ave Quick Reference Octave Version 1.1.1

sing Octave

	start interactive Octave session
file	run Octave on commands in *file*
`--help`	describe command line options

ping Octave

r `exit`	exit Octave
UPT	(*e.g.* `C-c`) terminate current command and return to top-level prompt

ing Help

	list all commands and built-in variables
ommand	briefly describe *command*
i	use Info to browse Octave manual
i *command*	search for *command* in Octave manual

on in Info

C-v	scroll forward one screenful
M-v	scroll backward one screenful
	redraw the display

e Selection in Info

	select the next node
	select the previous node
	select the 'up' node
	select the 'top' node
	select the directory node
	select the first node in the current file
	select the last node in the current file
	reads the name of a node and selects it
	kills the current node

ching in Info

	search for a string
	search forward incrementally
	search backward incrementally
	search index & go to corresponding node
	go to next match from last 'i' command

mand-Line Cursor Motion

	move back one character
	move forward one character
	move the the start of the line
	move to the end of the line
	move forward a word
	move backward a word
	clear screen, reprinting current line at top

rting or Changing Text

	insert a tab character
	delete character to the left of the cursor
	delete character under the cursor
	add the next character verbatim
	transpose characters at the point
	transpose words at the point

round optional arguments ... show one or more arguments

Killing and Yanking

`C-k`	kill to the end of the line
`C-y`	yank the most recently killed text
`M-d`	kill to the end of the current word
`M-DEL`	kill the word behind the cursor
`M-y`	rotate the kill ring and yank the new top

Command Completion and History

`TAB`	complete a command or variable name
`M-?`	list possible completions
`RET`	enter the current line
`C-p`	move 'up' through the history list
`C-n`	move 'down' through the history list
`M-<`	move to the first line in the history
`M->`	move to the last line in the history
`C-r`	search backward in the history list
`C-s`	search forward in the history list
`history` [`-q`] [*N*]	list *N* previous history lines, omitting history numbers if `-q`
`history -w` [*file*]	write history to *file* (`~/.octave_hist` if no *file* argument)
`history -r` [*file*]	read history from *file* (`~/.octave_hist` if no *file* argument)
`edit_history` *lines*	edit and then run previous commands from the history list
`run_history` *lines*	run previous commands from the history list
[*beg*] [*end*]	Specify the first and last history commands to edit or run.

If *beg* is greater than *end*, reverse the list of commands before editing. If *end* is omitted, select commands from *beg* to the end of the history list. If both arguments are omitted, edit the previous item in the history list.

Shell Commands

`cd` *dir*	change working directory to *dir*
`pwd`	print working directory
`ls` [*options*]	print directory listing
`getenv` (*string*)	return value of named environment variable
`system` (*cmd*)	execute arbitrary shell command string

Matrices

Square brackets delimit literal matrices. Commas separate elements on the same row. Semicolons separate rows. Commas may be replaced by spaces, and semicolons may be replaced by one or more newlines. Elements of a matrix may be arbitrary expressions, provided that all the dimensions agree.

[*x*, *y*, ...]	enter a row vector
[*x*; *y*; ...]	enter a column vector
[*w*, *x*; *y*, *z*]	enter a 2×2 matrix

Ranges

base : *limit*
base : *incr* : *limit*

Specify a range of values beginning with *base* with no elements greater than *limit*. If it is omitted, the default value of *incr* is 1. Negative increments are permitted.

Strings and Common Escape Sequences

A *string constant* consists of a sequence of characters enclosed in either double-quote or single-quote marks.

`\\`	a literal backslash
`\"`	a literal double-quote character
`\'`	a literal single-quote character
`\n`	newline, ASCII code 10
`\t`	horizontal tab, ASCII code 9

Index Expressions

var (*idx*)	select elements of a vector
var (*idx1*, *idx2*)	select elements of a matrix
scalar	select row (column) corresponding to *scalar*
vector	select rows (columns) corresponding to the elements of *vector*
range	select rows (columns) corresponding to the elements of *range*
:	select all rows (columns)

Global Variables

`global` *var1* ... Declare variables global.

Global variables may be accessed inside the body of a function without having to be passed in the function parameter list provided they are also declared global within the function.

Selected Built-in Variables

`EDITOR`	editor to use with `edit_history`
`Inf, NaN`	IEEE infinity, NaN
`LOADPATH`	path to search for function files
`PAGER`	program to use to paginate output
`ans`	last result not explicitly assigned
`eps`	machine precision
`pi`	π
`realmax`	maximum representable value
`realmin`	minimum representable value

`automatic_replot`	automatically redraw plots
`do_fortran_indexing`	Fortran-style indexing of matrices
`implicit_str_to_num_ok`	allow strings to become numbers
`output_max_field_width`	maximum numeric field width
`output_precision`	min significant figures displayed
`page_screen_output`	control whether output is paged
`prefer_column_vectors`	create column vectors by default
`resize_on_range_error`	automatic resizing of matrices
`save_precision`	digits stored by `save` command
`silent_functions`	suppress output from functions
`warn_divide_by_zero`	suppress divide by zero errors

`commas_in_literal_matrix`
control handling of spaces in matrices

`ignore_function_time_stamp`
ignore changes in function files during session

`ok_to_lose_imaginary_part`
allow complex to real conversion

`prefer_zero_one_indexing`
if ambiguous, prefer 0-1 style indexing

nmetic and Increment Operators

	addition
	subtraction
	matrix multiplication
	element by element multiplication
	right division, conceptually equivalent to `(inverse (y') * x')'`
	element by element right division
	left division, conceptually equivalent to `inverse (x) * y`
	element by element left division
	power operator
	element by element power operator
	negation
	unary plus (a no-op)
	complex conjugate transpose
	transpose
(-- *x*)	increment (decrement) *x*, return *new* value
(*x* --)	increment (decrement) *x*, return *old* value

gnment Expressions

expr	assign expression to variable
idx) = *expr*	assign expression to indexed variable

parison and Boolean Operators

operators work on an element-by-element basis. Both ents are always evaluated.

	true if *x* is less than *y*
	true if *x* is less than or equal to *y*
	true if *x* is greater than *y*
	true if *x* is greater than or equal to *y*
	true if *x* is equal to *y*
	true if *x* is not equal to *y*
	true if both *x* and *y* are true
	true if at least one of *x* or *y* is true
	true *bool* is false

t-circuit Boolean Operators

ors evaluate left-to-right, expecting scalar operands. nds are only evaluated if necessary, stopping once overall value can be determined. Operands are converted to by applying the `all` function.

	true if both *x* and *y* are true
	true if at least one of *x* or *y* is true

rator Precedence

a table of the operators in Octave, in order of ing precedence.

	statement separators
	assignment, groups left to right
	logical "or" and "and"
	element-wise "or" and "and"
= >= > !=	relational operators
	colon
	addition and subtraction
.* ./ .\	multiplication and division
	transpose
-+ -- !	unary minus, increment, logical "not"
	exponentiation

Statements

`for` *identifier* `=` *expr stmt-list* `endfor`

Execute *stmt-list* once for each column of *expr*. The variable *identifier* is set to the value of the current column during each iteration.

`while` (*condition*) *stmt-list* `endwhile`

Execute *stmt-list* while *condition* is true.

`break`	exit innermost loop
`continue`	go to beginning of innermost loop
`return`	return to calling function

`if` (*condition*) *if-body* [`else` *else-body*] `endif`

Execute *if-body* if *condition* is true, otherwise execute *else-body*.

`if` (*condition*) *if-body* [`elseif` (*condition*) *elseif-body*] `endif`

Execute *if-body* if *condition* is true, otherwise execute the *elseif-body* corresponding to the first `elseif` condition that is true, otherwise execute *else-body*.

Any number of `elseif` clauses may appear in an `if` statement.

`unwind_protect` *body* `unwind_protect_cleanup` *cleanup* `end`

Execute *body*. Execute *cleanup* no matter how control exits *body*.

Defining Functions

`function` [*ret-list*] *function-name* [(*arg-list*)]
function-body
`endfunction`

ret-list may be a single identifier or a comma-separated list of identifiers delimited by square-brackets.

arg-list is a comma-separated list of identifiers and may be empty.

Basic Matrix Manipulations

`rows` (*a*)	return number of rows of *a*
`columns` (*a*)	return number of columns of *a*
`all` (*a*)	check if all elements of *a* nonzero
`any` (*a*)	check if any elements of *a* nonzero
`find` (*a*)	return indices of nonzero elements
`sort` (*a*)	order elements in each column of *a*
`sum` (*a*)	sum elements in columns of *a*
`prod` (*a*)	product of elements in columns of *a*
`min` (*args*)	find minimum values
`max` (*args*)	find maximum values
`rem` (*x*, *y*)	find remainder of *x*/*y*
`reshape` (*a*, *m*, *n*)	reformat *a* to be *m* by *n*
`diag` (*v*, *k*)	create diagonal matrices
`linspace` (*b*, *l*, *n*)	create vector of linearly-spaced elements
`logspace` (*b*, *l*, *n*)	create vector of log-spaced elements
`eye` (*n*, *m*)	create *n* by *m* identity matrix
`ones` (*n*, *m*)	create *n* by *m* matrix of ones
`zeros` (*n*, *m*)	create *n* by *m* matrix of zeros
`rand` (*n*, *m*)	create *n* by *m* matrix of random values

Linear Algebra

`chol` (*a*)	Cholesky factorization
`det` (*a*)	compute the determinant of a matrix
`eig` (*a*)	eigenvalues and eigenvectors
`expm` (*a*)	compute the exponential of a matrix
`hess` (*a*)	compute Hessenberg decomposition
`inverse` (*a*)	invert a square matrix
`norm` (*a*, *p*)	compute the *p*-norm of a matrix
`pinv` (*a*)	compute pseudoinverse of *a*
`qr` (*a*)	compute the QR factorization of a matrix
`rank` (*a*)	matrix rank
`schur` (*a*)	Schur decomposition of a matrix
`svd` (*a*)	singular value decomposition
`syl` (*a*, *b*, *c*)	solve the Sylvester equation

Equations, ODEs, DAEs, Quadrature

`*fsolve`	solve nonlinear algebraic equations
`*lsode`	integrate nonlinear ODEs
`*dassl`	integrate nonlinear DAEs
`*quad`	integrate nonlinear functions

`perror` (*nm*, *code*) for functions that return numeric codes, print error message for named function and given error code

`*` See the on-line or printed manual for the complete list of arguments for these functions.

Signal Processing

`fft` (*a*)	Fast Fourier Transform using FFTPACK
`ifft` (*a*)	inverse FFT using FFTPACK
`freqz` (*args*)	FIR filter frequency response
`sinc` (*x*)	returns `sin` (π x)/(π x)

Image Processing

`colormap` (*map*)	set the current colormap
`gray2ind` (*i*, *n*)	convert gray scale to Octave image
`image` (*img*, *zoom*)	display an Octave image matrix
`imagesc` (*img*, *zoom*)	display scaled matrix as image
`imshow` (*img*, *map*)	display Octave image
`imshow` (*i*, *n*)	display gray scale image
`imshow` (*r*, *g*, *b*)	display RGB image
`ind2gray` (*img*, *map*)	convert Octave image to gray scale
`ind2rgb` (*img*, *map*)	convert indexed image to RGB
`loadimage` (*file*)	load an image file
`rgb2ind` (*r*, *g*, *b*)	convert RGB to Octave image
`saveimage` (*file*, *img*, *fmt*, *map*)	save a matrix to *file*

Sets

`create_set` (*a*, *b*)	create row vector of unique values
`complement` (*a*, *b*)	elements of *b* not in *a*
`intersection` (*a*, *b*)	intersection of sets *a* and *b*
`union` (*a*, *b*)	union of sets *a* and *b*

Strings

`strcmp` (*s*, *t*)	compare strings
`strcat` (*s*, *t*, ...)	concatenate strings

yle Input and Output

(*name*, *mode*)	open file *name*
(*file*)	close *file*
(*fmt*, ...)	formatted output to `stdout`
f (*file*, *fmt*, ...)	formatted output to *file*
f (*fmt*, ...)	formatted output to string
(*fmt*)	formatted input from `stdin`
(*file*, *fmt*)	formatted input from *file*
(*str*, *fmt*)	formatted input from *string*
(*file*, *len*)	read *len* characters from *file*
(*file*)	flush pending output to *file*
(*file*)	return file pointer position
d (*file*)	move file pointer to beginning
t	print a info for open files
(*file*, *size*, *prec*)	read binary data files
(*file*, *size*, *prec*)	write binary data files
file)	determine if pointer is at EOF

may be referenced either by name or by the number
d from `fopen`. Three files are preconnected when Octave
`stdin`, `stdout`, and `stderr`.

r Input and Output functions

le var ...	save variables in *file*
le	load variables from *file*
var)	display value of *var* to screen

ellaneous Functions

str)	evaluate *str* as a command
(*str*, ...)	evaluate function named by *str*, passing remaining args to called function
(*message*)	print message and return to top level
pattern	clear variables matching pattern
(*str*)	check existence of variable or function
	list current variables

nomials

(*p*)	companion matrix
a, *b*)	convolution
(*a*, *b*)	deconvolve two vectors
a)	create polynomial from a matrix
riv (*p*)	derivative of polynomial
duce (*p*)	integral of polynomial
l (*p*, *x*)	value of polynomial at *x*
lm (*p*, *x*)	value of polynomial at *x*
(*p*)	polynomial roots
e (*a*, *b*)	partial fraction expansion of ratio *a*/*b*

stics

ef (*x*, *y*)	correlation coefficient
, *y*)	covariance
a)	mean value
(*a*)	median value
)	standard deviation
)	variance

Basic Plotting

`gplot` [*ranges*] *expr* [*using*] [*title*] [*style*]	2D plotting
`gsplot` [*ranges*] *expr* [*using*] [*title*] [*style*]	3D plotting

ranges	specify data ranges
expr	expression to plot
using	specify columns to plot
title	specify line title for legend
style	specify line style

If *ranges* are supplied, they must come before the expression to plot. The *using*, *title*, and *style* options may appear in any order after *expr*. Multiple expressions may be plotted with a single command by separating them with commas.

`set` *options*	set plotting options
`show` *options*	show plotting options
`replot`	redisplay current plot
`closeplot`	close stream to `gnuplot` process
`purge_tmp_files`	clean up temporary plotting files
`automatic_replot`	built-in variable

Other Plotting Functions

`plot` (*args*)	2D plot with linear axes
`semilogx` (*args*)	2D plot with logarithmic x-axis
`semilogy` (*args*)	2D plot with logarithmic y-axis
`loglog` (*args*)	2D plot with logarithmic axes
`bar` (*args*)	plot bar charts
`stairs` (*x*, *y*)	plot stairsteps
`hist` (*y*, *x*)	plot histograms
`title` (*string*)	set plot title
`axis` (*limits*)	set axis ranges
`xlabel` (*string*)	set x-axis label
`ylabel` (*string*)	set y-axis label
`grid` [on|off]	set grid state
`hold` [on|off]	set hold state
`ishold`	return 1 if hold is on, 0 otherwise
`mesh` (*x*, *y*, *z*)	plot 3D surface
`meshdom` (*x*, *y*)	create mesh coordinate matrices
