

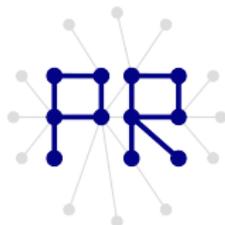
# Einführung in die Informatik II

## III.3 Visualisierung

Prof. Dr.-Ing. Marcin Grzegorzek<sup>1</sup>

Forschungsgruppe für Mustererkennung  
[www.pr.informatik.uni-siegen.de](http://www.pr.informatik.uni-siegen.de)

Institut für Bildinformatik  
Universität Siegen



---

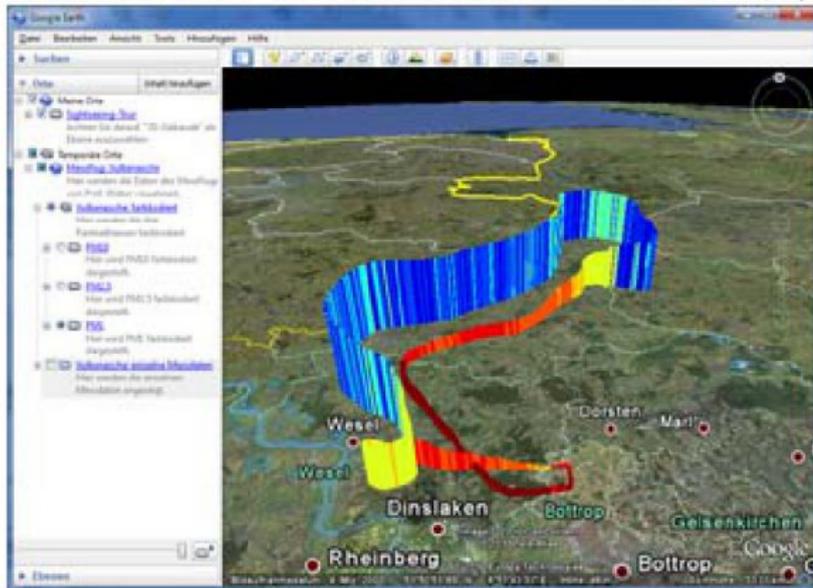
<sup>1</sup>Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

# Inhaltsverzeichnis

- I. MATLAB-Einführung
- II. Algorithmen
- III. MATLAB-Fortsetzung
  - 1. Internet und Werkzeuge
  - 2. Dateien
  - 3. **Visualisierung**
  - 4. Visualisierung von 3D-Daten
  - 5. Optimierung

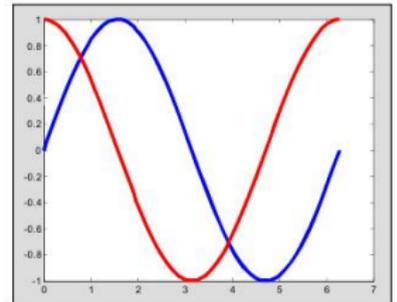
# Beispiel: Vulkanaschemessung

- Visualisierung mit google earth
- Datenbasis von Prof. Dr. rer. nat. Konradin Weber, FH-D



# Mehrere Kurven in einem Bild

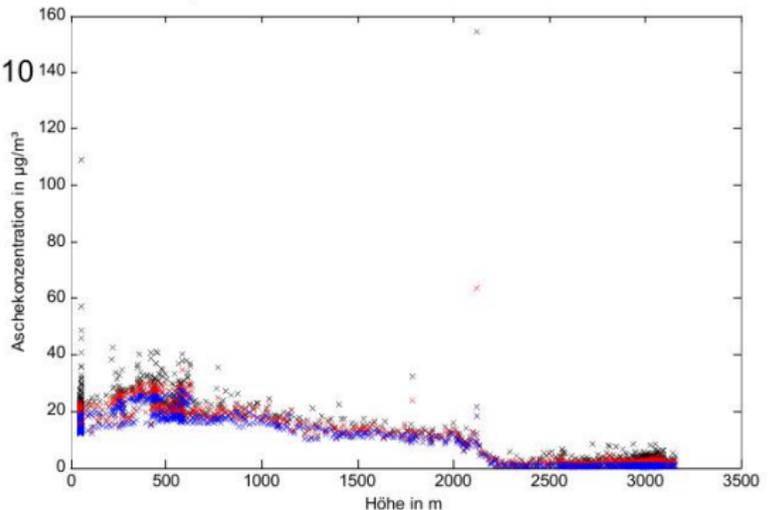
- Jeder neue `plot`-Befehl löscht den alten Plot:
  - `x=linspace(0,2*pi,100);`
  - `plot(x,sin(x)); % Sinuskurve erscheint`
  - `plot(x,cos(x)); % Sinuskurve verschwindet,  
% Cosinuskurve erscheint`
- Der Befehl `hold on` sorgt dafür, dass alle bestehenden Plots erhalten bleiben:
  - `plot(x,sin(x));`
  - `hold on;`
  - `plot(x,cos(x));`  
    % Beide Kurven sichtbar
- Löschen aller Plots mit
  - `clf % clear figure`



# Beispiel: Vulkanaschekonzentration

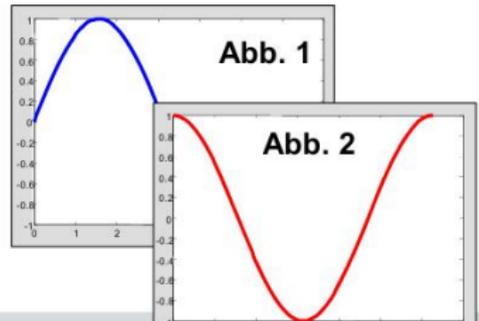
- Datenbasis: Zusammengeführte Sensordaten in CSV-Datei.
- PM10: Konzentration der Partikel in ppm ( $\mu\text{g}$  Asche pro  $\text{m}^3$  Luft) von Partikeln  $<10\mu\text{m}$

- Schwarz: PM10
- Rot: PM2,5
- Blau: PM1,0



# Erstellen mehrerer Bilder

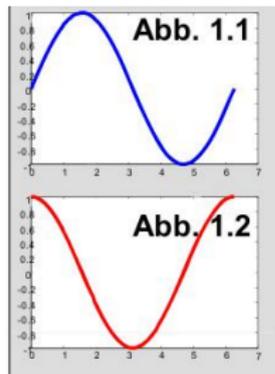
- Kurven können in verschiedene Abbildungen (figures) gezeichnet werden.
- Die Figur wird mit dem **figure**-Befehl angewählt:
  - `x=linspace(0,2*pi,100);`
  - `figure(1);`      `% Abbildung 1 ist aktiv`
  - `plot(x,sin(x));`
  - `figure(2);`      `% Abbildung 2 ist aktiv`
  - `plot(x,cos(x));`
- Die jeweils **aktive** Abbildung ist die zuletzt gewählte.
- Plotbefehle (`plot`, `hold on`, `clf`, usw.) beziehen sich immer auf die gerade aktive Abbildung.



# Unterplots in einem Bild

- Innerhalb einer Abbildung können mehrere Unterabbildungen gezeichnet werden.
- Der `subplot`-Befehl wählt die Zahl der Unterabbildungen und zugleich die aktive Unterabbildung:

- `x=linspace(0,2*pi,100);`
- `figure(1);`      % **Abbildung 1 ist aktiv**
- `subplot(2,1,1);` % **2 Unterplots in 2 Zeilen**  
% **und 1 Spalte.**  
% **Anwahl des 1. Unterplots**
- `plot(x,sin(x));`
- `subplot(2,1,2);` % **Anwahl des 2. Unterplots**
- `plot(x,cos(x));`

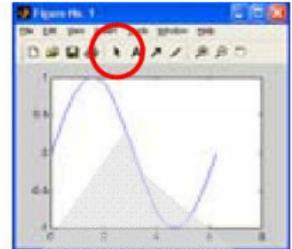


# Achsen, Gitter und Beschriftungen

- Die Gestaltung von Abbildungen kann mit folgenden Befehlen beeinflusst werden:
  - `axis equal`           % gleiche Achsenskalierung
  - `axis off`             % keine Achsen zeichnen
  - `grid on`             % Koordinatengitter zeichnen
  - `title ('Titel')`   % Titel der Grafik
  - `xlabel ('X')`       % X-Achsen-Beschriftung
  - `ylabel ('Y')`       % Y-Achsen-Beschriftung
- Griechische Buchstaben und Indizes sind mit `\_ ^` möglich
  - `xlabel ('\phi')`   % kleines phi
  - `ylabel ('\Phi')`   % großes phi
  - `title ('a^2+x_1+x_2^2')`

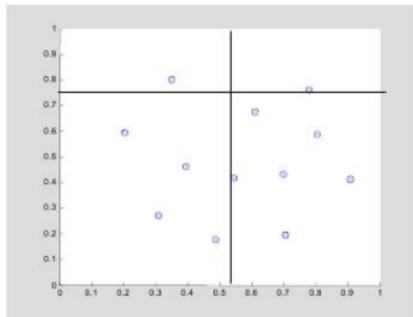
# Editieren von Plots

- Ploteditor einschalten über das Pfeilsymbol
- Veränderbare Attribute
  - Linien
    - Farbe
    - Strichstärke
    - Art
  - Marker
    - Farbe
    - Größe
    - Art
  - Achsenbeschriftung
    - Farbe
    - Font
    - Größe
  - Achsenunterteilung
    - Anzahl
    - Skala (lin/log)
    - Gitter
- Alle Attribute können auch per Programm eingestellt werden (fortgeschrittener Stoff)
- Grafiken können zur Weiterverarbeitung gespeichert werden



# ginput

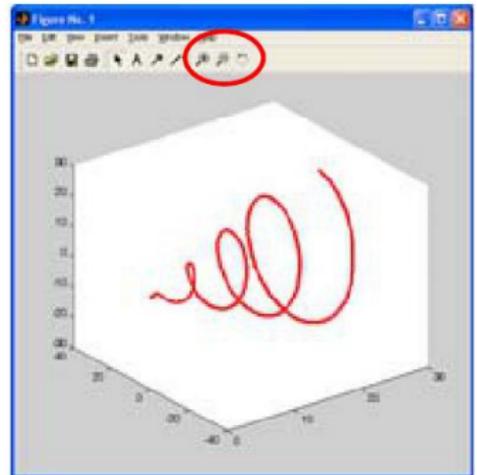
- In Abbildungen können Punkte interaktiv mit der Maus angeklickt und die Koordinaten abgefragt werden:
  - `[x,y,b]=ginput(n)`  
liefert n Punkte mit (x,y)-Koordinaten  
und gedrückter Maustaste 1 (Links), 2 (Mitte), 3 (Rechts)
- Beispiel: Zeichnen von Punkten mit der linken Maustaste, solange bis eine andere Maustaste gedrückt wird:
  - `figure(1);`  
`hold on;`  
`[x,y,b]=ginput(1);`  
`while b==1`  
    `plot(x,y,'o');`  
    `[x,y,b]=ginput(1);`  
`end;`





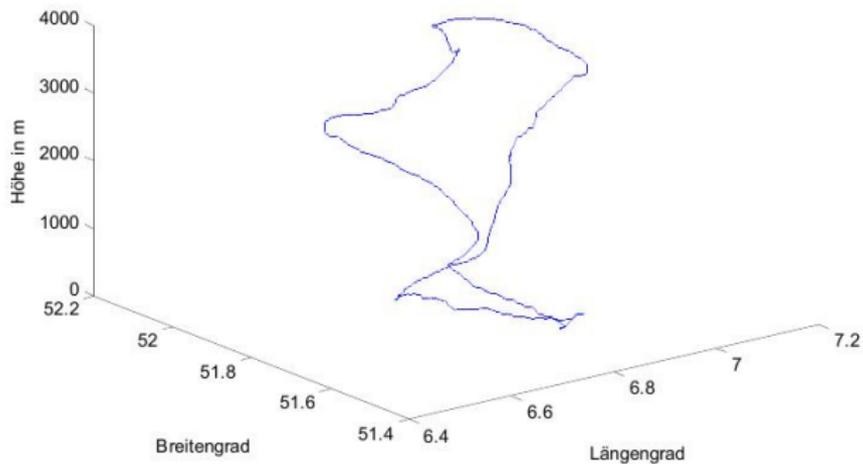
# 3-dimensionale Kurven

- `plot3` zeichnet eine Kurve in (x,y,z)-Koordinaten
  - `x=linspace(0,8*pi,400);`
  - `plot3(x,x.*sin(x),x.*cos(x),'r');`
- `view` verändert den Blickwinkel
  - `view(LG,BG);`
    - % Längen-
    - % und Breitengrad
- Mit dem Rotier-Tool auf der Menüleiste kann der Blickwinkel interaktiv verändert werden
- Mit den Zoom-in und Zoom-out-Tools kann man vergrößern oder verkleinern



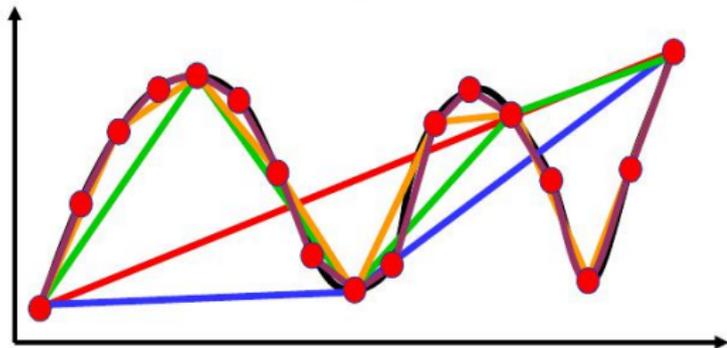
# Beispiel: Messflug Vulkanasche

- `plot3(l,b,h)`
- `comet3(l,b,h)`



# Abtastproblem beim Plotten

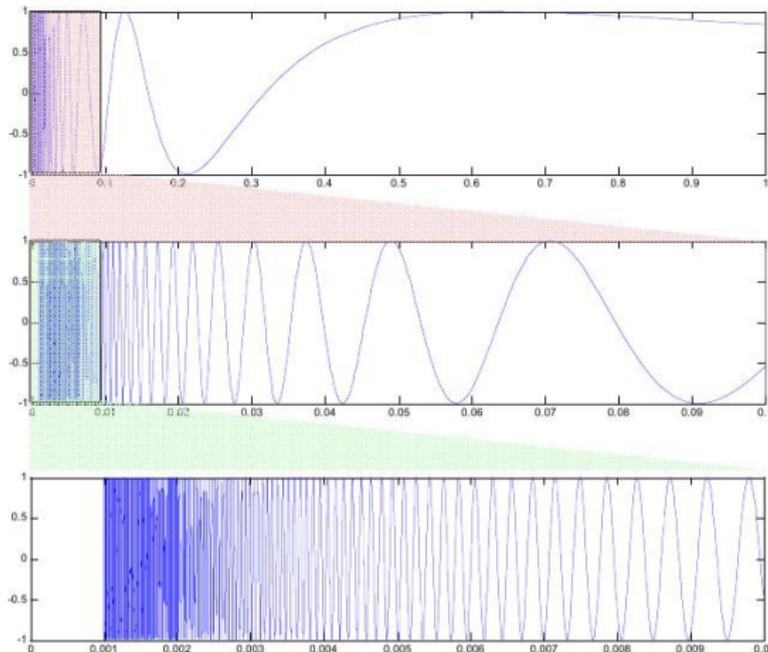
- Bei zu grober Abtastung der Funktionswerte wird die Funktion falsch dargestellt



- Moderne Zeichenalgorithmen steuern ihre Abtastschrittweite selbst (adaptiv)
- Einen 100 % zuverlässigen Zeichenalgorithmus gibt es nicht !

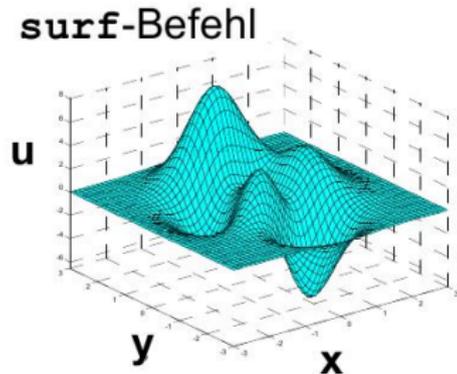
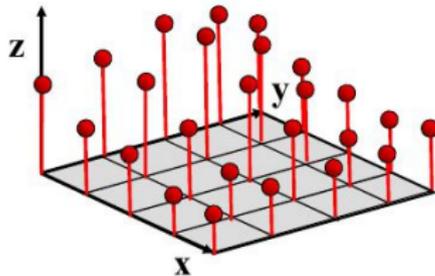
# Beispiel: $\sin(1/x)$

- Die Funktion schwingt bei Annäherung an 0 unendlich oft hin und her



# Flächenplots

- Flächen werden mit dem `surf`-Befehl gezeichnet. Dazu benötigt man:
  - Zwei Vektoren mit x- und y-Koordinaten für die Achsenbeschriftung
  - Eine Matrix mit Werten für jeden (x,y)-Punkt des davon aufgespannten Gitters

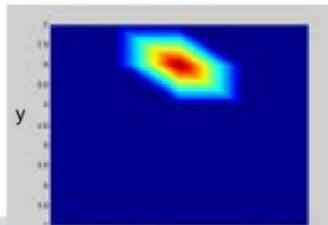


# Beispiel für den surf-Befehl

- Der **surf**-Befehl benötigt 3 Argumente:
  - Vektor der x-Koordinaten des Gitters
  - Vektor der y-Koordinaten des Gitters
  - Matrix der entsprechenden z-Werte
- Beispiel:
  - `xScale=[0 1 2 3 4 ];`
  - `yScale=[2 3 4 5 6 7];`
  - `zData=zeros(6,5);`
  - `zData(2,3)=1;`
  - `surf(xScale,yScale,zData);`
  - `shading interp;`  
% Flächen mit Farbverlauf

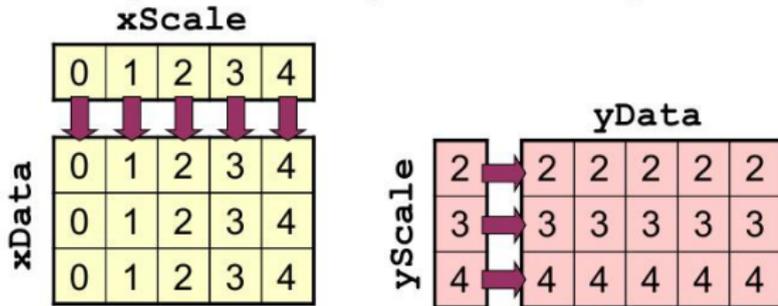
	0	1	2	3	4
2	0	0	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0

zData



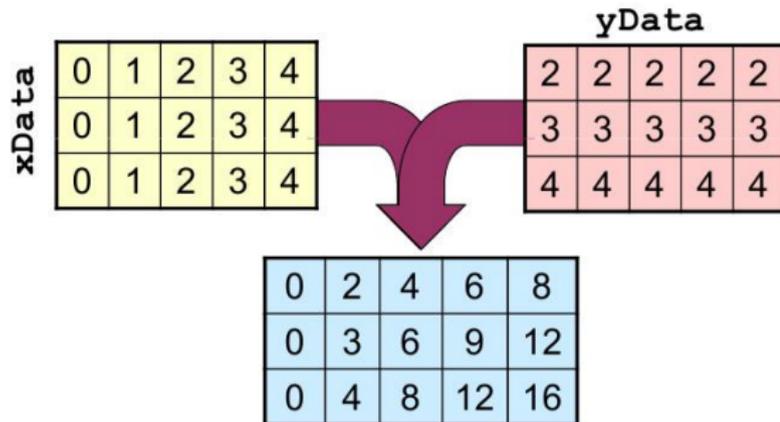
# Berechnen von x,y-Wertematrizen

- Im eindimensionalen Fall funktioniert
  - `plot(xScale, sin(xScale))` ;
- Im zweidimensionalen Fall funktioniert das nicht
  - `surf(xScale, yScale, xScale.*yScale)` ;  
ist keine Matrix !
- Der `meshgrid`-Befehl erzeugt Matrizen aus x- und y-Werten:
  - `[xData, yData]=meshgrid(xScale, yScale)` ;



# Elementweise Operation auf x,y-Wertematrizen

- Mit x,y-Wertematrizen kann man elementweise rechnen



`zData=xData.*yData`

# Beispiel für den surf-Befehl

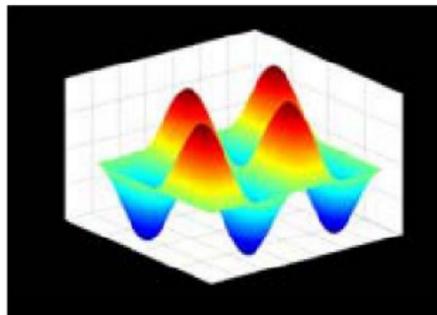
- Beispiel: Zeichne den Graphen der Funktion

$$z = \sin(x) \cdot \sin(y)$$

über

$$0 \leq x \leq 4\pi$$
$$0 \leq y \leq 2\pi$$

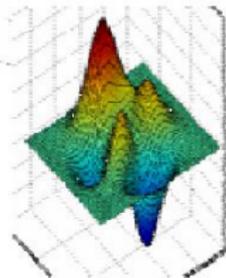
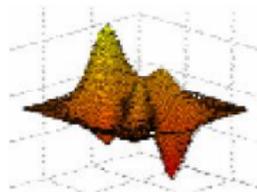
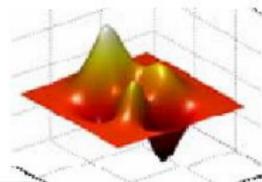
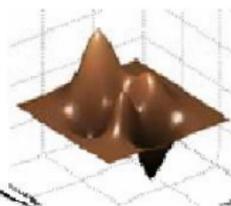
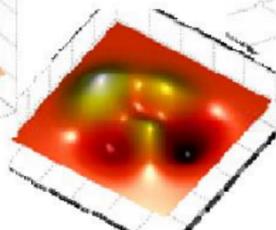
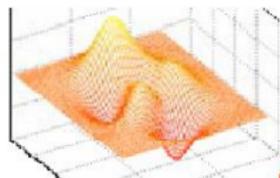
- `xScale=linspace(0,4*pi,200);`
- `yScale=linspace(0,2*pi,200);`
- `[xData,yData]=meshgrid(xScale,yScale);`
- `zData=sin(xData).*sin(yData);`
- `surf(xScale,yScale,zData);`
- `shading interp`



# Computergraphische Attribute

Für jede 3D-Grafik einstellbar:

- Farbe
- Blickwinkel
- Transparenz
- Projektion
- Beleuchtung
- Distanz
- Glättung



# Daten des Motors

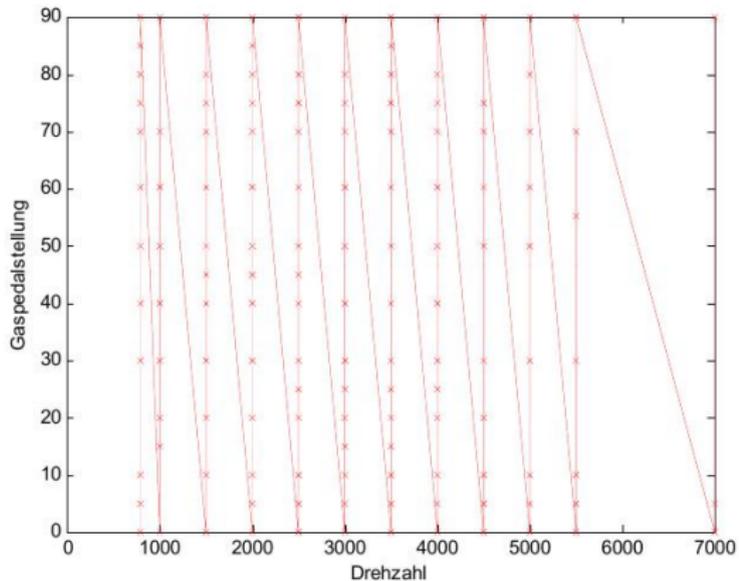
% Kennlinienfeld eines VW-Motors (1.6l, 75 PS)

VW\_Data=...

[...]

800.00	0.00	0.00	0.00	;...
800.00	5.00	0.00	0.00	;...
800.00	10.00	15.16	600.00	;...
800.00	85.00	83.77	290.00	;...
800.00	90.00	89.14	300.00	;...
1000.00	0.00	0.00	0.00	;...
1000.00	15.00	0.00	0.00	;...
1000.00	20.00	13.70	600.01	;...

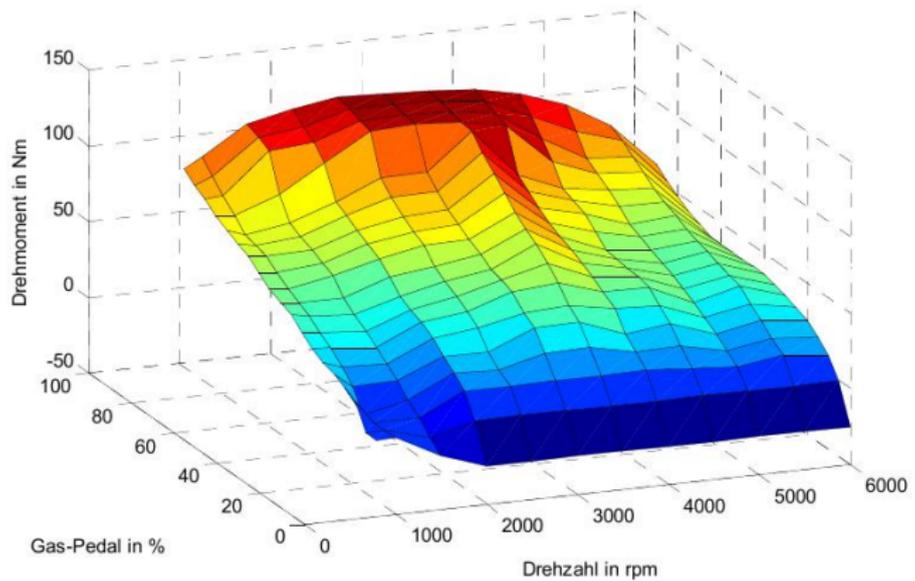
# Problem: nicht vollständige Daten



## Lösung: Interpolation fehlender Daten

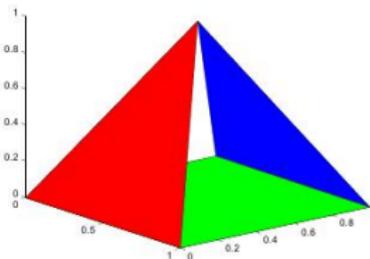
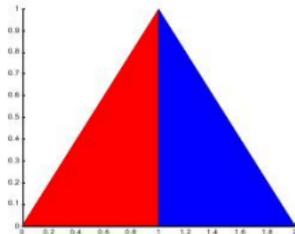
- `XGrid=[800; 1000; 1500; 2000; 2500; 3000; 3500; 4000; 4500; 5000; 5500; 6000];`
- `YGrid=0:5:90;`
- `Drehmoment=griddata(VW_Data(:,1),VW_Data(:,2),VW_Data(:,3),XGrid,YGrid);`
- `surf(XGrid,YGrid,Drehmoment)`
  
- **Der `griddata()`-Befehl**
- **X,Y,Z sind die Vektoren mit den Messdaten**
- **x,y sind Vektoren mit den Positionen zum Interpolieren**
- `M=griddata(X,Y,Z,x,y);`
- `surf(x,y,M)`

# Beispiel: Motorkennfeld



# Manueller Aufbau einer 3D-Grafik

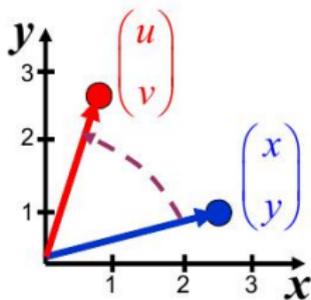
- `patch`-Befehl zeichnet farbig gefüllte Polygone in 2D und 3D
- 2D-Beispiel (`hold on`):
  - `patch([0,1,1],[0,0,1], 'r');`
  - `patch([1,2,1],[0,0,1], 'b');`
- 3D-Beispiel (`hold on`):
  - `patch([0,1,0.5],[0,0,0.5],[0,0,1], 'r');`
  - `patch([0,1,0.5],[1,1,0.5],[0,0,1], 'b');`
  - `patch([0,0,1,1],[0,1,1,0],[0,0,0,0], 'g');`



# Beispiel: Geometrie in der Ebene

- Die meisten geometrischen Abbildungen der Ebene können mit Hilfe von Matrizen dargestellt werden:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



- Beispiele :

- Streckung entlang der x-Achse  $\begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$

- Spiegelung an der y-Achse  $\begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$

- Drehung um den Winkel  $\alpha$   $\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$

- Scherung parallel zur x-Achse  $\begin{pmatrix} 0 & a \\ 1 & 1 \end{pmatrix}$

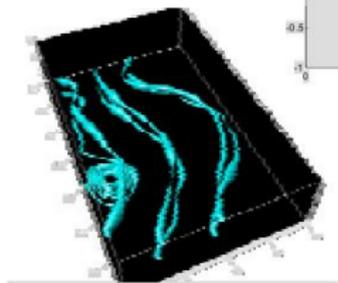
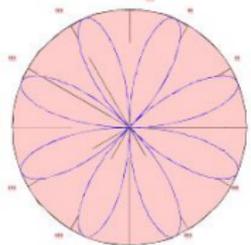
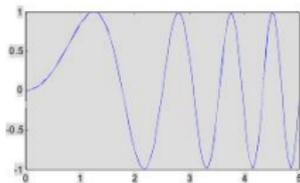
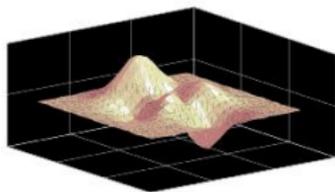
# Vereinfachte Zeichenbefehle

- Einige MATLAB-Funktionen dienen dem schnellen Zeichnen von Funktionsgraphen
  - `ezplot`        zeichnet 2D-Funktionsgraphen
  - `ezsurf`        zeichnet 3D-Funktionsgraphen (Flächen)
  - `ezplot3`        zeichnet 3D-Kurven
- Beispiele (Beachte: kein `.*`, `./`, `.^` erforderlich):
  - `>> ezplot`  
    `('cos(x)*sin(x)', [0,2*pi]);`
  - `>> ezsurf`  
    `('cos(x)*sin(y)', [0,2*pi], [0,2*pi]);`
  - `>> ezplot3`  
    `('x', 'x*sin(x)', 'x*cos(x)', [0,10*pi]);`
- Weitere Befehle (vgl. nachfolgende Folien):

<code>ezpolar</code>	<code>ezcontour</code>	<code>ezmesh</code>
<code>ezsurf</code>	<code>ezcontourf</code>	<code>ezmeshc</code>

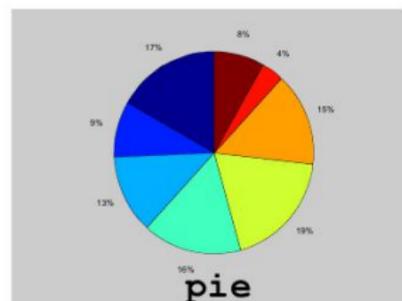
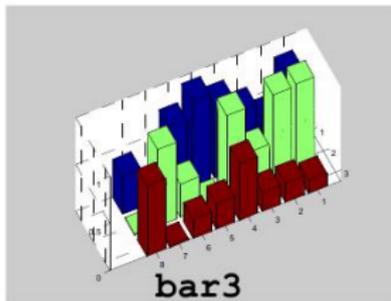
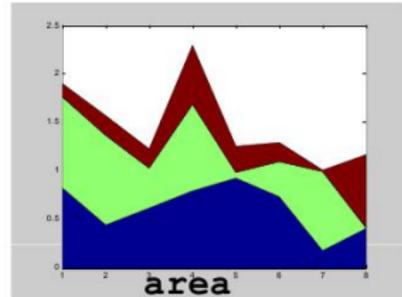
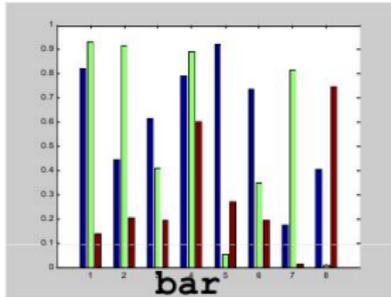
# Wissenschaftliche Visualisierung

- Veranschaulichung großer Datenmengen
- Gezielte Darstellung wichtiger Informationen
- Nutzung der Fähigkeiten des menschlichen Auges
- Visuelle Präsentation von Ergebnissen

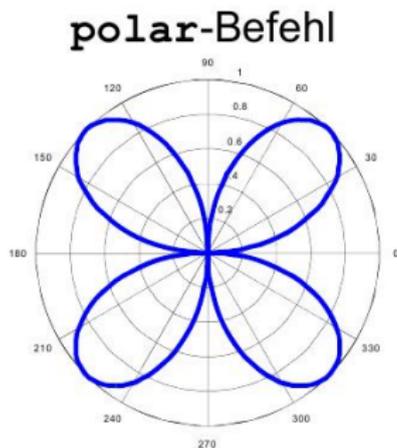
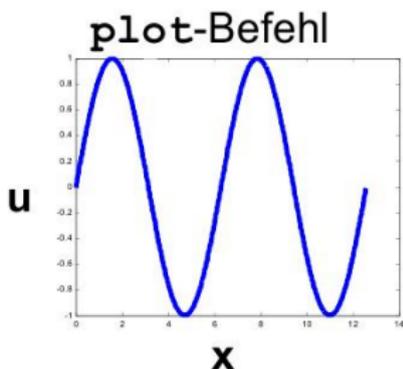


# Einfache Vektoren und Matrizen

- Darzustellende Daten:  $x_1, \dots, x_n$   
 $y_1, \dots, y_n$   
 $z_1, \dots, z_n$



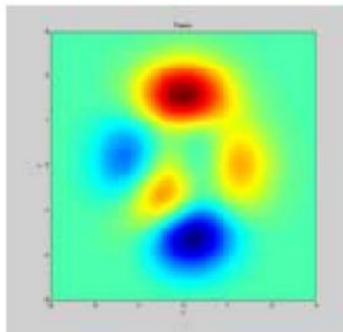
# Funktionsgraphen



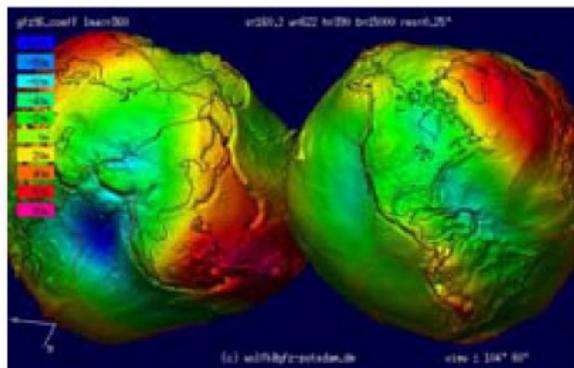
Weitere Befehle: **semilogx**, **semilogy**, **loglog**

# Farbfelder

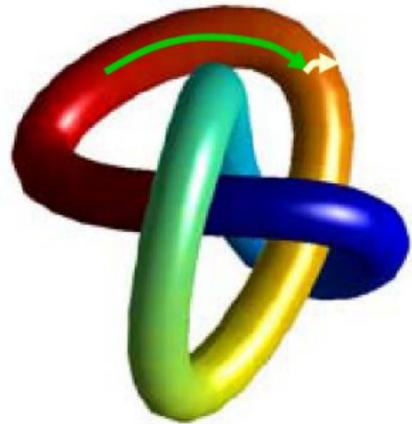
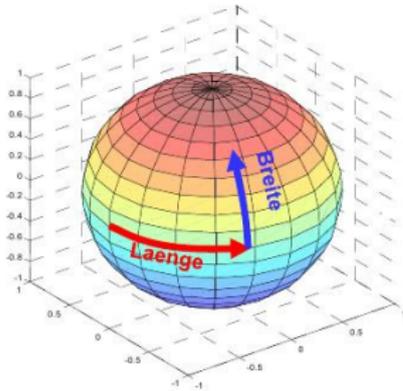
**surf**-Befehl



„Kartoffelplot“ des  
Erdgravitationsfeldes

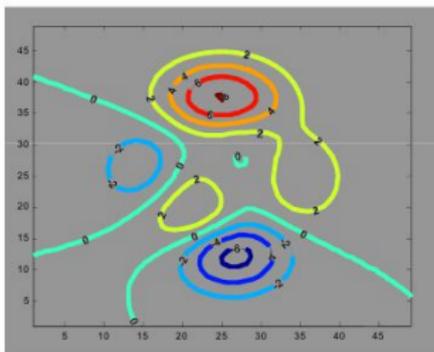


# Mehrdimensional parametrisierte Flächen

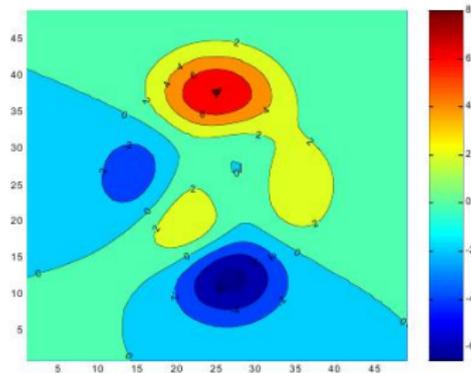


# Niveaulinienplots

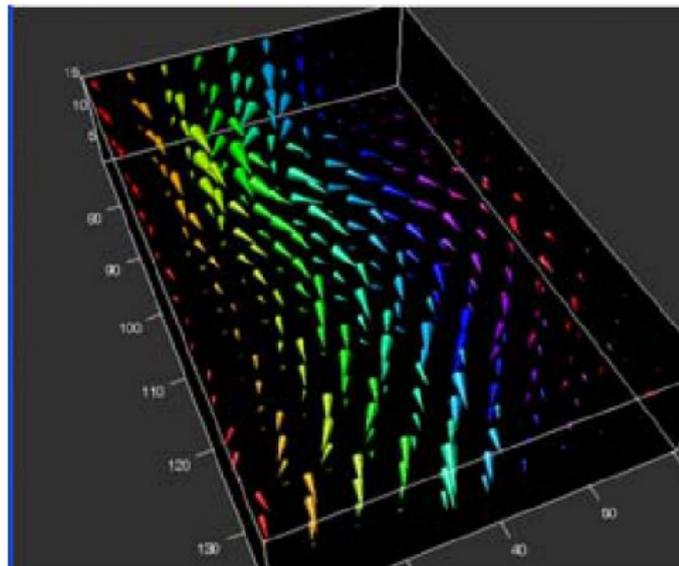
`contour`-Befehl



`contourf`-Befehl



# Ausblick: 3D-Vektorfelder



# Ausblick: Strömungsdarstellung

