

Leseprobe

Rainer Hagl

Informatik für Ingenieure

Eine Einführung mit MATLAB, Simulink und Stateflow

ISBN (Buch): 978-3-446-44363-1

ISBN (E-Book): 978-3-446-45116-2

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-44363-1>

sowie im Buchhandel.

Vorwort

Dieses Lehrbuch stellt eine Einführung in grundlegende Themen der Informatik für Ingenieure dar. Schwerpunkt ist die zeiteffiziente Analyse und der Entwurf von technischen Systemen im Ingenieurbereich mittels Software aus dem Bereich der computerunterstützten Entwicklung (Computer Aided Engineering, CAE). Das Thema Ingenieurinformatik wird beispielhaft anhand der Entwicklungsumgebung MATLAB, Simulink und Stateflow zur Analyse technischer Systeme und zur Entwicklung elektronischer Steuergeräte verdeutlicht. Diese Entwicklungsumgebung ist in der Industrie weltweit quasi als Standard für die beschriebenen Aufgaben etabliert. Dadurch ist ein hoher Praxisbezug gegeben.

Das Fachbuch ist insbesondere für die Bachelorausbildung von Studierenden der Ingenieurwissenschaften in folgenden Studienschwerpunkten konzipiert:

- Elektro- und Informationstechnik
- Mechatronik
- Maschinenbau
- Automatisierungstechnik
- Energietechnik
- Gebäudetechnik

Es eignet sich ebenso für technisch Interessierte, die sich in die Thematik einarbeiten wollen.

Zunächst wird in der Einführung auf die Entwicklungen eingegangen, die zum heutigen Einsatz von Computern geführt haben. Es wird die weite Verbreitung der computerunterstützten Entwicklung und Fertigung im Ingenieurbereich dargestellt. Grundlegende Kenntnisse, Zusammenhänge und Werkzeuge bei der Softwareentwicklung werden auf Basis der im technischen und naturwissenschaftlichen Bereich weit verbreiteten Sprache MATLAB gezeigt. Diese sind allgemein gültig und lassen sich weitestgehend auf andere Programmiersprachen übertragen. In vielen Projekten ist es hilfreich, grafische Bedienoberflächen zu verwenden. Dieses Themenfeld wird am Beispiel der in MATLAB eingebauten Möglichkeiten dargestellt. Die digitale Arbeitsweise von Computern führt zu mehr oder weniger starken Begrenzungen des Wertebereiches von Zahlen und damit einhergehend auch zu Einschränkungen der Berechnungsergebnisse. Insbesondere bei der Entwicklung elektronischer Steuergeräte, bei denen die Herstellkosten ein wichtiger Faktor sind, ist die Wertebereichsbegrenzung ein wichtiger Punkt bei der Entwicklung. Eine Einführung hierzu wird im Kapitel „Zahlenformate“ gegeben. Viele Berechnungsaufgaben können aufgrund ihrer Komplexität nur auf Computern mit Näherungsverfahren gelöst werden. Grundlegende Zusammenhänge werden im Kapitel „Numerische Integration“ behandelt.

Eine wichtige praktische Anwendung ist die Simulation dynamischer Systeme, welche beispielhaft anhand des Softwarepaketes Simulink gezeigt wird. In vielen Steuerungsaufgaben muss auf Ereignisse mit vorbestimmten Aktionen reagiert werden. Ereignisdiskrete Systeme werden daher in einem separaten Kapitel behandelt. Um die Arbeitsweise praxisnah darzustellen, wird in diesem Kapitel das Softwarepaket Stateflow verwendet. Manche Berechnungsaufgaben müssen aufgrund der vielen Einzelschritte auf mehrere Recheneinheiten verteilt werden, um in akzeptablen Wartezeiten Ergebnisse zu erhalten. Daher wird in einem eigenen Kapitel auf die Thematik „Paralleles Rechnen“ in Grundzügen eingegangen. Mathematische Umformungen und Vereinfachungen können computerunterstützt erfolgen. Im Kapitel „Symbolisches Rechnen“ wird einführend auf diese Möglichkeit eingegangen. Den Kapiteln zugeordnete Übungen erlauben eine Überprüfung des Lernfortschrittes. Weiteres Zusatzmaterial steht den Lesern unter <http://www.hanser-fachbuch.de/buch/Informatik+ fuer + Ingenieure / 9783446443631> zur Verfügung.

Im Buch haben sich sicherlich Fehler eingeschlichen. Vielleicht ist das eine oder andere auch nicht ganz verständlich. Über Rückmeldungen zu Fehlern oder Verbesserungsvorschläge würde ich mich sehr freuen, da diese zu einer kontinuierlichen Verbesserung der schriftlichen Unterlagen führen. Sie können mir diesbezüglich gerne eine E-Mail an rainer.hagl@fh-rosenheim.de senden. Für Ihre Unterstützung möchte ich mich bereits im Voraus bei Ihnen bedanken.

Danksagung

In den vergangenen Jahren wurde aufgrund von Erfahrungen in Vorlesungen und Übungen sowie technischen Weiterentwicklungen das Manuskript, das Basis für dieses Buch war, kontinuierlich angepasst. Diskussionen mit meinem Kollegen Prof. Zentgraf, mit dem ich die „Ingenieurinformatik“ an der Hochschule Rosenheim gemeinsam aufgebaut habe, waren sehr offen und hilfreich. Zu den Kapiteln 2 und 9 hat mein Kollege maßgebliche Anteile konzipiert und mir dankenswerterweise erlaubt, diese zu benutzen.

Für die kritische Durchsicht des Manuskriptes möchte ich mich zudem bei meinen Kollegen Prof. Franz Perschl, Prof. Martin Versen und Frau Julia Höllthaler sowie bei den Studierenden Herrn Johannes Hilverkus, Herrn Maximilian Lindinger, Herrn Florian Planthaler und Herrn Maximilian Stadler sehr herzlich bedanken.

Inhalt

1	Einführung	15
1.1	Historie Rechenmaschinen	18
1.2	Computerunterstützung bei der Lösung mathematischer Aufgaben	25
1.3	Modellbasierte Steuergeräteentwicklung	29
2	Grundlagen der Programmierung	35
2.1	Erste Schritte in MATLAB und Grundregeln	36
2.1.1	Bedienoberfläche	36
2.1.2	Wertezuweisung und Variablendefinition	39
2.1.3	Hilfeunterstützung und elektronische Dokumentation	44
2.1.4	Ein- und mehrdimensionale Felder	47
2.1.5	Arithmetische Operatoren für den Einstieg	49
2.1.6	Relationale und logische Operatoren	51
2.1.7	Sonderzeichen	53
2.1.8	MATLAB Editor	55
2.1.9	Programmbeispiel	61
2.1.10	Script und Function	64
2.1.11	Workspace und Gültigkeitsbereich von Variablen	73
2.1.12	Arbeitsverzeichnisse	75
2.1.13	Fehlersuche und Debugger	78
2.1.14	Freigabe und Initialisierung von Speicherbereichen	82
2.1.15	MATLAB Version	83
2.1.16	Auffinden des Verzeichnisses von Funktionen	84
2.2	Vektoren und Matrizen	85
2.2.1	Teilentnahmen von Elementen bei Vektoren und Matrizen	86
2.2.2	Automatisierte Bestimmung von Indizes	86
2.2.3	Automatisierte Bestimmung der Dimensionen	87
2.2.4	Vorbelegung	88
2.2.5	Automatisiertes Zusammenfügen von Vektoren und Matrizen .	89
2.3	Zeichenketten	90
2.3.1	Grundlagen	90
2.3.2	Klassenumwandlungen	92
2.3.3	Ausführung als MATLAB Anweisung	92
2.4	Structure Array	93
2.5	Cell Array	95
2.6	Objekte	96

2.7	Ablauf- und Kontrollstrukturen	98
2.7.1	If-Verzweigungen	98
2.7.2	Switch-Verzweigung	100
2.7.3	For-Schleife	101
2.7.4	While-Schleife	102
2.7.5	Schleifenunterbrechung (break)	103
2.7.6	Try/catch-Verzweigung	104
2.7.7	Pause	106
2.8	Text einlesen und ausgeben	106
2.9	Daten einlesen und speichern	109
2.9.1	Allgemein übliche Dateiformate	109
2.9.2	MATLAB spezifisches Dateiformat	111
2.10	Grafische Visualisierung	113
2.10.1	Zweidimensionale Visualisierung	114
2.10.2	Dreidimensionale Visualisierung	120
2.11	MATLAB Grundeinstellungen	126
2.11.1	Einrückungen	126
2.11.2	Autosave	127
2.11.3	Kopien von Grafiken in Dokumente	128

3 Grafische Bedienoberflächen 130

3.1	Grafische Elemente (Graphics Objects)	132
3.1.1	Eigenschaften (Properties)	133
3.1.2	Identifizierungskennzeichen (Handle)	136
3.1.3	Abfrage von Eigenschaften	139
3.1.4	Veränderung von Eigenschaften	142
3.1.5	Hierarchie grafischer Elemente	145
3.1.6	Ermittlung von Identifizierungskennzeichen (Handle)	146
3.1.7	Aktuelles Identifizierungskennzeichen (Handle)	148
3.1.8	Festlegung des Achssystems	149
3.1.9	Achsbeschriftungen	150
3.2	Entwicklung grafischer Bedienoberflächen	152
3.2.1	Beispielaufgabe	152
3.2.2	Programmatic GUI	156
3.2.3	Platzierung grafischer Bedienelemente	159
3.2.4	Callback	161
3.2.5	Menüleiste	161
3.2.6	Symbolleiste	164
3.2.7	Ablaufsteuerung	166
3.2.8	Entwicklungsumgebung Guide	167
3.2.9	Ausrichtung grafischer Bedienelemente (Alignment)	174
3.2.10	Eigenschaften grafischer Bedienelemente (Properties)	174
3.2.11	Tags	177
3.2.12	Callback Guide	183
3.2.13	Object Browser	185
3.2.14	Tab Order Editor	186

3.2.15	Datenorganisation	186
3.2.16	Beispiel	187
3.3	Kapselung der grafischen Bedienoberfläche	190
3.3.1	Callbacks als Funktion	191
3.3.2	Lokale Datenhaltung	192
3.4	Guide Template	195
3.4.1	Erzeugung von „function handles“	196
3.4.2	Datenverwaltung	201
3.4.3	Funktionsergänzungen	202
3.5	Animation	203
3.6	Eigenständige Applikationen (Apps)	204
4	Zahlenformate	206
4.1	Ganze Zahlen	206
4.2	Gleitkommazahlen und Festkommazahlen	213
4.3	Zahlenformate in MATLAB	217
4.4	Über- oder Unterschreitung des Wertebereiches	219
4.5	Auflösungsgrenzen bei Berechnungen	220
4.6	Komplexe Zahlen	222
5	Numerische Integration	223
5.1	Mathematische Problemstellung	224
5.2	Explizites Euler-Verfahren	226
5.3	Runge-Kutta-Verfahren	232
5.4	Berechnungsgenauigkeit und Berechnungsdauer	233
5.5	Einschritt- und Mehrschrittverfahren	236
5.6	Verfahren mit variabler Schrittweite	236
5.7	Steife Systeme	238
5.8	Numerische Integration mit MATLAB	239
6	Zeitgesteuerte Systeme (Simulink)	245
6.1	Modellerstellung	248
6.2	Eigenschaften von Blöcken	265
6.3	Simulation	268
6.4	Visualisierung und Weiterverarbeitung der Simulationsergebnisse ...	271
6.5	Dashboard-Blöcke	277
6.6	Externe Beeinflussung von Blockparametern	280
6.7	Hierarchisches Modell und Verbesserung der Übersichtlichkeit	283
6.8	Model Explorer	288
6.9	Physikalische Modellierung	288
6.10	Codegenerierung	294

7	Ereignisdiskrete Systeme (Stateflow)	295
7.1	Entwicklungsumgebung Stateflow	296
7.2	Beispielsystem	301
7.3	Flussdiagramme	302
7.3.1	Modellerstellung	305
7.3.2	Vorgefertigte Musterabläufe	315
7.3.3	Backtracking	318
7.3.4	Designrichtlinien	320
7.4	Zustandsdiagramme	320
7.4.1	Modellerstellung	323
7.4.2	Aktualisierungsbeispiel	333
7.4.3	Super Step	334
7.4.4	Flussdiagramm in einem Zustand	335
7.4.5	Designrichtlinien	337
7.4.6	Hierarchische Modelle	337
7.4.7	History Junction	341
7.4.8	Parallele Zustände	343
7.4.9	Events	345
7.4.10	Funktionsaufrufe	361
7.5	Tabellarische Beschreibung von Zustandsautomaten	366
7.5.1	Wahrheitstabellen	367
7.5.2	Zustandsübergangstabellen	371
7.6	Simulation und Debugging	378
8	Paralleles Rechnen	382
8.1	Vorarbeit serielle Codeoptimierung	385
8.2	Eingebaute Parallelisierung	387
8.3	Auswahl der Hardware-Ressourcen	388
8.4	Parallele for-Schleifen	390
8.5	Batch jobs und Cluster	392
9	Symbolisches Rechnen	399
9.1	Umformen von algebraischen Ausdrücken	400
9.2	Lösung von Gleichungen	401
9.2.1	Lineare Gleichungen	401
9.2.2	Nichtlineare Gleichungen	403
9.3	Taylorreihen	403
9.4	Laplace-Transformation	404
9.5	Integrieren von Funktionen	404
9.6	Differenzieren von Funktionen	405
9.7	Lösung von Differentialgleichungen	406
	Literatur	410
	Index	411

2

Grundlagen der Programmierung

Zur Programmierung von Computern gibt es, ähnlich wie bei natürlichen Sprachen, eine sehr große Anzahl an Programmiersprachen. Diese können sich, wie sogenannte „Maschinensprachen bzw. Assemblersprachen“, sehr nahe an den Abläufen in den elektronischen Schaltungen orientieren. Deutlich abstrakter aufgebaut sind sogenannte „Hochsprachen“, bei denen es meist nicht erforderlich ist, Kenntnisse über die Arbeitsweise der elektronischen Schaltungen zu besitzen. Hierzu zählen z. B. C, C++, Visual Basic^{®*}, Java[™], HTML, MATLAB[®] und Python[®]. Die Anweisungen von Hochsprachen müssen durch Interpreter oder Compiler in für die jeweils verwendete elektronische Schaltung ausführbare Anweisungen umgesetzt werden. „Hochsprachen“ haben zwei wesentliche Vorteile im Vergleich zu „Maschinensprachen bzw. Assemblersprachen“:

- Einfacher, auch für vollständig Unerfahrene, zu erlernen.
- Lösungen für Aufgaben sind damit einfacher und schneller zu formulieren, d. h., die Tätigkeit der Programmierung ist zeit- und kosteneffizienter.

Der wesentliche Unterschied zwischen Interpretern und Compilern bei der Umsetzung ist:

- Interpreter

Die im Programm enthaltenen Anweisungen werden während des Programmablaufes Schritt für Schritt analysiert und umgesetzt. Das Programm wird interpretiert.

- Compiler

Alle im Programm enthaltenen Anweisungen werden vor dem Programmablauf in einem separaten Vorgang vollständig analysiert und umgesetzt. Das Programm wird kompiliert. Erst danach kann der Programmablauf gestartet werden.

Der Vorteil von Interpretern ist, dass der separate Vorgang und zeitliche Aufwand für das Kompilieren entfällt. Entscheidender Nachteil ist der deutlich langsamere Programmablauf. Je öfter das gleiche Programm ausgeführt wird, umso vorteilhafter ist es bezüglich des Gesamtzeitaufwandes, Compiler zu verwenden. Die Grenzen zwischen beiden Lösungen der Umsetzung sind oft fließend und in einigen Hochsprachen werden beide Lösungen angeboten.

In diesem Kapitel werden die Grundlagen der Programmierung am Beispiel der Entwicklungsumgebung MATLAB dargestellt. Viele Methoden und Funktionen gibt es so oder in ähnlicher Weise auch in Programmierungsumgebungen anderer Hochsprachen. Ist ein Grund-

* Visual Basic[®] ist eine eingetragene Marke der Microsoft Corporation.

Java[™] ist eine eingetragene Marke der Oracle Corporation.

Python[®] ist eine eingetragene Marke der Python Software Foundation.

verständnis fürs Programmieren vorhanden, lassen sich vergleichsweise schnell andere Programmiersprachen erlernen.

Das Kapitel ist keine detaillierte Darstellung von MATLAB. Hierfür gibt es die in die MATLAB Entwicklungsumgebung integrierte Hilfe und Dokumentation. Im Kapitel werden vielmehr die wichtigsten Zusammenhänge dargestellt, um einen Überblick für den Einstieg zu erhalten.

■ 2.1 Erste Schritte in MATLAB und Grundregeln

MATLAB ist primär eine

- interaktive Programmierumgebung und gleichzeitig
- eine Programmiersprache

zur computerunterstützten Lösung insbesondere mathematischer und naturwissenschaftlicher Aufgaben. Ebenso wie bei anderen Hochsprachen können in MATLAB Programme erstellt und interpretiert oder kompiliert werden.

Im Internet gibt es viele sehr hilfreiche Schulungsunterlagen, die zum Eigenstudium sehr geeignet sind.



Einführungsvideo Englisch

<http://www.mathworks.de/videos/getting-started-with-MATLAB-68985.html>



Tutorials Englisch

http://www.mathworks.de/academia/student_center/tutorials



Beispiel- und Produktvideos in Englisch

<http://www.mathworks.de/products/MATLAB/demos.html>

2.1.1 Bedienoberfläche

Beim Start von MATLAB erscheint eine in mehrere Bereiche eingeteilte Bedienoberfläche (MATLAB Desktop) (Bild 2.1). Die wichtigsten Bereiche sind markiert.



Das Aussehen der Bedienoberfläche variiert abhängig von der Version und den Voreinstellungen! Es kann vom Anwender im Pull-Down-Menü „Layout“ verändert werden (Markierung in Bild 2.2). Nicht geübte Anwender sollten die Einstellung auf „Default“ stellen.

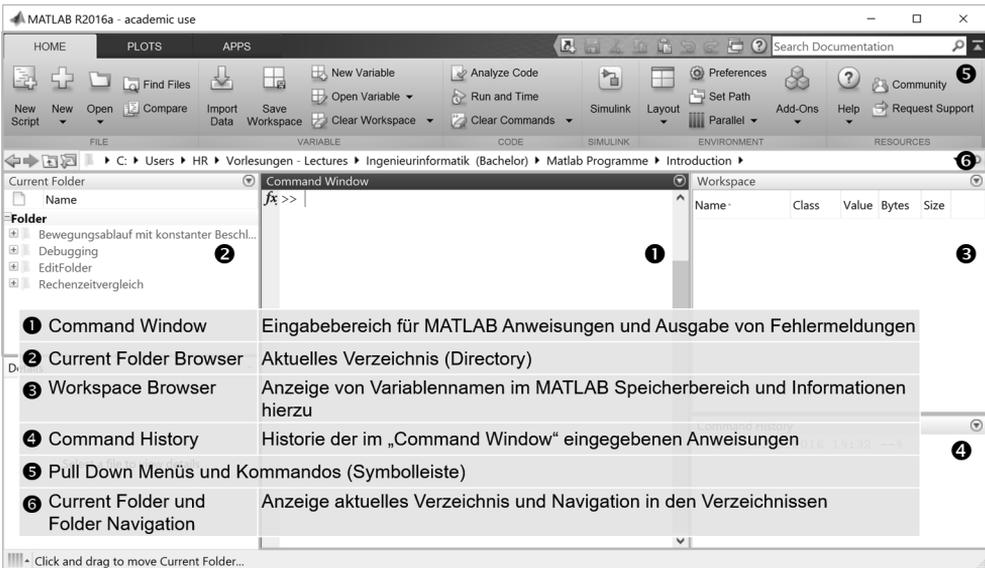


Bild 2.1 Oberfläche beim Programmstart (MATLAB Desktop) und wichtige Bereiche



Bild 2.2 Layout

MATLAB kann im einfachsten Fall wie ein Taschenrechner genutzt werden. Dazu erfolgt im „Command Window“ hinter dem „>>“-Zeichen (Eingabeaufforderung, Prompt) die Eingabe, was berechnet werden soll. Für Grundrechenarten werden die in Tabelle 2.1 gezeigten Symbole (MATLAB Operatoren) benutzt.

Tabelle 2.1 MATLAB Operatoren für die Grundrechenarten

MATLAB	Operator
Addition	+
Subtraktion	-
Multiplikation	*
Division	/

So können z. B. zwei Zahlen multipliziert werden:

2*3

Nach dem Drücken der Eingabetaste (Enter-Taste, ↵), im Folgenden nur noch mit „Enter“ bezeichnet, erhält man folgende Ergebnisanzeige:

```
ans =
     6
```



ans ist eine Abkürzung für „answer“. Gleichzeitig ist es eine Standardvariable von MATLAB, die immer dann benutzt wird, wenn keine eigene Variable zugewiesen wird. Eigene Variablenzuweisungen werden im nächsten Schritt erklärt.

Die neu angelegte Variable ans wird im „Workspace Browser“ angezeigt (siehe Bild 2.3).

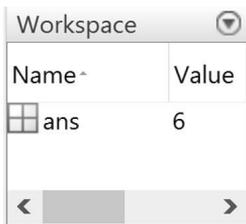


Bild 2.3 Workspace

Immer nach Betätigen der Eingabetaste wird die Eingabe vom MATLAB Interpreter dahingehend überprüft, ob es sich um eine gültige MATLAB Anweisung handelt, d. h. die MATLAB Syntax (Regeln, „Grammatik“) eingehalten wird. Ist dies der Fall, wird die Anweisung ausgeführt, ansonsten erfolgt eine Fehlermeldung.

Wie beim Taschenrechner gibt es vordefinierte Konstanten. Beim Taschenrechner finden sich diese auf vorbelegten Tasten. Bei MATLAB werden sie über Namen angesprochen. So wird z. B. π über den Namen „pi“ eingegeben. Gibt man „pi“ ein, erhält man als Antwort den Wert.

```
pi
ans =
  3.1416
```



Zu beachten ist, dass der Wert der Variablen ans sich dadurch auch automatisch ändert, wie dies in Bild 2.4 des „Workspace Browser“ dargestellt ist.

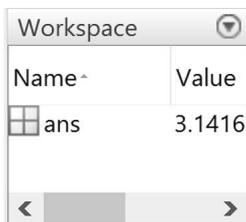


Bild 2.4 Aktualisierung

Damit kann, wie auf dem Taschenrechner, auch der Sinuswert bei 90 Grad ($\pi/2$) berechnet werden. Die Anweisung und Ergebnisanzeige haben folgendes Aussehen:

```
sin(pi/2)
ans =
    1
```

2.1.2 Wertezuweisung und Variablendefinition

Wertezuweisungen erfolgen mit dem Gleichheitszeichen (=). Die Befehlseingabe kann mit einem Komma (,) oder Semikolon (;) abgeschlossen werden. Nach Betätigung von „Enter“ erfolgt die Befehlsausführung. Zugewiesene Werte werden automatisch im MATLAB Arbeitsspeicher, dem „Workspace“, gespeichert. Berechnungen werden unmittelbar ausgeführt. Wird der Befehl ohne Semikolon oder mit einem Komma abgeschlossen, so erfolgt eine Anzeige des Befehlsergebnisses im „Command Window“. Bei einem Befehlsabschluss mit Semikolon erfolgt keine Anzeige. Die Variablen im Workspace werden im „Workspace Browser“ mit Wert (Value) und anderen beschreibenden Größen angezeigt. Die Größen, die angezeigt werden, sind abhängig von den durch den Nutzer im „Workspace Browser“ gewählten Einstellungen.



In MATLAB werden grundsätzlich englische Begriffe, Bezeichnungen und Notationen verwendet.

→ Dezimalzahlen: 0.5 **nicht:** 0,5



Als Variablennamen sind alphanumerische Zeichen, Unterstrich und nicht führende Nummern erlaubt.

In MATLAB werden folgende Zahlenformate unterstützt:

- Ganze Zahlen
Beispiel: -2, 0, 100
- Dezimalzahlen
Beispiel: -2.0, -1.25, 0.33
- Gleitkommazahlen
Beispiel: -1.25 10⁻³, 2 10⁹

Die Eingabedefinition (Notation) ist eine Trennung von Mantisse und Exponent mit dem Buchstaben e.

Eingabe in MATLAB für die Beispiele: -1.25e-3, 2e9

- Komplexe Zahlen
Beispiel: 3+2i, -1.25+4 10⁹i

- Die Eingabedefinition (Notation) ist eine Summe aus Realteil und Imaginärteil. Beim Imaginärteil wird ein i angehängt. i ist dabei der Platzhalter für die imaginäre Einheit ($i^2 = -1$) und eine interne Konstante von MATLAB.
- Eingabe in MATLAB für die Beispiele: $3+2i$, $-1.25+4e9i$

Es gibt alternative Eingabemöglichkeiten. So kann z. B. die Zahl 0.33 auch als .33, +0.33 oder +.33 eingegeben werden. Die rechnerinterne Darstellung von Zahlen und das Arbeiten mit komplexen Zahlen in MATLAB werden im Kapitel 4 (Zahlenformate) behandelt.

Am Beispiel aus Bild 1.15 sollen die ersten Schritte erklärt werden. Es wird von einer Masse von 1200 kg ausgegangen, auf die eine konstante Kraft von 800 N wirkt. Die Anweisungen für die Wertezuweisung zu den Variablen für die Masse (m) und die Kraft (F) lauten:

```
m=1200;
F=800;
```

Für die Berechnung der konstanten Beschleunigung ist die Anweisung:

```
a=F/m,
```

Da die Anweisung mit einem Komma abgeschlossen ist, erfolgt eine Ausgabe des berechneten Beschleunigungswertes.

```
a =
    0.6667
```

Nach diesen Anweisungen stellen sich das „Command Window“, der „Workspace Browser“ und die „Command History“ wie folgt dar (Bild 2.5).

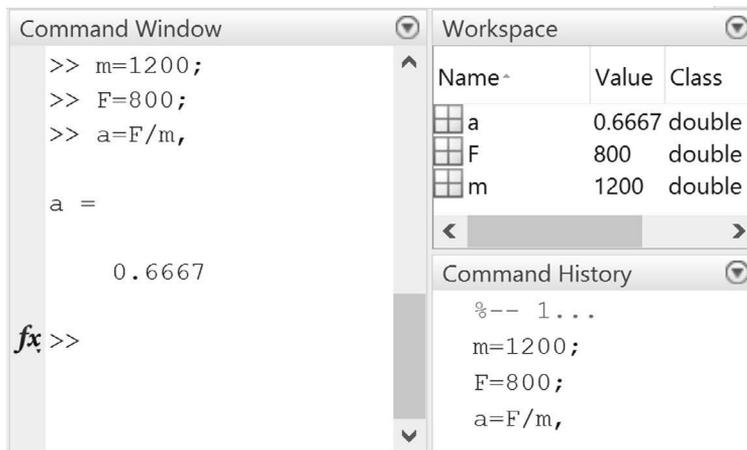


Bild 2.5 „Command Window“, „Workspace Browser“ und „Command History“

Allgemein gilt für Variablenzuweisungen (Variablendefinitionen) in MATLAB:

Name = Class(Value)

Im Deutschen werden folgende Begriffe verwendet:

Variablenname = Datentyp(Wert)



Der Name einer Variable sagt nichts über deren „Inhalt“ (Datentyp, Class) aus. Es kann sich unter anderem um eine Zahl, ein Textzeichen oder einen Wahrheitswert handeln.

MATLAB erlaubt es, dass der Datentyp (Class) nicht angegeben wird. Es wird automatisch der passende Datentyp gewählt. Zunächst soll nur die automatische Zuweisung des Datentyps benutzt werden. Im Kapitel 4 „Zahlenformate“ wird beispielhaft für Zahlen erläutert, wie Datentypen vom Nutzer definiert werden können und welche Auswirkungen dies auf den Speicherplatzbedarf hat. Bei Zahlenwerten wird von MATLAB automatisch der Datentyp (Class) „double“ gewählt.

Nachdem die Zuweisung eines Zahlenwertes oben bereits erläutert wurde, soll im Folgenden die Zuweisung von Textzeichen und Wahrheitswerten gezeigt werden. Die Zuweisung von Textzeichen oder Zeichenketten (Englisch: character oder string) erfolgt durch Setzen des zuzuweisenden Textes in Hochkomma:

```
s='Hochschule Rosenheim';
```

Der Variablenname ist s (Name: s), der Datentyp wird durch die Hochkommas automatisch character (Class: char) und der Wert ist Hochschule Rosenheim (Value: ‚Hochschule Rosenheim‘).

Die Zuweisung von Wahrheitswerten (Englisch: logical values), auch boolesche Werte genannt, die nur wahr (true) oder falsch (false) als Wert annehmen können, ist:

```
x=true;
```

Der Variablenname ist x (Name: x), der Datentyp ist logical (Class: logical) und der Wert ist 1 (Value: 1). Bei „false“ wäre der Wert 0.

Den Workspace nach all diesen Zuweisungen zeigt Bild 2.6. Dort ist tabellarisch für alle bis dato definierten Variablen der Variablenname (Name), der Datentyp (Class) und der Wert (Value) übersichtlich dargestellt. An den Icons am jeweiligen Namen ist auch bereits der Datentyp erkennbar.

Welche Größen angezeigt werden und deren Reihenfolge kann vom Nutzer ausgewählt werden. Wie man zur Auswahl der Größen gelangt, zeigt Bild 2.7.

Name	Value	Class	Bytes	Size
a	0.6667	double	8	1x1
F	800	double	8	1x1
m	1200	double	8	1x1
s	'Hochschule Rosenheim'	char	40	1x20
x	1	logical	1	1x1

Bild 2.6 Workspace nach Zuweisungen von Zahlen, eines Textes und eines Wahrheitswertes

Name	Value	Class	Bytes	Size
a	0.6667	double	8	1x1
F	800	double	8	1x1
m	1200	double	8	1x1
s	'Hochschule Rosenheim'	char	40	1x20
x	1	logical	1	1x1

- ✓ Name
- ✓ Value
- ✓ Size
- ✓ Bytes
- ✓ Class
- Min
- Max
- Range
- Mean
- Median
- Mode
- Var
- Std

- New Strg+N
- Save Strg+S
- Clear Workspace
- Refresh F5
- Choose Columns >
- Sort By >
- Paste Strg+V
- Select All Strg+A
- Print... Strg+P
- Page Setup...
- Minimize
- ☐ Maximize Strg+Umschalt+M
- 📌 Undock Strg+Umschalt+U
- ✕ Close Strg+W

Bild 2.7 Auswahl der Größen, die im „Workspace Browser“ angezeigt werden



Bei Namen von Variablen ist es aus Gründen der Übersichtlichkeit und der geringeren Gefahr von Verwechslungen und damit Fehlern vorteilhaft, keine Abkürzungen, sondern sogenannte „sprechende Namen“ zu verwenden. Dies bedeutet zwar zunächst mehr Schreibaufwand, aber steigert insgesamt die Arbeitseffizienz.

Beim obigen einfachen Beispiel für die Berechnung der Beschleunigung ist dies nicht zwingend erforderlich. Mit sprechenden Variablennamen würde die Berechnung beispielsweise folgendermaßen aussehen.

```
Masse=1200;
Kraft=800;
Beschleunigung=Kraft/Masse;
```

Entsprechendes gilt für später behandelte Programm- und Dateinamen.

In Programmiersprachen, wie MATLAB, C oder C++, werden Zuweisungen von rechts nach links gelesen. So bedeutet:

```
F=F+200,
```

dass zum aktuellen Wert in der Variablen für die Kraft (Name: F; Value: 800, wie oben bereits definiert) der Wert 200 addiert wird. Der Ergebniswert 1000 wird nach links übergeben. Nach der Ausführung der Anweisung (Enter) erscheint im „Command Window“:

```
F =
    1000
```

Der Wert für die Kraft ist dann 1000 (Value: 1000) und wird auch so im „Workspace Browser“ angezeigt.



Anweisungen sind nicht als Gleichung im mathematischen Sinne zu deuten. Die oben gezeigte Anweisung wäre mathematisch falsch.



Zahlenvariablen haben keine Einheiten, sondern nur einen Wert. Für die Kraft kann nicht $F = 100 \text{ N}$ zugewiesen werden. Derjenige, der eine Variable definiert, muss selbst entscheiden, in welcher Einheit der Wert angegeben wird, und sich diese „merken“ und „dokumentieren“. Bei physikalischen Größen wird grundsätzlich empfohlen bei Werten von Variablen SI-Einheiten (Système international d'unités, Internationales Einheitensystem) zu nutzen. Dabei können Basiseinheiten oder daraus abgeleitete Einheiten mit besonderem Namen verwendet werden (Tabelle 2.2 und Tabelle 2.3). Dies ist insbesondere bei komplexeren Programmen übersichtlicher und weniger fehleranfällig.

Tabelle 2.2 SI-Einheiten – Basisgrößen und Basiseinheiten

Basisgröße und Dimensionsname	Größensymbol	Einheit	Einheitenzeichen
Länge	l	Meter	m
Masse	m	Kilogramm	kg
Zeit	t	Sekunde	s
Stromstärke	I	Ampere	A
Thermodynamische Temperatur	T	Kelvin	K
Stoffmenge	n	Mol	mol
Lichtstärke	I_v	Candela	cd

Tabelle 2.3 SI-Einheiten – abgeleitete Einheiten mit besonderem Namen (Auszug)

Größe	Einheit	Einheitenzeichen	In anderen SI-Einheiten	In SI-Basiseinheiten
Winkel (Ebene)	Radian	rad	m/m	1
Frequenz	Hertz	Hz		1/s
Kraft	Newton	N	J/m	kg m/s ²
Druck	Pascal	Pa	N/m ²	kg/(m s ²)
Energie	Joule	J	Nm, Ws	kg m ² /(s ²)
Leistung	Watt	W	J/s, VA	kg m ² /(s ³)
Elektrische Ladung	Coulomb	C		A/s
Elektrische Spannung	Volt	V	W/A; J/C	kg m ² /(A s ³)
Elektrischer Widerstand	Ohm	Ω	V/A	kg m ² /(A ² s ³)

Bevor ein Variablenname in einer Anweisung auf der rechten Seite verwendet werden kann, muss der Variablen ein Wert zugewiesen werden. Im bereits benutzten Beispiel für die Berechnung der Beschleunigung ist folgende Reihenfolge nicht möglich und führt zu einer Fehlermeldung:

```
m=1200;
a=F/m;
Undefined function or variable 'F'.
F=800;
```

Die Variable ‚F‘ hat in der zweiten Anweisungszeile noch keinen Wert. Es nützt nichts, wie hier gezeigt, in der dritten Zeile der Variablen einen Wert zuzuweisen. Die zweite Zeile wird nicht mehr ausgeführt und damit kein Wert für die Beschleunigung berechnet.



Anweisungen werden von oben nach unten ausgeführt.

Fehlermeldungen werden grundsätzlich in roter Schrift in Englisch angezeigt.



Die Ursache von Fehlermeldungen ist immer zu beheben, bevor weitergearbeitet werden kann.

2.1.3 Hilfeunterstützung und elektronische Dokumentation

Die Eingabe von help im „Command Window“ liefert eine Übersicht zu Themengebieten, für die Informationen zur Verfügung stehen. Wird das gesuchte Themengebiet angeklickt, gibt es jeweils weitere Informationen.

Index

A

Ablaufstrukturen 98
Achsbeschriftung 150
Achssystem 149
Action 321
Action table 368
Adams-Bashforth-Verfahren 236
Align Objects 173
analytische Lösung 228
Anfangswert 224
Anfangswertproblem 224
Animation 203, 379
Anonymous Function 70
Anweisung 325
Arbeitsverzeichnis 75
arithmetische Operatoren
(Arithmetic operators) 49
ASCII 91
Assemblersprache 35
Auflösung 220
Ausgabeargumente 64
Ausgabengrößen 64
automatische Codegenerierung 29
axes 137, 158
Axes objects 132, 156

B

Base Workspace 73, 190, 192
Batch jobs 392
bedingte Aktion 303, 309
Bedingung 303, 309
benamte Felder (named fields) 93
Berechnungsfehler 228, 234
Berechnungsgenauigkeit 223, 235
Berechnungszeit 234
Binärzahlen 206
biologische Systeme 245

black box 64
Blockorientierte Modellierung 245
Breakpoint 379
broadcast 345
Browser-basierte Hilfe 45
built-in functions 64
Button Group 158

C

Callback 161, 166, 183, 191, 192, 196,
266, 314
Callback Function 266
Callbacks 314
case 100
catch 104
cd 76
cell 95
cell2mat 95
Cell Array 95, 96
cells 96
char 91
character 41
Chart 298, 323
Chart-level 327
Chart Properties 304
Check Box 157
children 146
Class(Value) 41
clear 82
Cluster 392
coeffs 400
Command Window 39
Comment (Kommentar) 331
Compiler 35
Computeralgebrasystem 27
Condition 303, 309, 331, 372
Condition Action 309, 331, 372

Condition table 368
Configuration Parameters 268
Connective junctions 302
Constant-Block 264
Core 382
createJob 394
createTask 394
Current directory 75
Current folder 75
Current working folder 75

D

Dashboard 342
Datentyp(Wert) 41
Debugger 379
Debug-Modus 81
Default Transition 302, 320, 373
delete 395
Dezimalzahlen 39, 206
diag 88
diff 405
Dimension 87
diskrete Elektronik 23
Diskretisierungsfehler 233, 234
disp 107
Distributed computing 383
Divisionsmethode 207
Dokumentation 44
double 217
dreidimensionale Visualisierung 120
dsolve 406

E

Edit Box 157
Editor 173
Eigenfrequenz 154
Eigenkreisfrequenz 154
Eigenschaften 241, 265
eigenständige Applikation (Apps) 204,
230
eindimensionale Felder 47
Eingabeargumente 64
Eingabedefinition (Notation) 40
Eingabegrößen 64

elektronische Steuergeräte 29
else 99
elseif 99
elseif-Verzweigung 99
Embedded Coder 294
end 89
Endlosschleife 103
ENIAC 21
ereignisdiskrete Systeme 29
eval 92, 93
evalc 93
evalin 93
Event (Ereignis) 331, 345
Event Name 346
Execution Order 327, 344
Exklusive Zustände 297, 343
expand 401
eye 88

F

false 41
fetchOutputs 395
fields 96
figure 132, 137
Figure object 132, 156
Figure Resize Box 173
Finanzsysteme 245
find 86
findobj 147
Flip Block 257
Flussdiagramm 29, 301, 302
Formatierungsoperatoren 108
For-Schleife 101
Foundation Library 289
fprintf 107
function 55, 64, 65
Function-Call-Subsystems 355
Function Functions 69
function handle 191
Function Handle 69, 240, 394
Function Workspace 73
Funktionsauswertung 227

G

ganze Zahlen 39, 206
gca (get current axes) 148
gcbo 184
gcf (get current figure) 148
gcp 389
geschlossene Lösung 26, 27
get 140, 147, 181
Gleitkommazahlen 39
globaler Fehler 234
Global Function 71
Global Variables 73
grafische Visualisierung 113
grid 150
guidata 201
Guide 167
Guide Template 195

H

Handle Graphics® 132
Hardware-in-the-Loop 31
Help 45
Help Browser 46
Hierarchie 146
High Byte 210
Hilfeunterstützung 44
History Junction 341
Hochsprachen 35
hold off 117
hold on 117

I

Icon 185
Identifizierungskennzeichen (Handle)
149, 191
If-Verzweigungen 98
importdata 111
Inf 50
Inherit 313
Initial condition 264
Inline Functions 73
input 106
int 400
int2str 92

int8 217
int16 217
int32 217
int64 217
integer numbers 206
integrierte Schaltkreise 23
Interpreter 35
intmax 218
intmin 218
isempty 74

J

Job 394

K

Kapselung 190
Klassenumwandlung 92
Knotenpunkte 301, 302
komplexe Zahlen 39, 206, 222
kontextsensitive Menüleiste 163
Kontrollstrukturen 98

L

Labs 385
laplace 404
LaTeX-Notation 117
Layout Editor 170
Least Significant Bit 210
Leibniz'sche Rechenmaschine 19
length 87
Line object 132
LineSpec 115
Linienart 115
Linienfarbe 115
List Box 157
load 113
local functions 71, 196, 202
logical values 41
logische Operatoren 51
logische Verzweigungen 98
lokale Datenhaltung 192
lokaler Fehler 234
Lösungsalgorithmus solver 242

Lösungsverfahren 230, 237, 239, 241
Low Byte 210

M

main function 70, 71, 196
Maple® 27
Markierungen 115
Maschinensprache 35
Masse 224
mat 111
Mathematica® 27
MATLAB Compiler 205
MATLAB Editor 55
MATLAB Grafikfenster 114
MATLAB Interpreter 92
MATLAB Lizenznummer 83
MATLAB Scheduler 393
MATLAB Syntax 92
MATLAB Version 83
MATLAB Versionsnummer 83
Matrix 48
Matrizen 85
mehrdimensionale Felder 47
Menu Editor 173
meshgrid 123
Mikroprozessor 23
Mittelpunktsregel 236
Model Browser 286
Model Explorer 313, 347, 358, 361, 369
Model Hierarchy 313
modellbasierte Entwicklung 29
Moore'sches Gesetz 25
Most Significant Bit 210
Multithreaded functions 387
Musterabläufe 315

N

Näherungslösung 223
Name 41
NaN 51
negative Zahlen 211
Nested Functions 73
New Script 55
normalized 160, 176

num2str 92
numerische Integration 239
numerische Lösung 27
numerische Lösungsverfahren 248
numerische Mathematik 26
numerisches Berechnungsverfahren 26
numerische Verfahren 223

O

Object Browser 173, 185
Objekt 96
objektorientierte Programmierung
(Object-oriented Programming) 96
Odefun 240
ones 88
OpenFcn 266
Opening Function 266
Options 241
otherwise 100

P

Panel 158
Parallel batch jobs 392
Parallel computing 382
Parallele Zustände 297, 343
Parallel pool 388
parcluster 393
parents 146
parpool 388
path 76
Pattern Wizard 315
pause 203
Pause 106
Periodendauer 154
Persistent Variables 74
Physikalische Modellierung (objekt-
orientierte Modellierung) 245
plot 114, 137
plot3 120
Pop-Up Menu 157
Port 313
Position Readouts 173
Preferences 171
Present working directory 75

- pretty 400
 - Private Functions 73
 - Programmatic GUI 156
 - Programmiersprache 35
 - properties 241, 265
 - Property Inspector 143, 173
 - Prozess 384
 - PS-Simulink Converter 290
 - Pushbutton 156, 164
 - pwd 76
- R**
- Radio Button 157
 - Rapid Control Prototyping 31
 - realmax 218
 - realmin 218
 - Recheneinheiten 382
 - reelle Zahlen 206
 - relationale Operatoren 51
 - Revision 83
 - Rotate Block 257
 - Rückverfolgung (Backtracking) 318
 - Run 173
 - Rundungsfehler 233
 - Runge-Kutta-Verfahren 232
- S**
- save 112
 - Schaltsignal 295
 - Schleifen 98
 - Schleifenunterbrechung (break) 103
 - Schlüsselwort (Keyword) 325
 - Schrittweite 228, 234, 235, 237
 - Scope 272, 361
 - Script 55, 64
 - Search Path 75
 - Separator 166
 - serielles Rechnen 382
 - set 143
 - Signalleitungen 255
 - Signal Logging 274
 - simplify 401
 - Simscape 288
 - Simulation 268, 378
 - Simulation Data Inspector 274
 - Simulink Coder 294
 - Simulink Library Browser 277, 298, 323
 - Simulink PS Converter 290
 - Simulink Start Page 298
 - single 217
 - Sinks 259
 - size 87
 - Slider 156
 - solve 401
 - solver 239, 241
 - Sonderzeichen 53
 - Spaltenvektor 48
 - Special character 54
 - Speicherbedarf 218
 - Stammbaum 146
 - Startzustand (Default State) 321
 - State 321
 - State Action 324, 326
 - State chart 320
 - Stateflow 296, 298
 - Stateflow Debugger 379
 - Stateflow Editor 298, 307, 323
 - Stateflow Layoutbereich 301
 - State Label 324
 - State-level 327
 - Statement 325
 - State name 324
 - State Transition Matrix 374
 - State Transition Table 366, 371
 - Static Text 157
 - str2double 183
 - strcmp 90
 - strfind 90
 - string 41, 90
 - struct 93
 - Structure Array 93, 96
 - Subfunctions 71
 - subplot 118
 - subs 400
 - Substate 337, 376
 - Subtraktionsmethode 207
 - Superstate 337, 376
 - Super Step 334
 - surf 120, 121
 - switch 100

Switch-Verzweigung 100
symbolische Lösung 27
symbolisches Lösungsverfahren 27
symbolisches Objekt 399
symbolisches Rechnen 399
Symbolleiste 164
Syntaxdefinition 53

T

Table 157
Tab Order Editor 173, 186
Task 393
taylor 403
technische Systeme 245
Template 202
Terminating Transition 319
termination junction 302
Textzeichen 41
Thread 384
tic 385
title 150
toc 385
Toggle Button 157, 164
Toolbar Editor 173
Toolbox 30
To Workspace 259
TRADIC 22
Transistor 22
Transition 301, 302, 321, 372
Transition action (Übergangsaktion) 331
Transition Label 303, 309, 326
Triggered Subsystems 355
true 41
Truth Table 366, 367
try 104

U

Übergangsbedingungen 302
uicontextmenu 159, 163
UIcontrol objects 156
uimenu 159
uint8 217
uint16 217
uint32 217

uint64 217
uipanel 159
uipushtool 159
uitable 159
uitoggletool 159
uitoolbar 159, 164
Unicode® 91
Units 159
Unterbrechungen 98
Unterprogramm 64

V

Variablendefinition 39, 41
Variablenname 41
Variablenzuweisung 41
variable Schrittweite 236
Variables Editor 85, 182
Vektoren 85
ver 83
Visualisierung 62
Vorbelegung 88
Vorlage (Template) 65, 196
Vorzeichenbit 211

W

Wahrheitstabelle 301
Wahrheitswert 41, 295, 310
wait 395
Wertebereich 206, 219
wertediskret 295
Wertezuweisung 39
While-Schleife 102
Worker 385, 393
Workspace 39
Workspace Browser 39

X

xlabel 150
XY Graph 259

Y

ylabel 150

Z

- Zahlenauflösung 221
- Zahlendarstellung 206
- Zeichenketten 41, 90
- Zeilenvektor 48
- zeitdiskret 295
- zeitgesteuerte Systeme 29
- Zeitlupe 203
- Zeitraffer 203
- zeros 67, 88
- Zuse Z3 19
- Zustand 301, 324, 372
- Zustandsautomaten 296, 320
- Zustandsdiagramme 301, 320
- Zustandsgraph 29
- Zustandsgröße 225
- Zustandsübergangstabelle 301, 371
- zweidimensionale Visualisierung
114
- Zweierkomplement 211