

MATLAB commands in numerical Python (NumPy)

Copyright ©2006 Vidar Bronken Gundersen

Permission is granted to copy, distribute and/or modify this document as long as the above attribution is kept and the resulting work is distributed under a license identical to this one.

The idea of this document (and the corresponding XML instance) is to provide a quick reference¹ for switching from MATLAB to an open-source environment, such as Python, Scilab, Octave and Gnuplot, or R for numeric processing and data visualisation.

Where Octave and Scilab commands are omitted, expect Matlab compatibility, and similarly where non given use the generic command.

Time-stamp: 2007-11-09T16:46:36 vidar

1 Help

| Language | MATLAB/Octave | Python | R |
|------------------------------------|---|---------------------|-------------------------|
| Browse help interactively | doc Octave: help -i % browse with Info | help() | help.start() |
| Help on using help | help help or doc doc | help | help() |
| Help for a function | help plot | help(plot) or ?plot | help(plot) or ?plot |
| Help for a toolbox/library package | help splines or doc splines | help(pylab) | help(package='splines') |
| Demonstration examples | demo | | demo() |
| Example using a function | | | example(plot) |

1.1 Searching available documentation

| Language | MATLAB/Octave | Python | R |
|---------------------------------------|---------------|---------------------------|---------------------|
| Search help files | lookfor plot | | help.search('plot') |
| Find objects by partial name | | | apropos('plot') |
| List available packages | help | help(); modules [Numeric] | library() |
| Locate functions | which plot | help(plot) | find(plot) |
| List available methods for a function | | | methods(plot) |

1.2 Using interactively

| Language | MATLAB/Octave | Python | R |
|----------------------|--------------------------|--|-------------------------------|
| Start session | Octave: octave -q | ipython -pylab or JupyterLab | RStudio |
| Auto completion | Octave: TAB or M-? | TAB | |
| Run code from file | foo(.m) | execfile('foo.py') or run foo.py | source('foo.R') |
| Command history | Octave: history | hist -n | history() |
| Save command history | diary on [...] diary off | | savehistory(file=".Rhistory") |
| End session | exit or quit | CTRL-D CTRL-Z # windows sys.exit() | q(save='no') |

2 Operators

| Language | MATLAB/Octave | Python | R |
|-------------------------|---------------|--------|--------------|
| Help on operator syntax | help - | | help(Syntax) |

¹References: Hankin, Robin. *R for Octave users* (2001), available from <http://cran.r-project.org/doc/contrib/R-and-octave-2.txt> (accessed 2005.07.24); Martelli, Alex. *Python in a Nutshell* (O'Reilly, 2003); Oliphant, Travis. *Guide to NumPy* (Trelgol, 2006); Hunter, John. *The Matplotlib User's Guide* (2005), available from <http://matplotlib.sf.net/> (accessed 2005.07.31); Langtangen, Hans Petter. *Python Scripting for Computational Science* (Springer, 2004); Ascher et al.: *Numeric Python manual* (2001), available from <http://numeric.scipy.org/numpy.pdf> (accessed 2005.06.25); Moler, Cleve. *Numerical Computing with MATLAB* (MathWorks, 2004), available from <http://www.mathworks.com/moler/> (accessed 2005.03.10); Eaton, John W. *Octave Quick Reference* (1996); Merrit, Ethan. *Demo scripts for gnuplot version 4.0* (2004), available from <http://gnuplot.sourceforge.net/demo/> (accessed 2005.07.24); Woo, Alex. *Gnuplot Quick Reference* (2004), available from <http://www.gnuplot.info/docs/gpcard.pdf> (accessed 2005.07.14); Venables & Smith: *An Introduction to R* (2005), available from <http://cran.r-project.org/doc/manuals/R-intro.pdf> (accessed 2005.07.25); Short, Tom. *R reference card* (2005), available from <http://www.rpad.org/Rpad/R-refcard.pdf> (accessed 2005.07.24).

2.1 Arithmetic operators

| | | | |
|--|----------------------------------|---|-------------------------------|
| Language | MATLAB/Octave | Python | R |
| Assignment; defining a number | <code>a=1; b=2;</code> | <code>a=1; b=1</code> | <code>a<-1; b<-2</code> |
| Addition | <code>a + b</code> | <code>a + b</code> or <code>add(a,b)</code> | <code>a + b</code> |
| Subtraction | <code>a - b</code> | <code>a - b</code> or <code>subtract(a,b)</code> | <code>a - b</code> |
| Multiplication | <code>a * b</code> | <code>a * b</code> or <code>multiply(a,b)</code> | <code>a * b</code> |
| Division | <code>a / b</code> | <code>a / b</code> or <code>divide(a,b)</code> | <code>a / b</code> |
| Power, a^b | <code>a .^ b</code> | <code>a ** b</code> | <code>a ^ b</code> |
| Remainder | <code>rem(a,b)</code> | <code>power(a,b)</code> <code>pow(a,b)</code> <code>a % b</code> <code>remainder(a,b)</code> <code>fmod(a,b)</code> | <code>a %% b</code> |
| Integer division | | | <code>a %/% b</code> |
| In place operation to save array creation overhead | Octave: <code>a+=1</code> | <code>a+=b</code> or <code>add(a,b,a)</code> | |
| Factorial, $n!$ | <code>factorial(a)</code> | | <code>factorial(a)</code> |

2.2 Relational operators

| | | | |
|-----------------------|------------------------|---|------------------------|
| Language | MATLAB/Octave | Python | R |
| Equal | <code>a == b</code> | <code>a == b</code> or <code>equal(a,b)</code> | <code>a == b</code> |
| Less than | <code>a < b</code> | <code>a < b</code> or <code>less(a,b)</code> | <code>a < b</code> |
| Greater than | <code>a > b</code> | <code>a > b</code> or <code>greater(a,b)</code> | <code>a > b</code> |
| Less than or equal | <code>a <= b</code> | <code>a <= b</code> or <code>less_equal(a,b)</code> | <code>a <= b</code> |
| Greater than or equal | <code>a >= b</code> | <code>a >= b</code> or <code>greater_equal(a,b)</code> | <code>a >= b</code> |
| Not Equal | <code>a ~= b</code> | <code>a != b</code> or <code>not_equal(a,b)</code> | <code>a != b</code> |

2.3 Logical operators

| | | | |
|----------------------------------|--|---|-----------------------------|
| Language | MATLAB/Octave | Python | R |
| Short-circuit logical AND | <code>a && b</code> | <code>a and b</code> | <code>a && b</code> |
| Short-circuit logical OR | <code>a b</code> | <code>a or b</code> | <code>a b</code> |
| Element-wise logical AND | <code>a & b</code> or <code>and(a,b)</code> | <code>logical_and(a,b)</code> or <code>a and b</code> | <code>a & b</code> |
| Element-wise logical OR | <code>a b</code> or <code>or(a,b)</code> | <code>logical_or(a,b)</code> or <code>a or b</code> | <code>a b</code> |
| Logical EXCLUSIVE OR | <code>xor(a, b)</code> | <code>logical_xor(a,b)</code> | <code>xor(a, b)</code> |
| Logical NOT | <code>~a</code> or <code>not(a)</code> | <code>logical_not(a)</code> or <code>not a</code> | <code>!a</code> |
| True if any element is nonzero | Octave: <code>~a</code> or <code>!a</code> <code>any(a)</code> | | |
| True if all elements are nonzero | <code>all(a)</code> | | |

2.4 root and logarithm

| | | | | |
|-------------------------------|-----------------------|-----------------------------|-----------------------|--------------------|
| Language | MATLAB/Octave | Python | R | |
| Square root | <code>sqrt(a)</code> | <code>math.sqrt(a)</code> | <code>sqrt(a)</code> | \sqrt{a} |
| Logarithm, base e (natural) | <code>log(a)</code> | <code>math.log(a)</code> | <code>log(a)</code> | $\ln a = \log_e a$ |
| Logarithm, base 10 | <code>log10(a)</code> | <code>math.log10(a)</code> | <code>log10(a)</code> | $\log_{10} a$ |
| Logarithm, base 2 (binary) | <code>log2(a)</code> | <code>math.log(a, 2)</code> | <code>log2(a)</code> | $\log_2 a$ |
| Exponential function | <code>exp(a)</code> | <code>math.exp(a)</code> | <code>exp(a)</code> | e^a |

2.5 Round off

| Language | MATLAB/Octave | Python | R |
|--------------------|-----------------------|--|-----------------------|
| Round | <code>round(a)</code> | <code>around(a)</code> or <code>math.round(a)</code> | <code>round(a)</code> |
| Round up | <code>ceil(a)</code> | <code>ceil(a)</code> | <code>ceil(a)</code> |
| Round down | <code>floor(a)</code> | <code>floor(a)</code> | <code>floor(a)</code> |
| Round towards zero | <code>fix(a)</code> | <code>fix(a)</code> | |

2.6 Mathematical constants

| Language | MATLAB/Octave | Python | R |
|------------------|---------------------|---|---------------------|
| $\pi = 3.141592$ | <code>pi</code> | <code>math.pi</code> | <code>pi</code> |
| $e = 2.718281$ | <code>exp(1)</code> | <code>math.e</code> or <code>math.exp(1)</code> | <code>exp(1)</code> |

2.6.1 Missing values; IEEE-754 floating point status flags

| Language | MATLAB/Octave | Python | R |
|---------------------|------------------|-------------------------|---|
| Not a Number | <code>NaN</code> | <code>nan</code> | |
| Infinity, ∞ | <code>Inf</code> | <code>inf</code> | |
| Infinity, $+\infty$ | | <code>plus_inf</code> | |
| Infinity, $-\infty$ | | <code>minus_inf</code> | |
| Plus zero, $+0$ | | <code>plus_zero</code> | |
| Minus zero, -0 | | <code>minus_zero</code> | |

2.7 Complex numbers

| Language | MATLAB/Octave | Python | R |
|----------------------------|-----------------------|--|--|
| Imaginary unit | <code>i</code> | <code>z = 1j</code> | <code>1i</code> |
| A complex number, $3 + 4i$ | <code>z = 3+4i</code> | <code>z = 3+4j</code> or <code>z = complex(3,4)</code> | <code>z <- 3+4i</code> |
| Absolute value (modulus) | <code>abs(z)</code> | <code>abs(3+4j)</code> | <code>abs(3+4i)</code> or <code>Mod(3+4i)</code> |
| Real part | <code>real(z)</code> | <code>z.real</code> | <code>Re(3+4i)</code> |
| Imaginary part | <code>imag(z)</code> | <code>z.imag</code> | <code>Im(3+4i)</code> |
| Argument | <code>arg(z)</code> | | <code>Arg(3+4i)</code> |
| Complex conjugate | <code>conj(z)</code> | <code>z.conj(); z.conjugate()</code> | <code>Conj(3+4i)</code> |

$$i = \sqrt{-1}$$

2.8 Trigonometry

| Language | MATLAB/Octave | Python | R |
|--------------------------------|------------------------|-------------------------|-------------------------|
| Arctangent, $\arctan(b/a)$ | <code>atan(a,b)</code> | <code>atan2(b,a)</code> | <code>atan2(b,a)</code> |
| Hypotenuse; Euclidean distance | | <code>hypot(x,y)</code> | |

$$\sqrt{x^2 + y^2}$$

2.9 Generate random numbers

| Language | MATLAB/Octave | Python | R |
|----------------------------------|-----------------------------|---|--------------------------------------|
| Uniform distribution | <code>rand(1,10)</code> | <code>random.random((10,))</code> <code>random.uniform((10,))</code> | <code>runif(10)</code> |
| Uniform: Numbers between 2 and 7 | <code>2+5*rand(1,10)</code> | <code>random.uniform(2,7,(10,))</code> | <code>runif(10, min=2, max=7)</code> |
| Uniform: 6,6 array | <code>rand(6)</code> | <code>random.uniform(0,1,(6,6))</code> | <code>matrix(runif(36),6)</code> |
| Normal distribution | <code>randn(1,10)</code> | <code>random.standard_normal((10,))</code> | <code>rnorm(10)</code> |

3 Vectors

Language
 Row vector, $1 \times n$ -matrix
 Column vector, $m \times 1$ -matrix

MATLAB/Octave
`a=[2 3 4 5];`
`adash=[2 3 4 5]';`

Python
`a=array([2,3,4,5])`
`array([2,3,4,5])[:,NewAxis]`
`array([2,3,4,5]).reshape(-1,1)`
`r_[1:10,'c']`

R
`a <- c(2,3,4,5)`
`adash <- t(c(2,3,4,5))`

3.1 Sequences

Language
`1,2,3, ... ,10`

MATLAB/Octave
`1:10`

`0.0,1.0,2.0, ... ,9.0`
`1,4,7,10`
`10,9,8, ... ,1`
`10,7,4,1`
 Linearly spaced vector of n=7 points
 Reverse
 Set all values to same scalar value

`0:9`
`1:3:10`
`10:-1:1`
`10:-3:1`
`linspace(1,10,7)`
`reverse(a)`
`a(:) = 3`

Python
`arange(1,11, dtype=Float)`
`range(1,11)`
`arange(10.)`
`arange(1,11,3)`
`arange(10,0,-1)`
`arange(10,0,-3)`
`linspace(1,10,7)`
`a[::-1] or`
`a.fill(3), a[:] = 3`

R
`seq(10) or 1:10`
`seq(0,length=10)`
`seq(1,10,by=3)`
`seq(10,1) or 10:1`
`seq(from=10,to=1,by=-3)`
`seq(1,10,length=7)`
`rev(a)`

3.2 Concatenation (vectors)

Language
 Concatenate two vectors

MATLAB/Octave
`[a a]`
`[1:4 a]`

Python
`concatenate((a,a))`
`concatenate((range(1,5),a), axis=1)`

R
`c(a,a)`
`c(1:4,a)`

3.3 Repeating

Language
`1 2 3, 1 2 3`
`1 1 1, 2 2 2, 3 3 3`
`1, 2 2, 3 3 3`

MATLAB/Octave
`[a a]`

Python
`concatenate((a,a))`
`a.repeat(3) or`
`a.repeat(a) or`

R
`rep(a,times=2)`
`rep(a,each=3)`
`rep(a,a)`

3.4 Miss those elements out

Language
 miss the first element
 miss the tenth element
 miss 1,4,7, ...
 last element
 last two elements

MATLAB/Octave
`a(2:end)`
`a([1:9])`
`a(end)`
`a(end-1:end)`

Python
`a[1:]`
`a[-1]`
`a[-2:]`

R
`a[-1]`
`a[-10]`
`a[-seq(1,50,3)]`

3.5 Maximum and minimum

Language
 pairwise max
 max of all values in two vectors

MATLAB/Octave
`max(a,b)`
`max([a b])`
`[v,i] = max(a)`

Python
`maximum(a,b)`
`concatenate((a,b)).max()`
`v,i = a.max(0),a.argmax(0)`

R
`pmax(a,b)`
`max(a,b)`
`v <- max(a) ; i <- which.max(a)`

3.6 Vector multiplication

Language
 Multiply two vectors
 Vector dot product, $u \cdot v$

MATLAB/Octave
`a.*a`
`dot(u,v)`

Python
`a*a`
`dot(u,v)`

R
`a*a`

4 Matrices

Language
 Define a matrix

MATLAB/Octave
`a = [2 3;4 5]`

Python
`a = array([[2,3],[4,5]])`

R
`rbind(c(2,3),c(4,5))`
`array(c(2,3,4,5), dim=c(2,2))`

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

4.1 Concatenation (matrices); rbind and cbind

Language
 Bind rows
 Bind columns

MATLAB/Octave
`[a ; b]`
`[a , b]`

Python
`concatenate((a,b), axis=0)`
`vstack((a,b))`
`concatenate((a,b), axis=1)`
`hstack((a,b))`
`concatenate((a,b), axis=2)`
`dstack((a,b))`
`concatenate((a,b), axis=None)`
`concatenate((r_[1:5],r_[1:5])).reshape(2,-1)`
`vstack((r_[1:5],r_[1:5]))`

R
`rbind(a,b)`
`cbind(a,b)`
`cbind(1:4,1:4)`

Bind slices (three-way arrays)
 Concatenate matrices into one vector
 Bind rows (from vectors)
 Bind columns (from vectors)

`[a(:), b(:)]`
`[1:4 ; 1:4]`
`[1:4 ; 1:4]'`

4.2 Array creation

Language
 o filled array
 o filled array of integers
 1 filled array
 Any number filled array
 Identity matrix
 Diagonal
 Magic squares; Lo Shu
 Empty array

MATLAB/Octave
`zeros(3,5)`
`ones(3,5)`
`ones(3,5)*9`
`eye(3)`
`diag([4 5 6])`
`magic(3)`

Python
`zeros((3,5),Float)`
`zeros((3,5))`
`ones((3,5),Float)`
`identity(3)`
`diag((4,5,6))`
`a = empty((3,3))`

R
`matrix(0,3,5) or array(0,c(3,5))`
`matrix(1,3,5) or array(1,c(3,5))`
`matrix(9,3,5) or array(9,c(3,5))`
`diag(1,3)`
`diag(c(4,5,6))`

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

4.3 Reshape and flatten matrices

| Language | MATLAB/Octave | Python | R | |
|--|--------------------------------|--|---|--|
| Reshaping (rows first) | <code>reshape(1:6,3,2)'</code> | <code>arange(1,7).reshape(2,-1)</code> <code>a.setshape(2,3)</code> | <code>matrix(1:6,nrow=3,byrow=T)</code> | $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ |
| Reshaping (columns first) | <code>reshape(1:6,2,3)</code> | <code>arange(1,7).reshape(-1,2).transpose()</code> | <code>matrix(1:6,nrow=2)</code> <code>array(1:6,c(2,3))</code> | $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ |
| Flatten to vector (by rows, like comics) | <code>a'(:)</code> | <code>a.flatten()</code> or | <code>as.vector(t(a))</code> | $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 5 & 3 & 6 \end{bmatrix}$ |
| Flatten to vector (by columns) | <code>a(:)</code> | <code>a.flatten(1)</code> | <code>as.vector(a)</code> | |
| Flatten upper triangle (by columns) | <code>vech(a)</code> | | <code>a[row(a) <= col(a)]</code> | |

4.4 Shared data (slicing)

| Language | MATLAB/Octave | Python | R |
|-----------|--------------------|---------------------------|--------------------|
| Copy of a | <code>b = a</code> | <code>b = a.copy()</code> | <code>b = a</code> |

4.5 Indexing and accessing elements (Python: slicing)

| Language | MATLAB/Octave | Python | R | |
|--------------------------------|--|--|---|---|
| Input is a 3,4 array | <code>a = [11 12 13 14 ... 21 22 23 24 ... 31 32 33 34]</code> | <code>a = array([[11, 12, 13, 14], [21, 22, 23, 24], [31, 32, 33, 34]])</code> | <code>a <- rbind(c(11, 12, 13, 14), c(21, 22, 23, 24), c(31, 32, 33, 34))</code> | $\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$ |
| Element 2,3 (row,col) | <code>a(2,3)</code> | <code>a[1,2]</code> | <code>a[2,3]</code> | $\begin{matrix} a_{23} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \end{matrix}$ |
| First row | <code>a(1,:)</code> | <code>a[0,]</code> | <code>a[1,]</code> | |
| First column | <code>a(:,1)</code> | <code>a[:,0]</code> | <code>a[,1]</code> | |
| Array as indices | <code>a([1 3],[1 4]);</code> | <code>a.take([0,2]).take([0,3], axis=1)</code> | | $\begin{bmatrix} a_{11} & a_{14} \\ a_{31} & a_{34} \end{bmatrix}$ |
| All, except first row | <code>a(2:end,:)</code> | <code>a[1:,:]</code> | <code>a[-1,]</code> | $\begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$ |
| Last two rows | <code>a(end-1:end,:)</code> | <code>a[-2:,:]</code> | | $\begin{bmatrix} a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$ |
| Strides: Every other row | <code>a(1:2:end,:)</code> | <code>a[:,2:,:]</code> | | $\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$ |
| Third in last dimension (axis) | | <code>a[... ,2]</code> | | |
| All, except row,column (2,3) | | | <code>a[-2,-3]</code> | $\begin{bmatrix} a_{11} & a_{13} & a_{14} \\ a_{31} & a_{33} & a_{34} \end{bmatrix}$ |
| Remove one column | <code>a(:,[1 3 4])</code> | <code>a.take([0,2,3],axis=1)</code> | <code>a[, -2]</code> | $\begin{bmatrix} a_{11} & a_{13} & a_{14} \\ a_{21} & a_{23} & a_{24} \\ a_{31} & a_{33} & a_{34} \end{bmatrix}$ |
| Diagonal | | <code>a.diagonal(offset=0)</code> | | $\begin{bmatrix} a_{11} & a_{22} & a_{33} & a_{44} \end{bmatrix}$ |

4.6 Assignment

| Language | MATLAB/Octave | Python | R |
|--|---|---|---|
| Clipping: Replace all elements over 90 | <pre>a(:,1) = 99 a(:,1) = [99 98 97]'</pre> | <pre>a[:,0] = 99 a[:,0] = array([99,98,97]) (a>90).choose(a,90) a.clip(min=None, max=90)</pre> | <pre>a[,1] <- 99 a[,1] <- c(99,98,97) a[a>90] <- 90</pre> |
| Clip upper and lower values | | <pre>a.clip(min=2, max=5)</pre> | |

4.7 Transpose and inverse

| Language | MATLAB/Octave | Python | R |
|-------------------------|--------------------------------|---------------------------------|------------------------------|
| Transpose | <pre>a'</pre> | <pre>a.conj().transpose()</pre> | <pre>t(a)</pre> |
| Non-conjugate transpose | <pre>a.' or transpose(a)</pre> | <pre>a.transpose()</pre> | |
| Determinant | <pre>det(a)</pre> | <pre>linalg.det(a) or</pre> | <pre>det(a)</pre> |
| Inverse | <pre>inv(a)</pre> | <pre>linalg.inv(a) or</pre> | <pre>solve(a)</pre> |
| Pseudo-inverse | <pre>pinv(a)</pre> | <pre>linalg.pinv(a)</pre> | <pre>ginv(a)</pre> |
| Norms | <pre>norm(a)</pre> | <pre>norm(a)</pre> | |
| Eigenvalues | <pre>eig(a)</pre> | <pre>linalg.eig(a)[0]</pre> | <pre>eigen(a)\$values</pre> |
| Singular values | <pre>svd(a)</pre> | <pre>linalg.svd(a)</pre> | <pre>svd(a)\$d</pre> |
| Cholesky factorization | <pre>chol(a)</pre> | <pre>linalg.cholesky(a)</pre> | |
| Eigenvectors | <pre>[v,1] = eig(a)</pre> | <pre>linalg.eig(a)[1]</pre> | <pre>eigen(a)\$vectors</pre> |
| Rank | <pre>rank(a)</pre> | <pre>rank(a)</pre> | <pre>rank(a)</pre> |

4.8 Sum

| Language | MATLAB/Octave | Python | R |
|--------------------------|------------------------|------------------------------|------------------------------|
| Sum of each column | <pre>sum(a)</pre> | <pre>a.sum(axis=0)</pre> | <pre>apply(a,2,sum)</pre> |
| Sum of each row | <pre>sum(a')</pre> | <pre>a.sum(axis=1)</pre> | <pre>apply(a,1,sum)</pre> |
| Sum of all elements | <pre>sum(sum(a))</pre> | <pre>a.sum()</pre> | <pre>sum(a)</pre> |
| Sum along diagonal | | <pre>a.trace(offset=0)</pre> | |
| Cumulative sum (columns) | <pre>cumsum(a)</pre> | <pre>a.cumsum(axis=0)</pre> | <pre>apply(a,2,cumsum)</pre> |

4.9 Sorting

| Language | MATLAB/Octave | Python | R |
|----------------------------------|--|---|---------------------------------|
| Example data | <code>a = [4 3 2 ; 2 8 6 ; 1 4 7]</code> | <code>a = array([[4,3,2],[2,8,6],[1,4,7]])</code> | |
| Flat and sorted | <code>sort(a(:))</code> | <code>a.ravel().sort()</code> <i>or</i> | <code>t(sort(a))</code> |
| Sort each column | <code>sort(a)</code> | <code>a.sort(axis=0)</code> <i>or</i> <code>msort(a)</code> | <code>apply(a,2,sort)</code> |
| Sort each row | <code>sort(a')'</code> | <code>a.sort(axis=1)</code> | <code>t(apply(a,1,sort))</code> |
| Sort rows (by first row) | <code>sortrows(a,1)</code> | <code>a[a[:,0].argsort(),:]</code> | |
| Sort, return indices | | <code>a.ravel().argsort()</code> | <code>order(a)</code> |
| Sort each column, return indices | | <code>a.argsort(axis=0)</code> | |
| Sort each row, return indices | | <code>a.argsort(axis=1)</code> | |

$$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 8 & 6 \\ 1 & 4 & 7 \\ 1 & 2 & 2 \\ 3 & 4 & 4 \\ 6 & 7 & 8 \\ 1 & 3 & 2 \\ 2 & 4 & 6 \\ 4 & 8 & 7 \\ 2 & 3 & 4 \\ 2 & 6 & 8 \\ 1 & 4 & 7 \\ 1 & 4 & 7 \\ 2 & 8 & 6 \\ 4 & 3 & 2 \end{bmatrix}$$

4.10 Maximum and minimum

| Language | MATLAB/Octave | Python | R |
|--------------------|-----------------------------|--|---|
| max in each column | <code>max(a)</code> | <code>a.max(0)</code> <i>or</i> <code>amax(a [,axis=0])</code> | <code>apply(a,2,max)</code> |
| max in each row | <code>max(a')</code> | <code>a.max(1)</code> <i>or</i> <code>amax(a, axis=1)</code> | <code>apply(a,1,max)</code> |
| max in array | <code>max(max(a))</code> | <code>a.max()</code> <i>or</i> | <code>max(a)</code> |
| return indices, i | <code>[v i] = max(a)</code> | | <code>i <- apply(a,1,which.max)</code> |
| pairwise max | <code>max(b,c)</code> | <code>maximum(b,c)</code> | <code>pmax(b,c)</code> |
| | <code>cummax(a)</code> | <code>a.ptp(); a.ptp(0)</code> | <code>apply(a,2,cummax)</code> |

4.11 Matrix manipulation

| Language | MATLAB/Octave | Python | R |
|----------------------------------|---|---|---|
| Flip left-right | <code>fliplr(a)</code> | <code>fliplr(a)</code> <i>or</i> <code>a[:,::-1]</code> | <code>a[,4:1]</code> |
| Flip up-down | <code>flipud(a)</code> | <code>flipud(a)</code> <i>or</i> <code>a[::-1,]</code> | <code>a[3:1,]</code> |
| Rotate 90 degrees | <code>rot90(a)</code> | <code>rot90(a)</code> | |
| Repeat matrix: [a a a ; a a a] | <code>repmat(a,2,3)</code> <i>Octave:</i> <code>kron(ones(2,3),a)</code> | <code>kron(ones((2,3)),a)</code> | <code>kronecker(matrix(1,2,3),a)</code> |
| Triangular, upper | <code>triu(a)</code> | <code>triu(a)</code> | <code>a[lower.tri(a)] <- 0</code> |
| Triangular, lower | <code>tril(a)</code> | <code>tril(a)</code> | <code>a[upper.tri(a)] <- 0</code> |

4.12 Equivalent to "size"

| Language | MATLAB/Octave | Python | R |
|--------------------------------|---|---|-----------------------------|
| Matrix dimensions | <code>size(a)</code> | <code>a.shape</code> <i>or</i> <code>a.getshape()</code> | <code>dim(a)</code> |
| Number of columns | <code>size(a,2)</code> <i>or</i> <code>length(a)</code> | <code>a.shape[1]</code> <i>or</i> <code>size(a, axis=1)</code> | <code>ncol(a)</code> |
| Number of elements | <code>length(a(:))</code> | <code>a.size</code> <i>or</i> <code>size(a[, axis=None])</code> | <code>prod(dim(a))</code> |
| Number of dimensions | <code>ndims(a)</code> | <code>a.ndim</code> | |
| Number of bytes used in memory | | <code>a.nbytes</code> | <code>object.size(a)</code> |

4.13 Matrix- and elementwise- multiplication

| Language | MATLAB/Octave | Python | R |
|--|------------------------|--|--|
| Elementwise operations | <code>a .* b</code> | <code>a * b</code> or <code>multiply(a,b)</code> | <code>a * b</code> |
| Matrix product (dot product) | <code>a * b</code> | <code>matrixmultiply(a,b)</code> | <code>a %*% b</code> |
| Inner matrix vector multiplication $a \cdot b'$ | | <code>inner(a,b)</code> or | |
| Outer product | | <code>outer(a,b)</code> or | <code>outer(a,b)</code> or <code>a %o% b</code> |
| Cross product | | | <code>crossprod(a,b)</code> or <code>t(a) %*% b</code> |
| Kronecker product | <code>kron(a,b)</code> | <code>kron(a,b)</code> | <code>kronecker(a,b)</code> |
| Matrix division, $b \cdot a^{-1}$ | <code>a / b</code> | | |
| Left matrix division, $b^{-1} \cdot a$ (solve linear equations) | <code>a \ b</code> | <code>linalg.solve(a,b)</code> | <code>solve(a,b)</code> |
| Vector dot product | | <code>vdot(a,b)</code> | |
| Cross product | | <code>cross(a,b)</code> | |

$$\begin{bmatrix} 1 & 5 \\ 9 & 16 \end{bmatrix} \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix} = \begin{bmatrix} 5 & 11 \\ 11 & 25 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 4 & 8 & 12 & 16 \end{bmatrix} \begin{bmatrix} 10 & 14 \\ 14 & 20 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 4 & 6 & 8 \\ 3 & 6 & 4 & 8 \\ 9 & 12 & 12 & 16 \end{bmatrix}$$

$$Ax = b$$

4.14 Find; conditional indexing

| Language | MATLAB/Octave | Python | R |
|----------------------------------|--------------------------------|--|---|
| Non-zero elements, indices | <code>find(a)</code> | <code>a.ravel().nonzero()</code> | <code>which(a != 0)</code> |
| Non-zero elements, array indices | <code>[i j] = find(a)</code> | <code>(i,j) = a.nonzero()</code> <code>(i,j) = where(a!=0)</code> | <code>which(a != 0, arr.ind=T)</code> |
| Vector of non-zero values | <code>[i j v] = find(a)</code> | <code>v = a.compress((a!=0).flat)</code> <code>v = extract(a!=0,a)</code> | <code>ij <- which(a != 0, arr.ind=T); v <- a[ij]</code> |
| Condition, indices | <code>find(a>5.5)</code> | <code>(a>5.5).nonzero()</code> | <code>which(a>5.5)</code> |
| Return values | | <code>a.compress((a>5.5).flat)</code> | <code>ij <- which(a>5.5, arr.ind=T); v <- a[ij]</code> |
| Zero out elements above 5.5 | <code>a .* (a>5.5)</code> | <code>where(a>5.5,0,a)</code> or <code>a * (a>5.5)</code> | |
| Replace values | | <code>a.put(2,indices)</code> | |

5 Multi-way arrays

| Language | MATLAB/Octave | Python | R |
|----------------------|--|---|---|
| Define a 3-way array | <code>a = cat(3, [1 2; 1 2],[3 4; 3 4]);</code> <code>a(1, :, :)</code> | <code>a = array([[[1,2],[1,2]], [[3,4],[3,4]]])</code> <code>a[0,...]</code> | |

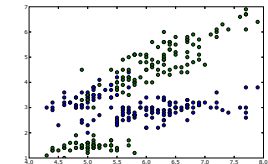
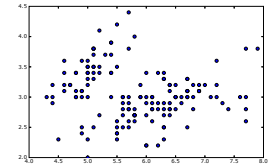
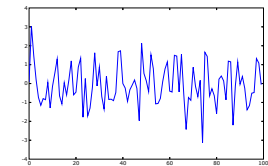
6 File input and output

| Language | MATLAB/Octave | Python | R |
|------------------------------|--|---|---|
| Reading from a file (2d) | <code>f = load('data.txt')</code> | <code>f = fromfile("data.txt")</code> <code>f = load("data.txt")</code> | <code>f <- read.table("data.txt")</code> |
| Reading from a file (2d) | <code>f = load('data.txt')</code> | <code>f = load("data.txt")</code> | <code>f <- read.table("data.txt")</code> |
| Reading from a CSV file (2d) | <code>x = dlmread('data.csv', ' ');</code> | <code>f = load('data.csv', delimiter=',')</code> | <code>f <- read.table(file="data.csv", sep=";")</code> |
| Writing to a file (2d) | <code>save -ascii data.txt f</code> | <code>save('data.csv', f, fmt='%.6f', delimitwrier='te(f;'),file="data.txt")</code> | |
| Writing to a file (1d) | | <code>f.tofile(file='data.csv', format='%.6f', sep=',')</code> | |
| Reading from a file (1d) | | <code>f = fromfile(file='data.csv', sep=',')</code> | |

7 Plotting

7.1 Basic x-y plots

| Language | MATLAB/Octave | Python | R |
|--|--|---|---|
| 1d line plot | <code>plot(a)</code> | <code>plot(a)</code> | <code>plot(a, type="l")</code> |
| 2d scatter plot | <code>plot(x(:,1),x(:,2),'o')</code> | <code>plot(x[:,0],x[:,1],'o')</code> | <code>plot(x[,1],x[,2])</code> |
| Two graphs in one plot | <code>plot(x1,y1, x2,y2)</code> | <code>plot(x1,y1,'bo', x2,y2,'go')</code> | <code>plot(x1,y1)</code> |
| Overplotting: Add new plots to current | <code>plot(x1,y1)</code> <code>hold on</code> <code>plot(x2,y2)</code> | <code>plot(x1,y1,'o')</code> <code>plot(x2,y2,'o')</code> <code>show()</code> # as normal | <code>matplot(x2,y2,add=T)</code> |
| subplots | <code>subplot(211)</code> | <code>subplot(211)</code> | |
| Plotting symbols and color | <code>plot(x,y,'ro-')</code> | <code>plot(x,y,'ro-')</code> | <code>plot(x,y,type="b",col="red")</code> |



7.1.1 Axes and titles

Language
Turn on grid lines
1:1 aspect ratio

Set axes manually
Axis labels and titles

Insert text

MATLAB/Octave
`grid on`
`axis equal`
Octave:
`axis('equal')`
`replot`
`axis([0 10 0 5])`
`title('title')`
`xlabel('x-axis')`
`ylabel('y-axis')`

Python
`grid()`
`figure(figsize=(6,6))`

`axis([0, 10, 0, 5])`

`text(2,25,'hello')`

R
`grid()`
`plot(c(1:10,10:1), asp=1)`

`plot(x,y, xlim=c(0,10), ylim=c(0,5))`
`plot(1:10, main="title",`
`xlab="x-axis", ylab="y-axis")`

7.1.2 Log plots

Language
logarithmic y-axis
logarithmic x-axis
logarithmic x and y axes

MATLAB/Octave
`semilogy(a)`
`semilogx(a)`
`loglog(a)`

Python
`semilogy(a)`
`semilogx(a)`
`loglog(a)`

R
`plot(x,y, log="y")`
`plot(x,y, log="x")`
`plot(x,y, log="xy")`

7.1.3 Filled plots and bar plots

Language

MATLAB/Octave

Python

R

Filled plot

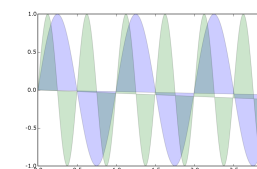
`fill(t,s,'b', t,c,'g')`
Octave: % fill has a bug?

`fill(t,s,'b', t,c,'g', alpha=0.2)`

`plot(t,s, type="n", xlab="", ylab="")`
`polygon(t,s, col="lightblue")`
`polygon(t,c, col="lightgreen")`

Stem-and-Leaf plot

`stem(x[,3])`



```
5 5
6 71
7 033
8 00113345567889
9 0133566677788
10 32674
```

7.1.4 Functions

Language
Defining functions

MATLAB/Octave
`f = inline('sin(x/3) - cos(x/5)')`

Python

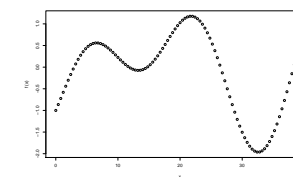
R
`f <- function(x) sin(x/3) - cos(x/5)` $f(x) = \sin\left(\frac{x}{3}\right) - \cos\left(\frac{x}{5}\right)$

Plot a function for given range

`ezplot(f,[0,40])`
`fplot('sin(x/3) - cos(x/5)',[0,40])`
Octave: % no ezplot

`x = arrayrange(0,40,.5)`
`y = sin(x/3) - cos(x/5)`
`plot(x,y, 'o')`

`plot(f, xlim=c(0,40), type='p')`



7.2 Polar plots

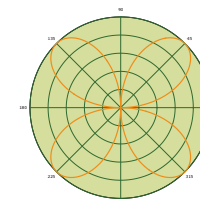
Language

MATLAB/Octave
`theta = 0:.001:2*pi;`
`r = sin(2*theta);`

Python
`theta = arange(0,2*pi,0.001)`
`r = sin(2*theta)`

R

$$\rho(\theta) = \sin(2\theta)$$



`polar(theta, rho)`

`polar(theta, rho)`

7.3 Histogram plots

Language

MATLAB/Octave
`hist(randn(1000,1))`
`hist(randn(1000,1), -4:4)`
`plot(sort(a))`

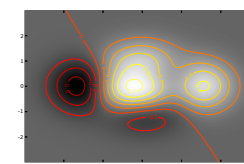
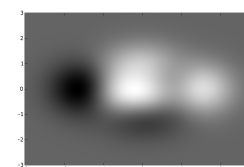
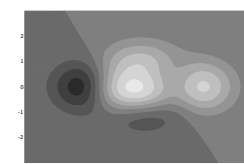
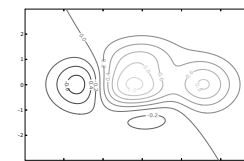
Python

R
`hist(rnorm(1000))`
`hist(rnorm(1000), breaks= -4:4)`
`hist(rnorm(1000), breaks=c(seq(-5,0,0.25), seq(0.5,5,0.5)), freq=F)`
`plot(apply(a,1,sort),type="l")`

7.4 3d data

7.4.1 Contour and image plots

| Language | MATLAB/Octave | Python | R |
|--|--|--|--|
| Contour plot | <code>contour(z)</code> | <pre>levels, colls = contour(Z, V, origin='lower', extent=(-3,3,-3,3)) clabel(colls, levels, inline=1, fmt='%1.1f', fontsize=10)</pre> | <code>contour(z)</code> |
| Filled contour plot | <code>contourf(z); colormap(gray)</code> | <pre>contourf(Z, V, cmap=cm.gray, origin='lower', extent=(-3,3,-3,3))</pre> | <pre>filled.contour(x,y,z, nlevels=7, color=gray.colors)</pre> |
| Plot image data | <pre>image(z) colormap(gray)</pre> | <pre>im = imshow(Z, interpolation='bilinear', origin='lower', extent=(-3,3,-3,3))</pre> | <pre>image(z, col=gray.colors(256))</pre> |
| Image with contours Direction field vectors | <code>quiver()</code> | <pre># imshow() and contour() as above quiver()</pre> | |



7.4.2 Perspective plots of surfaces over the x-y plane

Language

MATLAB/Octave

```
n=-2:.1:2;
[x,y] = meshgrid(n,n);
z=x.*exp(-x.^2-y.^2);
```

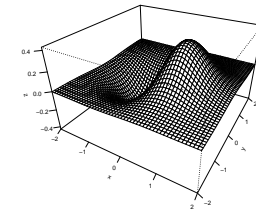
Python

```
n=arrayrange(-2,2,.1)
[x,y] = meshgrid(n,n)
z = x*power(math.e,-x**2-y**2)
```

R

```
f <- function(x,y) x*exp(-x^2-y^2)
n <- seq(-2,2, length=40)
z <- outer(n,n,f)
```

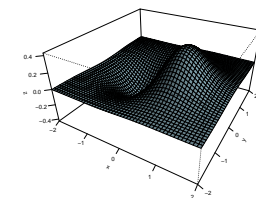
$$f(x,y) = xe^{-x^2-y^2}$$



Mesh plot

mesh(z)

```
persp(x,y,z,
      theta=30, phi=30, expand=0.6,
      ticktype='detailed')
```



Surface plot

surf(x,y,z) or surf1(x,y,z)
Octave: % no surf1()

```
persp(x,y,z,
      theta=30, phi=30, expand=0.6,
      col='lightblue', shade=0.75, ltheta=120,
      ticktype='detailed')
```

7.4.3 Scatter (cloud) plots

Language

MATLAB/Octave

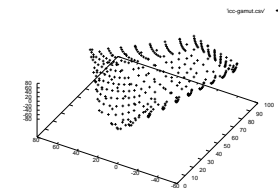
Python

R

3d scatter plot

plot3(x,y,z,'k+')

cloud(z~x*y)



7.5 Save plot to a graphics file

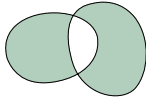
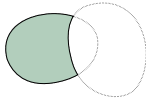
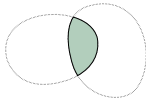
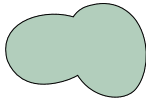
| | | | |
|-------------------------------|--|---------------------------------|--|
| Language | MATLAB/Octave | Python | R |
| PostScript | <code>plot(1:10)</code> <code>print -depsc2 foo.eps</code> Octave: <code>gset output "foo.eps"</code> <code>gset terminal postscript eps</code> <code>plot(1:10)</code> | <code>savefig('foo.eps')</code> | <code>postscript(file="foo.eps")</code> <code>plot(1:10)</code> <code>dev.off()</code> |
| PDF | | <code>savefig('foo.pdf')</code> | <code>pdf(file='foo.pdf')</code> |
| SVG (vector graphics for www) | | <code>savefig('foo.svg')</code> | <code>devSVG(file='foo.svg')</code> |
| PNG (raster graphics) | <code>print -dpng foo.png</code> | <code>savefig('foo.png')</code> | <code>png(filename = "Rplot%03d.png")</code> |

8 Data analysis

8.1 Set membership operators

| | | | |
|---------------------|--|--|--|
| Language | MATLAB/Octave | Python | R |
| Create sets | <code>a = [1 2 2 5 2];</code> <code>b = [2 3 4];</code> | <code>a = array([1,2,2,5,2])</code> <code>b = array([2,3,4])</code> <code>a = set([1,2,2,5,2])</code> <code>b = set([2,3,4])</code> | <code>a <- c(1,2,2,5,2)</code> <code>b <- c(2,3,4)</code> |
| Set unique | <code>unique(a)</code> | <code>uniqueid(a)</code> <code>unique(a)</code> <code>set(a)</code> | <code>unique(a)</code> |
| Set union | <code>union(a,b)</code> | <code>union1d(a,b)</code> <code>a.union(b)</code> | <code>union(a,b)</code> |
| Set intersection | <code>intersect(a,b)</code> | <code>intersect1d(a)</code> <code>a.intersection(b)</code> | <code>intersect(a,b)</code> |
| Set difference | <code>setdiff(a,b)</code> | <code>setdiff1d(a,b)</code> <code>a.difference(b)</code> | <code>setdiff(a,b)</code> |
| Set exclusion | <code>setxor(a,b)</code> | <code>setxor1d(a,b)</code> <code>a.symmetric_difference(b)</code> | <code>setdiff(union(a,b),intersect(a,b))</code> |
| True for set member | <code>ismember(2,a)</code> | <code>2 in a</code> <code>setmember1d(2,a)</code> <code>contains(a,2)</code> | <code>is.element(2,a)</code> or <code>2 %in% a</code> |

[1 2 5]



8.2 Statistics

| | | | |
|-------------------------|------------------------|---|--------------------------------|
| Language | MATLAB/Octave | Python | R |
| Average | <code>mean(a)</code> | <code>a.mean(axis=0)</code> <code>mean(a [,axis=0])</code> | <code>apply(a,2,mean)</code> |
| Median | <code>median(a)</code> | <code>median(a)</code> or <code>median(a [,axis=0])</code> | <code>apply(a,2,median)</code> |
| Standard deviation | <code>std(a)</code> | <code>a.std(axis=0)</code> or <code>std(a [,axis=0])</code> | <code>apply(a,2,sd)</code> |
| Variance | <code>var(a)</code> | <code>a.var(axis=0)</code> or <code>var(a)</code> | <code>apply(a,2,var)</code> |
| Correlation coefficient | <code>corr(x,y)</code> | <code>correlate(x,y)</code> or <code>corrcoef(x,y)</code> | <code>cor(x,y)</code> |
| Covariance | <code>cov(x,y)</code> | <code>cov(x,y)</code> | <code>cov(x,y)</code> |

8.3 Interpolation and regression

| | | | |
|-----------------------------------|--|--|---|
| Language | MATLAB/Octave | Python | R |
| Straight line fit | <code>z = polyval(polyfit(x,y,1),x)</code> <code>plot(x,y,'o', x,z ,'-')</code> | <code>(a,b) = polyfit(x,y,1)</code> <code>plot(x,y,'o', x,a*x+b,'-')</code> | <code>z <- lm(y~x)</code> <code>plot(x,y)</code> <code>abline(z)</code> <code>solve(a,b)</code> |
| Linear least squares $y = ax + b$ | <code>a = x\y</code> | <code>linalg.lstsq(x,y)</code> | |
| Polynomial fit | <code>polyfit(x,y,3)</code> | <code>polyfit(x,y,3)</code> | |

8.4 Non-linear methods

8.4.1 Polynomials, root finding

| | | | |
|--------------------------|---|---|---|
| Language | MATLAB/Octave | Python | R |
| Polynomial | | <code>poly()</code> | |
| Find zeros of polynomial | <code>roots([1 -1 -1])</code> | <code>roots()</code> | <code>polyroot(c(1,-1,-1))</code> |
| Find a zero near $x = 1$ | <code>f = inline('1/x - (x-1)')</code> <code>fzero(f,1)</code> | | $x^2 - x - 1 = 0$ $f(x) = \frac{1}{x} - (x - 1)$ |
| Solve symbolic equations | <code>solve('1/x = x-1')</code> | | $\frac{1}{x} = x - 1$ |
| Evaluate polynomial | <code>polyval([1 2 1 2],1:10)</code> | <code>polyval(array([1,2,1,2]),arange(1,11))</code> | |

8.4.2 Differential equations

| | | | |
|---|----------------------|-----------------------------------|---|
| Language | MATLAB/Octave | Python | R |
| Discrete difference function and approximate derivative | <code>diff(a)</code> | <code>diff(x, n=1, axis=0)</code> | |
| Solve differential equations | | | |

8.5 Fourier analysis

| | | | |
|---------------------------|----------------------|--|-----------------------------------|
| Language | MATLAB/Octave | Python | R |
| Fast fourier transform | <code>fft(a)</code> | <code>fft(a)</code> or <code>ifft(a)</code> | <code>fft(a)</code> |
| Inverse fourier transform | <code>ifft(a)</code> | <code>ifft(a)</code> or <code>convolve(x,y)</code> | <code>fft(a, inverse=TRUE)</code> |
| Linear convolution | | | |

9 Symbolic algebra; calculus

| | | | |
|---------------|-----------------------|--------|---|
| Language | MATLAB/Octave | Python | R |
| Factorization | <code>factor()</code> | | |

10 Programming

| | | | |
|-------------------------------|--|--------------------------------|--|
| Language | MATLAB/Octave | Python | R |
| Script file extension | .m | .py | .R |
| Comment symbol (rest of line) | % | # | # |
| Import library functions | Octave: % or # % must be in MATLABPATH Octave: % must be in LOADPATH | from pylab import * | library(RSvgDevice) |
| Eval | string='a=234'; eval(string) | string="a=234" eval(string) | string <- "a <- 234" eval(parse(text=string)) |

10.1 Loops

| | | | |
|--------------------------|--|--|--|
| Language | MATLAB/Octave | Python | R |
| for-statement | for i=1:5; disp(i); end | for i in range(1,6): print(i) | for(i in 1:5) print(i) |
| Multiline for statements | for i=1:5 disp(i) disp(i*2) end | for i in range(1,6): print(i) print(i*2) | for(i in 1:5) { print(i) print(i*2) } |

10.2 Conditionals

| | | | |
|----------------------------------|-----------------------------|---------------|--|
| Language | MATLAB/Octave | Python | R |
| if-statement | if 1>0 a=100; end | if 1>0: a=100 | if (1>0) a <- 100 |
| if-else-statement | if 1>0 a=100; else a=0; end | | |
| Ternary operator (if?true:false) | | | ifelse(a>0,a,0) $a > 0 ? a : 0$ |

10.3 Debugging

| | | | |
|-------------------------------------|------------------------|---------|-------------|
| Language | MATLAB/Octave | Python | R |
| Most recent evaluated expression | ans | | .Last.value |
| List variables loaded into memory | whos or who | | objects() |
| Clear variable <i>x</i> from memory | clear x or clear [all] | | rm(x) |
| Print | disp(a) | print a | print(a) |

10.4 Working directory and OS

| | | | |
|--|---------------------------------------|----------------------|----------------------------|
| Language | MATLAB/Octave | Python | R |
| List files in directory | dir or ls | os.listdir(".") | list.files() or dir() |
| List script files in directory | what | grep.grep("*.py") | list.files(pattern=".r\$") |
| Displays the current working directory | pwd | os.getcwd() | getwd() |
| Change working directory | cd foo | os.chdir('foo') | setwd('foo') |
| Invoke a System Command | !notepad Octave: system("notepad") | os.system('notepad') | system("notepad") |
| | | os.popen('notepad') | |

²This document is still draft quality. Most shown 2d plots are made using Matplotlib, and 3d plots using R and Gnuplot, provided as examples only.

³Version numbers and download URL for software used: Python 2.4.2, <http://www.python.org/>; NumPy 0.9.5, <http://numeric.scipy.org/>; Matplotlib 0.87, <http://matplotlib.sf.net/>; IPython 0.7.1, <http://ipython.scipy.org/>; R 2.1.1, <http://www.r-project.org/>; Octave 2.1.50, <http://www.octave.org/>; Scilab 4.0, <http://www.scilab.org/>; Gnuplot 4.0, <http://www.gnuplot.info/>.

⁴For referencing: Gundersen, Vidar Bronken. *MATLAB commands in numerical Python* (Oslo/Norway, 2005), available from: <http://mathesaurus.sf.net/>

⁵Contributions are appreciated: The best way to do this is to edit the XML and submit patches to our tracker or forums.