

Anwendung von (Mathematica und) Matlab in der Physikalischen Chemie

3. Interaktive Kommandozeile

Albert-Ludwigs-Universität Freiburg



**UNI
FREIBURG**

Dr. Till Biskup

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

Einführung: Interaktive Kommandozeile

Grundaspekte der Programmierung

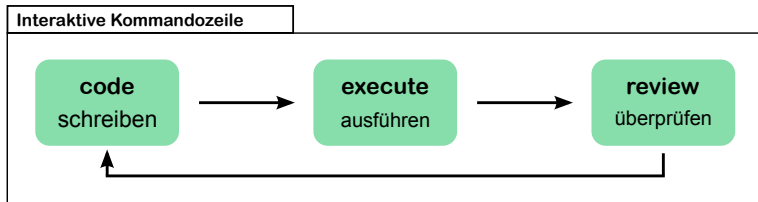
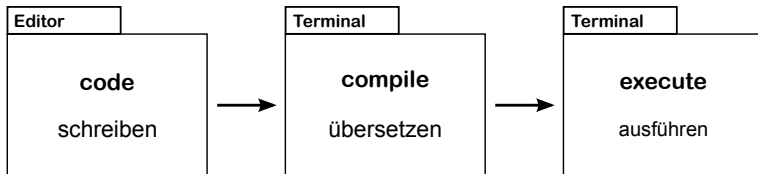
Matlab als Taschenrechner

Weitere Aspekte

Kosmetik und Komfort

Interaktive Kommandozeile

Vergleich zur konventionellen Programmierung



Vorteile

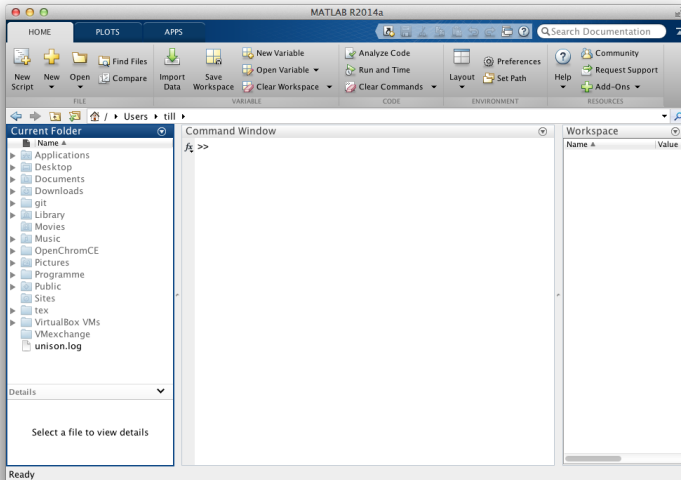
- ▶ Direkte Rückmeldung
 - Der Nutzer sieht sofort das Ergebnis eines Befehls
- ▶ „Rapid Prototyping“
 - Gut geeignet, um „mal schnell“ etwas auszuprobieren
 - Kurzer Zyklus zwischen Befehlseingabe und Ergebnis

Nachteile

- ▶ Ungeeignet für komplexere Abläufe
 - Schleifen können eingesetzt werden... aber mühsam
- ▶ Normalerweise keine Speicherung
 - Die Befehlshistorie ist begrenzt.

Interaktive Kommandozeile

Matlab: Ein erster Eindruck



- ▶ Integrierte grafische Nutzeroberfläche
 - Übersicht über die definierten Variablen
 - Zugriff auf die Hilfe (in eigenem Fenster)
 - Übersicht über die Befehlshistorie
 - Plots funktionieren direkt aus der Kommandozeile
 - Computermouse als intuitive Bedienungshilfe
- ▶ Alle Befehle auf der Kommandozeile erreichbar
 - Matlab: Alles, was im Matlab-Suchpfad liegt
 - Andere Sprachen: ggf. mit „import“ o.ä. arbeiten
- ▶ Ausgabeformat steuerbar
 - Z.B.: Darstellung von Fließkommazahlen
- ☞ Die einfach bedienbare grafische Oberfläche war wesentlich am Erfolg von Matlab beteiligt

- ▶ Matlab ist eine Programmiersprache
 - Große Ähnlichkeit mit C, Pascal, ...
 - Nimmt dem Nutzer viele Programmierdetails ab
 - Fokus auf Anwendung (und Mathematik)
- ▶ Matlab versucht nicht, seinen grundlegenden Charakter als Programmiersprache zu verbergen.
 - Ursprung als einfache Schnittstelle zu Fortran-Routinen
 - Viel näher an „normalen“ Programmiersprachen als beispielsweise Mathematica
- ☞ Grundaspekte der Programmierung notwendig, um mit Matlab auf der Kommandozeile zu arbeiten.

Allgemeine Aspekte

- ▶ Das Gleichheitszeichen dient der Zuweisung
 - „Zuweisungsoperator“
 - Mathematische Gleichheit wird durch „==“ abgefragt
- ▶ Zuweisungen werden von rechts nach links gelesen
 - Der Ausdruck rechts des Gleichheitszeichens wird der Variable links davon zugewiesen
 - Mehrfache Zuweisungen sind (in Matlab) nicht erlaubt

Matlab-Spezifika

- ▶ Variablen müssen nicht vordefiniert werden
- ▶ Variablen sind Matrizen (wenn nicht anders angegeben)

Listing 1: Die interaktive Kommandozeile in Matlab

```
1 >> 1+2
2
3 ans =
4
5      3
6
7 >> a=5; b=3;
8 >> c=a*b
9
10 c =
11
12     15
13
14 >>
```

- ▶ Das letzte Ergebnis wird immer in „ans“ gespeichert.
- ▶ Ein Semikolon unterdrückt die Ausgabe.

| Operation | Matlab-Befehl |
|------------------|-----------------------|
| Grundrechenarten | $+$, $-$, $*$, $/$ |
| Potenz | $^$ |
| Quadratwurzel | <code>sqrt</code> |
| n -te Wurzel | <code>nthroot</code> |

Anmerkungen

- ▶ Der Multiplikationsoperator „ $*$ “ muss immer explizit ausgeschrieben werden.
- ▶ Matlab beherrscht grundlegende Operatorrangfolge („Punkt vor Strich“)
- ▶ Terme können durch *runde* Klammern gruppiert werden

| Funktion | Matlab-Befehl |
|---------------------|---|
| trigonometrisch | <code>sin, cos, tan</code> <code>asin, acos, atan</code> |
| hyperbolisch | <code>sinh, cosh, tanh</code> <code>asinh, acosh, atanh</code> |
| Exponentialfunktion | <code>exp</code> |
| Logarithmen | <code>log, log10, log2</code> |
| Vorzeichen | <code>sign</code> |

Anmerkungen

- ▶ Argumente für die trigonometrischen Funktionen in Radians

| Funktion | Matlab-Befehl |
|----------------------------------|--------------------|
| Runden auf nächste ganze Zahl | <code>round</code> |
| Runden in Richtung 0 | <code>fix</code> |
| Runden in Richtung $-\infty$ | <code>floor</code> |
| Runden in Richtung $+\infty$ | <code>ceil</code> |
| Rest (Vorzeichen des Zählers) | <code>rem</code> |
| Rest (Vorzeichen des Nenners) | <code>mod</code> |
| größter gemeinsamer Teiler | <code>gcd</code> |
| kleinstes gemeinsames Vielfaches | <code>lcm</code> |

| Funktion | Matlab-Befehl |
|--|--------------------|
| Betrag | <code>abs</code> |
| Argument (Winkel der Polarkoordinaten) | <code>angle</code> |
| Realteil | <code>real</code> |
| Imaginärteil | <code>imag</code> |
| komplex-konjugierte Zahl | <code>conj</code> |

Anmerkungen

- ▶ Komplexe Zahlen werden mittels „i“ (bzw. „j“) angegeben
- ▶ Niemals „i/j“ als Laufvariable in Schleifen verwenden!

| Konstante | Matlab-Befehl |
|--|---------------------------------|
| komplexe Zahl (i) | <code>i</code> , <code>j</code> |
| Kreiszahl (π) | <code>pi</code> |
| Maschinengenauigkeit (ε) | <code>eps</code> |

Anmerkungen

- ▶ Die Schreibweise der komplexen Zahl als „j“ kommt aus der Elektrotechnik.
- ▶ `pi` ist nur so exakt wie die numerische Genauigkeit (ε).
 - `sin(pi)` gibt deshalb in Matlab *nicht* 0 zurück...
- ☞ Mehr zur numerischen Genauigkeit (ε) später

- ▶ Alle Variablen sind für Matlab Matrizen.
 - Solange man nicht explizit etwas anderes sagt...
 - Matlab = MATrix LABoratory
- ▶ Matrixoperationen sind in Matlab schnell, der Rest nicht.
- ▶ Eckige Klammern dienen der Definition:

Listing 2: Definition von Matrizen und Vektoren in Matlab

```
% Ein Zeilenvektor  
a = [1 2 3]  
% Ein Spaltenvektor  
b = [1; 2; 3]  
% Eine Matrix  
c = [1 0; 0 1];
```

- ▶ Matrizen werden in der Reihenfolge Zeile-Spalte indiziert
- ▶ Zugriff auf Elemente über runde Klammern
 - Alle Elemente einer Zeile/Spalte über „:“

Listing 3: Zugriff auf Elemente von Matrizen und Vektoren in Matlab

```
% Eine 2x3-Matrix  
m = [1 2 3; 4 5 6];  
  
% Zugriff auf das zweite Element der ersten Zeile  
m(1,2)  
  
% Zugriff auf die zweite Zeile  
m(2,:)   
  
% Zugriff auf die erste Spalte  
m(:,1)
```


Operatoren

| Operator | Bedeutung |
|-----------------------------------|-------------------------------------|
| $+$, $-$, $*$, $/$, $^$ | Matrix-Operationen |
| $.\+$, $.-$, $.*$, $./$, $.^$ | elementweise Operationen |
| \backslash | Division von links |
| $:$ | Zugriff auf Bereiche (Spalte/Zeile) |
| $'$ | Adjungieren |
| $.'$ | Transponieren |

Anmerkungen

- ▶ „ $.\+$ “ und „ $.-$ “ sind identisch mit „ $+$ “ und „ $-$ “.
- ▶ Arithmetische Operatoren führen Matrix-Operationen aus

Funktionen

| Funktion | Bedeutung |
|------------------------|-------------------------|
| <code>cross</code> | Kreuzprodukt |
| <code>dot</code> | Punktprodukt |
| <code>kron</code> | Kronecker-Tensorprodukt |
| <code>eig, eigs</code> | Eigenwerte |
| <code>diag</code> | Diagonale |
| <code>transpose</code> | Transponieren |

Anmerkungen

- ▶ „*“ führt eine Matrixmultiplikation durch.

Spezielle Matrizen

| Matrix | Bedeutung |
|--------|--------------------------|
| ones | Matrix aus Einsen |
| zeros | Matrix aus Nullen |
| eye | Einheitsmatrix |
| diag | Diagonalmatrix |
| rand | Matrix aus Zufallszahlen |

Anmerkungen

- ▶ „eye“ ist ein Wortspiel für „I“.
 - Anfangs unterschied Matlab nicht zwischen Groß- und Kleinschreibung, und „i“ war vergeben (imaginäre Zahl).

Inline-Funktionen

Listing 4: Definition einer Inline-Funktion

```
>> fun = @(x) 3*x^3+5*x^2+8*x+17;  
>> fun(3)
```

```
ans =
```

```
167
```

- ▶ Für wiederkehrende mathematische Zusammenhänge
 - Auch in Abhängigkeit mehrerer Variablen.
 - Einfache Zusammenhänge, in einer Zeile formulierbar
- ▶ Definition über @-Zeichen
 - Variablen der Funktion nach dem @ in runden Klammern
 - Variablenname für die Zuweisung beliebig

Plots

Listing 5: Einfaches Beispiel eines Plot-Befehls

```
x=0:0.1:10; y=sin(x)
plot(x,y);
```

- ▶ Intuitiver Befehl für Vektoren/Funktionen einer Variablen
 - Vektoren (x, y) müssen gleich lang sein.
 - Bei nur einem Vektor wird gegen dessen Index geplottet.
- ▶ Jeder `plot`-Befehl überschreibt das aktuelle Grafikfenster.
 - Abhilfe (I): Paare von Vektoren in einem `plot`-Befehl.
 - Abhilfe (II): `hold on`, `hold off`

☞ Details ausführlicher in einer späteren Lektion

Und Vieles mehr...

- ▶ Es gibt noch viele weitere Aspekte.
 - Der Kurs erhebt keinerlei Anspruch auf Vollständigkeit.
 - Die Auswahl ist (notwendigerweise) subjektiv.
- ▶ Die Matlab-Hilfe ist ein guter Startpunkt.
 - Einführende Kapitel geben einen guten Überblick.
- ☞ Vieles läuft über (ernsthaftere) Programmierung.
 - Dazu kommen wir noch später.
 - Hier geht es erstmal um einen ersten Eindruck.
 - Matlab taugt auch als komfortabler „Taschenrechner“.

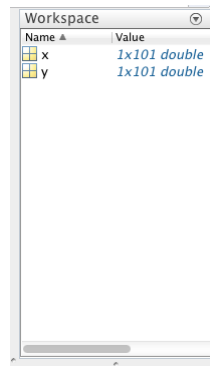
Listing 6: Standard-Ausgabeformat der Matlab-Kommandozeile

```
>> pi  
  
ans =  
  
    3.1416
```

- ▶ Normalerweise werden nur vier Dezimalstellen angezeigt.
 - Die interne Rechengenauigkeit ist natürlich viel höher.
 - Die Ausgabe rundet die letzte angezeigte Dezimale...
- ▶ Befehl zur Kontrolle der Anzeige: `format`
 - Optionen (u.a.): `long`, `short`, `rat`, `compact`
- 👉 Details in der Matlab-Hilfe: `doc format`

Übersicht über die Variablen – grafisch

- ▶ Vorteil der grafischen Oberfläche
 - Fenster „Workspace“ mit Variablen
 - Doppelklick auf Variable öffnet einen Variableneditor (wozu auch immer...)
- ▶ Übersicht über
 - Name
 - Typ (über Symbol vor dem Namen)
 - Größe („Value“)
 - Minimum und Maximum
- ▶ Nachteile
 - Typ nur über Symbol
 - Scrollen notwendig



Übersicht über die Variablen – programmatisch

Listing 7: Übersicht über die definierten Variablen

```
>> who
```





```
Your variables are:
```

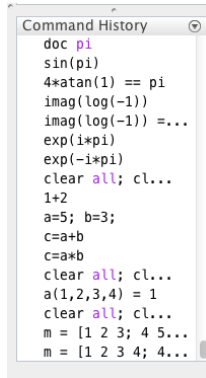
```
x  y
```

```
>> whos
```

| Name | Size | Bytes | Class | Attributes |
|------|-------|-------|--------|------------|
| x | 1x101 | 808 | double | |
| y | 1x101 | 808 | double | |

- ▶ Zwei Befehle
 - who, whos
- ▶ Zusätzliche Optionen für die beiden Befehle

- ▶ Matlab legt eine Befehlshistorie an
 - Für jeden Nutzer separat
 - Standard: letzte 25.000 Befehle
 - Einstellbar (\Rightarrow „Preferences“)
- ▶ Anzeige der Historie in der GUI möglich
 - Muss ggf. eingestellt werden
- ▶ Historie über Pfeiltasten erreichbar
 -  ruft die Historie auf
 -  und  zum Blättern
 - Anfangsbuchstabe(n) und  für gezielte Suche



```
Command History
doc pi
sin(pi)
4*atan(1) == pi
imag(log(-1))
imag(log(-1)) =...
exp(i*pi)
exp(-i*pi)
clear all; cl...
1+2
a=5; b=3;
c=a+b
c=a*b
clear all; cl...
a(1,2,3,4) = 1
clear all; cl...
m = [1 2 3; 4 5...
m = [1 2 3 4; 4...
```

☞ Kann das Leben mitunter sehr vereinfachen...

Hilfe, mein Matlab müllt zu...

► Problem

- Alle Variablen auf der Kommandozeile sichtbar/erreichbar
- Kommandozeile unübersichtlich vollgeschrieben
- Ggf. unzählige (Grafik-)Fenster offen

► Lösung

- Gezielt Variablen löschen, Fenster schließen, ...

| Matlab-Befehl | Beschreibung |
|--------------------|----------------------------|
| <code>clear</code> | Variable(n) löschen |
| <code>close</code> | (Grafik-)Fenster schließen |
| <code>clc</code> | Kommandozeile aufräumen |

Listing 8: Der ultimative Aufräumbefehl...

```
clear all; close all; clc
```

- ▶ **Drei auf einen Streich:**
 - Löscht alle definierten Variablen (`clear all`).
 - Schließt alle offenen (Grafik-)Fenster (`close all`).
 - Räumt die Kommandozeile auf (`clc`).
- ▶ **Löscht nicht die Befehlshistorie**
- ▶ **Mit Bedacht einsetzen...**
 - Alle Variablen zu löschen, muss keine gute Idee sein.
 - Nicht unbedingt geeignet als Beginn eines Skripts...



...Zeit für eigene praktische Arbeit...

Vorschau: **Skripte und Funktionen**

- ▶ Editor
- ▶ Befehle/Funktionen