

# **Eine Einführung in Scilab 6.0**

Peter Beater

Schrittweises Kennenlernen des frei verfügbaren  
Berechnungsprogramms mit vielen Beispielen und  
Aufgaben

Prof. Dr.-Ing. Peter Beater  
Fachhochschule Südwestfalen  
Fachbereich Maschinenbau-Automatisierungstechnik  
Lübecker Ring 2  
5 94 94 Soest

## 1. Auflage - Manuskript vom 20. 2. 2017

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

© Dr. Peter Beater, 2017

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften. Text und Abbildungen wurden mit größter Sorgfalt erarbeitet. Verlag und Autor können jedoch für eventuell verbliebene fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Herstellung und Vertrieb - Books on Demand GmbH - D-22848 Norderstedt – ISBN 9783743172975

## **Zu diesem Buch**

Eine Reihe von Änderungen und Verbesserungen in Scilab haben mich veranlasst, den Vorgänger dieses Buches zu überarbeiten, an die aktuelle Version 6.0 anzupassen und ein wenig zu erweitern. Dabei wollte ich aber den Umfang nicht zu sehr anschwellen lassen, damit es sich nach wie vor um eine Einführung handelt. Sie soll in den ersten Wochen eines Semesters Werkzeuge für die eigentliche Aufgabe zur Verfügung stellen, z. B. Datenauswertungen bei der experimentellen Modellbildung für numerische Simulation.

Das in diesem Buch verwendete Querformat hat sich bewährt. Klassische Buchlayouter werden monieren, dass die Zeilen nun zu lang sein, um einfach gelesen werden zu können. Die große Anzahl an Buchstaben pro Zeile macht jedoch den Ausdruck der Scilab-Programme weniger unübersichtlich, da Fortsetzungszeilen weitgehend entfallen können. Ebenso habe ich positive Rückmeldungen über die Bindung als Ringbuch erhalten, da sie es ermöglicht, das Buch einfach aufzuschlagen und flach neben die Tastatur zu legen.

Der Inhalt dieses Buches basiert auf den Erfahrungen aus vielen Vorlesungen und Seminaren, in denen Scilab für numerische Berechnungen eingesetzt wurde. Die Benutzeroberfläche, die es ab Version 5.4 gibt und die in V 6.0 etwas erweitert wurde, wird hauptsächlich verwendet, ohne die früheren Möglichkeiten der Konsolensteuerung zu vernachlässigen.

# Inhalt

<b>Zu diesem Buch .....</b>	<b>III</b>
<b>1 Scilab als Taschenrechner .....</b>	<b>1</b>
1.1 Eingabe einzelner Befehle.....	1
1.2 Speichern und Laden von Daten .....	14
<b>2 Steuerkonstrukte .....</b>	<b>17</b>
<b>3 Skripte und Funktionen.....</b>	<b>19</b>
3.1 Skripte .....	19
3.2 Beispiele zu Skripten.....	26
3.3 Funktionen.....	28
3.4 Weitere Möglichkeiten bei Funktionen.....	30
3.5 Globale und lokale Variable .....	33
3.6 Darstellung einer Anweisung als mathematische Formel in OpenOffice .....	36
3.7 Beispiele zu Funktionen .....	39
<b>4 Vektoren und Matrizen.....</b>	<b>41</b>
4.1 Vektoren .....	41
4.1.1 Vektorfunktionen .....	44
4.2 Matrizen .....	48
4.2.1 Matrizenfunktionen .....	50

4.3 Listen.....	52
<b>5 Tabellen und Interpolation.....</b>	<b>56</b>
<b>6 Formatierte Ausgabe auf dem Bildschirm.....</b>	<b>58</b>
<b>7 X-Y- und 3D-Grafiken .....</b>	<b>61</b>
7.1 Plotten von Daten mit <code>plot</code> .....	61
7.2 Der Plot-Editor .....	67
7.3 Weitere Plotbefehle für x-y-Grafiken .....	68
7.3.1 Zeichnen von Stufenfunktionen mit <code>plot2d2</code> .....	69
7.3.2 Zeichnen von Balken mit <code>plot2d3</code> .....	72
7.3.3 Zeichnen von Funktionen mit <code>fplot2d</code> .....	73
7.3.4 Zeichnen von Histogrammen mit <code>histplot</code> .....	74
7.3.5 Zeichnen mehrerer Grafiken in ein Fenster mit <code>subplot</code> .....	75
7.4 3D-Grafiken .....	76
7.4.1 3D-Plot mit <code>plot3d</code> .....	77
7.4.2 3D-Plot mit <code>mesh</code> .....	78
7.4.3 3D-Plot mit <code>surf</code> .....	81
7.4.4 3D-Trajektorie mit <code>param3d1</code> .....	84
7.4.5 Zeichnen von Höhenlinien mit <code>contour</code> .....	85
7.5 Export von Grafiken.....	87
7.6 Grafikeigenschaften programmieren.....	89
7.7 Kommentare zu Grafikformaten, die Scilab exportieren kann .....	96
7.7.1 Grafikformate – Rastergrafiken.....	96

7.7.2 Grafikformate – Vektorgrafiken.....	99
<b>8 Numerische Mathematik .....</b>	<b>102</b>
8.1 Polynome.....	102
8.2 Berechnen von Polynomkoeffizienten mit <code>polyfitPB.sci</code> .....	105
8.3 Beispiel für die Funktion <code>datafit</code> .....	109
8.4 Optimieren mit <code>fminsearch</code> .....	112
8.5 Manchmal geht leider etwas daneben ... ..	115
<b>9 Klassische Regelungstechnik mit Scilab.....</b>	<b>120</b>
9.1 Systemdefinition mittels Übertragungsfunktion .....	120
9.2 Filterberechnungen und weitere Möglichkeiten .....	127
<b>10 Einrichten von Scilab .....</b>	<b>129</b>
<b>11 Was alles noch nicht gesagt wurde .....</b>	<b>133</b>
<b>12 Übungen .....</b>	<b>135</b>
12.1 Erste Eingaben .....	135
12.2 Skript und Funktion zur Berechnung einer Düse .....	137
12.3 Beispiel zur Grenzen der Fließkommazahlen .....	140
12.4 Übungen zu Vektoren.....	141
12.5 Übungen zu Matrizen .....	142
12.6 Kennfeldberechnung und Darstellung einer Blende .....	143
12.7 Graphische Darstellung und Nullstellenbestimmung von Funktionen .....	144
12.8 Optimierung mit <code>fminsearch</code> .....	146

12.9 Optimale Auslegung einer Bremse .....	148
12.10 Auslegung eines PI-Reglers .....	154
<b>13 Übersichten über Scilab-Befehle.....</b>	<b>156</b>
<b>14 Lösungen zu den Übungen.....</b>	<b>172</b>
14.1 Erste Eingaben .....	172
14.2 Skript und Funktion zur Berechnung einer Düse .....	174
14.3 Beispiel zur Grenzen der Fließkommazahlen .....	178
14.4 Übungen zu Vektoren.....	181
14.5 Übungen zu Matrizen .....	182
14.6 Kennfeldberechnung und Darstellung einer Blende .....	182
14.7 Graphische Darstellung und Nullstellenbestimmung von Funktionen .....	186
14.8 Optimierung mit fminsearch .....	191
14.9 Optimale Auslegung einer Bremse .....	197
14.10 Auslegung eines PI-Reglers .....	203

# 1 Scilab als Taschenrechner

Fast alle Ingenieuraufgaben werden heute nicht mehr mit Papier und Bleistift oder Rechenschieber, sondern mittels PC erledigt. Um das dafür sehr gut geeignete Berechnungsprogramm Scilab kennen zu lernen, bedienen wir es in diesem Kapitel durch einzelne Eingaben im sogenannten Konsolenfenster wie einen Taschenrechner.

## 1.1 Eingabe einzelner Befehle

In Bild 1-1 ist die Oberfläche nach der Installation dargestellt. In der Mitte befindet sich die Konsole zur Befehlseingabe. Links ist ein Datei-Browser vorhanden. Der Inhalt wird durch einen Klick auf die Schaltfläche *Filter* aktualisiert. Rechts oben befindet sich der Variablen-Browser. Im Fenster Mitte rechts sind die bisherigen Befehle aufgelistet. Unten rechts ist neu in Version 6.0 der *Newsfeed*. Die einzelnen Fenster können gelöst (unlock) oder auch geschlossen werden.

Diese Oberfläche wurde mit der Version 5.4 eingeführt. Sie rückt die Befehlseingabe etwas in den Hintergrund und ermöglicht einen besseren Überblick über die bisherigen Berechnungsschritte und Ergebnisse. Die dazu in älteren Versionen erforderlichen Tastaturbefehle sind nach wie vor verfügbar.



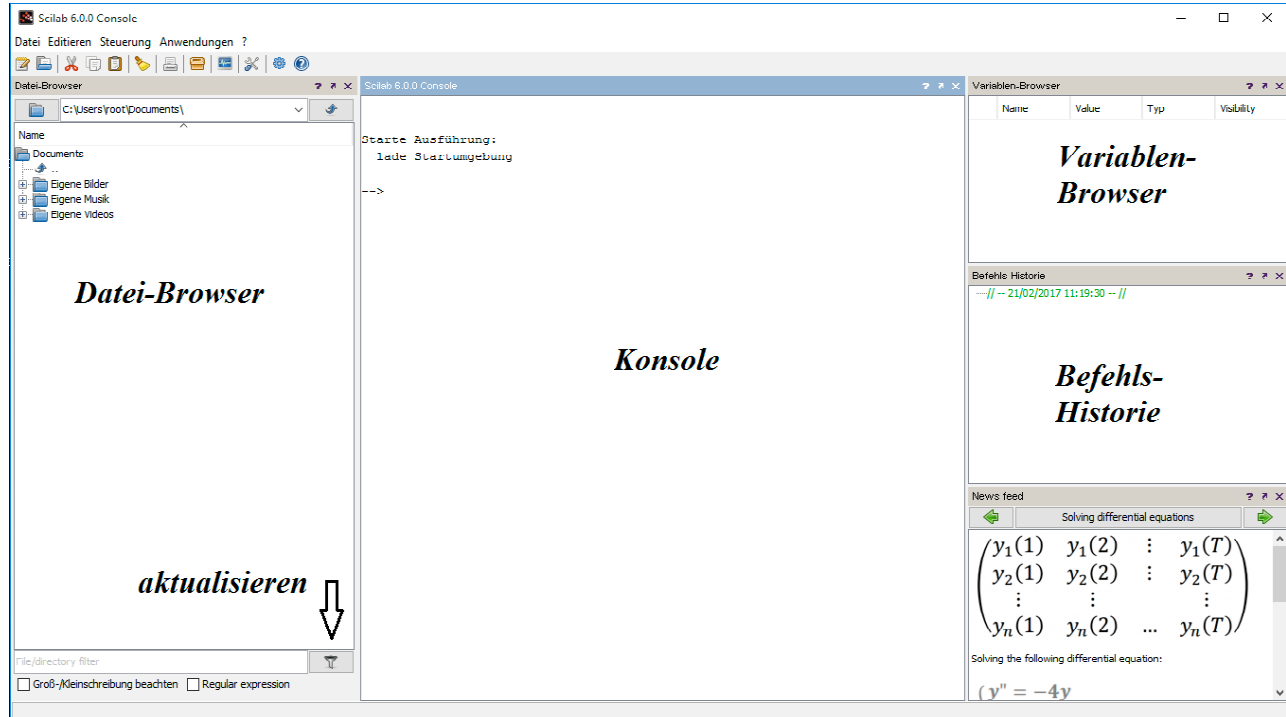
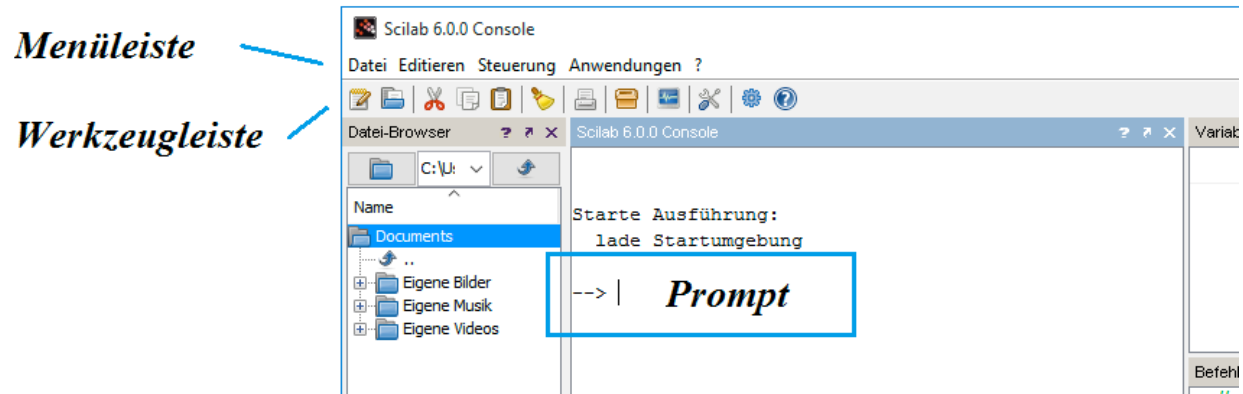


Bild 1-1 Scilab-Bildschirm mit fünf Fenstern





**Bild 1-2 Scilab-Bildschirm mit Menü- und Werkzeugleiste und Prompt**

Bild 1-2 zeigt Menü- und Werkzeugleiste sowie den blinkenden *Prompt*.

Immer wenn das Symbol

--> |

erscheint – der sogenannte Scilab-Prompt – und die Einfügemarke des Cursors sich dort befindet, können wir im Konsolenfenster eine Eingabe machen, die mit Betätigung der *Eingabetaste* ( oder ) abgeschlossen wird.

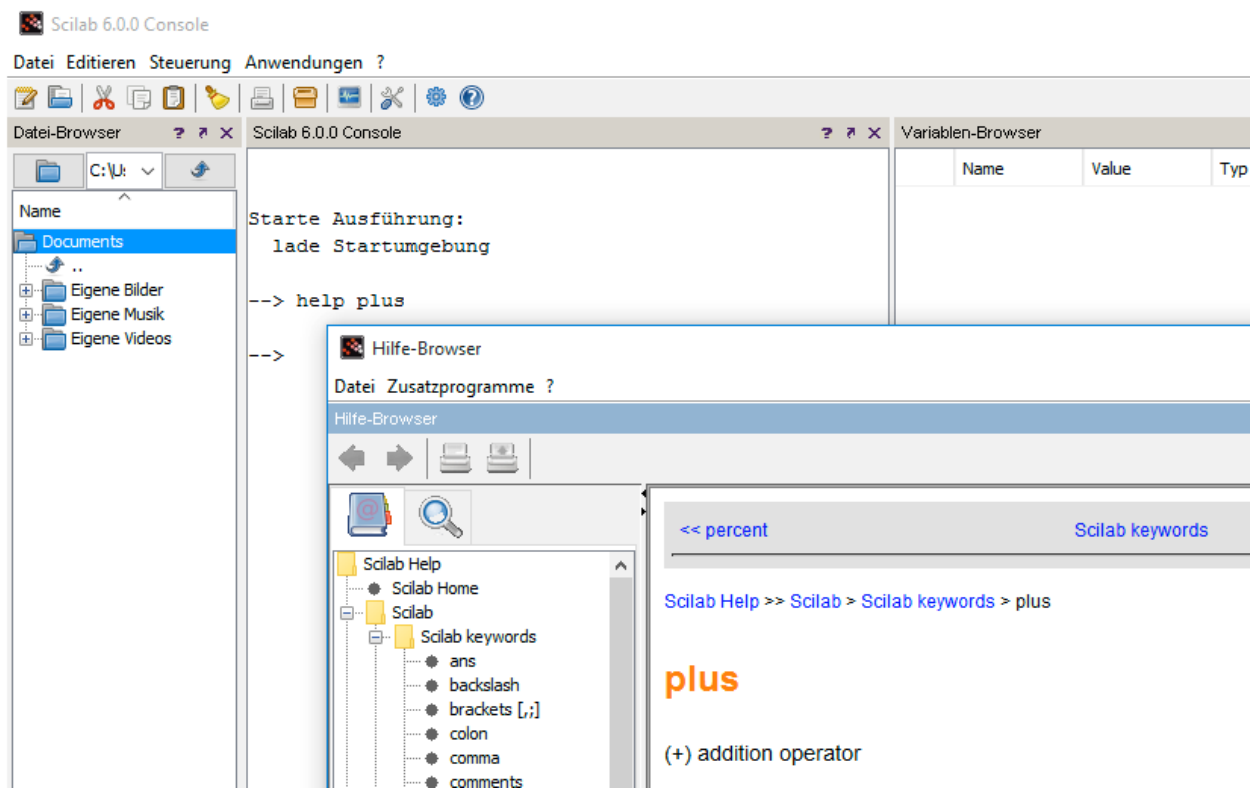
Mit den Cursor-Tasten  $\uparrow$  oder  $\downarrow$  kann man frühere Befehle zurückholen und bearbeiten oder noch einmal ausführen. Wenn wir die ersten Buchstaben eines Befehls eingeben und dann die Taste  $\uparrow$  betätigen, springt Scilab zu dem letzten Befehl, der mit dieser Zeichenkette begann. Mit den Cursor-Tasten  $\rightarrow$  oder  $\leftarrow$  kann die eingegebene Befehlszeile bearbeitet werden. Das rechte mittlere Fenster (Befehls-Historie) listet die bisherigen Befehle auf. Durch Doppelklicken werden sie noch einmal ausgeführt. Wir können auch im Fenster der Befehls-Historie einen Befehl markieren, ihn in die Zwischenablage kopieren und von dort aus in einen Text einfügen. Geben wir am Prompt einen oder mehrere Buchstaben ein und drücken dann die TAB-Taste, erscheint ein Menü mit Befehlen, die mit dem bzw. den eingegebenen Buchstaben anfangen.



Im einfachsten Fall führen wir nur eine Rechenoperation aus. Das Ergebnis wird der Variable **ans** zugewiesen und automatisch angezeigt, wie der folgende Bildschirmausdruck zeigt. Das Betätigen der Eingabetaste ( $\leftarrow$  oder **Enter**) am Ende der ersten Zeile wird auf dem Bildschirm nicht dargestellt. Es wird im Folgenden daher auch nicht mehr gesondert angegeben.

```
-->1 + 1 Enter  
ans =
```

2.

Bei Unklarheiten kann die *Online-Hilfe* mit **help xzy** aufgerufen werden, wobei xzy das Thema ist, zu dem Hilfe gesucht wird. Eine zweite Suchmöglichkeit bietet der Befehl **apropos**, der *Schlüsselwörter* in den Hilfe-Dateien sucht.

**Bild 1-3 Ausgabe des Hilfe-Browsers nach dem Befehl help plus**

Den Hilfe-Browser können wir im Windows-System auch mit einem Klick in der Befehlszeile auf  aufrufen. Die Lupe  dient dann zum weiteren Suchen. Mit der Tastenkombination „Steuerung +“ können wir die Schriftgröße vergrößern, mit „Steuerung -“ verkleinern.

Bei längeren Rechnungen ist es sinnvoll, Variablen zu verwenden. Wie bei fast allen Programmiersprachen handelt es sich bei Scilab-Kommandos *nicht* um *Gleichungen*, sondern um *Zuweisungen*. Das heißt, dass der Ausdruck rechts vom *Zuweisungszeichen* = ausgewertet und dann der Variable links des Zuweisungszeichens zugewiesen wird. Wie bei allen Rechenprogrammen üblich, werden Nachkommastellen nach dem *Dezimalpunkt* eingegeben.

```
-->a = 2
a =
    2.

-->a = 3 * a + 5
a =
   11.

-->a = a - 11
a =
    0.
```

Um den Strom durch einen Ohmschen Widerstand zu berechnen, weisen wir zuerst der Variable R und dann der Variable u einen Wert zu und berechnen den Strom i. Dabei unterscheidet Scilab zwischen Groß- und Kleinschreibung: R und r sind unterschiedliche Variable.

```
-->R = 10
R =

   10.
```

```
-->u = 0.5
u =

    0.5

-->i = u / R
i =

    0.05
```

Das „Echo“ können wir mit einem Semikolon am Befehlsende unterdrücken. Es ist auch möglich, mehrere Zuweisungen in eine Zeile zu schreiben oder eine Zeile in der folgenden fortzusetzen, indem die erste mit einem Leerzeichen und drei Punkten ... beendet wird.

```
-->R = 10; u = 0.5; i = u/R
i =

    0.05
```

das ; unterdrückt die Ausgabe

Ein Name beginnt mit einem Buchstaben oder einem der Sonderzeichen `#_ $ ?`, gefolgt von Buchstaben, Ziffern oder den Sonderzeichen `#_ $ ! ?`. Der Variablenname durfte in Version 5.5 nicht länger als 24 Zeichen sein. Diese Einschränkung ist ebenso wie das Verbot deutscher Sonderzeichen wie `Ü` oder `ß` nicht mehr vorhanden. Leerräume sind in Programmiersprachen nicht vorgesehen und dürfen daher auch in SciLab nicht verwendet werden.


Den Wert einer Variablen erhält man, wenn man im Konsolenfenster nur den Namen eingibt und die Zeile mit Betätigung der Eingabetaste abschließt. Eine zweite Möglichkeit ist der Doppelklick auf die Variable im rechts oben angeordneten Variablen-Browser, der auch Plots ermöglicht (rechte Maustaste).

Bei Scilab ist es nicht nötig, Variablen vor dem ersten Gebrauch zu definieren. Das hat Vorteile, da Berechnungen sehr schnell eingegeben werden können. Die Gefahr besteht aber darin, dass durch Schreibfehler neue Variablen definiert werden, die dann zu falschen Ergebnissen führen.

Die bereits definierten Variablen sehen wir rechts im Variablen-Browser (wenn er „verschwunden“ ist: Aufruf in der Scilab-Menüleiste unter **Anwendungen** / **Variablen-Browser**). Wir können sie auch mit dem Befehl **who** auflisten. Es erscheinen dann auch Systemvariablen, Bibliotheksnamen und vordefinierte Konstanten.

Der Befehl **whos** gibt die Variablen und deren Dimension in das Konsolenfenster aus. Außerdem zeigt er vom Benutzer erstellte und geladene Funktionen mit der korrekten Schreibweise an.

Mit dem Befehl **clear** werden alle vom Nutzer definierten Variablen gelöscht, wobei die Systemvariablen erhalten bleiben; **clear a b x** löscht die Variablen a, b und x.

Zum Löschen des Konsolenfensters steht auch ein Icon in der Werkzeugleiste zur Verfügung  oder die Abfolge **Editieren** / **Konsole leeren**. Der entsprechende Befehl heißt **clc**.

Scilab beherrscht die vier Grundrechenarten und berücksichtigt dabei auch den Vorrang der Multiplikation und Division vor Addition und Subtraktion.

```
-->Grundrechenarten = 1 + 2 - 3 * 4 / 5
Grundrechenarten =

    0.6
```

Das Multiplikationszeichen darf bei Scilab *nicht* weggelassen werden.

Neben der „normalen“ Division mit / gibt es die in erster Linie für Matrizenrechnungen vorgesehene „links-seitige Matrizendivision“, mit \, wobei die Zeile

```
-->X = A \ B
```

die Lösung  $\underline{X}$  zur Gleichung  $\underline{A} \cdot \underline{X} = \underline{B}$  liefert.

Zahlen in der sogenannten wissenschaftlichen Darstellung zur Basis 10 mit Mantisse a und Exponenten b werden als **aeb** oder in Scilab auch als aDb geschrieben.

$$5 \cdot 10^{-0,7} = a \cdot 10^b \rightarrow 5e(-0.7) \text{ .}$$

```
-->expEingabe = 5e-7
expEingabe =
```

```
0.0000005
```

Scilab hat zwei verschiedene Formate, um Zahlen *anzuzeigen*: Das variable Format und das Exponentialformat. Mit dem Befehl **format** kann das Format und die Zahl der Stellen gewählt werden. Die interne Zahlendarstellung ändert der Befehl **format** nicht!

```
-->format("v", 25), 5e-7 / 1.11      // variables Format
ans =
```

```
0.0000004504504504504504
```

```
-->format("e", 25), 5e-7 / 1.11      // Exponentialformat
ans =
```

```
4.504504504504503790D-07
```



```
-->format("e", 15), 5e-7 / 1.11
ans =

4.50450450D-07
```

"v" bzw. "e" steht für die Ausgabeart, die Zahl danach für die maximale Stellenzahl. Der Vorgabewert ist 10 Stellen, also `format("v",10)`.

Auch Potenzrechnung, das Logarithmieren, die Winkelfunktionen und die Quadratwurzel sind verfügbar. Wie bei allen Rechenprogrammen wird in der Trigonometrie im Bogenmaß und nicht im Gradmaß gerechnet, mit  $2\pi \text{ rad} = 360^\circ$ .

Wichtige Konstanten, z. B. die Eulersche Zahl  $e$ , die Kreiszahl  $\pi$  oder die imaginäre Einheit  $i$  mit  $i^2 = -1$  sind ebenfalls bereits definiert und werden mit einem % direkt vor dem Namen aufgerufen, z. B. %e, %pi oder %i.

```
-->test1 = log10(10 ^ 3) // Logarithmus zur Basis 10
test1 =
3.
-->test1a = log10(10 ** 3)
test1a =
3.
-->test2 = log(%e ^ 2) // Logarithmus zur Basis e
test2 =
2.
```

```
-->test2a = log(exp(2)) // exp für Exponentialfunktion
test2a =

2.

-->test3 = acos(cos(%pi / 4)) / %pi // acos als Umkehrfkt. zu cos
test3 =

0.25

-->test4 = sqrt(3) ^ 2 // Quadratwurzel zur zweiten Potenz
test4 =

3.
```

Etwas Vorsicht ist bei der Eingabe von negativen Zahlen nötig. Um ganz sicher zu sein, sollte man diese in ( ) setzen.

```
-->neg_zahl_1=-1 ^ 2, neg_zahl_2=(-1) ^ 2, neg_zahl_3= -5 + (-1)
neg_zahl_1 =

- 1.
neg_zahl_2 =

1.

neg_zahl_3 =

- 6.
```

Bei manchen Rechnungen benötigen wir das Vorzeichen einer Variablen. Dazu dient die Funktion **sign( )**. Sie liefert bei positiven Zahlen als Ergebnis +1, bei Null 0 und bei negativen Zahlen -1.

```
-->sign(11)
ans =
    1.

-->sign(0)
ans =
    0.
-->sign(-0.3)
ans =
   - 1.
```

Bei dem komplexen Argument A liefert  $\text{sign}(A) = A ./ \text{abs}(A)$ .

Den Betrag einer Zahl können wir mit der Funktion **abs( )** berechnen.

```
-->abs(-0.3)
ans =

    0.3
```

Imaginäre oder komplexe Zahlen werden als Summe aus Realteil und Imaginärteil eingegeben, wobei der Imaginärteil mit der in Scilab fest definierten imaginären Einheit **%i** multipliziert wird. Die in der Technik übliche Darstellung der imaginären Einheit als **j** wird von Scilab nicht unterstützt. Der Betrag einer komplexen Zahl **z** wird mit dem Befehl **abs(z)**, der Realteil mit **real(z)**, der Imaginärteil mit **imag(z)** berechnet.

```
-->imag_zahl = 4 * %i
imag_zahl =

    4.i

-->imag_zahl * imag_zahl
ans =

    - 16.
```

Gelegentlich müssen wir Text abspeichern. Dazu gibt es in Scilab den Datentyp **string**, der bei der Eingabe durch Zollzeichen " " angegeben wird. Möglich sind auch Hochkommata: ' '.

```
-->text_als_string = "Dies ist Text";
-->text_als_string
text_als_string =
    Dies ist Text
```

Bei Zeichenketten müssen identische Zeichen am Anfang und am Ende verwendet werden; eine Mischung, z. B. "gemischt mal Zoll mal Hochkomma" führt bei Scilab 6.0 zu einer Fehlermeldung. Da das Hochkomma ' auch der Operator für das Transponieren einer Matrix ist, wird das Zollzeichen " empfohlen.