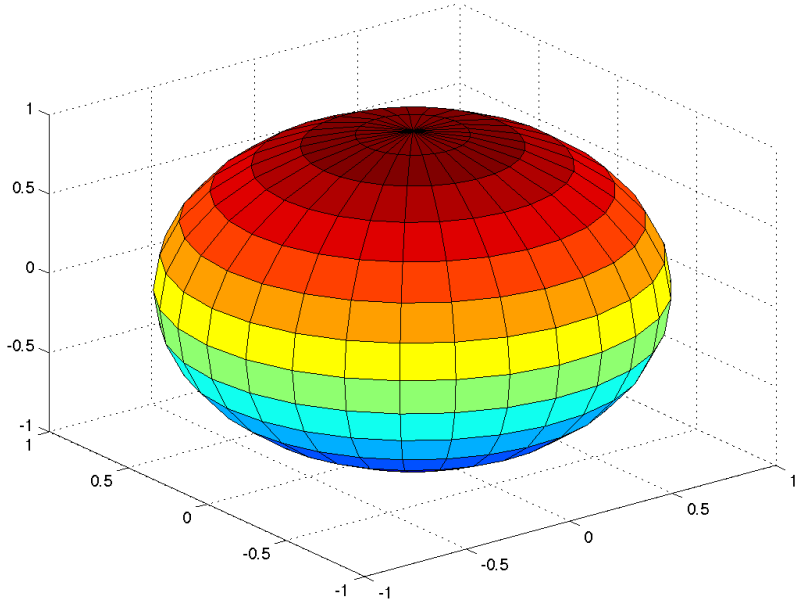
```
>> ezplot3(@cos,@sin,...
           @(t)sin(2*t),...
           [0,2*pi]);
```



3D Plot Beispiel: Parametrisierte Fläche

Analog zur räumlichen Kurve kann auch eine parametrisierte Oberfläche $s : \boldsymbol{R}^2 \longrightarrow \boldsymbol{R}^3$ dargestellt werden. Hierbei wird mit `meshgrid` ein Gitter erzeugt, die Koordinaten auf dem Gitter berechnet und die Oberfläche mit `surf` dargestellt.

```
>> [s,t]=meshgrid([-pi:pi/30:pi]);
>> x=cos(s).*sin(t);
>> y=sin(s).*sin(t);
>> z=cos(t);
>> surf(x,y,z);
```

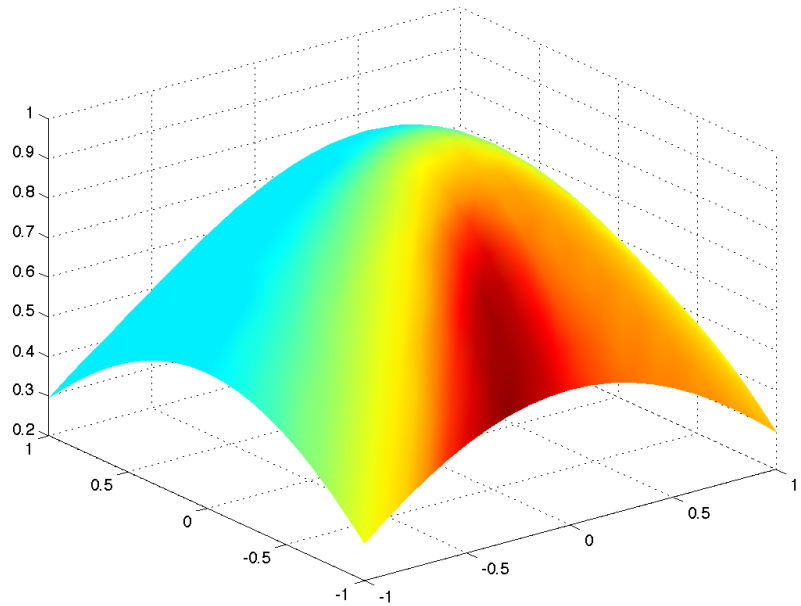


3D Plot Beispiel: Beleuchteter Graph einer Funktion

Um Funktionen $f : \boldsymbol{R}^2 \longrightarrow \boldsymbol{R}$ darzustellen kann wiederum die Funktion `surf` verwendet werden. Im folgenden Beispiel wurde der Plotbefehl `surfl` verwendet, bei dem der Graph als schattierte Oberfläche dargestellt wird.

Mit der Funktion `shading` kann die Art der Schattierung bei 3D Plotts festgelegt werden.

```
>> [x,y]=meshgrid(-1:.1:1);
>> z=cos(x).*cos(y);
>> surfl(x,y,z);
>> shading interp;
```

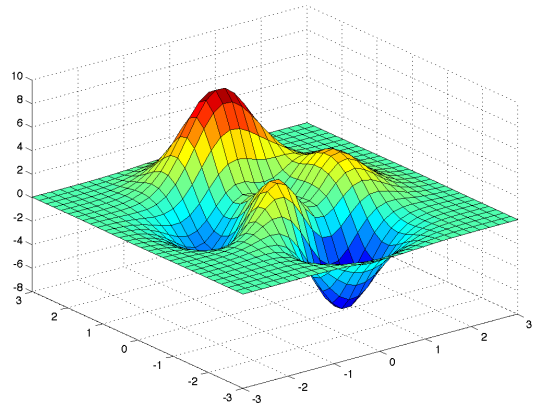


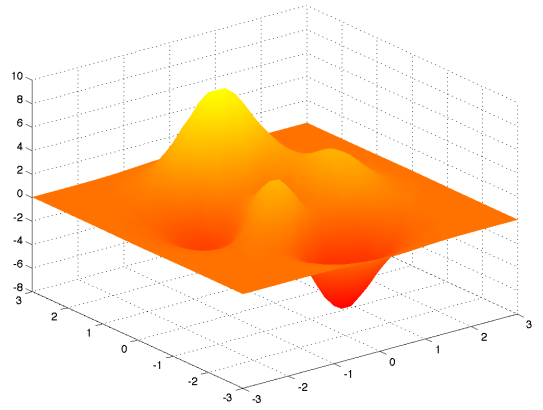
Farbgestaltung von 3D Plots

Bei Plots in 3D ist oft die farbliche Gestaltung wichtig um Details zu erkennen. Zur Änderung der Farbgestaltung können z.B. folgende Funktionen benutzt werden:

- `shading`: legt die Farbgestaltung fest
- `colormap`: legt die Farbcodierung fest
- `spinmap`: Animiert die Farbgebung eines Graphen

```
>> [x,y,z]=peaks(30);
>> surf(x,y,z);

>> figure;
>> surf(x,y,z);
>> shading interp;
>> colormap autumn;
```

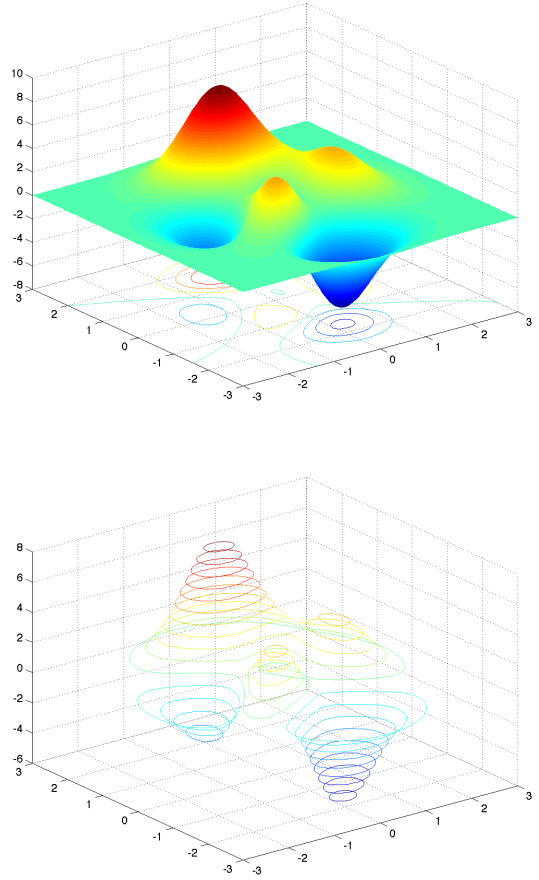




Schnittbilder

Eine weitere Möglichkeit reelwertigen Funktionen $f : \boldsymbol{R}^2 \longrightarrow \boldsymbol{R}$ darzustellen besteht darin, Schnitte der Funktionen darzustellen, z.B. Höhenlinien bzw. farbcodierte Schnitte. Höhenlinenen können z.B. mit der Funktion `surfc` zusammen mit der Oberfläche dargestellt. Sollen nur Höhenlinien zu bestimmten Niveaus dargestellt werden, so kann die Funktion `contour3` verwendet werden.

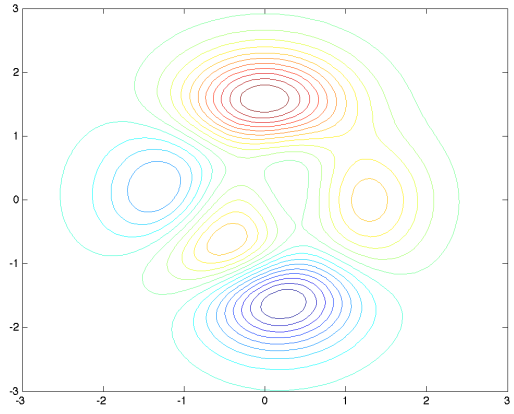
```
>> [x,y,z]=peaks(100);
>> surfc(x,y,z);

>> figure;
>> contour3(x,y,z,20);
```

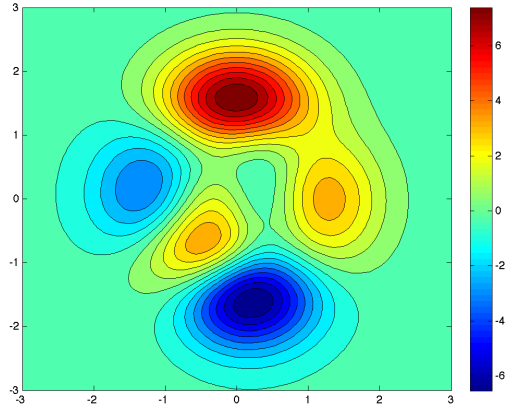


2D Schnittbilder

Sollen die Höhenlinien als 2D Bild dargestellt werden (Kartendarstellung), so können die Funktionen `contour` oder `contourf` verwendet werden.
Für diese Darstellung ist es meist sinnvoll mit `colorbar` die verwendete Farbcodierung mit anzugeben.

```
>> [x,y,z]=peaks(100);
>> contour(x,y,z,20);

>> figure;
>> contourf(x,y,z,20);
>> colorbar
```

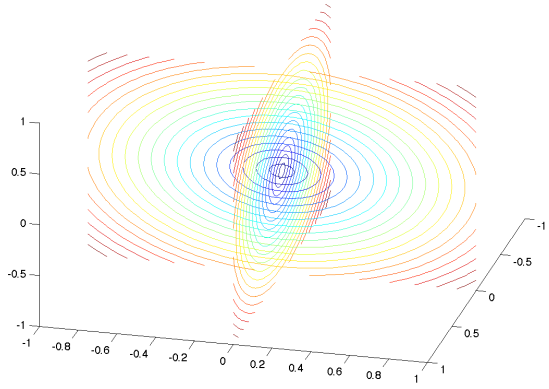


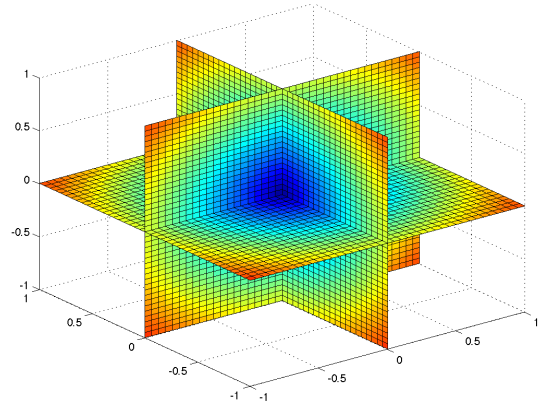


Höhenlinien auf Schnittebenen: Beispiel

Sollen Funktionen $f : \boldsymbol{R}^3 \longrightarrow \boldsymbol{R}$ dargestellt werden, so geht dies nur indem man sich Niveaus auf Flächen anschaut.
Die Funktionen `slice` und `countourslice` erzeugt eine solche Darstellung. Dazu werden auf einem 3D-Gitter die Funktionswerte berechnet und zusammen mit den Achsabschnitten der Niveauflächen an die Funktion übergeben.
Der Ansichtswinkel kann, wie bei den anderen Funktionen zur räumlichen Darstellung mit dem Befehl `view` festgelegt werden.

```
>> [x,y,z]=meshgrid(-1:.05:1);
>> v=sqrt(x.^2+y.^2+z.^2);
>> contourslice(x,y,z,v,0,0,[],20);
>> view([4,1,3]);

>> figure;
>> slice(x,y,z,v,0,0,0);
```



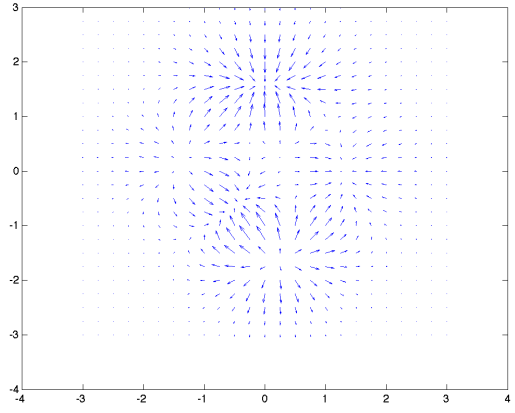


Vektorfelder in 2D

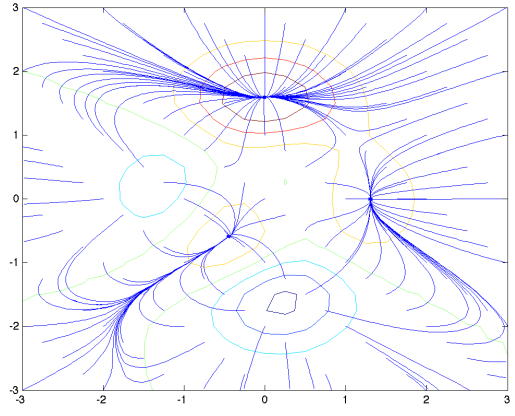
Vektorfelder $f : \boldsymbol{R}^2 \longrightarrow \boldsymbol{R}^2$ können als ebene Vektorfelder oder mit Hilfe von Stromlinien dargestellt werden. Für die Darstellung als Vektorfeld wird auf einem Gitter die Funktion ausgewertet und mit der Funktion `quiver` dargestellt.

Stromlinien können mit der Funktion `streamline` unter Angabe von Startpunkten gezeichnet werden.

```
>> [x,y,z]=peaks(25);
>> hx=x(1,2)-x(1,1);
>> hy=y(2,1)-y(1,1);
>> [gx,gy]=gradient(z,hx,hy);
>> quiver(x,y,gx,gy);
```



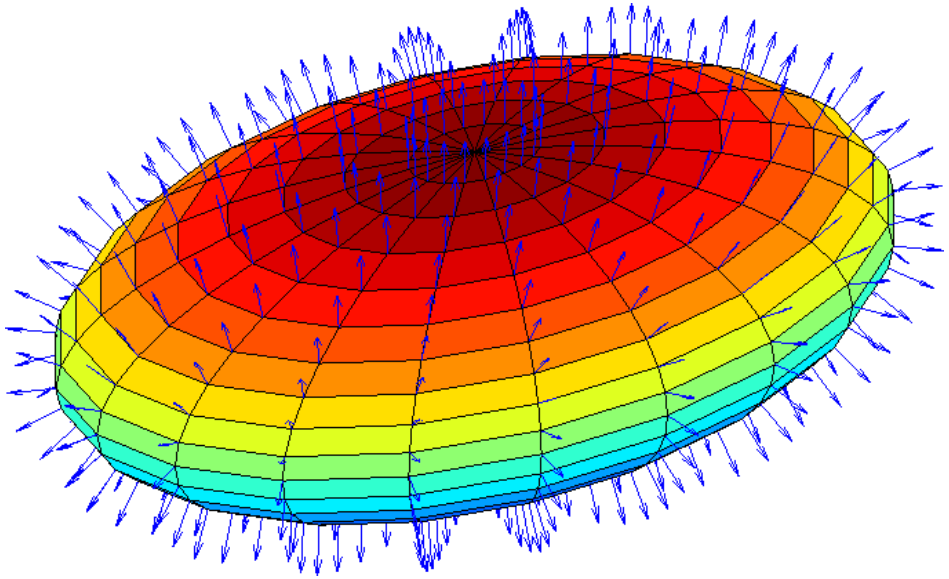
```
>> xs=x(1:4:end);
>> ys=y(1:4:end);
>> contour(x,y,z);
>> streamline(x,y,gx,gy,xs,ys);
```



Vektorfelder in 3D

Dreidimensionale Vektorfelder $f : \boldsymbol{R}^2 \longrightarrow \boldsymbol{R}^3$ können analog zu 2D Vektorfeldern mit `quiver3` dargestellt werden. Somit kann man z.B. die Normalen einer Oberfläche darstellen.

```
>> [x,y,z]=ellipsoid(0,0,0,3,2,1,20);
>> [ny,nx,nz]=surfnorm(x,y,z);
>> quiver3(x,y,z,nx,ny,nz);
>> hold on;
>> surf(x,y,z);
>> axis equal;
```



Animationen

Um Animationen zu erstellen, werden verschiedene Plots hintereinander "gehängt", d.h. nach dem Erzeugen eines Plots wird das Bild mit dem Befehl `getframe` zu einer Struktur hinzugefügt. Anschließend können die gespeicherten Bilder mit `movie` animiert werden.

```
t=[0:.1:10];
x=[0:.01:2*pi];
j=1;
for s=t
    y=sin(s+x);
    plot(x,y);
    F(j)=getframe;
    j=j+1;
end
movie(F,1)
movie2avi(F,'sin.avi');
```

Mit `movie2avi` kann die Animation als avi-Datei gespeichert werden.