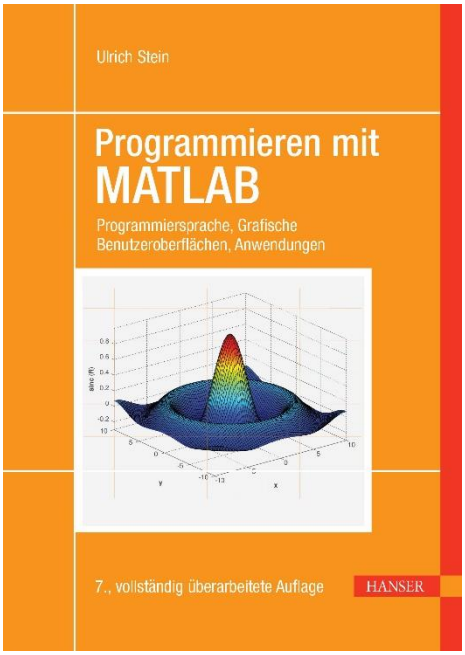
HANSER



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

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 Programm-Code 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 Bedieneroberfläche 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

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.

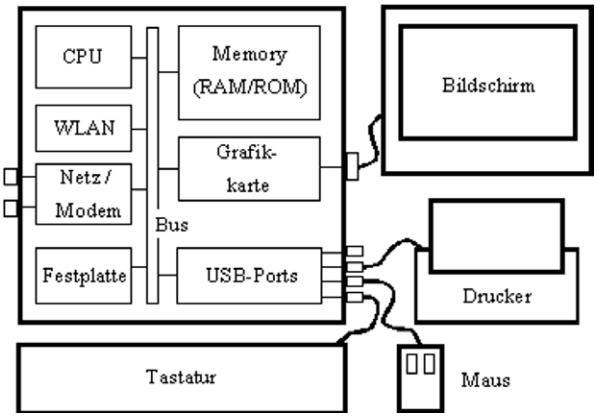


Bild 1.1 Hardware-Komponenten

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 Programmdaten 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\“.

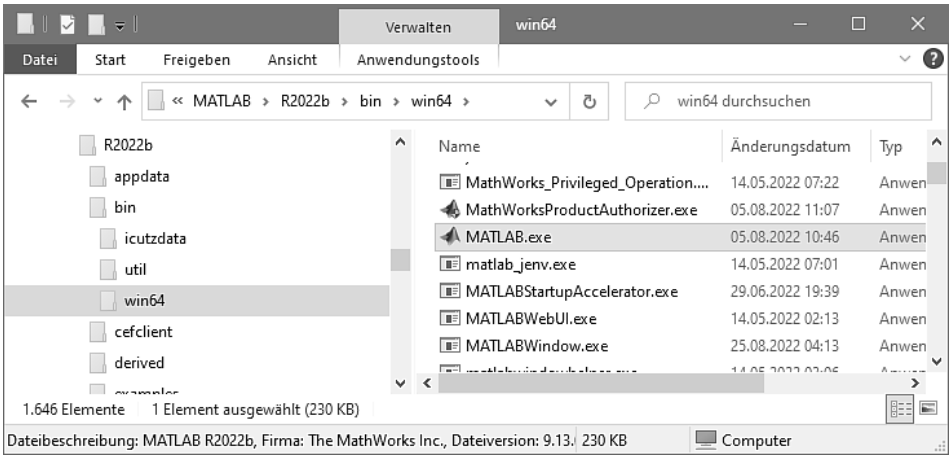


Bild 1.2 MATLAB-Verzeichnis

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.

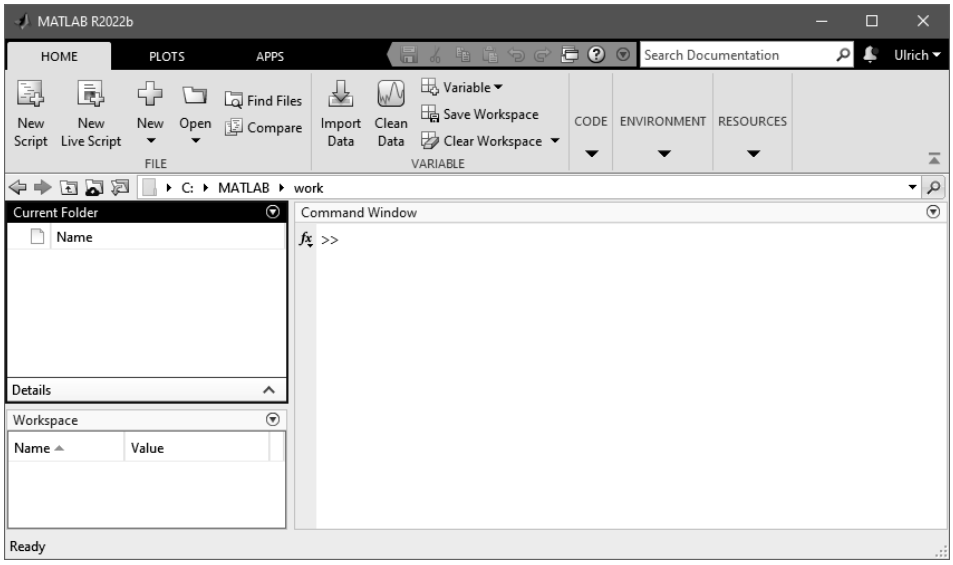


Bild 1.4 MATLAB-Desktop

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

C

D

E

F

G

Q

R

S

T

U

V

W

X

Y

Z