



# MATLAB für Naturwissenschaftler

## 8. Grafiken

Till Biskup

Lehrstuhl für Physikalische Chemie und Didaktik  
Universität des Saarlandes



Motivation

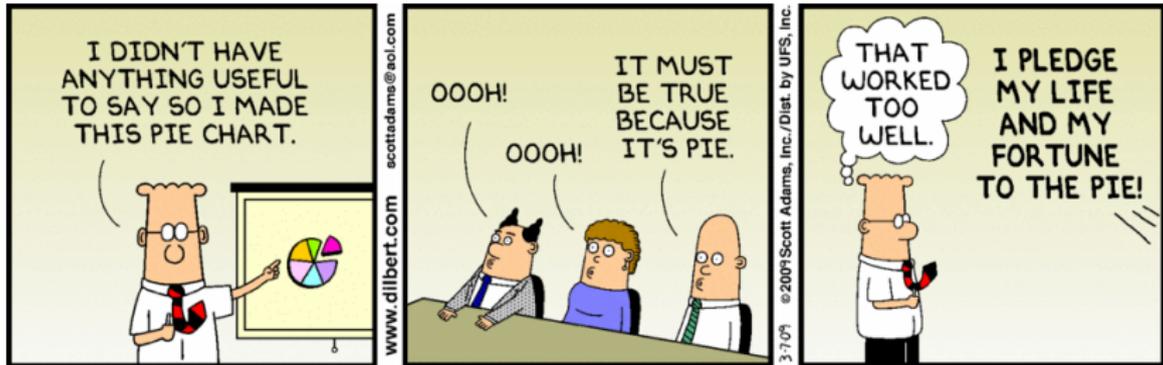
Formatierung von Abbildungen

Plot-Befehle in MATLAB

Abbildungen exportieren

# Motivation

Ein Bild sagt mehr als tausend Worte



- ☛ Excel ist gut geeignet für Business-Grafiken, aber nicht zur Darstellung wissenschaftlicher Daten.

© Scott Adams, <http://dilbert.com/2009-03-07/>



## Warum sind grafische Darstellungen relevant?

- ▶ Ein Bild sagt mehr als tausend Worte
  - Abbildungen dienen der schnellen Übersicht
  - Übersichtliche und ansprechende Darstellung
  - Ähnlich viel Zeit investieren wie in den begleitenden Text
- ▶ Auswertung und grafische Darstellung beeinflussen sich
  - Ein erster Eindruck der Daten steht oft am Anfang.
  - Manche Auswertung erst durch die Darstellung möglich.
- ☞ Auswertungen mit MATLAB erstellen,  
die wissenschaftlichen Kriterien entsprechen  
(und den nervigen Betreuer zufriedenstellen).



### Formatierung von Abbildungen

- ▶ Konventionen in den Naturwissenschaften
  - Diskrete Datenpunkte (normalerweise) nicht verbinden
  - Formelgrößen *kursiv* setzen
  - Einheiten aufrecht und *nie* in eckigen Klammern
  - Achsenbeschriftungen: *Größe* / Einheit
- ▶ MATLAB unterstützt grundlegende  $\LaTeX$ -Formatierung
  - kursiver Text: „`\it Text`“
  - hochgestellter Text: „`^{\text{Text}}`“
  - tiefgestellter Text: „`_{\text{Text}}`“
- ▶ Hinweis zu Sonderzeichen
  - MATLAB unterstützt (noch) kein Unicode
  - Sonderzeichen sind mitunter betriebssystemabhängig



- ▶ Korrekte und vollständige Achsenbeschriftungen
  - Größe *kursiv*, Einheit aufrecht
  - Schrägstrich „/“ als Trenner zwischen Größe und Einheit
  - Einheiten *nie* in eckigen Klammern

### Notation physikalischer Größen

Der Wert einer physikalischen Größe kann als Produkt eines Zahlenwertes und einer Einheit ausgedrückt werden:

$B = 1 \text{ mT}$	Größe mit Wert
$\{B\} = 1$	Zahlenwert
$[B] = \text{mT}$	Einheit

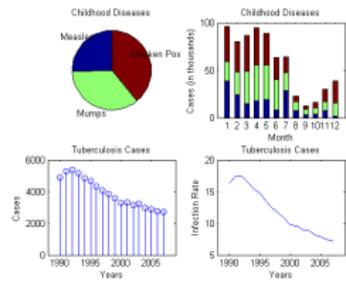
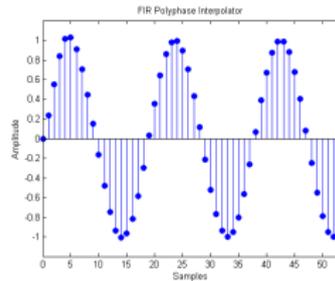
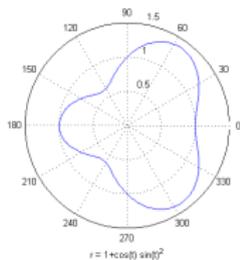
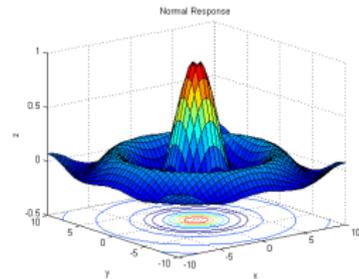
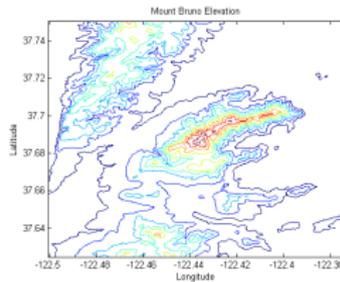
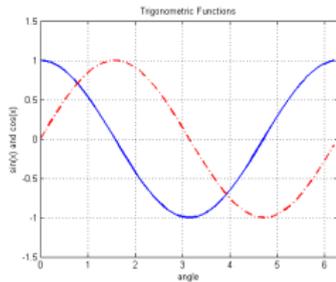


### Formatierung von Abbildungen

- ▶ Vorhandenen Platz sinnvoll nutzen
  - MATLAB hat mitunter eigene Vorstellungen...
- ▶ Vergleich mehrerer Abbildungen
  - Achsen mit identischem Wertebereich
  - Achsenformatierungen konsistent
  - Daten ggf. (identisch) skalieren
  - Konsistentes Farbschema
- ▶ Abbildungen für Präsentationen
  - Achsenbeschriftung ausreichend groß
  - Liniendicke und Farbe präsentationskompatibel
  - Weniges ist nerviger und hinderlicher als fehlende oder unlesbare Achsenbeschriftungen

# Plot-Befehle in MATLAB

## Galerie von Abbildungstypen in MATLAB



Hinweis: Diese Abbildungen haben viel zu kleine Achsenbeschriftungen für eine Präsentation...

<http://www.mathworks.de/discovery/gallery.html>



### Grundlegende Plot-Befehle in MATLAB

- ▶ Eindimensionale Abbildungen in MATLAB: `plot`
  - Eine Dimension:  $f(x)$  gegen  $x$  auftragen
  - Häufigste (und einfachste) Darstellungsform
- ▶ Achsen beschriften: `xlabel`, `ylabel`
  - Wichtig: Auf korrekte Formatierung achten
  - Größe und Einheit (wenn es eine Einheit gibt)
- ▶ Legende: `legend`
  - Box innerhalb der Achsen
  - Beschreibung jeder einzelnen „Kurve“
  - Position (in gewissen Grenzen) kontrollierbar

☞ Details und weitere Plot-Befehle in der MATLAB-Hilfe



### Grundlegende Plot-Befehle in MATLAB: `plot`

#### Listing 1: Beispiele für den `plot`-Befehl

```
1 % Define x,y vectors
2 x = 0:0.1:2*pi;
3 y = sin(x);
4
5 % Plot y = f(x)
6 plot(x,y);
7
8 % Same plot, but with different colouring
9 % "k" - black (from cmyk)
10 % "-" - solid line
11 plot(x,y,'k-');
12
13 % Same plot, but with different colouring and line style
14 % "r" - red (from rgb)
15 % "x" - crosses, no solid line connecting the data points
16 plot(x,y,'rx');
```



### Linienstile, -Marker und -Farben in MATLAB

- ▶ Können auf zwei Wegen angegeben werden
  - Als drittes Argument nach  $x$  und  $y$  (als String)
  - Als Schlüssel-Wert-Paare
- ▶ Schlüssel
  - `Color`, `LineStyle`, `Marker`
- ▶ Farben
  - Kürzel: `r`, `g`, `b`, `c`, `m`, `y`, `k`, `w`
  - RGB-Tripel als Vektor mit Werten zwischen 0 und 1
- ▶ Automatische Abfolge von Linienfarben und -Stilen
  - Bei mehreren Linien in einem Plotbefehl
  - Reihenfolge kann vom Nutzer festgelegt werden



### Linienstile in MATLAB

Kürzel	Linienstil
-	durchgezogen
--	gestrichelt
:	gepunktet
-.	gestrichpunktet
none	keine Verbindungslinie

- ▶ Standard ist die durchgezogene Linie.
- ▶ Wissenschaftliche Darstellungen
  - Punkte in der Regel nicht verbinden
  - Stattdessen Marker setzen (die Punkte sind viel zu klein)



### Linienmarker in MATLAB

Kürzel	Marker
+	Pluszeichen
o	Kreis
*	Stern
.	Punkt
x	Kreuz
s, square	Quadrat
d, diamond	Raute
^, v, >, <	Dreieck (nach oben, unten, rechts, links)
p, pentagram	Pentagramm
h, hexagram	Hexagramm
none	kein Marker



## Grundlegende Plot-Befehle in MATLAB: xlabel, ylabel

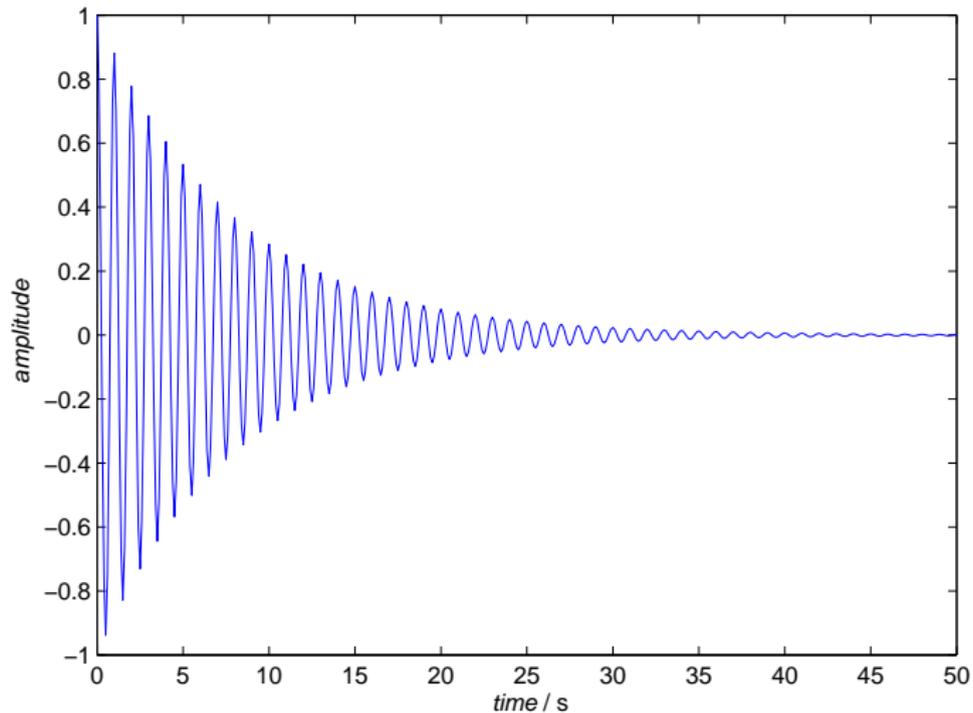
### Listing 2: Beispiele für Achsenbeschriftungen (xlabel, ylabel)

```
1 % Define x,y vectors and plot y = f(x)
2 t = 0:0.1:50; % time in seconds
3 A = cos(t*(2*pi)).*exp(-0.02*t*(2*pi)); % damped oscillation
4 plot(t,A);
5
6 % Set x and y labels
7 xlabel('\it time / s');
8 ylabel('\it amplitude');
```

- ▶ Auf korrekte Formatierung achten
  - Größe / Einheit
- ▶ MATLAB unterstützt grundlegende  $\text{\LaTeX}$ -Formatierung
  - kursiver Text: „ $\{\it \text{Text}\}$ “

# Plot-Befehle in MATLAB

## Grundlegende Plot-Befehle in MATLAB





## Grundlegende Plot-Befehle in MATLAB: legend

### Listing 3: Beispiele für eine Legende (legend)

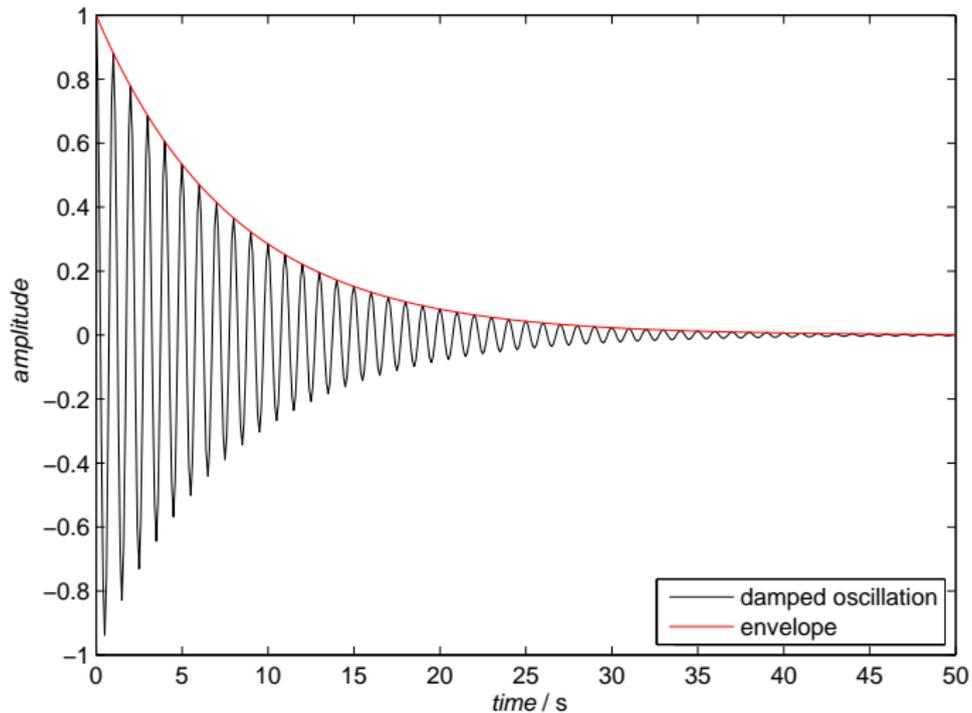
```
1 % Plot damped oscillation and envelope
2 plot(...
3     t,A,'k-',...
4     t,exp(-0.02*t*(2*pi)), 'r-' ...
5     );
6
7 % Plot legend
8 legend({'damped oscillation','envelope'});
9
10 % Plot legend at specific location
11 legend({'damped oscillation','envelope'},'Location','SouthEast');
```

### ► Position der Legende

- Zusätzlicher Parameter 'Location' (wenig intuitiv)
- Vier Ecken über Himmelsrichtungen
- Wenn nicht angegeben: „beste“ Position (laut MATLAB)

# Plot-Befehle in MATLAB

## Grundlegende Plot-Befehle in MATLAB





### Problem

- ▶ Jeder `plot`-Befehl löscht das aktuelle Grafikfenster.

### Lösungen für mehrere Linien im gleichen Fenster

- ▶ Mehrere  $x, y$ -Wertepaare im `plot`-Befehl
  - ✓ MATLAB permutiert Linienfarbe/-Stil automatisch
  - ✗ wird im `plot`-Befehl schnell unübersichtlich
- ▶ `hold on, hold off`
  - ✓ Auch nachträglich (bei aktivem Fenster) anwendbar
  - ✗ Linienfarbe/-Stil muss manuell angegeben werden
- ▶ Low-level-Routine: `line`
  - ✓ Viele Freiheiten
  - ✗ Keinerlei automatische Anpassung der Achsen



## Subplots: Mehrere Plots in einem Fenster

### Listing 4: Beispiele für Subplots

```
1 % Divide figure into 2x1 grid, active axis in grid position 1
2 subplot(2,1,1)
3 plot(t,A,'k-');
4
5 % Active axis in grid position 2
6 subplot(2,1,2)
7 plot(t,A,'r:');
```

- ▶ Nummerierung der Subplots zeilenweise
  - Erst alle Subplots einer Zeile, dann nächste Zeile
- ▶ Aufruf von `subplot` aktiviert nur die jeweilige Achse
  - Eigentlicher Plot erst durch nachfolgende Befehle
  - Hilfreich zum Wechseln zwischen Achsen in Subplots



## Abbildungen in MATLAB sind Objekte

- ▶ Grafik-Objekte verhalten sich ähnlich wie Strukturen
  - Hierarchisch verschachtelt
  - Jedes Grafik-Objekt hat Eigenschaften (*properties*)
  - Jedes Objekt hat eine Referenz (*handle*) für den Zugriff
- ▶ Standard-Hierarchie eines MATLAB-Abbildungsfensters
  - Grafikfenster
  - Achse
  - Linie (und andere Objekte innerhalb der Achse)
- ▶ Standard-Referenzen (*handles*) in MATLAB:
  - `gcf` – aktives Abbildungsfenster (*current figure*)
  - `gca` – aktive Achse (*current axes*)
  - `gco` – aktives Grafikobjekt (*current object*)



## Kontrolle der Eigenschaften

- ▶ Über „Getter“ und „Setter“
  - Funktionen `get` und `set`
  - Erstes Argument ist jeweils die Objektreferenz
  - Eigenschaften werden durch Schlüssel-Wert-Paare gesetzt
- ▶ Sehr viele Eigenschaften
  - Kontrollieren das Aussehen in relativ großem Detail
  - Details zu den Eigenschaften in der MATLAB-Hilfe
- ☞ Auf den ersten Blick nicht sehr intuitiv
- ☞ Seit MATLAB 2014b große Änderungen
- ☞ Wird noch wichtig beim Export der Grafiken



### Abbildungen aus MATLAB exportieren

- ▶ MATLAB unterstützt Export in diverse Grafikformate
  - Vektorisiert: EPS, PDF
  - Bitmap: PNG, JPG, ...
  - Vektorgrafiken sind *immer* zu bevorzugen  
(einfache Nachbearbeitung mit anderen Programmen)
- ▶ Grundsätzlich zwei Wege zum Export von Abbildungen
  - Grafisch über die MATLAB-GUI bzw. das Menü des Fensters
  - Über die Kommandozeile
- ▶ Befehle zum Speichern von Abbildungen in MATLAB
  - `saveas`, `print`
- ☞ Export führt mitunter zu überraschenden Ergebnissen



### Abbildungen aus MATLAB exportieren

- ▶ Vorteile von Vektorgrafiken
  - Ermöglichen Nachbearbeitung (oft notwendig)
  - Benötigen weniger Speicherplatz
  - Sind beliebig skalierbar
- ▶ Strategie für die Erstellung von Abbildungen
  - Möglichst viel in MATLAB automatisieren
  - Export als Vektorgrafik (über Routine)
  - Nachbearbeitung in externem Vektorgrafikprogramm
- ▶ Umfangreiche Kontrolle des Aussehens möglich
  - Papierformat, Schriftart und -größe, ...
  - Eigenschaften der Abbildungen über `set` setzen
- ▶ Tipp: Eigene Routine zum Export von Abbildungen



## Warum Abbildungen automatisiert exportieren?

- ▶ Sorgt für ein möglichst konsistentes Aussehen.
- ▶ Erleichtert den Reexport nach Änderungen an den Daten.

## Warum Abbildungen als PDF-Dateien exportieren?

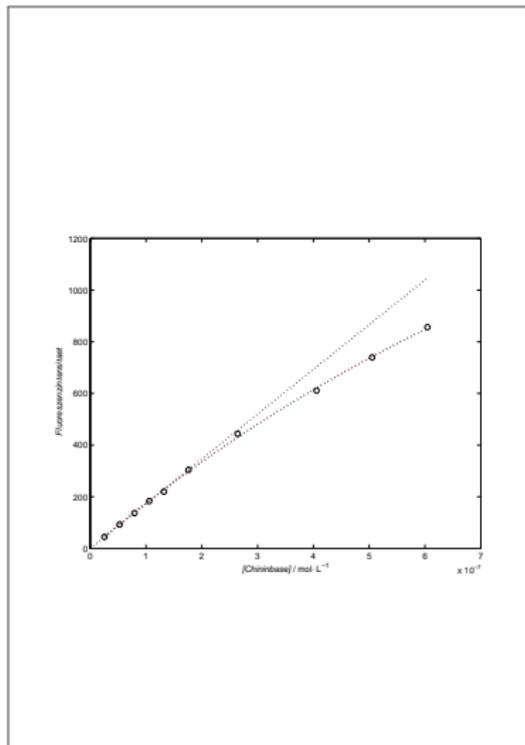
- ▶ PDF-Dateien sind (in der Regel) vektorisiert, können also beliebig skaliert und einfach nachbearbeitet werden.

### Listing 5: Grundlegender Abbildungsexport als PDF-Datei

```
1 print(gcf, 'erster-test.pdf', '-dpdf');
```

# Abbildungen exportieren

Der erste Versuch – noch nicht ganz das gewünschte Ergebnis





### Listing 6: Anpassungen der Seitengröße

```
1 % Anpassung der Seitengroesse
2 set(gcf,'paperunits','centimeters');
3 set(gcf,'papersize',[16 10]);
4
5 % Anpassung der Positionierung auf der Seite
6 set(gcf,'paperpositionmode','auto');
7 set(gcf,'Units','centimeters');
8
9 % Anpassung der Groesse der Achsen
10 set(gca,'Units','centimeters');
11 set(gca,'OuterPosition',[0 0 16 10]);
12
13 % Positionierung auf dem Papier
14 oldpos = get(gcf,'Position');
15 set(gcf,'Position',[oldpos([1 2]) 16 10]);
```

- ☛ Die Reihenfolge der Befehle ist nicht immer egal.
- ☛ Manchmal erschließt sich die Logik nicht zwangsläufig...



*...Zeit für eigene praktische Arbeit...*

## Vorschau: Lineare und nichtlineare Regression

- ▶ Allgemeines zu Regression und Kurvenanpassung
- ▶ Lineare Regression
- ▶ Nichtlineare Regression