

## Kurzreferenzen

### Scilab-Kurzreferenz<sup>1</sup>

Die vorliegende Scilab-Referenz stellt eine Auswahl der gebräuchlichsten Kommandos und Funktionen dar. Der Schwerpunkt liegt auch hier vor allem auf solchen Kommandos, die wir in diesem Kurs auch genutzt haben. Verwenden Sie bitte die Referenz lediglich als Übersicht, eben als „erste Hilfe“. Nutzen Sie sie, wenn Sie eine Aufgabe aus einem bestimmten Themengebiet lösen wollen, jedoch nicht wissen, mit welchem Kommando sie bearbeitet werden kann. In jedem Fall sollten Sie anschließend noch die ausführlicheren Hilfefunktionen von Scilab oder die Online-Hilfe zu Rate ziehen.

Ergänzend werden zum jeweiligen Thema die Dateien angegeben, die online zum Buch verfügbar sind. Dabei sind jedoch nur solche Programme aufgeführt, die nicht lediglich einfache Illustrationen anhand von Beispielen darstellen.

### Allgemeine Kommandos

#### Allgemeine Informationen und Hilfe

`help` - Online-Hilfe, Text erscheint im Hilfe-Browser  
`format` - gibt das Ausgabeformat auf dem Bildschirm an

Das Kommando `format` erlaubt im Gegensatz zu MATLAB nur zwei Einstellungen: `type` für variables Format, `long` für die Zahl der auszugebenden Stellen. Umstellungen sind möglich durch Eingabe des Befehls `format([type],[long])`. Die Einstellung `type` kann bedeuten:

`v` - variables Format (voreingestellt, d.h. je nach Zweckmäßigkeit Exponential- oder Gleitkommadarstellung) oder  
`e` - immer Exponentialdarstellung.

Die Einstellung `long` gibt die Zahl der Stellen an (voreingestellt sind zehn Stellen).

```
-->format('v',16),  
-->format('v',12),  
-->format('e',12),
```

Wenn der Formatstring weggelassen wird, wird nur die Zahl der angezeigten Ziffern verändert, der Typ des Formats bleibt bestehen:

---

<sup>1</sup> Beruht auf dem Hilfe-Browser von Scilab.

## 2 Kurzreferenzen

---

-->format(12) //12 Ziffern zur Anzeige

Mittels format() ohne Bezeichner wird nur das gerade eingestellte Format angezeigt.

- exec - veranlasst die Ausführung einer Script-Datei
- mode - gibt an, ob ein Echo der Befehle angezeigt wird
- clc - löscht das Kommandofenster
- pause - Programmunterbrechung, ermöglicht den Aufruf lokaler Variablen (Rückkehr mit resume, abort, quit oder return)
- abort - Programmabbruch
- resume - äquivalent zu return
- stacksize - legt die Größe des Arbeitsspeichers (stack) fest

### Arbeiten mit dem Workspace

- who - Liste aller verfügbaren Variablen
- whos - Liste der vom Anwender benutzten Variablen. Langform, liefert zusätzliche Informationen zu diesen Variablen (z.B. Speichergröße)
- who\_user - Liste der vom Anwender benutzten Variablen
- clear - löscht Variablen und Funktionen aus dem Speicher
- quit - beendet die Scilab-Sitzung oder die Programmunterbrechung
- pwd - zeigt das aktuelle Arbeitsverzeichnis an

### Arbeiten mit Kommandos und Funktionen

- what - listet alle Scilab-Kommandos auf (Kurzform)
- edit - erlaubt das Editieren der sce/sci-Dateien

### Daten laden oder speichern

- load - lädt Daten aus einer binären Datei
- save - speichert Daten in eine binäre Datei
- loadmatfile - lädt Daten einer binären MAT- oder ASCII-Datei in den Workspace
- savematfile - speichert Daten aus dem Workspace in eine binäre MAT- oder ASCII-Datei

## Operatoren und spezielle Zeichen

### Arithmetische Operatoren und Matrixoperatoren

- + Plus
- Minus
- \* Matrixmultiplikation
- . \* Array-Multiplikation
- . ^ Potenzieren von Arrays („normales“ Potenzieren)
- ^ oder hat Potenzieren von Matrizen
- \ Backslash oder linksseitige Division (Lösung von Gleichungssystemen)
- / Slash oder rechtsseitige Division (Lösung von Gleichungssystemen)
- ./ Division von Arrays („normale“ Division)
- .\ linksseitige Division von Arrays
- ' Transponieren einer Matrix

### Relationale Operatoren

== gleich  
 ~= ungleich  
 < kleiner als  
 > größer als  
 <= kleiner als oder gleich  
 >= größer als oder gleich

### Logische Operatoren

&& logisches UND  
 || logisches ODER  
 (sogenannte „Short-circuit-Operationen“ mit verkürzter Auswertung; bei Arrays sind stattdessen die Operationen & beziehungsweise | zu verwenden)  
 ~ logisches Komplement (NOT)  
 xor logisches EXCLUSIV-ODER  
 or Aussage ist wahr, wenn mindestens ein Element ungleich null ist  
 and Aussage ist wahr, wenn alle Elemente ungleich null sind

### Spezielle Zeichen

[ ] - eckige Klammern für Darstellung von Matrizen  
 ... - Fortsetzung in nächster Zeile  
 ; - Semikolon, verhindert als Abschluss der Befehlszeile die Ausgabe auf der Kommandozeile  
 // - Beginn eines Kommentars  
 ' - Transposition. X' ist die konjugiert komplexe Matrix zu X, X.' ist die nichtkonjugierte Transponierte.  
 = - Zuordnung. B = A speichert die Elemente von A in B.  
 % - spezielle vordefinierte Zeichen, z.B. %pi, %e, %inf  
 \$ - letzter Index

### Punktuation

. Dezimalpunkt: 325/100, 3.25 und .325e1 bedeuten dasselbe.  
 . Array-Operationen. Elementweises Multiplizieren, Potenzieren, Dividieren usw.: \*, .^, ./ .  
 Zum Beispiel ist C = A ./ B die Matrix mit den Elementen  
 $c(i,j) = a(i,j)/b(i,j)$ .  
 : Doppelpunkt  
 J:K ist dasselbe wie [J, J+1, ..., K].  
 J:K ist leer, wenn J > K.  
 J:D:K ist dasselbe wie [J, J+D, ..., J+m\*D], wobei  $m = \text{fix}((K-J)/D)$ .  
 J:D:K ist leer, wenn D > 0 und J > K oder wenn D < 0 und J < K.  
 Doppelpunkt (colon) als Separator  
 COLON(J,K) ist dasselbe wie J:K und colon(J,D,K) ist dasselbe wie J:D:K.  
 - Die Darstellung mittels Doppelpunkt kann benutzt werden, um Zeilen, Spalten und Elemente eines Vektors, einer Matrix oder eines Arrays herauszuheben.  
 A(:) sind alle Elemente eines Vektors, geschrieben als einzelne Spalte.

- Auf der *linken* Seite einer Zuordnung kann der Doppelpunkt benutzt werden, um bestimmte Zeilen, Spalten oder Elemente eines Vektors, einer Matrix oder eines Arrays herauszuheben, wobei die Gestalt von A erhalten bleibt:

$A(:, J)$  ist die J-te Spalte von A,

$A(J:K)$  ist  $[A(J); A(J+1); \dots; A(K)]$ ,

$A(:, J:K)$  ist  $[A(:, J), A(:, J+1), \dots, A(:, K)]$  usw.

## Elementare mathematische Funktionen

sqrt - Quadratwurzel

### Trigonometrie

sin - Sinus  
sind - Sinus, wenn das Argument in Grad („degree“) angegeben wird  
cos - Kosinus  
cosd - Kosinus, wenn das Argument in Grad angegeben wird  
tan - Tangens  
tand - Tangens, wenn das Argument in Grad angegeben wird  
asin - Arkussinus (Inverse des Sinus)  
asind - Arkussinus, Ausgabe in Grad  
acos - Arkuskosinus (Inverse des Kosinus)  
acosd - Arkuskosinus, Ausgabe in Grad  
atan - Arkustangens (Inverse des Tangens)  
atand - Arkustangens, Ausgabe in Grad

### Exponentialfunktionen, Logarithmen und Hyperbelfunktionen

exp - Exponentialfunktion  
sinh - Hyperbelsinus  
cosh - Hyperbelkosinus  
tanh - Hyperbeltangens  
asinh - Areasinus (Inverse des Hyperbelsinus)  
acosh - Areakosinus (Inverse des Hyperbelkosinus)  
atanh - Areatangens (Inverse des Hyperbeltangens)  
log - natürlicher Logarithmus  
log10 - dekadischer Logarithmus (Basis 10)  
log2 - dualer Logarithmus (Basis 2)

### Weitere wichtige Funktionen

erf - Fehlerintegral  
gamma - Gamma-Funktion  
besselj - BESSEL-Funktion erster Art

### Komplexe Zahlen und Funktionen

real - Realteil  
imag - Imaginärteil  
abs - Absolutwert (Betrag)  
conj - komplex konjugierter Wert

polar - Polarkoordinaten einer komplexen Zahl (Winkel in Radiant)  
phasemag - Polarkoordinaten einer komplexen Zahl (Winkel in Grad, Logarithmus des Betrags)

### Runden und Rest

fix - Runden zum nächstniedrigeren Wert hin  
round - Runden zum nächsten Nachbarn  
modulo - Modulus (vorzeichenbehafteter Rest nach der Division)  
pmodulo - positiver Modulus (positiver Rest nach der Division)  
sign - Vorzeichen

## Matrizen und Manipulationen mit Matrizen

### Elementare Matrizen

zeros - Array aus Nullen  
ones - Array aus Einsen  
eye - Einismatrix (Diagonalelemente eins, sonst null)  
rand - erzeugt gleichförmig verteilte Zufallszahlen:  
    rand(t, 'uniform') liefert Matrix der Dimension t mit gleichverteilten Zufallszahlen.  
    rand(t, 'normal') liefert Matrix der Dimension t mit normalverteilten Zufallszahlen.  
sprand - erzeugt gleichförmig verteilte Zufallszahlen einer schwach besetzten Matrix  
linspace - Vektor mit linear verteilten Abständen  
logspace - Vektor mit logarithmisch verteilten Abständen  
meshgrid - X-Y-Array für 3-D-Plots  
: - Vektor mit linear verteilten Abständen und Index in einer Matrix  
toeplitz - erzeugt eine TOEPLITZ-Matrix

### Elementare Array-Information

size - Größe einer Matrix  
length - Länge eines Vektors  
diag - listet die Diagonalelemente einer Matrix auf  
trace - Summe der Diagonalelemente einer Matrix  
disp - Anzeige einer Matrix oder eines Textes  
isempty - wahr für eine leere Matrix  
isequal - wahr, wenn Arrays gleich sind  
type - zeigt den Variablentyp an (z.B. reell, komplex, Boole, String)  
sum - Summe der Elemente einer Matrix  
cumsum - kumulative Summe der Elemente einer Matrix (mit Angabe der Zwischenergebnisse)  
\$ - letzter Index in einer Variablenreihe oder Matrix (auch Abschluss einer Ablaufstruktur)  
max - größte Komponente einer Matrix oder eines Vektors  
min - kleinste Komponente einer Matrix oder eines Vektors  
rank - Zahl der linear unabhängigen Zeilen oder Spalten einer Matrix  
det - Determinante einer quadratischen Matrix  
sort - sortiert Elemente einer Matrix nach steigender Größe

### Matrixmanipulation

mtlb\_fliplr - Umkehrung einer Matrix von links nach rechts  
rot90 - Drehung einer Matrix um 90 Grad im mathematisch positiven Drehsinn (entgegen dem Uhrzeiger)

find - findet zugehörige Indizes zu einem Element einer Matrix (das von null verschieden ist)  
inv - inverse Matrix

### Schwach besetzte Matrizen

sparse - erzeugt eine schwach besetzte Matrix  
mtlb\_sparse - wandelt die Ausgabe einer schwach besetzten Matrix in das MATLAB-Format um  
full - wandelt eine schwach besetzte Matrix in eine volle Matrix um  
sprand - erzeugt gleichförmig verteilte Zufallszahlen einer schwach besetzten Matrix  
PlotSparse - visualisiert eine schwach besetzte Matrix (Default-Farbe der Matrixpunkte ist weiß!)

[Datei auf Webseite](#)

Scilab\_spy - visualisiert eine schwach besetzte Matrix

### Spezielle Variablen und Konstanten

ans - letzte Antwort  
mtlb\_realmax - größte positive Gleitkommazahl  
mtlb\_realmin - kleinste positive Gleitkommazahl  
number\_properties - Eigenschaften der Gleitkommazahlen  
%eps - Genauigkeit der Gleitkommarechnung (Abstand zweier Gleitkommazahlen)  
%pi - Zahl  $\pi$  ( $\approx 3.141592\dots$ )  
%i - imaginäre Einheit  
%inf - unendlich, z.B. bei  $1/0$   
%nan - Not-a-Number, Ergebnis undefinierter Operationen wie  $0/0$  (keine gültige Zahl)  
isnan - wahr für Not-a-Number  
%t - True  
%f - False  
%e - EULERSche Konstante  $e$  ( $\approx 2.718281\dots$ )

### Polynome

poly - erzeugt ein halbsymbolisches Polynom aus den angegebenen Wurzeln oder aus den Koeffizienten  
inv\_coeff - erzeugt die halbsymbolische Darstellung eines Polynoms aus seinen Koeffizienten  
convol - Polynommultiplikation, Ausgabe als Polynomkoeffizienten (einer der Vektoren muss transponiert vorliegen: convol(pa',pb))  
\* - Polynommultiplikation in halbsymbolischer Darstellung  
pdiv - Polynomdivision  
oder  
/ - Polynomdivision  
numer - Zähler einer halbsymbolischen Polynommatrix  
denom - Nenner einer halbsymbolischen Polynommatrix  
roots - Berechnung der Wurzeln eines Polynoms  
horner - Berechnung des Wertes (der Werte) eines Polynoms (Polynom muss in halbsymbolischer Darstellung vorliegen)  
derivat - Ableitung eines Polynoms  
datafit - Anpassung von Messdaten mit Hilfe eines Polynoms  
polyfit - Kurvenanpassung durch ein Polynom  
polyval - Berechnung des Wertes (der Werte) eines Polynoms

Beide Funktionen sind nicht Bestandteil des Scilab-Pakets, sondern gehören zur Statistik-Toolbox und können als Freeware getrennt erworben werden<sup>2</sup>  
`intpoly` - analytische Polynomintegration  
 Diese Funktion ist ebenfalls nicht Bestandteil des Scilab-Pakets und kann als Freeware getrennt erworben werden<sup>3</sup>.

## Funktionen von Funktionen

(Der Name der Funktion muss als Funktionsstring angegeben werden.)

`fsolve` - Nullstellensuche bei Funktionen einer Variablen  
`fminsearch` - Minimumsuche bei Funktionen einer Variablen  
`fplot2d` - grafische Darstellung einer Funktion zwischen spezifizierten Grenzen (Unterschied zu `plot`!)  
`feval` - führt eine Funktion aus, die in einem String beschrieben ist  
`execstr` - führt den Code aus, der durch einen String dargestellt wird  
`ode` - Lösungsverfahren für Differentialgleichungen  
`intg` - bestimmtes Integral einer Funktion

## Integration und Differentiation

`inttrap` - Integration von Einzelwerten mit Trapezregel  
`intg` - bestimmtes Integral einer Funktion  
`integrate` - bestimmtes Integral, Funktion wird in Formel direkt eingegeben  
`diff` - Differenz, benutzt zur Bildung der Ableitung  
`ode` - Lösungsverfahren für Differentialgleichungen

*Datei auf Webseite*

`trap.sce` - Trapezregel, angewandt auf beliebige Funktion  
`trap_iter.sce` - iterative Trapezregel, mit möglicher Ausgabe von Zwischenergebnissen und Grafik  
`simp_iter.sce` - iterative SIMPSONsche Regel, mit möglicher Ausgabe von Zwischenergebnissen und Grafik  
`romberg.sce` - numerische Integration nach ROMBERG, mit möglicher Ausgabe von Zwischenergebnissen und Grafik  
`gaussquad.sce` - numerische Integration nach GAUß-Verfahren, bis zu fünf Integrationsintervalle möglich  
`dgl_euler.sce` - Lösung einer Dgl. nach dem EULERSchen Verfahren

## Datenanalyse, Interpolation und Approximation

`max` - größte Komponente einer Matrix oder eines Vektors  
`min` - kleinste Komponente einer Matrix oder eines Vektors  
`interp1` - Interpolation  
`gsort` - sortiert Elemente einer Matrix nach steigender Größe

<sup>2</sup> Baudin M, Holtsberg A, *stixbox* (Statistik-Toolbox zu Scilab),  
<http://forge.scilab.org/index.php/p/stixbox/source/tree/master/macros/polyfit.sci>

<sup>3</sup> Urroz G E, *my SCILAB page*, <http://www.neng.usu.edu/cee/faculty/gurro/Scilab.html>

- cgs - iterative Lösungsmethode für lineare Gleichungssysteme (LGS) mit „konjugierter Gradientenmethode“
- gmres - iterative Lösungsmethode für LGS mit der „Methode der Residuen“ (für unsere Zwecke empfohlen)

*Datei auf Webseite*

newt\_intp.sce - Berechnung von dividierten Differenzen und NEWTON-Polynomen  
 lagrange.sce - Berechnung von LAGRANGE-Polynomen

### FOURIER-Transformation und Wavelets

- fft - Diskrete FOURIER-Transformation, basierend auf dem schnellen Fast-FOURIER-Algorithmus
- fftshift - verschiebt die Null-Frequenz-Komponente in die Mitte des Spektrums
- Ctfstfft - Kurzzeit-(ST-)FOURIER-Transformation eines Signalvektors (Time Frequency Toolbox notwendig)
- cwt - Kontinuierliche Wavelet-Transformation eines Signalvektors (Wavelet Toolbox notwendig)

### Ablaufstrukturen („Kontrollstrukturen“)

#### Flusssteuerung

- if - Bedingte Ausführung eines Kommandos:  
     if ... elseif ... else ... end
- else - alternative Entscheidung zu if
- elseif - weitere alternative Entscheidung zu if
- end - Abschluss eines Blocks, der mit einem der Befehle for, while, select, oder if beginnt
- for - Beginn einer Zählschleife (Endwert steht fest): for ... end
- while - Beginn einer Wiederholschleife (Endwert unbestimmt): while ... end
- break - Abbruch einer while- oder for-Schleife
- select - Fallunterscheidung: Umschalten zwischen verschiedenen Fällen eines Ausdrucks:  
     select ... case ... case... ... else ... end
- case - mögliche Schalterstellung, gehört zur select-Anweisung
- return - Rückkehr aus der aktuellen Funktion oder Rückkehr vom pause-Modus zum nächstniedrigeren Niveau

#### Berechnung und Ausführung

- eval - führt einen String als ein Scilab-Kommando aus
- feval - führt eine Funktion aus, die in einem String beschrieben ist (Der Name der Funktion muss in Hochkommas oder als Handle angegeben werden.)
- halt - wartet auf eine Eingabe von der Tastatur
- tic() - startet eine Zeitmessung
- toc() - gibt den aktuellen Zeitwert aus

#### Debugging

- setbpt - setzt einen Breakpoint
- dispbpt - zeigt alle Breakpoints an
- delbpt - löscht alle Breakpoints



## Scripts, Funktionen und Variablen

`function` - fügt eine neue Funktion hinzu  
`global` - Definition einer globalen Variablen  
`exists` - prüft, ob die angegebenen Variablen oder Funktionen definiert sind  
`argn` - gibt die Zahl der Eingabe-/Ausgabe-Argumente in einem Funktionsaufruf an

## Zeichenketten-(String-)Funktionen

`evstr` - führt das als String hinterlegte Kommando aus  
`execstr` - führt das als String hinterlegte Kommando aus (ähnlich wie `evstr`, aber für mehr als einen Ausgang geeignet)  
`sprintf` - wandelt Zahl in String um (emuliert die aus C bekannte `sprintf`-Funktion)  
`msprintf` - wandelt und formatiert Daten in einen String  
`dec2hex` - wandelt ganzzahlige Dezimalzahl in String um  
`hex2dec` - wandelt String in ganzzahlige Dezimalzahl um  
`hex2num` - wandelt Hex-String in Gleitkommazahl um  
`msprintf` - wandelt formatierte Daten in einen String um  
`mscanf`, `mfscanf`, `msscanf` - liest formatierte Daten ein (aus der Standard-Eingabe, aus einem Datenstrom oder aus einem String)

*Datei auf Webseite*

`dechex.sce` - wandelt eine Dezimalzahl in eine Hexadezimalzahl um (Nachkommastellen sind erlaubt)  
`dec_fraction.sce` - Umwandlung der Nachkommastellen einer Dezimalzahl in eine Hexadezimalzahl `n`

## Dateneingabe und -ausgabe

`disp` - Anzeige einer Matrix oder eines Textes  
`mfprintf` - schreibt formatierte Daten in eine Datei  
`mfscanf` - liest formatierte Daten aus einer Datei  
`savewave` - schreibt Daten in eine Wave-Datei  
`loadwave` - liest Daten aus einer Wave-Datei

## Grafik

### Elementare xy-Graphen:

`plot` - Darstellung von diskreten Werten in kartesischen Koordinaten  
`plot2d` - grafische Darstellung von Funktionen in kartesischen Koordinaten, alternatives Kommando  
`fplot2d` - grafische Darstellung einer Funktion zwischen spezifizierten Grenzen (Unterschied zu `plot`!)  
`logflag` - Parameter innerhalb des Kommandos `plot2d` und anderer Grafikfunktionen zur Festlegung der Achsenskalierung:  
   `nn` - beide Achsen linear  
   `ll` - beide Achsen logarithmisch  
   `ln` und `nl` - eine Achse logarithmisch  
`figure` - erzeugt ein neues Grafikfenster ohne Inhalt, speziell:  
   `figure('background', -2)` mit weißem Hintergrund

`close` - schließt das aktuelle Grafikenfenster  
`xdel` - löscht ein Grafikenfenster, speziell:  
    `xdel(winsid())` löscht alle vorhandenen Grafiken.  
`clf` - löscht den Inhalt des aktuellen Grafikenfensters oder setzt es auf Default-Werte  
`polarplot` - Zeichnung in Polarkoordinaten  
`xgrid` - Gitterlinien in der Grafik  
`grayplot` - zweidimensionale grafische Darstellung („Farbplot“) der Werte einer Matrix  
`get` - liest Objekteigenschaften aus (z.B. liest `get(gca(), 'data_bounds')` die Achsenskalierung einer gegebenen Grafik aus)  
`gca` - liest Objekteigenschaften des Achsensystems einer vorhandenen Abbildung aus:  
    Mit dem Kommando `da.auto_clear = 'on'` werden vorhandene Grafiken generell automatisch gelöscht, bevor eine neue Grafik generiert wird (wie in MATLAB).  
    Mit `da.auto_clear = 'off'` bleiben die Grafiken generell erhalten (Default).  
`set` - setzt Objekteigenschaften fest  
    (z.B. legt `set(gca(), 'data_bounds', [x_start, y_start; x_end, y_end])` die Achsenskalierung einer gegebenen Grafik neu fest)  
`PlotSparse` - visualisiert eine schwach besetzte Matrix (Default-Farbe der Matrixpunkte ist weiß!)

[Datei auf Webseite](#)

`comp.sce` - Kompass-Plot (Vektoren in komplexer Ebene)  
`Scilab_spy` - visualisiert eine schwach besetzte Matrix

### Dreidimensionale Grafik

`param3d` - dreidimensionale Parameterdarstellung einer Linie

### Darstellung von Flächen

`meshgrid` - erzeugt Punktgitter auf der xy-Ebene  
`surf` - erzeugt eine Oberfläche  
`plot3d` - erzeugt eine Oberfläche  
`colormap` - verändert die Farbe der Oberfläche (Benutzung anders als in MATLAB!)  
`contour2d` - erzeugt Höhenlinien in der xy-Ebene  
`contour` - erzeugt dreidimensionale Höhenlinien  
`rotate_axes` - interaktive Drehung von Achsen einer Grafik

### Achsensteuerung

`xgrid` - Gitterlinien  
`subplot` - erzeugt Graphen in bestimmten Positionen  
`zoom_rect` - Steuerung der Achsenskalierung einer Grafik  
`mtlb_axis` - Steuerung der Achsenskalierung einer Grafik (emuliert die MATLAB-Funktion)  
`square` - steuert die Achsen einer Grafik in einem quadratischen Fenster

### Bezeichnungen an Graphen

`legend` - Legende eines Graphen  
`title` - Titel  
`xlabel` - Beschriftung der x-Achse  
`ylabel` - Beschriftung der y-Achse  
`xtitle` - fasst `title`, `xlabel` und `ylabel` zusammen  
`text` - Platzierung von Text

`get` - liefert die Eigenschaften eines Grafikobjekts

### Einfache Dialogboxen

`x_choose` - erzeugt eine Auswahl-Dialogbox

### Detaillierte Funktionsbeschreibung des `plot`-Kommandos

`plot` - Graph in kartesischen Koordinaten

- `plot(x,y)` gibt eine Grafik als durchgezogene Linie aus, die den Vektor `y` über dem Vektor `x` darstellt.
- `plot(x,y,x,y1)` gibt eine Grafik aus, die den Vektor `y` über dem Vektor `x` und `y1` über `x` darstellt.
- `plot(y)` zeichnet die Spalten von `y` über ihren Indizes.  
Unterschiedliche Linienformen, Symbole und Farben werden mittels `plot(x,y,S)` erzeugt, wobei `S` eine Zeichenkette (String) ist, die aus einem der folgenden Elemente besteht (in Hochkommas eingeschlossen):

<code>k</code>	schwarz („black“)		
<code>y</code>	gelb („yellow“)	<code>.</code>	Punkt - durchgezogene Linie
<code>m</code>	magenta	<code>o</code>	Kreis : punktierte Linie
<code>c</code>	cyan	<code>+</code>	Plus - . strichpunktierte Linie
<code>r</code>	rot	<code>*</code>	Stern -- gestrichelte Linie
<code>g</code>	grün	<code>s</code>	Quadrat („square“)
<code>b</code>	blau	<code>d</code>	Rhombus („diamond“)
<code>w</code>	weiß	<code>x</code>	liegendes Kreuz
<code>k</code>	schwarz („black“)		
<code>^</code>	Dreieck (nach obenweisend)		
<code>&lt;</code>	Dreieck (nach linksweisend)		
<code>&gt;</code>	Dreieck (nach rechtsweisend)		
<code>p</code>	Pentagramm		

Um detailliertere Informationen zu erhalten, sollten Sie unbedingt die Hilfsfunktion in Scilab unter dem Stichwort `GlobalProperty` aufrufen.

#### Beispiele:

- `plot(x,y,'c+:')` zeichnet eine punktierte cyanfarbene Linie, bei der zusätzlich jeder Datenpunkt ein Pluszeichen erhält.
- `plot(x,y,'bd')` zeichnet blaue „Diamant“-Symbole an jedem Datenpunkt, aber keine Linie.

`plot2d` - Graph in kartesischen Koordinaten (alternatives Scilab-Kommando)

- `plot2d(x,y)` gibt eine Grafik als durchgezogene Linie aus, die den Vektor `y` über dem Vektor `x` darstellt.
- `plot2d([x,x], [y,y1])` gibt eine Grafik aus, die den Vektor `y` über dem Vektor `x` und `y1` über `x` darstellt. `x`, `y` und `y1` müssen hierbei *Spaltenvektoren* sein.
- `plot2d(y)` zeichnet die Spalten von `y` über ihren Indizes.  
Unterschiedliche Linienformen, Symbole und Farben werden mittels `plot2d(x,y,opt_args)` erzeugt, wobei `opt_args` optionale Argumente symbolisiert, die folgende Eigenschaften festlegen (in Hochkommas eingeschlossen):


`logflag` - logarithmische Darstellung  
`logflag='ll'` - beide Achsen logarithmisch  
`logflag='ln'` - x-Achse logarithmisch  
`logflag='nl'` - y-Achse logarithmisch  
`logflag='nn'` - keine Achse logarithmisch  
`style` - falls positiv, wird die Linienfarbe festgelegt, falls negativ, wird die Form der Einzelpunkte festgelegt (`plot2d(x,y,style=xx)`).

*Positive Werte:*

1 schwarz	2 dunkelblau	
3 hellgrün	4 himmelblau	5 hellrot
6 purpur	7 hellrot	8 weiß
9 hellblau	10 blau	11 dunkelblau
12 himmelblau	13 dunkelgrün	14 dunkelgrün
15 hellgrün	16 dunkelgrün	17 dunkelblaugrün
18 blaugrün	19 dunkelrot	20 dunkelrot
21 rot	22 dunkelpurpur	23 hellpurpur
24 dunkelrotbraun	25 dunkelrotbraun	26 rotbraun
27 dunkelorange	28 pink	29 pink

(Unterschiedliche Zahlen bedeuten auch leicht unterschiedliche Linienfarben, trotz evtl. gleicher Bezeichnung.)

*Negative Werte:*


  
`xx` = -0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14

Weitere Details sind der Hilfedatei zu `plot2d` zu entnehmen.

*Beispiele:* `plot2d(x,y, style=17)` zeichnet eine punktierte dunkelblaugrüne Linie.  
`plot2d(x,y, style=-7)` zeichnet einzelne Datenpunkte aus kleinen, auf der Spitze stehenden Dreiecken.  
`plot2d(x,y, style=21, logflag='ll')` zeichnet eine rote Linie in doppelt-logarithmischer Darstellung.

### Scilab-Colormaps für dreidimensionale Oberflächen

Aufruf mittels „figure handle“ (hier `h1`) in der Form.

```
h1 = figure(1, 'background', -2); surf(Xgrid, Ygrid, z); h1.color_map = xx(n);
```

`xx` steht für eine der Farbschattierungen

`autumncolormap`, `bonecolormap`, `coolcolormap`, `coppercolormap`, `graycolormap`,  
`hotcolormap`, `hsvcolormap`, `jetcolormap`, `oceancolormap`, `pinkcolormap`, `rain-`  
`bowcolormap`, `springcolormap`, `summercolormap`, `whitecolormap`, `wintercolormap`,

`n` kennzeichnet die Farbbreite.