Anwendung von (Mathematica und) Matlab in der Physikalischen Chemie

10. Projekt: Vorstellung und Pflichtenheft

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Sommersemester 2016

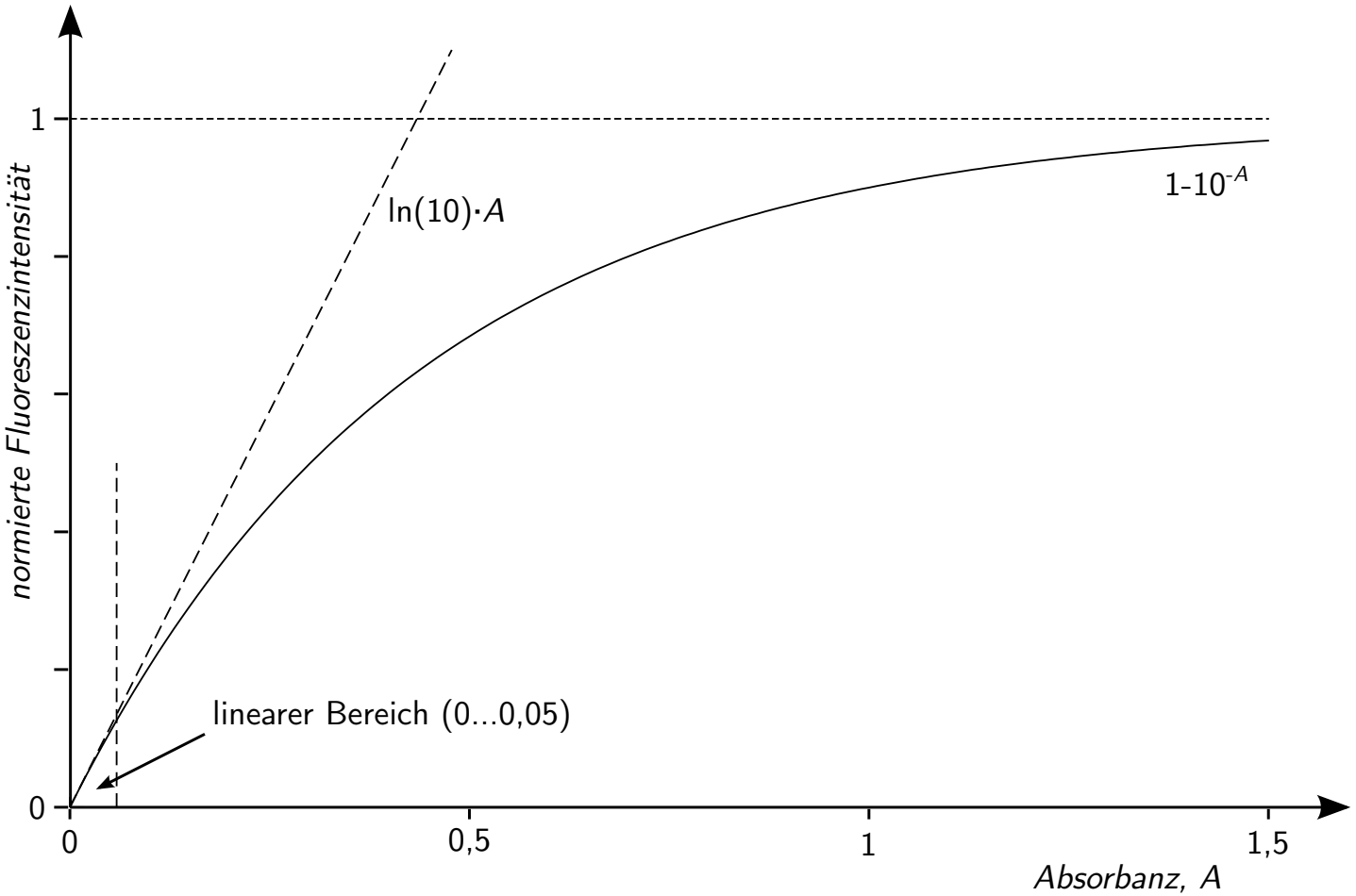
Ein reales Beispiel: Fluoreszenz-Versuch aus dem PCG

- Ausgangslage
  - Daten wurden alle gemessen
  - Daten liegen als Textdateien (ASCII) vor
- Zielstellung
  - Vollständige Auswertung gemäß Fragestellung
  - Abbildungen, die den Assistenten zufriedenstellen (und den wissenschaftlichen Standards entsprechen)
- Vorgehen
  1. Pflichtenheft erstellen (was muss getan werden?) ✔
  2. Notwendige Grundlagen von Matlab aneignen ✔
  3. Auswertung gemäß Pflichtenheft in Matlab programmieren

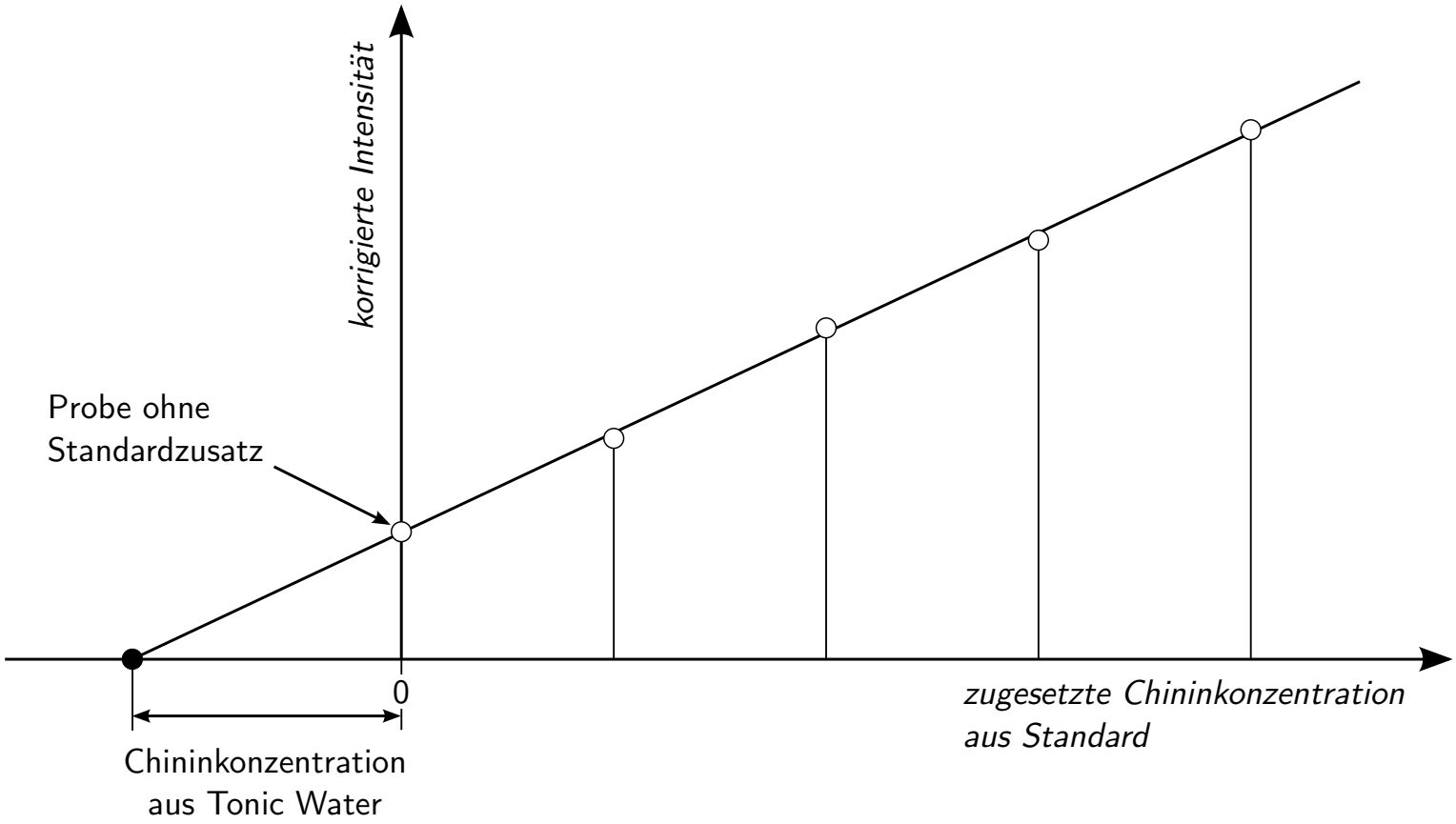
Kurze Wiederholung: Der Fluoreszenz-Versuch im PCG

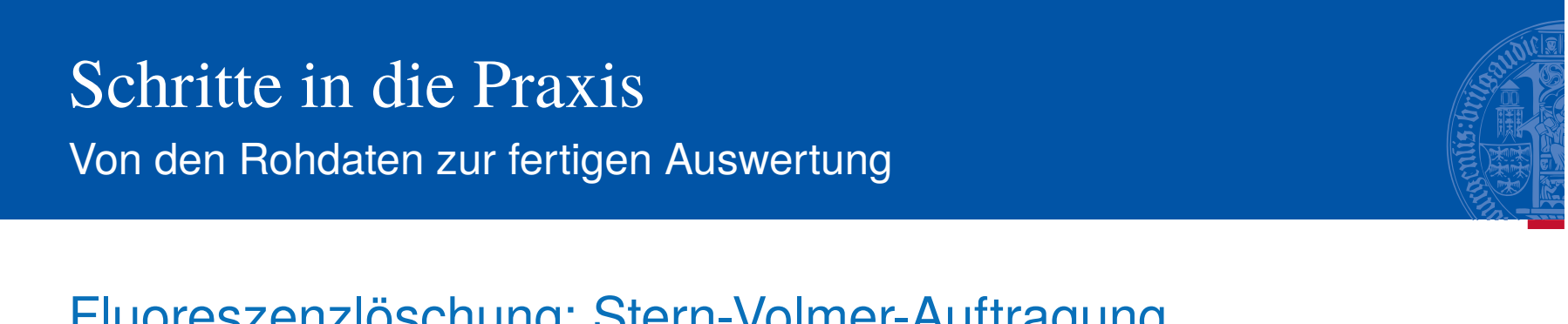
1. Anregungs- und Emissionsspektren
   - Spektren darstellen
   - Maximum hervorheben
2. Konzentrationsabhängigkeit der Fluoreszenz
   - Intensität als Funktion der Konzentration darstellen
   - Lineare und nichtlineare Kurvenanpassung
3. Bestimmung des Chiningehalts von Tonic Water
   - Lineare Regression
4. Dynamische Fluoreszenzlöschung (Stern-Volmer)
   - Lineare Regression mit festem y-Achsen-Abschnitt

Fluoreszenzintensität als Funktion der Konzentration

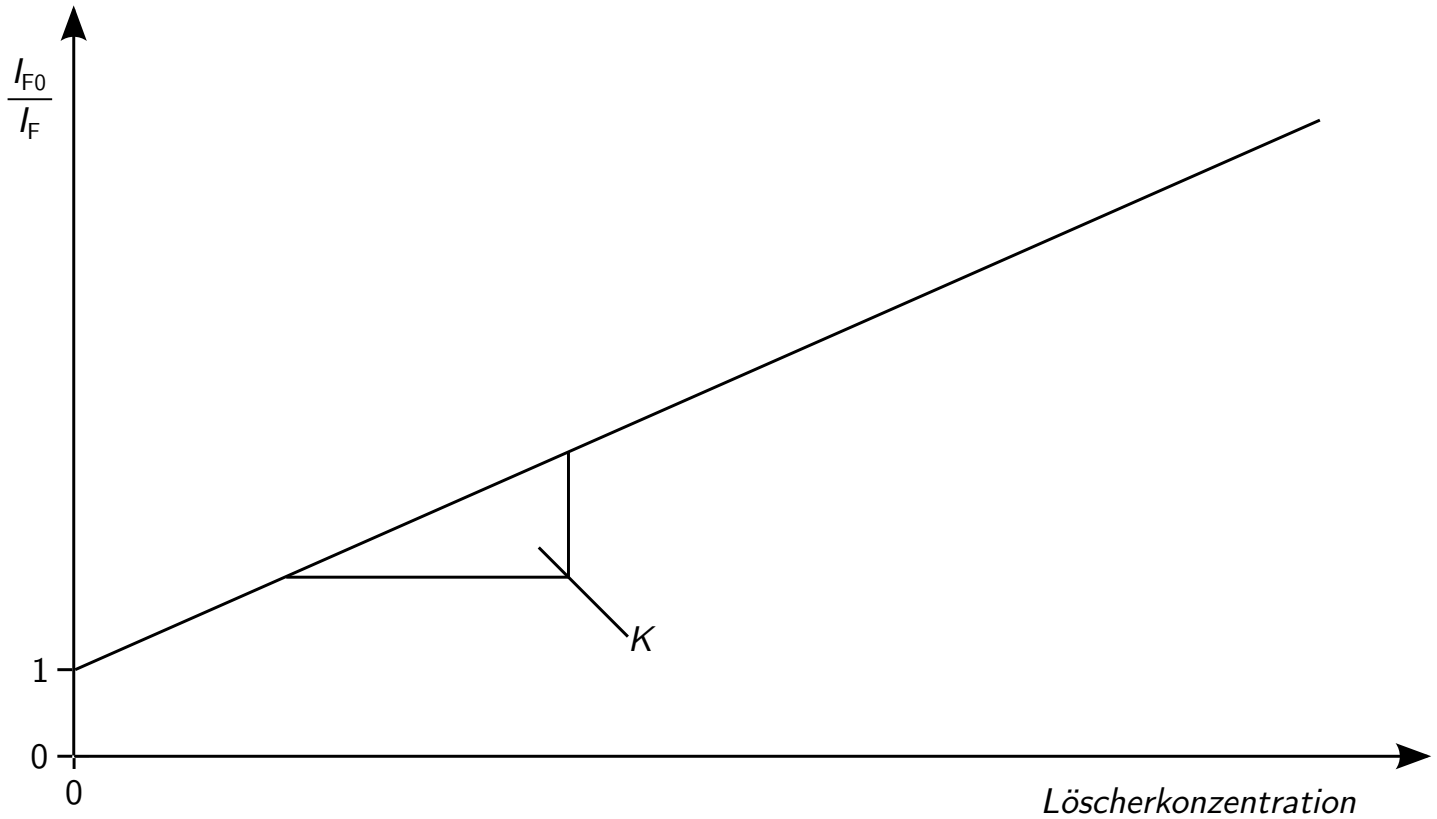


Konzentrationsbestimmung durch Standardaddition





Fluoreszenzlöschung: Stern-Volmer-Auftragung



Pflichtenheft: Einzelne Schritte der Auswertung

- Daten einlesen
  - Daten importieren in Matlab
- Spektren darstellen
  - Daten in Matlab grafisch darstellen (plotten)
  - Achsenbeschriftungen gemäß Vorgaben
  - Abbildungen aus Matlab exportieren
- Intensitäten für eine Wellenlänge aus mehreren Spektren
  - Mehrere Spektren einlesen
  - Zugriff auf einen bestimmten Wert in einem Vektor
- Lineare und nichtlineare Kurvenanpassung
  - Matlab-Routinen zur Kurvenanpassung

Übersicht

UNI FREIBURG

Daten importieren

Daten verarbeiten

- Zugriff auf die eingelesenen Daten
- Umrechnung von Werten
- Regression und Anpassung von Kurven

Daten darstellen

- Formatierung von Abbildungen
- Grundlegende Plot-Befehle im Matlab
- Abbildungen aus Matlab exportieren

Import von Textdateien (ASCII): `importdata`

- Einsatzgebiet
  - Textdateien mit Kopfzeilen
- Voraussetzung
  - Länge des Dateikopfes ist bekannt
  - Daten: Identische Spaltenzahl für jede Reihe
  - Daten: Punkt als Dezimaltrennzeichen
- Parameter der Funktion
  - Dateiname
  - *Optional*: Trennzeichen für die einzelnen Datenspalten
  - *Optional*: Zahl der Kopfzeilen der Datei

☛ Wir werden *diese* Funktion nachher brauchen.

Import von Textdateien (ASCII): importdata

Listing 2: Beispiel einer mit `importdata` lesbaren Textdatei

```
1  PE FL                    SPECTRUM    ASCII       PEDS        1.60
2    -1
3  1A-01.SP
4  13/05/02
5  13:55:40.00
6  13/05/02
7  13:56:11.00
8
9
10 400.000000
11 1
12 #DATA
13 270.000000  7.640000
14 270.500000  7.712395
15 271.000000  7.775626
16 271.500000  7.830389
17 272.000000  7.877049
18 272.500000  7.925246
19 273.000000  7.979488
20 273.500000  8.035135
```

Import von Textdateien (ASCII): importdata

Listing 3: Aufruf der importdata-Funktion

```
1 % Aufruf ohne optionale Parameter
2 data = importdata('filename');
3
4 % Aufruf mit Spaltentrennzeichen (hier: Tabulator)
5 data = importdata('filename','\t');
6
7 % Aufruf mit Spaltentrennzeichen und Anzahl Kopfzeilen
8 data = importdata('filename','\t',13);
```

- Wichtige Hinweise
  - Zahl der Kopfzeilen kann nur *gemeinsam* mit dem Spaltentrennzeichen angegeben werden.
  - Rückgabeparameter (`data`) ist ein `struct`
    Felder: `data`, `textdata`, `colheaders`

Daten: Währung der empirischen Wissenschaften

- Grundlage und Ausgangspunkt empirischer Wissenschaft
  - Daten sind nicht notwendigerweise „offensichtlich“.
  - Messung zur „Aufnahme“ von Daten
- Daten überdauern, Interpretationen ändern sich
  - Daten nach bestem Wissen und Gewissen aufnehmen
  - Rohdaten *niemals* wegwerfen
- Verantwortung des Wissenschaftlers
  - Saubere Datenaufnahme und -dokumentation
  - Nachvollziehbarkeit der Datenaufnahme und -verarbeitung
  - Reproduzierbarkeit der Ergebnisse

Zugriff auf die eingelesenen Daten

- `importdata` liefert ein `struct` zurück
  - Eigentliche Daten stehen im Feld `data`
  - `data` in diesem Fall Matrix mit zwei Spalten

Listing 4: Zugriff auf die Daten aus `importdata`

```
1  % Import data; separator: tabulator; header lines: 42
2  data = importdata('datafile.txt','\t',42);
3  % Note: "data" is a structure with three fields
4  %       data       - numeric; contains the actual data
5  %       textdata   - cell array; header lines
6  %       colheaders - cell array; column headers
7
8  % Plot y = f(x)
9  % x is in the 1st column of data.data
10 % y is in the 2nd column of data.data
11 plot(data.data(:,1),data.data(:,2));
```

Zugriff auf die eingelesenen Daten

- Zugriff auf Intensitätswert über Wellenlängenachse
  - Gefragt sei nach der Intensität bei 448 nm
  - Zugriff über „logische Indizierung“

Listing 5: Logische Indizierung

```
1 % Import fluorescence spectrum; separator: tabulator; header lines: 42
2 data = importdata('1E-01.sp','\t',42);
3
4 % Assume that data.data(:,1) is the wavelength axis
5 %          and data.data(:,2) the corresponding intensities
6
7 % Get intensity at 448 nm
8 Int448 = data.data(data.data(:,1)==448,2)
```

Zugriff auf die eingelesenen Daten

- Handhabung mehrerer Datensätze
  - Gefragt seien elf Datenpunkte, je aus einem Datensatz
  - Fluoreszenzintensität gegen Konzentration auftragen

Listing 6: Mögliche Verarbeitung mehrerer Datensätze

```
1 % Cell array with filenames
2 fileNames = {'2E-01.sp','2E-02.sp',...,'2E-11.sp'};
3
4 % For each filename, load file and get intensity at 448 nm
5 for k=1:length(fileNames)
6     data = importdata(fileNames{k},'\t',42);
7     Int448(k) = data.data(data.data(:,1)==448,2);
8 end
```

☛ Tipp: Initialisierung von `Int448` (Warnung im Editor)

Umrechnung von Werten

- Normierung der Fluoreszenzintensität auf die Leerprobe
  - Abziehen des Wertes der Leerprobe

Listing 7: Matrixoperationen – eine Stärke von Matlab

```
1 % Assume Int448 to be a vector of fluorescence intensities at 448 nm
2 % Assume the first element of Int448 to be the blank
3 normInt448 = Int448-Int448(1);
```

- Anmerkungen
  - `Int448` ist ein Vektor mit mehreren Elementen.
  - `Int448(1)`, der erste Wert von `Int448`, ist die Leerprobe.

Regression und Anpassung von Kurven

- Allgemeines zur Kurvenanpassung
  - Häufig über Minimierung der Summe der Fehlerquadrate (*least squares fit*)
  - *Erst überlegen*, welches Modell man anpassen möchte
  - Zahl der Parameter ist entscheidend
- Arten von Kurvenanpassungen im Fluoreszenzversuch
  - Linear (ein und zwei Parameter)
  - Nichtlinear

☞ Matlab stellt diverse Möglichkeiten zur Verfügung.

Matlab-Routinen zur Kurvenanpassung

- Polynome
  - `polyfit`, `polyval`
- Lineare Gleichungssysteme
  - `\`, `lscov`
- Allgemeine nichtlineare Kurvenanpassungen
  - `fminsearch`
  - (kommerzielle) Toolboxen

Lineare Regression mit zwei Parametern

- Funktion: $y = m \cdot x + c$
- Matlab-Funktion: `polyfit`, `polyval`

Listing 8: Lineare Regression mit zwei Parametern

```
1 % Assume data x,y
2 % Fit polynomial of first order, f(x)=y=m*x+c
3 coefficients = polyfit(x,y,1);
4
5 % Get regression curve with calculated coefficients
6 regression   = polyval(coefficients,x);
```

Lineare Regression durch den Ursprung

- Funktion: $y = m \cdot x$
- Matlab-Funktion: `lscov` oder `\`

Listing 9: Lineare Regression durch den Ursprung

```
1 % Assume data x,y
2 % Use system of linear equations, A*m = B => y = m*x
3 % Solve using "lscov" to get slope
4 m = lscov(x(:),y(:));
5
6 % Second parameter gives error estimate
7 [m,sm] = lscov(x(:),y(:));
```

- Anmerkungen
  - `lscov` ist empfindlich auf die Dimension der Vektoren.
  - `x(:),y(:)` umgeht das Problem.

Nichtlineare Kurvenanpassung

- Funktion: $y = a \cdot (1 - 10^{-b \cdot x})$
- Matlab-Funktion: `fminsearch`

Listing 10: Nichtlineare Kurvenanpassung

```
% Assume data x,y
% Set starting values for coefficients c
c = [1000 5e-4];

% Define model to fit the data, with vector of coefficients c
% Model: y = c(1)*(1-10^(-c(2)*x))
model = @(c)c(1).*(1-10.^(-c(2).*x));

% Define fit function as sum of residual least suqares
fitfun = @(c)sum((y-model(c)).^2);

% Get coefficients using fminsearch
coeff = fminsearch(fitfun,c);
```

Daten darstellen

- Abbildungen sind wichtig
  - „Ein Bild sagt mehr als tausend Worte...“
  - *Charakteristika* der Daten darstellen und hervorheben
  - Eine gute Abbildung darf Zeit kosten.
- Allgemeine Hinweise
  - Vorhandenen Platz möglichst ideal ausnutzen
  - Beschriftungen: vollständig, korrekt und ausreichend groß
  - Abbildungen durchnummerieren
  - Verweise aus dem Text
- Tipps zu Abbildungsunterschriften
  - Zusammenfassung im ersten Satz (ggf. fett hervorheben)
  - Alle wichtigen Informationen in die Abbildungsunterschrift

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}“
  - tiefgestellter Text: „_{Text}“
- Hinweis zu Sonderzeichen
  - Matlab unterstützt (noch) kein Unicode
  - Sonderzeichen sind mitunter betriebssystemabhängig

Grundlegende Plot-Befehle im Matlab

- Eindimensionale Daten darstellen: `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

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

Listing 11: Beispiel für den Grafikexport mit `saveas`

```
1 % Save current figure ("gcf") as PNG file (bitmap)
2 saveas(gcf,'myFigure.png','png');
```

Listing 12: Beispiel für den Grafikexport mit `print`

```
1 % Save current figure ("gcf") as PDF file (vectorised)
2 print(gcf,'-dpdf','myFigure.pdf');
```

- Umfangreiche Kontrolle des Aussehens möglich
  - Papierformat, Schriftart und -größe, ...
  - Eigenschaften der Abbildungen über `set` setzen
- Tipp: Eigene Routine zum Export von Abbildungen

Ausblick

Wie geht's weiter?

...Zeit für eigene praktische Arbeit...

„Lunch atop a Skyscraper", Charles C. Ebbets, 1932