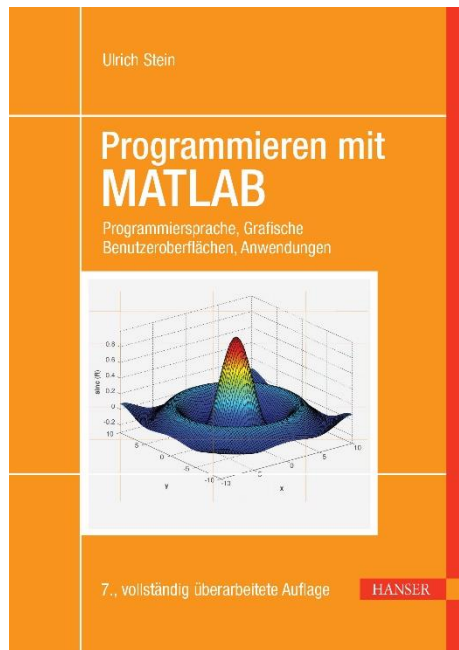


HANSER



Leseprobe

zu

Programmieren mit MATLAB

von Ulrich Stein

Print-ISBN: 978-3-446-47703-2

E-Book-ISBN: 978-3-446-47788-9

Weitere Informationen und Bestellungen unter

<https://www.hanser-kundencenter.de/fachbuch/artikel/9783446477032>

sowie im Buchhandel

© Carl Hanser Verlag, München

Vorwort zur 7. Auflage

Die erste Auflage dieses Lehrbuchs erschien im Jahr 2007. Im Rahmen des Umbaus vom Abschluss Dipl.-Ing. auf Bachelor und Master hatten wir damals, am Department „Maschinenbau und Produktion“ an der HAW Hamburg, das Modul „Angewandte Informatik“ von der Programmiersprache C auf MATLAB® umgestellt – mit der Vorgabe, möglichst alles, was Kernighan und Ritchie [Kern 1988] in der Sprache C behandeln, mithilfe von MATLAB zu lehren.

Gut, bei den Zeigern haben wir versagt. Aber den Rest der Lernziele und Lehrinhalte konnten wir ohne größere Probleme übertragen. Nur fehlte ein Lehrbuch zum Programmieren mit MATLAB. Hier half der Fachbuchverlag Leipzig im Carl Hanser Verlag. Und so entstand dieses Buch, das sich in der Stoffauswahl an unserer Informatik-Vorlesung orientiert, gedacht für die Studentinnen und Studenten der Ingenieurwissenschaften in den ersten Semestern – und jetzt hoffentlich auch für Sie.

Unser Weg durch die Informatik beginnt mit den elementaren Prinzipien der Datenverarbeitung und vermittelt dann anhand der in MATLAB integrierten Programmiersprache systematisch die Grundkenntnisse des Programmierens. Darauf aufbauend vertiefen Anwendungen das Wissen und führen in die weitergehenden numerischen Verfahren von MATLAB ein. Wo für spezielle Probleme das Basismodul von MATLAB nicht mehr ausreicht, wird für eine intensivere Nutzung auf die MATLAB-Toolboxen™ (Add-Ons) verwiesen – deren Installation zum Verständnis dieses Buches jedoch nicht notwendig ist.

Anwendungsbeispiele, zahlreiche Abbildungen und Übungsaufgaben zum Stoff fördern das Verständnis. Das Werk ist sowohl für Programmierneulinge als auch für Umsteiger von anderen Programmiersprachen, wie C oder Java, geeignet.

In diesem Buch wird in MATLAB programmiert. Im Vordergrund steht dabei aber nicht primär das Programm MATLAB, sondern die Vermittlung von Programmierkenntnissen, die auch in vergleichbaren Sprachen die Basis der Programmierung bilden. Das bedeutet, dass nicht immer Wert darauf gelegt wurde, den Programmcode in Bezug auf die Stärken von MATLAB zu optimieren. Auch ist dies kein Referenz-Handbuch für MATLAB. Die MATLAB-Funktionen werden meist nur so weit vorgestellt, wie es für die aktuelle Aufgabenstellung nötig ist. Für eine vollständige Definition der Funktionen sei auf die MATLAB-Hilfe verwiesen.

Dieses Buch soll das Programmieren lehren, aber nicht zum Informatiker ausbilden. Deshalb fehlen Themen wie Automatentheorie, formale Sprachen, Petrinetze, Maschinensprache, Netzwerke und auch die beliebten Umwandlungen vom Dezimal- zum Dualsystem. So auf Du und Du mit Prozessor, Speicher oder serieller Schnittstelle sind Ingenieure heute meist nicht mehr. Informationen zu diesen Themen finden Sie in Büchern zur Informatik, wie im „Informatik-Handbuch“ von P. Rechenberg und G. Pomberger [Rech 2006].

Leider mussten auch die verketteten Listen wegfallen – dazu fehlen in MATLAB die Zeiger. Wie man dies (und andere Spezialitäten) auch in MATLAB realisieren kann, steht in meinem zweiten Lehrbuch „Objektorientierte Programmierung mit MATLAB“ [Stei 2015].

Frau Dipl.-Ing. Erika Hotho, vom Fachbuchverlag Leipzig, machte mir vor über fünfzehn Jahren den Vorschlag, ein Buch über das Programmieren mit MATLAB zu schreiben. Vielen Dank für ihr Engagement und Dank auch an all die Mitarbeiter in Leipzig, die mich jahrelang kompetent betreut haben. Zu Beginn des Jahres 2019 wurde die Abteilung in Leipzig aufgelöst und die Betreuung wechselte ins Stammhaus von Hanser nach München. Danke dort an Frau Natalia Silakova und an Frau Christina Kubiak. Dank auch an alle Kollegen, die mich zu diesem Projekt ermutigten und hilfreiche Tipps gaben. Und einen besonderen Dank an meine Frau, Elfriede Neubauer, die mir bei der stilistischen Überarbeitung eine große Hilfe war. Danke auch für die vielen Zuschriften und die nahezu einhellig positive Reaktion der Leser.

Für die siebte Auflage habe ich einige Teile des Buches umgestellt und an neue MATLAB-Funktionalitäten angepasst. Beispielsweise wird jetzt der Bereich objektorientierte Programmierung stärker hervorgehoben und das dritte Kapitel wurde von GUIDE auf das neue Tool App Designer umgestellt.

Die im Buch beschriebenen und abgebildeten Abläufe beziehen sich auf die Bedienoberfläche der Version **MATLAB R2022b**. Andere MATLAB-Versionen präsentieren sich dem Anwender zum Teil mit einer leicht abgewandelten Oberfläche. Lassen Sie sich dadurch aber nicht verwirren. Die vorgestellten Programme wurden mit verschiedenen Versionen von MATLAB 7.0.1 bis MATLAB R2022b getestet. Erweiterungen und die Lösungen der Aufgaben finden Sie auf

plus.hanser-fachbuch.de

Ich wünsche den Lesern, dass Ihnen das Programmieren neben der Lernarbeit auch Spaß macht und dass Ihnen möglichst viel vom hier präsentierten Stoff im wirklichen Leben bei Problemlösungen nützt. Und nicht verdrängen oder vergessen: Informatik kann auch Schaden anrichten. Deshalb sollte jeder, der programmiert, sich überlegen, ob er sein Tun verantworten kann und will.

Lübeck/Hamburg, im Januar 2023

Ulrich Stein

Inhalt

Vorwort zur 7. Auflage	V
-------------------------------------	----------

1 Einführung	1
---------------------------	----------

1.1 Hello, world.	1
1.2 Datenverarbeitung	3
1.2.1 Hardware	3
1.2.2 Software	5
1.2.3 Datentypen	6
1.2.4 Editieren.	8
1.2.5 Programmausführung	8
1.3 Erster Kontakt mit MATLAB.	9
1.3.1 Der MATLAB-Desktop.	9
1.3.2 MATLAB als Taschenrechner.	10
1.3.3 Zahlen- und Textdarstellung	12
1.3.4 Variablen und Datentypen	14
1.3.5 Vektoren und Matrizen	19
1.3.6 MATLAB aufräumen.	22
1.3.7 Fragen	22
1.3.8 Aufgaben	23

2 Programmstrukturen	24
-----------------------------------	-----------

2.1 Funktionen	24
2.1.1 Eine Black Box.	24
2.1.2 Eingangs- und Rückgabeparameter	26
2.1.3 Funktionen in MATLAB	26

2.1.4	Funktionsbeispiel: <i>Umfang</i>	29
2.1.5	Stack, Funktionsparameter	31
2.1.6	Ablaufprotokoll	35
2.1.7	MATLAB-Arbeitsverzeichnis	36
2.1.8	Fragen	38
2.1.9	Aufgaben	38
2.2	Ein- und Ausgabe	39
2.2.1	I/O-Kanäle	40
2.2.2	Einfache Ausgabe	40
2.2.3	Formatierte Ausgabe	41
2.2.4	Einfache Eingabe.	43
2.2.5	Ein-/Ausgabe-Beispiel: <i>UmfangInput</i>	45
2.2.6	varargs-Mechanismus	46
2.2.7	Fragen	47
2.2.8	Aufgaben	47
2.3	Ablaufstrukturen	48
2.4	Verzweigungen.	49
2.4.1	Bedingungen	49
2.4.2	Vergleiche	50
2.4.3	Logische Verknüpfungen	51
2.4.4	Alternative	52
2.4.5	<i>if-else</i> -Beispiele	55
2.4.6	Fallunterscheidung	58
2.4.7	Fragen	59
2.4.8	Aufgaben	59
2.5	Schleifen	60
2.5.1	Schleifenbedingung	60
2.5.2	Zählschleife	61
2.5.3	Summen- und Produkt-Bildung.	64
2.5.4	Iteration und Rekursion.	68
2.5.5	Verschachtelte Schleifen	69
2.5.6	Wiederholschleife	71

2.5.7	Taylorreihe, Beispiel: e-Funktion.	73
2.5.8	Numerische Verfahren	75
2.5.9	Schleifen verlassen	77
2.5.10	Fragen	78
2.5.11	Aufgaben	78
2.6	Felder	80
2.6.1	Matrizen.	80
2.6.2	Matrix-Beispiel: <i>sinPlot</i>	83
2.6.3	Matrizen erzeugen	85
2.6.4	Der <code>:</code> -Operator und <i>linspace</i>	86
2.6.5	Analyse von Feldern.	88
2.6.6	<i>meshgrid</i>	90
2.6.7	Matrix-Operatoren	92
2.6.8	Verknüpfungen	94
2.6.9	Cell-Arrays.	94
2.6.10	Fragen	96
2.6.11	Aufgaben	97
2.7	Grafik.	98
2.7.1	Grafiktypen	98
2.7.2	2D-Grafik	99
2.7.3	3D-Grafik	107
2.7.4	Mehrere Plots in einer <i>figure</i>	110
2.7.5	3D-Kurven	112
2.7.6	Grafik-Handle	113
2.7.7	Fragen	116
2.7.8	Aufgaben	117
2.8	Strukturen und Klassen	118
2.8.1	Strukturierte Daten.	118
2.8.2	Datenfelder	119
2.8.3	<i>struct</i> -Variablen	120
2.8.4	<i>struct</i> ändern	122
2.8.5	Objektorientierte Programmierung (OOP).	124
2.8.6	Objekt-Arrays, Suchen	132

2.8.7	CAD-Drahtmodell	134
2.8.8	Fragen	139
2.8.9	Aufgaben	139
2.9	Dateien	140
2.9.1	Dateizugriff	141
2.9.2	Dateien auslesen	142
2.9.3	Dateien beschreiben	143
2.9.4	Excel-Dateien	144
2.9.5	MAT-Files	146
2.9.6	Fragen	146
2.9.7	Aufgaben	147
2.10	Strings	148
2.10.1	Character-Arrays vs. <i>string</i> -Klasse	148
2.10.2	String-Funktionen	150
2.10.3	String-Evaluation	153
2.10.4	Fragen	154
2.10.5	Aufgaben	155
3	GUI	157
3.1	Grafische Benutzeroberfläche	157
3.1.1	Das große Warten – Callbacks	158
3.1.2	Einführung in den App Designer	159
3.1.3	Fragen	164
3.1.4	Aufgabe	164
3.2	GUI-Elemente	164
3.2.1	Fenster und Maus	164
3.2.2	Die Klasse <i>app1</i>	165
3.2.3	Text-Ausgabefeld „Label“	167
3.2.4	Text-Eingabefeld „Edit Field (Text)“	169
3.2.5	GUI-Grafikobjekt	171
3.2.6	Drop-Down-Menü	174
3.2.7	Fragen	176
3.2.8	Aufgaben	176

3.3	GUI-Menüs	177
3.3.1	Menu Bar	177
3.3.2	Context Menu	180
3.3.3	Fragen	182
3.3.4	Aufgabe	183
3.4	Standarddialoge	183
3.4.1	Standarddialog-Typen	183
3.4.2	Aufgaben	186
3.5	Callback-Interaktionen	187
3.5.1	Tastatur-Interaktion	187
3.5.2	Timer-Aktion	188
3.5.3	Maus-Interaktion	191
3.5.4	Fragen	194
3.5.5	Aufgaben	194
4	Anwendungen	195
4.1	Akustik: Signalverarbeitung	195
4.1.1	Schwingungen	195
4.1.2	Fourier-Transformation	200
4.1.3	Audio-Funktionen	204
4.1.4	Fragen	206
4.1.5	Aufgaben	206
4.2	Bildverarbeitung	208
4.2.1	RGB-Farbmodell	208
4.2.2	Grafikformate	209
4.2.3	Bilder einlesen	210
4.2.4	Bilder bearbeiten	213
4.2.5	Hoch- und Tiefpass	217
4.2.6	Fragen	221
4.2.7	Aufgaben	221
4.3	Spiel: Projekt Labyrinth	222
4.3.1	Projektstruktur	222
4.3.2	Datenbasis	223

4.3.3	Spiel laden	225
4.3.4	Spielfeld zeichnen.	230
4.3.5	Spielablauf.	232
4.3.6	Fragen	233
4.3.7	Aufgaben	234
4.4	Mathematik: Funktionen.	235
4.4.1	Polynome	235
4.4.2	Kurvendiskussion.	236
4.4.3	Polynom-Fit, Lineare Regression.	239
4.4.4	Datenauswertung	242
4.4.5	Nullstellen	244
4.4.6	Newton-Verfahren.	248
4.4.7	Numerische Integration.	251
4.4.8	Vektorfelder.	253
4.4.9	Fragen	256
4.4.10	Aufgaben	256
4.5	Physik: Differentialgleichungen	257
4.5.1	Federschwingung	258
4.5.2	Differentialgleichungen	259
4.5.3	Numerische Lösung	261
4.5.4	Ungedämpfte Schwingungen.	266
4.5.5	Gedämpfte Schwingungen	269
4.5.6	Erzwungene Schwingungen.	273
4.5.7	Randwertproblem	276
4.5.8	Fragen	281
4.5.9	Aufgaben	282
4.6	Technische Mechanik	283
4.6.1	Zentrales Kraftsystem	283
4.6.2	Lineare Gleichungssysteme.	285
4.6.3	Zusatzaufgabe	287
4.6.4	Gaußsches Eliminationsverfahren	288
4.6.5	Multivariate Regression.	291
4.6.6	Fragen	293
4.6.7	Aufgaben	294

4.7	Regelungstechnik	295
4.7.1	Stehpendel	296
4.7.2	Stabilität	300
4.7.3	Eigenwerte und Eigenvektoren	300
4.7.4	Regelung	304
4.7.5	Control System Toolbox™	307
4.7.6	Simulink®	311
4.7.7	Fragen	317
4.7.8	Aufgaben	317
4.8	Prozess-Kommunikation, Internet	318
4.8.1	COM, OLE und ActiveX.	318
4.8.2	Kontakt zum Internet Explorer	319
4.8.3	Java Virtual Machine (JVM).	324
4.8.4	Fragen	327
4.8.5	Aufgaben	328
4.9	MEX – C in MATLAB	328
4.9.1	C	328
4.9.2	DLL	331
4.9.3	C vs. MATLAB	333
4.9.4	Parameterübergabe.	335
4.9.5	Fragen	338
4.9.6	Aufgabe	338
5	Programmierhilfen	339
5.1	Das Programm läuft nicht!	339
5.2	Der Debugger	344
5.3	Weitere MATLAB-Tools	346
5.3.1	M-Lint Code Analyzer.	346
5.3.2	Profiler	346
5.3.3	Dependency Report.	347
5.3.4	Help Report	347
5.3.5	File Comparison Report	347
5.4	Fragen	347

6	Befehlsübersicht.....	348
	Literatur	359
	Index	362

1

Einführung

Es wird eine Weile dauern, bis Sie flüssig programmieren können. Lassen Sie sich bitte nicht durch die Fülle des Stoffs entmutigen! Und da es motiviert, wenn man möglichst bald ein paar Erfolge erzielt, sollen Sie hier, im Kapitel „Einführung“, zum Warmwerden bereits erste, kleine Schritte ins Reich der Programmierung machen, bevor Sie in Kapitel 2 systematisch die Programmstrukturen erlernen.

■ 1.1 Hello, world

Ein paar Fragen am Anfang dieses Buches – und der Versuch, darauf möglichst einfache und dennoch richtige Antworten zu geben:

Wie lernt man Programmieren?

Im Standardwerk zur Programmiersprache C, dem Buch „The C Programming Language“ [Kern 1988] (auf Deutsch: „Programmieren in C“), schrieben Brian Kernighan und Dennis Ritchie vor über 30 Jahren zu Beginn des ersten Kapitels:

*Eine neue Programmiersprache lernt man nur,
wenn man in ihr Programme schreibt.*

Die erste Programmieraufgabe ist für alle Sprachen dieselbe:

Ein Programm soll folgende Wörter ausgeben:

hello, world

Danach folgen eine halbe Seite Erklärungen, fünf Zeilen Programm-Code, ein Compiler- und ein Programmaufruf, bis der Text endlich auf dem Bildschirm erscheint.

In MATLAB erreicht man dies über einen einzigen Befehl:

```
>> fprintf( 'Hello, world\n' );  
Hello, world
```

Zu MATLAB kommen wir aber erst später. Und die Frage, wie und wo man dies in MATLAB eingibt, wird in den folgenden Kapiteln ausführlich behandelt.

Programmieren erlernt man wie eine Fremdsprache – und zwar, indem man die Sprache ausübt und nicht etwa die Beschreibungen auswendig lernt! Sie müssen sich in der Sprache wohlfühlen und mit der Zeit lernen, sich „darin auszudrücken“. Das heißt, Sie sollen fähig sein, ein technisches Problem in der Sprache zu formulieren und mit den Mitteln der Sprache zu lösen.

Deshalb müssen Sie Programme schreiben, die in diesem Buch erklärten Beispiele nachprogrammieren, sie auch variieren. Und wenn nichts so läuft, wie es auf dem geduldigen Papier geschrieben steht, sollten Sie einen Blick in Kapitel 6 „Programmierhilfen“ werfen.

Warum „Hello, world“ und nicht „Hallo, Welt“?

Deutsch – Englisch. Viele Programmierer verwenden in ihrem Programm-Code ein wildes Gemisch an deutschen und englischen Ausdrücken. Ich muss zugeben – im privaten Umfeld meide ich diese hässlichen Konstrukte wie die Pest. Aber in meinen Computerprogrammen, da pflegen Deutsch und Englisch eine enge Partnerschaft.

MATLAB hat eine englischsprachige Oberfläche. Die Programmiersprache von MATLAB verwendet englische Begriffe. Auch die Dokumentation ist in Englisch abgefasst und die angegebenen Beispiele haben englische Bezeichner. So erhalten Variablen und eigene Funktionen oft auch englische Namen. Diese Fachausdrücke sind nun einmal älter und eindeutiger, die deutschen Übersetzungen oft eher verwirrend und kurios.

Was ist eigentlich MATLAB?

MATLAB[®], eine Abkürzung für MATrix LABoratory, wurde von Cleve Moler und Jack Little in den 1970er-Jahren entwickelt und wird von deren Firma „The MathWorks, Inc.“ vertrieben. MATLAB gehört weltweit zu den bekanntesten Tools zur Berechnung und Simulation komplexer mathematischer und technischer Probleme sowie zur grafischen Darstellung der Ergebnisse.

Die Funktionalität von MATLAB kann auf zwei Arten genutzt werden: zum einen als interaktive Berechnungs- und Simulationsumgebung und zum anderen über den Aufruf von selbst geschriebenen MATLAB-Programmen. Die Programmiersprache von MATLAB, um die es in diesem Buch hauptsächlich geht, verwendet weitgehend die bekannte mathematische Notation. Kontroll- und Datenstrukturen sind ähnlich definiert wie in C – obwohl MATLAB ursprünglich in der Sprache FORTRAN entwickelt wurde.

Der Basis-Datentyp in MATLAB ist die Matrix. Matrizenrechnungen sind deshalb eine der Stärken von MATLAB. In der Industrie und an Hochschulen wird MATLAB

für vielfältige Aufgaben eingesetzt, insbesondere in der Regelungstechnik und der technischen Mechanik. Zusätzlich zum Basismodul sind eine ganze Reihe von Erweiterungen für MATLAB erhältlich, die sogenannten Toolboxen™ (Add-Ons). Dazu gehört beispielsweise Simulink®, eine grafische Oberfläche, mit der man komplexe Systeme modellieren und simulieren kann.

Pro Jahr erscheinen von MATLAB zwei neue Versionen (Releases), ein „Release a“ im Frühjahr und ein „Release b“ im Herbst, wie R2022b. Infos zu Unterschieden zwischen den einzelnen Versionen finden Sie im Internet unter:

www.de.mathworks.com/help/matlab/release-notes.html

Weitere Informationen zu MATLAB und den verfügbaren Toolboxen gibt es im Internet auf den Seiten von „The MathWorks, Inc.“: *www.mathworks.com/* bzw. *www.mathworks.de/* und speziell im Buch und auf den Seiten von Cleve Moler, siehe [Mole 2010].

■ 1.2 Datenverarbeitung

Bevor wir zum eigentlichen Programmieren kommen, wollen wir einen Blick darauf werfen, was wir als Voraussetzung benötigen. Man kann Programme zwar auch mit Bleistift und Papier entwerfen. Wenn Sie dieses Buch lesen, sollten Sie jedoch besser bereits an einem Computer sitzen, um Beispiele und Anwendungen direkt auszuprobieren und sofort eine Rückmeldung über gemachte Fehler zu erhalten.

Und Sie werden am Anfang Fehler machen! Sogar an Stellen, von denen Sie bisher gar nicht wussten, dass man dort Fehler machen kann. Sie können mir glauben: Ihr Deutschlehrer war um einiges toleranter, als MATLAB es sein wird!

1.2.1 Hardware

Der Computer – auch Rechner genannt – ist Ihnen sicher nicht vollkommen unbekannt. Sie haben mit Tastatur und Maus den einen oder anderen Text erstellt und ihn mit einem Drucker aufs Papier gebracht. Sie waren schon im Internet zum Surfen oder zum Chatten. Sie haben sich Filme auf dem Bildschirm angeschaut und Musikstücke auf der Festplatte gespeichert oder auf eine CD oder DVD gebrannt. Damit kennen Sie bereits einen großen Teil von dem, was man Hardware nennt – die „harten“ Bestandteile Ihres Computers, also das, was man sehen und anfassen kann.

Oft gliedert man die Hardware noch in die primären Bestandteile, das heißt, die Teile, die unbedingt nötig sind, damit der Computer funktioniert, wie Tower,

Bildschirm, Tastatur, Maus (oder alles zusammen in einem Notebook oder Tablet), und in die Peripheriegeräte, die man seltener benötigt und die eventuell nicht ständig mit dem Computer verbunden sind, wie Netzwerk, Drucker, Scanner, Kamera, Lautsprecher, MP3-Player, USB-Stick und weitere Geräte an der USB- oder einer anderen Schnittstelle.

In unserer Aufzählung kam als erster und oft schwerster Bestandteil der **Tower**, also das Gehäuse des Rechners, der noch einiges an Innenleben zu bieten hat. Im Tower (oder im Inneren eines Notebooks) befinden sich primär die Komponenten, die beim Austausch von Daten oder bei der gegenseitigen Ansteuerung schnell zusammenarbeiten müssen.

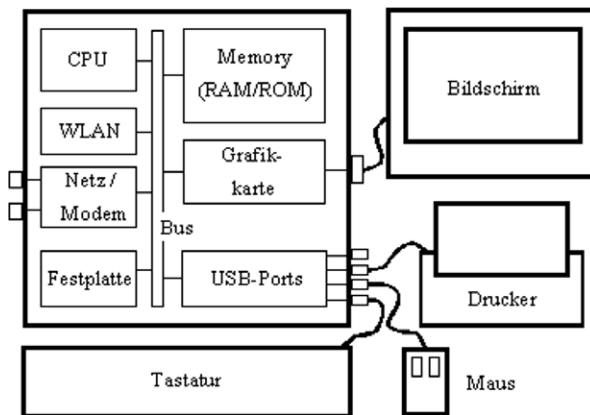


Bild 1.1 Hardware-Komponenten

Auch wenn diese inneren Werte beim Kauf eines neuen Rechners wichtig sind und den Hauptteil des Preises ausmachen, sollen sie hier nur kurz aufgelistet werden:

- CPU (Central Processing Unit/Prozessor): die zentrale Recheneinheit, das Herz des Rechners, das die Programme ausführt, vom Typ Intel, AMD, ...;
- Systemtakt: legt fest, wie viele Operationen pro Sekunde das System ausführen kann, aktuell im GHz-Bereich;
- Speicherbausteine (Memory): dienen zum kurzzeitigen, schnellen Zwischenspeichern von Daten (RAM, aktuell im GByte-Bereich, aufrüstbar) oder enthalten unveränderliche Startdaten für das System (ROM, BIOS);
- Bus: zentrales, schnelles Kommunikationssystem für den Datenaustausch zwischen den einzelnen Komponenten im Tower, auch zur Aufnahme von Erweiterungskarten;
- Grafikkarte: zur Aufbereitung der Daten für den Bildschirm;
- USB-Schnittstelle: Anschlüsse für externe Geräte, wie Tastatur, Maus, ...;
- Festplatten (Hard Disc): langfristiger Datenspeicher;

- CD/DVD-Laufwerk bzw. -Brenner: Speicherung auf externe Medien;
- Netzwerkkarte (LAN): Anschluss an ein Computer-Netzwerk;
- WLAN (Wireless-LAN): Funkschnittstelle zu einem Computer-Netzwerk, damit oft auch Zugang zu einem Internet-Modem, zu Druckern, etc.;
- Bluetooth: zur Kommunikation mit nahen Geräten, wie Maus oder Handy.

1.2.2 Software

Als Software bezeichnet man die Programmteile eines Rechners. Dies sind sozusagen die „Ideen“ – also die Vorschläge, was man mit der Hardware alles anstellen könnte. Auch wenn man Ideen nicht anfassen kann, so sind sie im Allgemeinen doch irgendwo lokalisierbar, bei den Menschen im Kopf und bei den Rechnern als Daten auf einem Speichermedium, wie CD, Festplatte, BIOS oder auch als Virus im Anhang einer E-Mail.

Um zu verstehen, wie aus den Programmdateien die Aktionen werden, die einen Rechner steuern und kontrollieren, müssen wir den Vorgang betrachten, der einen Rechner „zum Leben erweckt“ – den Boot-Vorgang oder kurz: das Booten.

Nach dem Einschalten des Rechners startet die CPU mit der Abarbeitung des BIOS an einer bestimmten Adresse im ROM-Datenbereich. Als Erstes wird ein Test des Systems durchgeführt und dann nach möglichen bootfähigen Betriebssystemen (Windows, Linux, MAC OS, ...) gesucht, entweder auf der Festplatte oder einem externen Medium, wie DVD, Memory-Stick etc. Werden mehrere Betriebssysteme gefunden, bleibt dem Anwender die Auswahl überlassen. Auf den meisten privaten Rechnern ist jedoch nur ein Betriebssystem vorhanden, das in diesem Fall automatisch gestartet wird.

Jetzt, wenn das Laufzeitsystem aktiv ist, übernimmt das Betriebssystem die Kontrolle über den Rechner, und Sie als Anwender können die **Anwenderprogramme** starten, zum Beispiel Word, Computerspiele oder auch MATLAB. Diese Programme befinden sich auf der Festplatte (oder auf einer CD bzw. einem sonstigen externen Medium) in einem bestimmten **Verzeichnis**. Beispielsweise liegt die Programmdatei „MATLAB.exe“ unter MS-Windows bei einer Standardinstallation (der Version R2022b) auf der Partition C der Festplatte im Verzeichnis „C:\Programme\MATLAB\R2022b\bin\win64“.

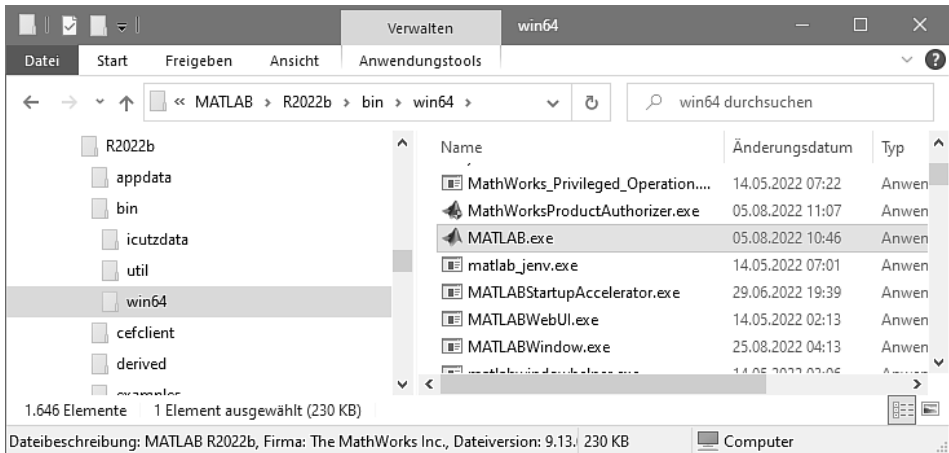


Bild 1.2 MATLAB-Verzeichnis

Beim Starten eines Programms (zum Beispiel durch einen Doppelklick mit der Maus) lädt das Betriebssystem den **Programm-Code** in den Hauptspeicher und reserviert zusätzlich weiteren Speicher des Rechners (RAM) für **Daten**, wie Stack und Heap. Diese Speicherbereiche werden während des Programmablaufs temporär benötigt, um zum Beispiel Zwischenergebnisse festzuhalten.

Das Betriebssystem hat noch weitere Aufgaben neben dem Starten von Anwenderprogrammen – die Verwaltung der Benutzer und deren Zugriffsrechte, die Benutzerschnittstelle mit der Kontrolle über die Eingabe- und Ausgabegeräte wie Tastatur und Maus, die Dateiverwaltung, die Speicherverwaltung für Programme und Daten, das Task-Management zu den laufenden Prozessen und einiges mehr.

1.2.3 Datentypen

Auf den Speichermedien werden alle Daten, ob Programme oder Textdateien, in binärer (zweiwertiger) Form abgelegt. Es gibt nur zwei Zustände für eine Dateneinheit: 0 oder 1. Diese Informationseinheit nennt man ein **Bit**. Zum Arbeiten mit den Daten, also auch beim Programmieren, hat es sich als nützlich erwiesen, nicht direkt auf die einzelnen Bits zuzugreifen, sondern mehrere Bits zur Darstellung unterschiedlicher logischer Datentypen zusammenzufassen, wie

- **Buchstaben** und sonstige Zeichen für Texte,
- **Zahlen** (reell, ganzzahlig, ...) für Rechenoperationen,
- **Wahrheitswerte** (logische Werte: wahr/falsch) für Entscheidungen,
- **komplexere Datentypen** (Vektoren, Matrizen, Strukturen, Klassen, ...).

Als Datenverarbeitung bezeichnet man im weitesten Sinn jede Operation mit Daten, zum Beispiel die Addition zweier Zahlen oder die Ausgabe eines Textes auf dem Drucker. Zur „Verarbeitung“ müssen die Daten im **Speicher** des Computers (RAM) vorhanden sein. Je nach Datentyp benötigen die Daten unterschiedlich große Speicherbereiche.

Zur Darstellung von **Zeichen** verwendet man normalerweise eine Speicherlänge von 8 Bits = 1 Byte. Damit können $2^8 = 256$ Zeichen unterschieden werden. In der **ASCII-Tabelle** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) ist vereinbart, auf welche Zeichen die Zahlen 0 bis 127 verweisen. So hat zum Beispiel der Buchstabe „A“ den ASCII-Wert 65, in hexadezimaler Schreibweise „0x41“ ($65 = 4 * 16 + 1$), und das Leerzeichen „ “ den Wert 32 = „0x20“.

ASCII-Wert: (n) = 0xLR		R						
		0	1	2	3	4	5	...
L	...							
	2	(32)	(33)	" (34)	# (35)	\$ (36)	% (37)	
	3	0 (48)	1 (49)	2 (50)	3 (51)	4 (52)	5 (53)	
	4	@ (64)	A (65)	B (66)	C (67)	D (68)	E (69)	
	...							

Bild 1.3 ASCII-Tabelle (Ausschnitt)

Texte aus mehreren Buchstaben benötigen mehrere Bytes, je nachdem wie viele Zeichen der Text enthält. Die ASCII-Werte von 128 bis 255 sind für spezielle Zeichen reserviert, die sich je nach Land oder Betriebssystem unterscheiden können (Codepages). Asiatische Sprachen kommen mit den 256 darstellbaren Zeichen pro Byte nicht aus. Hier verwendet man pro Zeichen ein Double-Byte (16 Bits) beziehungsweise die Kodierung durch Unicode. Die ASCII-Tabelle und weitere Informationen zu den Kodierungen finden Sie beispielsweise in der Internet-Bibliothek www.wikipedia.de.

Zahlen benötigen je nach Typ (ganzzahlig, reell, mit/ohne Vorzeichen, ...) einen Speicher von 1 bis 8 Bytes und zum Teil mehr. Der Typ *uint8* (unsigned integer 8 Bits) kann mit seinen 256 Werten die vorzeichenlosen ganzen Zahlen von 0 bis 255 darstellen. Da man heutzutage mit dem Speicherplatz nicht mehr ganz so sparsam umgehen muss, verwenden die meisten Sprachen standardmäßig inzwischen die größeren Formate wie *int32* (ganzzahlig, 32 Bits = 4 Bytes) und *double*, also reelle Zahlen in „doppelter Genauigkeit“, die in MATLAB 8 Bytes belegen.

Komplexere Datentypen setzen sich aus den einfacheren Typen zusammen und benötigen deshalb einen größeren Speicherbereich.

1.2.4 Editieren

Dieses Buch beschreibt, wie man eigene Computerprogramme erstellt. **Anwenderprogramme** entstehen aus Textzeilen, die vor dem eigentlichen Programmstart als Textdateien auf der Festplatte in einem Verzeichnis (Arbeitsverzeichnis) abgespeichert werden. Zum Erzeugen dieser Textdateien (Editieren) können Sie jeden **Text-Editor** verwenden, der reinen Text-Code (ASCII-Code) erzeugt, unter Windows beispielsweise den „notepad.exe“ (nicht aber Microsoft Word, da dieses Programm ein spezielles, binäres Dateiformat anlegt). Die verschiedenen Programmiersprachen (MATLAB, C, Java, Visual Basic, ...) stellen meist eigene Editoren zur Verfügung, die an die Besonderheiten der jeweiligen Sprache angepasst sind und beispielsweise die Elemente der Sprache durch eine entsprechende Farbgebung kennzeichnen (Syntax-Highlighting).

1.2.5 Programmausführung

Es gibt prinzipiell zwei Arten, wie Programme im Computer ablaufen:

a) kompilierte Programme:

Hierbei wird vor der Programmausführung aus der Textdatei (mit den Anweisungen des Programms) eine sogenannte ausführbare Datei erzeugt – durch Kompilieren und Linken. Unter Windows haben diese ausführbaren Dateien die Endung “.exe“, weshalb man sie auch als exe-Dateien bezeichnet. exe-Dateien kann man mit einem Doppel-Klick der Maus direkt starten, ohne die Programmierumgebung aufrufen zu müssen. Beim Kompilieren und Linken werden außerdem noch umfangreiche Checks durchgeführt, die prüfen, ob im Programmtext die Regeln der Programmiersprache eingehalten wurden. Aktuell sind C und die Erweiterung C++ die in der Industrie am häufigsten verwendeten Sprachen, die kompilierte Programme erzeugen.

b) interpretierte Programme:

Interpretierte Programme benötigen zur Ausführung immer die Programmierumgebung der verwendeten Sprache. Beim Ablauf des Programms wird jede Anweisung der Textdatei zur Laufzeit interpretiert und ausgeführt. Da der Programmtext vor der Ausführung nicht als Ganzes überprüft wird, werden Programmierfehler bei interpretierten Programmen erst zur Laufzeit erkannt. In MATLAB gibt es jedoch Tools, um die Syntax des Programm-Codes vor der

Ausführung zu testen. Typische Vertreter der interpretierten Sprachen sind MATLAB und die verschiedenen Basic-Dialekte (wobei es hier auch kompilierbare Varianten gibt).

Zu MATLAB gibt es eine Toolbox (MATLAB Compiler™), die es erlaubt, aus Ihrem Code eine kompilierte exe-Version zu erstellen, die dann auf anderen Computern ohne explizit installierte MATLAB-Anwendung lauffähig ist.

■ 1.3 Erster Kontakt mit MATLAB

1.3.1 Der MATLAB-Desktop

Nach dem Starten von MATLAB erscheint auf dem Bildschirm die Standard-Entwicklungsumgebung, der MATLAB-Desktop, der in mehrere Fenster aufgeteilt ist.

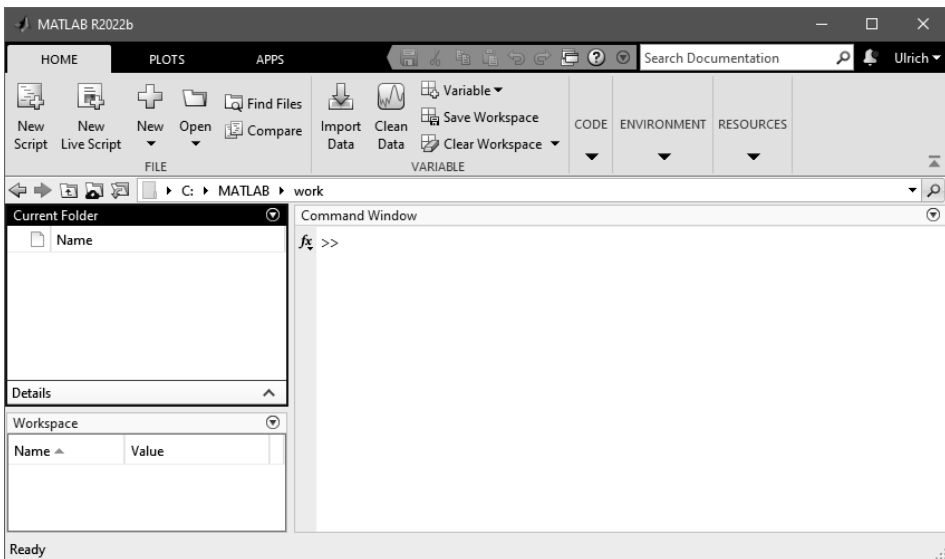


Bild 1.4 MATLAB-Desktop

In dem von mir verwendeten Layout (einstellbar unter Home + Environment + Layout + Two Column) erscheint im rechten Fenster das „**Command Window**“, in dem Sie Ihre MATLAB-Befehle eingeben können. Im Fenster links oben mit dem Titel „**Current Folder**“ haben Sie Zugriff auf alle Dateien im aktuellen Verzeichnis. Darunter im Fenster „**Workspace**“ sehen Sie alle Variablen, die Sie bisher angelegt

haben. Ein Doppelklick auf eine Variable öffnet den „Workspace-Editor“. Die „Command History“ erreichen Sie im Command Window über die Taste „Cursor nach oben ↑“. In der angezeigten Liste können Sie vorher ausgeführte Befehle auswählen, um sie zu wiederholen.

Sie sollten sich angewöhnen, immer in einem definierten **Arbeitsverzeichnis** zu arbeiten. Dieses Verzeichnis wird oberhalb des Command Windows angezeigt. Zum Wechseln des aktuellen Verzeichnisses dienen die Schaltknöpfe über dem Fenster „Current Folder“.

Sie können den MATLAB-Desktop beliebig anpassen. Der Menü-Befehl „Home + Layout + Default“ stellt Ihnen bei Bedarf die Originalkonfiguration wieder her. Über den Befehl „Home + Add-Ons“ haben Sie Zugriff auf weitere Module und die Toolboxen (Add-Ons), die für Ihr System verfügbar sind.

1.3.2 MATLAB als Taschenrechner

Um Ihnen einen ersten Eindruck von MATLAB zu vermitteln, soll in diesem Abschnitt der mathematische Berechnungsteil von MATLAB vorgestellt werden. Gehen Sie nach dem Start von MATLAB in das rechte Fenster, das Command Window. Dort können Sie hinter dem **Prompt** `>>` (der Eingabeaufforderung) einzelne MATLAB-Befehle eintippen. Versuchen Sie es mit folgendem Aufruf:

```
>> 1+1
```

Nach dem Drücken der Eingabetaste (Return/Enter-Taste) antwortet MATLAB:

```
ans = 2
>>
```



MATLAB verwendet für die Systemantwort (answer) die Standardvariable *ans*, falls der Benutzer zum Speichern eines Ergebnisses nicht eine eigene Variable angegeben hat (siehe Abschnitt 1.3.4 „Variablen und Datentypen“).

Sie können MATLAB wie einen Taschenrechner bedienen mit den gewohnten mathematischen Symbolen „+“, „-“, „*“ und „/“, der Potenzierung „^“ oder den runden Klammern:

```
>> (20 + 4 * 4) / 9
ans = 4
>> 3^2
ans = 9
```

MATLAB stellt, wie andere Programmierumgebungen auch, eine Vielzahl von **Funktionen** zur Verfügung, die Sie in den eigenen Programm-Code einbauen

Index

Symbole

|| ODER 51
~ NICHT 51
%-Kommentar 26
\ Links-Division 93, 285, 293
\n Neue Zeile 41
:-Operator 86
. Punkt-Operator 125
/ Rechts-Division 93
&& UND 51

A

Ablaufprotokoll 35
Ablaufstruktur 48
abs 74
Abtasttheorem 201
Achsenbeschriftung 116
Ackermann-Funktion 308
actxserver 319
Add-Ons V, 10, 330
Amplitude 196
Anfangsbedingungen 259
Anfangswertproblem 260
angle 207
Antwort unterdrücken 14, 30
Anwenderprogramm 5
App Designer 157
– Einführung 159
Arbeitsverzeichnis 10
Arduino 144
Argumente 26
arguments 33

Array 19, 80
ASCII-Tabelle 7
Aspect Ratio 212
audioinfo 205
audioread 205
audiorecorder 206
audiowrite 205
Ausgabe
– einfach 40
– formatiert 41
Ausgleichsgerade 239
axes 113
axis 111
axis image 212

B

Backslash 41
Bedingungen 49
Beispiel
– e-Funktion 73
– Ein-/Ausgabe UmfangInput 45
– Funktion Umfang 29
– if-else 55
– Matrix, sinPlot 83
Bilder
– bearbeiten 213
– einlesen 210
Binden 330
Bit 6
bmp-Format 209
Bogenmaß 286
Boot-Vorgang 5

break 77
Breakpoint 344
BVP 277
bvp4c 278
bvpinit 278
Byte 7

C

C 1, 328
C++ 124, 329
CAD
- Beispiel 193
- Drahtmodell 134
calendarDuration 152
Callback-Interaktionen 187
Callbacks 158
cart2pol 255
cart2sph 255
cast 17, 79, 216
cd 36
ceil 243
cell 95
cell2mat 96
Cell-Array 94
char 17
Character-Array 21, 148
class 127
clc 22
clear all 22
clf 106
colorbar 110
colormap 110
color style marker 103
COM 318
Command-Window 164
Compiler 9, 329
Component Browser 163
compose 151
COM-Server 319
Context Menu 177, 180
continue 78
contour3 110
contourf 110
Control System Toolbox 307

conv 238
cos 286
cosd 286
curl 254
Cursor-Tasten 188

D

date 152
Dateien 140
- auslesen 142
- beschreiben 143
Dateiname 27, 36
Datenauswertung 242
Datenfelder 119
Datenkapselung 129
Datentyp 6, 17, 119
- Kennzeichner 41
Datenverarbeitung 3
datetime 152
Debugger 344
delete 324
Design View 160
det 93
deval 269
DGL 259
- numerische Lösung 261
diag 85
diff 254
Differentialgleichungen 259
- gewöhnliche 260
- partielle 260
Digitalisierung 196
digraph 106
disp 40
divergence 254
DLL 331
doc-Funktion 11
Document Object Model 321
DOM 321
double 17
Drop-Down-Menü 174
duration 152
Dynamic Link Library 331

E

Edit Field (Text) 169
 Editieren 8
 e-Funktion 73
 eig 300
 Eigenschaften 124
 Eigenvektoren 300
 Eigenwerte 300
 Eingabe 43
 Eingangsparameter 26
 elseif 55
 eps 51
 error 78
 errorldg 183
 Euler-Verfahren 261
 eval 153
 Excel 144
 Exceptions 342
 exe-Version 9
 eye 85

F

Fallunterscheidung 58
 Farben invertieren 214
 Fast Fourier Transform 200
 fclose 142
 Fehlermeldung 12, 340
 Felder 19, 80
 – Analyse 88
 Fensterfunktion 202
 fft 200
 fft2 217
 fgetl 142
 fgets 143
 figure 100, 104, 113
 File-Identifizier 141
 Filter 217
 findobj 114
 floor 243
 Flussdiagramm 49
 fopen 141
 for 61, 86
 format 40
 Format-Anweisung 41

Fortsetzungszeichen 31, 57
 Fourier-Reihe 200
 Fourier-Transformation 200
 fplot 99
 fprintf 41, 143
 fread 143
 fscanf 143
 function 26
 Function Argument Validation 33
 Function Handle 99, 244
 Funktion 24
 – anonym 99, 244
 – Aufruf 28
 – Kopf 27
 – Körper 27
 – Name 27, 36
 – Parameter 31
 – privat 28
 – Rumpf 332
 – Unterfunktion 49
 fwrite 144
 fzero 244

G

Gaußsches Eliminationsverfahren 288
 gca 114
 (gcf 114
 gco 114
 get 114, 322
 ginput 191
 Gleichungssysteme, lineare 285
 global 18, 35, 301, 341
 Grad 286
 gradient 254
 Grafik 98
 – 2D 99
 – 3D 107
 – Export 100
 – Handle 113
 – mehrere 104
 grafische Benutzeroberfläche 157
 graph 106
 Graufilter 215
 grid on 84

Grundton 198

GUI 157

– Elemente 164

– Grafikobjekt 171

– Klasse app1 165

– Menüs 177

GUIDE 157

H

Handle-Klassen 126

Hardware 3

Header-Datei 332

height 88

Hello, world 1, 41, 330

help-Funktion 11

Heron-Verfahren 250

hidden on/off 109

H-Line 26

Hochpass 217

hold on/off 104

I

if-Abfrage 53

if-else

– Abfrage 53

– verschachtelt 54

ifft2 218

imag 14

image 211, 231

imfinfo 209

imread 210

imwrite 212

input 43, 154

inputdlg 183

int2str 151

int32 17

integral 252

Integration, numerisch 251

Interfaces 318

Internet 318

interp2 91

inv 93

invoke 321

I/O-Kanäle 40

isa 18, 32

ischar 45, 151

isnumeric 45

isreal 237

Iteration 68, 249

J

Java

– in MATLAB 337

– Virtual Machine 324

JVM 324

K

KeyPressFcn 232

Klasse 127

Kommentare 26

Kompilieren 330

Komponenten 120

Konstruktor 127

Kurven

– 3D 112

Kurvendiskussion 236

L

Label 167

lasterr 343

Layout-Editor 159

legend 105

length 88, 150, 151

Lineare Regression 239, 292

Linken 330

linspace 87

Lint 346

Literale 13

Live Editor 31

load 146

logical 17

logischer Ausdruck 50

logische Verknüpfung 51

loglog 106

lower 151

Is 37
L-Value 28

M

MAT-Files 146
MATLAB 2
– Arbeitsverzeichnis 36
– aufräumen 22
– Desktop 9
– Editor 30
– Funktion 26
– Hilfe 11
Matrix 80
– Funktionen 85
– Operatoren 92
– Verknüpfungen 94
Matrizen 19
Maus-Interaktion 191
max 242
mean 242
Menu Bar 177
mesh 109
meshgrid 90, 108
Methoden 126, 318
methods 126
MEX 328
mexFunction 332
m-File 27
min 242
Mittelwert 242
mlapp-File 165
mod 243
movie 283
msgbox 183
Multivariate Regression 241, 291

N

nargin 46
Newton-Verfahren 248
n-Fakultät 66
nnz 88
norm 93
Nullstellen 236, 244
num2cell 96

num2str 45, 151
numel 88
Numerische Verfahren 75, 244, 248, 251, 261

O

Obertöne 198
Objekt 124
– Array 132
Objektorientierte Programmierung 124
ODE 260
ode45 264
OLE 319
ones 85
OOP 124
Optimization Toolbox 243
orderfields 123
Output-Vektor 305

P

Parameterübergabe 335
Pass-by-value 27
PDE 260
persistent 18
Phase 196
place 308
Platzhalter 41
plot 83, 100
plot3 112
Plots, mehrere 110
pol2cart 255
Pol-Vorgabe 308
polyder 237
polyfit 239
Polygonzugverfahren 261
polyint 238
Polynome 235
Polynom-Fit 239
polyval 235
Präprozessor 332
private 129
Profiler 269, 346
Programmausführung 8
Programme

- interpretierte 8
- kompilierte 8
- Programmstruktur 24
- Prompt 10
- properties 124
- Prozess-Kommunikation 318
- public 129
- Punkt-Operator 21, 120
- Push Button 160
- pwd 36

Q

- quad 252
- Quadratische Gleichungen 236
- questdlg 183
- quiver 254

R

- rand 85, 257
- randn 85, 257
- Random Walk 257
- Randwertproblem 276
- readcell 145
- readmatrix 145
- readtable 146
- real 14
- Rechteck-Funktion 198
- Rechteckregel 252
- Regelung 304
- regress 293
- Rekursion 68
- return 78
- RGB-Farbmodell 186, 208
- rmfield 123
- roots 113, 236, 244
- round 243
- Rückführvektor 305, 307
- Rückgabeparameter 26
- Runge-Kutta-Verfahren 264

S

- Sample-Rate 197, 205
- save 146

- Schleife 49
 - äußere 70
 - Bedingung 60
 - innere 70
 - verlassen 77
 - verschachtelt 69
- Schnittstelle 129
- Schranke epsi 74
- Schrittweite 261
- Schwingungen 195
 - erzwungen 273
 - gedämpft 269
 - ungedämpft 266
- Scope 34
- Script 34
- semilogx 106
- semilogy 106
- Sequenz 27, 48
- set 115
- short-circuit 52
- sign 207, 243
- Signal Processing Toolbox 220, 243
- Signalverarbeitung 195, 220
- Simulink 311
- sin 286
- sind 286
- size 88
- Skalarfeld 253
- Skalarprodukt 93
- Software 5
- Solver 264
- sort 243
- Sortieren 243
- sound 205
- Spalte 80
- Spaltentrenner 19, 80
- Speicherklasse 18
- sph2cart 255
- split 151
- sprintf 150
- sqrt 11, 25
- sscanf 151, 229
- Stabilität 300
- Stack 32
- Standardabweichung 243
- Standarddialoge 183

Statistics Toolbox 293
 std 243
 stderr 141
 stdout 141
 str2double 151
 str2num 44, 154
 strcmp 151, 335
 Strg c 340
 string-Klasse 149
 Strings 14, 148
 – Evaluation 153
 – Funktionen 150
 strlen 151
 strlenlength 150
 strncmp 151
 struct 120
 – ändern 122
 Strukturen 120
 Stützstellen 261
 subplot 110
 Suchen 132
 Summen- und Produkt-Bildung 64
 surf 91, 107
 switch 58
 Syntax 43
 Syntax-Highlighting 341

T

Tastatur-Interaktion 187
 Taylorreihe 73
 text 112
 Text 14, 148
 – Ausgabefeld 167
 – Darstellung 12
 – Eingabefeld 169
 Text-Editor 8
 Tiefpass 219
 Timer-Aktion 188
 title 105
 Toolbox V, 10, 220, 243, 253, 295, 311, 331
 Totschleife 340
 Transposition 93
 triu, tril 93
 try-catch 343
 Tupel 80

Typumwandlung 17
 Typzuweisung, automatisch 18

U

Überladen 129
 Überschreiben 131
 uigetfile 183, 226
 uint8 208, 216
 uiputfile 183
 uisetcolor 183
 upper 151

V

varargin 46
 varargs-Mechanismus 26, 46, 332
 Variablen 14
 – Definition 15
 – Deklaration 334
 – Name 15
 Vektor 19, 81
 Vektorfelder 253
 Vererbung 130
 Vergleiche 50
 – Operatoren 50
 Verzeichnis 5
 Verzweigung 49
 view 111
 Visible 322

W

Wertzuweisung 15, 30, 48
 which 38
 while 71
 whos 17
 width 88
 Wiederholschleife 71
 writematrix 144
 writetable 146

X

xlabel 105
 xline 105

xlsread *145*
xlswrite *145*
XTick *115*
XTickLabel *115*

Y

ylines *105*

Z

Zahlen
- Darstellung *12*
- imaginäre *14*
Zählschleife *61*
Zeile *80*
Zeilentrenner *19, 80*
Zeilenumbruch *31, 57*
Zeilenvorschub *41*
Zelle *19, 80*
zeros *82, 85*
Zugriffsart *141*
Zustandsvektor *300*