Anwendung von (Mathematica und) Matlab in der Physikalischen Chemie

6. Grundlegende Dokumentation

Albert-Ludwigs-Universität Freiburg

Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Wintersemester 2017/2018

Übersicht

UNI FREIBURG

Dokumentation

- Motivation: Warum dokumentieren?
- Arten von Dokumentation

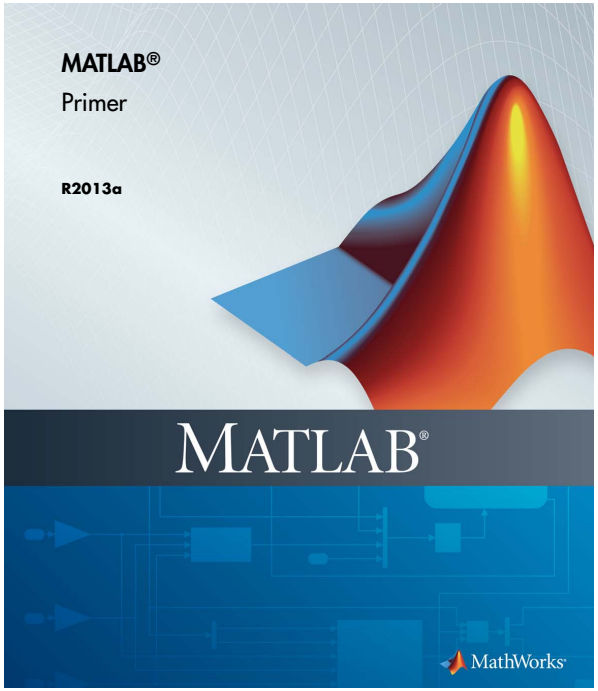
Dokumentation im Code

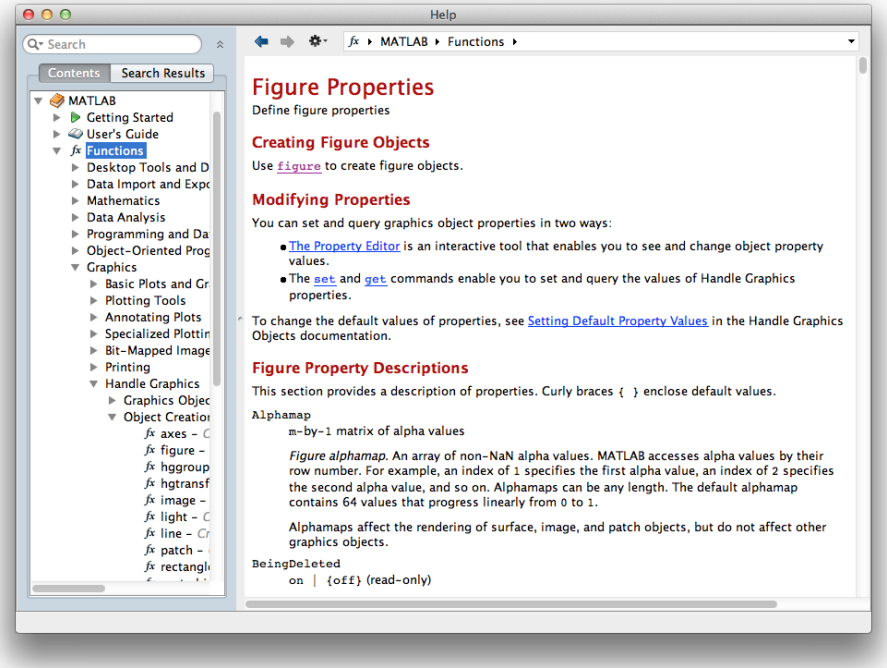
- Kein Code ohne Dokumentation
- Grundlegende Struktur eines Kommentarkopfes
- Dokumentationshilfen

Probleme von Dokumentation

Dokumentation

Entscheidend für gute Code-Qualität





Listing 1: linux-2.2.16/fs/buffer.c

```
1 /*
2  * We used to try various strange things. Let's not.
3  */
```

Real programmers don't comment their code.
If it was hard to write, it should be hard to read.

Warum dokumentieren?

- Weil andere das Programm nutzen/verstehen wollen.
- Weil man sich selbst nach zwei Monaten nicht mehr daran erinnern kann, was man da programmiert hat.
- Weil wir in aller Regel zu schlecht programmieren.
- Weil Dokumentation (gerade von Konzepten) für die weitere Entwicklung sehr hilfreich ist.

☛ Weil nur sauberer (dokumentierter) Code Zukunft hat.

Viele verschiedene Arten

- Im Quellcode
  - Schnittstellen-Dokumentation
  - Quellcode-Dokumentation (einzelne Zeilen)
- Im gleichen Verzeichnis wie die Funktionen/Toolbox
  - `README`
  - `INSTALL`
- Nutzerhandbuch
  - Beschreibung der Verwendung jeder Funktion
- Konzepte
- Beispiele

Unterschiedliche Einteilung

- nach Zielgruppe
  - Programmierer
  - Anwender
- nach Inhalt
  - Quellcode-Dokumentation
  - Schnittstellen-Dokumentation
  - Dokumentation der Konzepte
  - Installation, Bedienung, ... (Anwenderdokumentation)
- nach Medium
  - im Quellcode
  - in Dateien neben dem Quellcode
  - getrennt vom Quellcode (Webseite, ...)

Dokumentation im Code

Kein Code ohne Dokumentation

UNI FREIBURG

Listing 2: linux-2.2.16/lib/vsprintf.c

```
1 /* vsprintf.c -- Lars Wirzenius & Linus Torvalds.
2  *
3  * Wirzenius wrote this portably, Torvalds fucked it up :-)
4  */
```

Listing 3: linux-2.2.16/fs/buffer.c

```
1 /*
2  * After several hours of tedious analysis, the following
3  * hash function won.   Do not mess with it... -DaveM
4  */
```

Listing 4: linux-2.0.38/arch/m68k/atari/atafb.c

```
1 /* Nobody will ever see this message :-) */
2 panic("Cannot initialize video hardware\n");
```

Dokumentation im Code

Kein Code ohne Dokumentation

UNI FREIBURG

Real Programmers don't need comments–
the code is obvious.

— Ed Post, 1983

- Nicht dokumentierter Code ist (oft) wertlos.
- Fehlende Kommentare erschweren das Lesen von Code.
- Dokumentation im Code sollte kurz, prägnant und informativ sein.

☛ Beispiel gefällig?

Ed Post, „Real Programmers Don't Use Pascal", Datamation 29(7), 1983

Dokumentation im Code

Kein Code ohne Dokumentation: Ein Beispiel

UNI FREIBURG

Listing 5: Undokumentierter Code

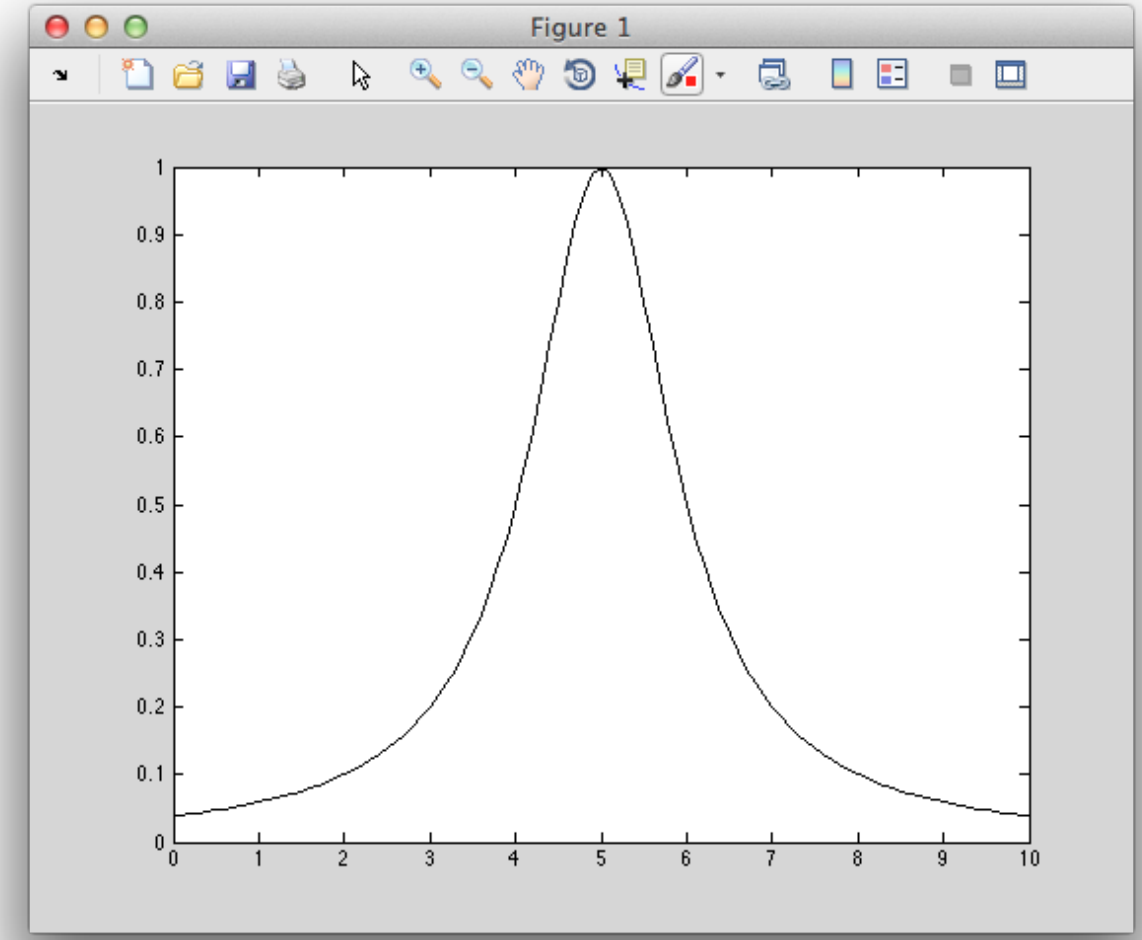
```
1 fun = @(x,s,t)s^2/4./(s^2/4+(x-t).^2);
2 s = 2;
3 t = 5;
4 x = 0:0.1:10;
5 y = fun(x,s,t);
6 plot(x,y,'k-');
```

- Keinerlei Kommentare
- Keine Gliederung des Codes durch Leerzeilen
- Keine oder wenige sprechende Variablennamen

- Irgendeine Idee, was der Code macht?
- Wie könnte der Code vernünftig dokumentiert aussehen?

Dokumentation im Code

Kein Code ohne Dokumentation: Ein Beispiel



Dokumentation im Code

Kein Code ohne Dokumentation: Ein Beispiel

UNI FREIBURG

Listing 6: Dokumentierter Code (lorentzian.m)

```
1  % LORENTZIAN Plot Lorentzian curve with fixed height.
2
3  % Copyright (c) 2013-14, Till Biskup <till.biskup@physchem.uni-freiburg.de>
4  % 2014-01-14
5
6  % Define Lorentzian with fixed height
7  %   s - width of the curve (FWHM)
8  %   t - position of the maximum
9  Lorentzian = @(x,s,t)s^2/4./(s^2/4+(x-t).^2);
10
11 % Define values for Lorentzian curve
12 width  = 2;
13 maxpos = 5;
14
15 % Define x,y vectors
16 x = 0:0.1:10;
17 y = Lorentzian(x,width,maxpos);
18
19 % Plot
20 plot(x,y,'k-');
```

Dokumentation im Code

Kein Code ohne Dokumentation

UNI FREIBURG

Regel

Keine Datei (Routine, Skript) ohne Dokumentation am Anfang.

- Dokumentation ist eine Frage der Disziplin.
- Zum „Rapid Prototyping“ eignet sich die Kommandozeile.
  Skripte werden dokumentiert!
- Dokumentation muss zur Routine werden.
  Im Nachhinein zu dokumentieren ist keine Option,
  weil es dann nie gemacht wird.

Zwei Arten von Dokumentation im Code

1. Kommentarköpfe von Funktionen
   - kurze Zusammenfassung der Aufgaben der Routine
   - präzise Beschreibung der Schnittstelle
   - Autor, Copyright und Datum der letzten Änderung
   - ggf. Hinweise auf Besonderheiten

2. (Meist) einzeilige Kommentare im Code
   - kurze Erklärungen zum folgenden Code-Abschnitt
   - wichtig: nicht das Offensichtliche dokumentieren

☛ Bis auf die Beschreibung der Schnittstelle gilt der Kommentarkopf für Skripte ebenso.

Dokumentation im Code

Grundlegende Struktur eines Kommentarkopfes

UNI FREIBURG

Ein Kommentarkopf besteht mindestens aus drei Teilen

1. Kurze Zusammenfassung der Aufgaben der Routine
   - Mindestens ein Satz, besser ein kurzer Absatz, der die Aufgabe der Routine präzise beschreibt.
2. Präzise Beschreibung der Schnittstelle
   - Funktionsaufruf
   - Dokumentation der Ein- und Ausgabeparameter (Typ, Bedeutung)
3. Autor, Copyright und Datum der letzten Änderung
   - Wichtig: Datum der letzten Änderung immer anpassen!

☛ Der Kommentarkopf liefert dem Nutzer ein Maximum an Information auf minimalem Raum.

Dokumentation im Code

Grundlegende Struktur eines Kommentarkopfes

UNI FREIBURG

Listing 7: Dokumentationsblock zu Beginn einer Routine

```
1  function spectrum = simCO2spectrum(x,fwhm,pos,height)
2  % SIMCO2SPECTRUM Function for simulating CO2 spectra using Lorentzians.
3  %
4  % Usage
5  %   spectrum = simCO2spectrum(x,fwhm,pos,height)
6  %
7  %   x        - vector
8  %              wavelength axis for spectrum
9  %
10 %   fwhm     - vector
11 %              spectral width of each Lorentzian
12 %
13 %   pos      - vector
14 %              position of each Lorentzian
15 %
16 %   height   - vector
17 %              height of each Lorentzian
18 %
19 %   spectrum - vector
20 %              calculated spectrum
21 %              same length as x
22 %
23 % The vectors fwhm, pos, and height have to be of the same size.
```

Dokumentation im Code

Grundlegende Struktur eines Kommentarkopfes

UNI FREIBURG

Listing 8: Copyrighthinweis mit einem Autor

```
1 % Copyright (c) 2011-16, Till Biskup <till.biskup@physchem.uni-freiburg.de>
2 % 2016-07-18
```

Listing 9: Copyrighthinweis mit mehreren Autoren

```
1 % Copyright (c) 1997-2005, A. Kabelschacht
2 % Copyright (c) 2006-2016, Till Biskup <till.biskup@physchem.uni-freiburg.de>
3 % 2016-07-18
```

- Die Angabe einer Email-Adresse ist optional.
- Der Copyright-Hinweis wird durch eine Leerzeile vom Kommentarkopf getrennt.

Dokumentation im Code

Konkrete Hinweise für Matlab

UNI FREIBURG

- Der erste Kommentarblock einer Datei wird beim Aufruf des Befehls `help` ausgegeben.
  - Ähnliches gilt für andere Sprachen und im Zusammenhang mit automatischer Generierung der Dokumentation (Doxygen, Javadoc etc.)
- Da der Copyright-Hinweis nicht ausgegeben werden soll, wird dieser durch eine Leerzeile vom ersten Kommentarblock abgetrennt.
- Symbolworte, die vom `help`-Befehl erkannt werden
  - `see also`

Dokumentation im Code

Ein Wort zur Sprache

UNI FREIBURG

- Kommentare im Code idealerweise auf Englisch
  - Wissenschaft ist international.
  - Japanische oder russische Kommentare im Quellcode sind (für die meisten von uns) leider wenig hilfreich...
- Aber: Lieber deutsche als keine Kommentare!
  - Es ist (meist) einfacher, einen Übersetzer zu finden, als einen undokumentierten Code zu verstehen.
- Grundsätzlich nie Sonderzeichen (z.B. Umlaute)
  - Unicode hat sich (leider) immer noch nicht durchgesetzt.
  - Beschränkung auf den ASCII-7-bit-Zeichensatz

Dokumentation im Code

Der ASCII-7-bit-Zeichensatz

UNI FREIBURG

Hex	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	NUL ^@ 0	SOH ^A 1	STX ^B 2	ETX ^C 3	EOT ^D 4	ENQ ^E 5	ACK ^F 6	BEL ^G 7	BS ^H 8	TAB ^I 9	LF ^J 10	VT ^K 11	FF ^L 12	CR ^M 13	SO ^N 14	SI ^O 15
10	DLE ^P 16	DC1 ^Q 17	DC2 ^R 18	DC3 ^S 19	DC4 ^T 20	NAK ^U 21	SYN ^V 22	ETB ^W 23	CAN ^X 24	EM ^Y 25	SUB ^Z 26	ESC ^[27	FS ^\ 28	GS ^] 29	RS ^^ 30	US ^? 31
20	32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
30	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
40	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
50	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
60	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
70	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

- Zeichen 20_{Hex} bis $7e_{Hex}$ (32 bis 126) sind „druckbar“.

Dokumentation im Code

Dokumentationshilfen

UNI FREIBURG

- Saubere Dokumentation im Code zahlt sich aus:
  - Automatische Generierung von Dokumentation aus den Kommentaren im Quellcode.
- Matlab
  - `help`-Befehl gibt ersten Kommentarblock aus.
  - Interne Funktionen sind gute Vorlagen/Beispiele.
  - Unterstützt das „Publizieren“ von Funktionen
- Dokumentationshilfen für andere Sprachen als Matlab
  - Doxygen, Javadoc, ROBODoc
  - Liefern Ideen für eine sinnvoll strukturierte Dokumentation
  - Erlauben die automatische Erstellung in diversen Formaten (LaTeX, DocBook, RTF, ...)

Häufige Probleme von Dokumentation

Veraltete/falsche Dokumentation

UNI FREIBURG

Problem

- Dokumentation hinkt der Entwicklung des Codes hinterher
  - Problem gerade bei Kommentarzeilen im Code:
    im schlimmsten Fall falsch und irreführend!

Lösungsansatz

- Keine unnötigen Kommentare im Code
  - Offensichtliches nicht dokumentieren
  - Offensichtliche Programmierung der Dokumentation
    einer weniger offensichtlichen Lösung vorziehen
- Beim Ändern von Code immer auf Kommentare achten

☛ Letztlich eine Frage der (persönlichen) Disziplin.

Häufige Probleme von Dokumentation

Dokumentation als Ersatz für Codeverbesserung

UNI FREIBURG

Problem

- Dokumentation statt Verbesserung von Code
  - Unklare Namen (Variablen, Funktionen, ...)
  - Wenig offensichtliche Strukturen

Lösungsansatz

- Sprechende Namen für Variablen, Funktionen, ...
  - Gute Namen zu finden kostet Zeit – spart aber mehr Zeit
- Offensichtliche Verwendung von Code-Strukturen
  - `switch/case` statt `if/elseif`
  - intuitive Formulierung von Bedingungen

☛ Code sollte *lesbar* sein – im eigentlichen Sinne

Häufige Probleme von Dokumentation

Fehlende Dokumentation der Installation/Bedienung

UNI FREIBURG

Problem

- Fehlende Dokumentation der Installation und Bedienung
  - Nichts ist schlimmer als ein Verzeichnis voller Skripte/Funktionen ohne jegliche Beschreibung.

Lösungsansatz

- Externe Dokumentation
  - Extern zu den Funktionen
  - ggf. in eigenem Unterverzeichnis (`doc`)
  - reine Textdateien, kurz und prägnant
  - mindestens `README`, ggf. `INSTALL`

Häufige Probleme von Dokumentation

Fehlende Dokumentation der Konzepte und Ideen

UNI FREIBURG

Problem

- Fehlende Dokumentation der Konzepte und Ideen
  - Schnittstellen-Dokumentation meist nicht ausreichend
  - Grundlegende Konzepte im Kontext dokumentieren
  - Statisches Dokument oft zu unflexibel

Lösungsansatz

- Konzeptionelle Dokumentation in einem Wiki
  - Flexibel
  - Erlaubt einfache Aktualisierungen
  - Geeignet als primäre Informationsquelle für Anwender.
- `README` für kleine Projekte

Ausblick

Wie geht's weiter?

...Zeit für eigene praktische Arbeit...

Vorschau: Datenein- und -Ausgabe

- Text- und Binärdateien
- High- und Low-Level-Routinen