



MATLAB für Naturwissenschaftler

12. Ausblick:
MATLAB im echten Leben

Till Biskup

Lehrstuhl für Physikalische Chemie und Didaktik
Universität des Saarlandes



Motivation zur Verwendung von MATLAB

- ▶ Wir haben Daten gemessen und wollen diese Daten auswerten und (sinnvoll) darstellen.

Warum MATLAB

- ▶ Komplexe Auswertungen
 - Nicht mehr einfach bzw. per Hand durchführbar
 - Automatisierung durch Programmierung einzelner Schritte
 - Reproduzierbarkeit und Nachvollziehbarkeit
- ▶ Einfache Erlernbarkeit von MATLAB
 - Programmiersprache mit Ähnlichkeiten zu C und Pascal
 - Grafische Oberfläche
 - Relativ einfach und schnell erste Erfolge



Programmierkenntnisse sind (fast) unerlässlich

- ▶ Kernaspekte der Physikalischen Chemie:
 - Vertieftes Verständnis gemessener Daten
 - Simulation basierend auf den physikalischen Grundlagen

Programmieren ist kein Teil des Studiums

- ▶ Die wenigsten Chemiker können gut programmieren.
- ▶ Einfach zu erlernende Sprachen sind hilfreich.
 - Geschwindigkeit spielt eher selten eine Rolle.
 - MATLAB & Co. sind ein guter Einstieg.



Automatisierung von Routineaufgaben

Toolboxen: Satz aufeinander bezogener Funktionen

Interaktive Datenauswertung (Nutzerschnittstellen)

Ausblick: Programmieren größerer Projekte



Warum Automatisierung?

- ▶ Konsistenz der Ergebnisse
- ▶ Zeitökonomie: der Computer kann nicht denken...

Wann Automatisierung?

- ▶ Wenn klar ist, dass die Aufgabe häufiger auftritt.
- ▶ Wenn die Implementierung weniger Zeit braucht.

Welche Aufgaben eignen sich dafür?

- ▶ Einfache Aufgaben
- ▶ Aufgaben, die keiner Nutzerinteraktion bedürfen



PCG-Assistent: Versuchsauswertung

- ▶ Pro Semester \gg 15 Gruppen
- ▶ Der Fluoreszenz-Versuch war sehr dankbar...
(Messdaten elektronisch, einfach automatisierbar)

Darstellung spektroskopischer Daten

- ▶ Import in MATLAB in der Regel einfach programmierbar.
- ▶ Schnelle Darstellung (mit korrekten Achsenbeschriftungen) hilft für einen ersten Eindruck.
- ▶ „Einfachstes“ Beispiel: UV/Vis-Spektrum



Grundlegend zwei Konzepte:

1. ein Skript für jeden Datensatz
2. eine Toolbox aus Funktionen, die generisch jeden Datensatz verarbeiten kann

☛ Beide Konzepte haben Vor- und Nachteile.

Toolbox

In sich geschlossene Sammlung von Funktionen (Routinen) für eine bestimmte Aufgabe



Toolboxen in MATLAB

- ▶ MATLAB selbst ist sehr modular aufgebaut.
- ▶ Viele hilfreiche zusätzliche Funktionalität ist über Toolboxen realisiert.
- ▶ Toolboxen sind in sich geschlossene Sammlungen von Funktionen für eine bestimmte Aufgabe.
- ▶ MATLAB bietet Unterstützung bei der Entwicklung eigener Toolboxen.
- ▶ Es gibt grundsätzlich zwei Arten von Toolboxen
 1. kommerziell (meist von MathWorks selbst)
 2. nichtkommerziell (meist frei im Netz verfügbar)



Kommerzielle Toolboxen

- ▶ Optimization Toolbox
- ▶ Global Optimization Toolbox
- ▶ ...

Nichtkommerzielle Toolboxen

- ▶ EzyFit Toolbox
- ▶ EasySpin
- ▶ DEER Analysis
- ▶ trEPR Toolbox, TA Toolbox
- ▶ ...



Toolboxen zur Simulation von Spektren



EasySpin – *by Stefan Stoll*

MATLAB toolbox for simulating and fitting
Electron Paramagnetic Resonance (EPR) spectra.

- ▶ *De-facto*-Standard für die EPR-Spektrensimulation
- ▶ Komplette kommandozeilenbasiert
- ▶ Implementiert viele verschiedene Algorithmen
- ▶ Gemeinsame Schnittstelle für alle Simulationen



Toolboxen mit grafischen Schnittstellen





Nutzerschnittstelle

Abstrakte Schicht zwischen dem Nutzer und den eigentlichen Routinen, die dem Nutzer die Bedienung erleichtert.

Zwei Arten von Nutzerschnittstellen

- ▶ Textbasierte Schnittstelle
command line interface, CLI
 - ▶ grafische Schnittstelle
graphical users interface, GUI
- ☞ Jede dieser Schnittstellen hat ihre Vor- und Nachteile.



Textbasierte Nutzerschnittstelle (CLI)

- ▶ Menüs und Nutzereingaben in einer Textkonsole
- ▶ Vollständig deterministisch (bis auf Nutzereingaben)
- ▶ Linear: immer nur eine Entscheidungsmöglichkeit
- ▶ Strukturiert, aber mit wenig Freiheiten

Grafische Nutzerschnittstelle (GUI)

- ▶ Grafische Anordnung von Bedienelementen
- ▶ Reihenfolge der Ereignisse unvorhersehbar
- ▶ Nichtlinear: beliebige Entscheidungsmöglichkeiten
- ▶ Große Freiheit: Alles (implementierte) jederzeit möglich



Beispiel: Textbasierte Nutzerschnittstelle (CLI)

Listing 1: Programm zur Simulation von EPR-Spektren

```
1 Do you wish to simulate or to fit?
2 [f] Fit
3 [s] Simulate
4 [q] Quit
5 Your choice (default: [f]): s
6
7 Do you wish to load experimental data?
8 [y] Yes
9 [n] No
10 Your choice (default: [n]): n
11
12 The simulation parameters currently chosen:
13 g          2.0200    2.0200    2.0200
14 D          3900.0000
15 E          130.0000
16 mwFreq     9.7000
17 nPoints    361.0000
18 Range      260.0000    440.0000
19 Temperature 0.0000    0.4500    0.5500
20 Method     matrix
```



Beispiel: Grafische Nutzerschnittstelle (GUI)





*Code as if whoever maintains your program
is a violent psychopath who knows where you live.*

– *Anonymous*





Was ist ein größeres Projekt?

- ▶ Zeitaufwand
 - Programmierung braucht deutlich länger als eine Woche
- ▶ Komplexität:
 - mehrere Funktionen
 - andere Anwender als der Programmierer

Was sind typische Probleme?

- ▶ Mangelnde Wiederverwertbarkeit
 - fehlende Dokumentation
 - Funktionen nicht ausreichend robust
- ▶ Korrektheit schwer zu überprüfen



Programmieren ist ein Handwerk

- ▶ Informatik hat relativ wenig mit Programmieren zu tun...
- ▶ Programmieren lässt sich lernen.
 - Entscheidend ist die eigene Motivation.
 - Viel läuft über Praxis und Erfahrung.
 - Es gibt viele gute Bücher zum Thema.
- ▶ (Selbst-)Disziplin ist entscheidend.
 - Schlechte Programmierung zahlt sich nicht aus.
 - Meist ist man selbst der nächste Anwender...
- ☞ Code wird in der Praxis viel zu häufig neu geschrieben
– aufgrund mangelnder Wiederverwertbarkeit.



Vorlesung (Wahlpflicht, M.Sc.): „Programmierkonzepte in der Physikalischen Chemie“



👉 voraussichtlich im Wintersemester 2019/20 👈

<https://www.till-biskup.de/de/lehre/programmierkonzepte/>

© Scott Adams, <http://dilbert.com/strip/2009-03-21>