

# MATLAB commands in numerical Python

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.

Contributor: Gary Ruben

The idea of this document (and the corresponding XML instance) is to provide a quick reference<sup>1</sup> for switching to open-source mathematical computation environments for computer algebra, numeric processing and data visualisation. Examples of well known systems are MATLAB, IDL, R, SPlus, with their open-source counterparts Octave, Scilab, FreeMat, Python (NumPy and matplotlib modules), and Gnuplot. Or CAS tools like Mathematica, Maple, MuPAD, with Axiom and Maxima as open alternatives.

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

	Browse help interactively
MATLAB	<code>doc</code>
Octave	<code>help -i % browse with Info</code>
Scilab	<code>help</code>
R	<code>help.start()</code>
Python	<code>help()</code>
gnuplot	<code>help</code> or <code>?</code>
IDL	<code>?</code>
Axiom	<code>)hd</code>
	Help on using help
MATLAB	<code>help help</code> or <code>doc doc</code>
R	<code>help()</code>
Python	<code>help</code>
IDL	<code>?help</code>
Axiom	<code>)help ?</code>
	Help for a function
MATLAB	<code>help plot</code>
R	<code>help(plot)</code> or <code>?plot</code>
Python	<code>help(plot)</code> or <code>?plot</code>
gnuplot	<code>help plot</code> or <code>?plot</code>
IDL	<code>?plot</code> or <code>man, 'plot</code>
Maxima	<code>describe(keyword)\$</code>
Maple	<code>?keyword</code>
Mathematica	<code>?keyword</code>
MuPAD	<code>?keyword</code>
	Help for a toolbox/library package
MATLAB	<code>help splines</code> or <code>doc splines</code>
R	<code>help(package='splines')</code>
Python	<code>help(pylab)</code>
	Demonstration examples
MATLAB	<code>demo</code>
Scilab	<code>demoplay();</code>
R	<code>demo()</code>
IDL	<code>demo</code>
	Example using a function
R	<code>example(plot)</code>
Maxima	<code>example(factor);</code>

### 1.1 Searching available documentation

	Search help files
MATLAB	<code>lookfor plot</code>
R	<code>help.search('plot')</code>
	Find objects by partial name
R	<code>apropos('plot')</code>
Scilab	<code>apropos plot</code>
Axiom	<code>)what operations pattern</code>
Maxima	<code>describe(pattern)\$</code>
Maple	<code>?keyword</code>
Mathematica	<code>?*pattern*</code>
MuPAD	<code>?*pattern*</code>

<sup>1</sup>References: 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); Greenfield, Jędrzejewski & Laidler. *Using Python for Interactive Data Analysis* (2005), pp.125–134, available from <http://stdas.stsci.edu/perry/pydatatut.pdf> (accessed 2005.07.29); Brisson, Eric. *Using IDL to Manipulate and Visualize Scientific Data*, available from <http://scv.bu.edu/Tutorials/IDL/> (accessed 2005.07.31); Wester, Michael (ed). *Computer Algebra Systems: A Practical Guide* (1999), available from [http://www.math.unm.edu/~wester/cas\\_review.html](http://www.math.unm.edu/~wester/cas_review.html) (accessed 2005.08.14).

List available packages	
MATLAB	<code>help</code>
R	<code>library()</code>
Python	<code>help(); modules [Numeric]</code>
Locate functions	
MATLAB	<code>which plot</code>
Scilab	<code>whereis plot</code>
R	<code>find(plot)</code>
Python	<code>help(plot)</code>
List available methods for a function	
R	<code>methods(plot)</code>

## 1.2 Using interactively

Start session	
Octave	<code>octave -q</code>
R	<code>Rgui</code>
Python	<code>ipython -pylab</code>
IDL	<code>idlde</code>
bc	<code>bc -lq</code>
gnuplot	<code>pgnuplot</code>
Auto completion	
Octave	<code>TAB</code> or <code>M-?</code>
Scilab	<code>! //</code> commands in history
Python	<code>TAB</code>
Run code from file	
MATLAB	<code>foo(.m)</code>
Scilab	<code>exec('foo.sce')</code>
R	<code>source('foo.R')</code>
Python	<code>execfile('foo.py')</code> or <code>run foo.py</code>
gnuplot	<code>load 'foo.gp'</code>
IDL	<code>@'foo.idlbatch" or .run 'foo.pro'</code>
Maxima	<code>batch("foo.mc")</code>
Command history	
Octave	<code>history</code>
Scilab	<code>gethistory</code>
R	<code>history()</code>
Python	<code>hist -n</code>
IDL	<code>help,/rec</code>
Axiom	<code>)history )show</code>
Save command history	
MATLAB	<code>diary on [..] diary off</code>
Scilab	<code>diary('session.txt') [..] diary(0)</code>
R	<code>savehistory(file=".Rhistory")</code>
IDL	<code>journal,'IDLhistory'</code>
Axiom	<code>)hist )write foo.input</code>
End session	
MATLAB	<code>exit</code> or <code>quit</code>
R	<code>q(save='no')</code>
Python	<code>CTRL-D</code>
	<code>CTRL-Z # windows</code>
	<code>sys.exit()</code>
gnuplot	<code>exit</code> or <code>quit</code>
IDL	<code>exit</code> or <code>CTRL-D</code>
Axiom	<code>)quit</code>
Maxima	<code>quit();</code>
Maple	<code>quit</code>
Mathematica	<code>Quit[]</code>
MuPAD	<code>quit</code>
Derive	<code>[Quit]</code>
REDUCE	<code>quit;</code>
bc	<code>quit</code>

## 2 Operators

Help on operator syntax	
MATLAB	<code>help -</code>
Scilab	<code>help symbols</code>
R	<code>help(Syntax)</code>

### 2.1 Arithmetic operators

Assignment; defining a number	
MATLAB	<code>a=1; b=2;</code>
R	<code>a&lt;-1; b&lt;-2</code>
Python	<code>a=1; b=1</code>
IDL	<code>a=1 &amp; b=1</code>
bc	<code>a=1; b=1</code>

Addition	
<b>Generic</b>	$a + b$
MATLAB	$a + b$
R	$a + b$
Python	$a + b$ or <code>add(a,b)</code>
gnuplot	$a + b$
IDL	$a + b$
bc	$a + b$
Subtraction	
<b>Generic</b>	$a - b$
MATLAB	$a - b$
R	$a - b$
Python	$a - b$ or <code>subtract(a,b)</code>
gnuplot	$a - b$
IDL	$a - b$
bc	$a - b$
Multiplication	
<b>Generic</b>	$a * b$
MATLAB	$a * b$
R	$a * b$
Python	$a * b$ or <code>multiply(a,b)</code>
gnuplot	$a * b$
IDL	$a * b$
bc	$a * b$
Division	
<b>Generic</b>	$a / b$
MATLAB	$a / b$
R	$a / b$
Python	$a / b$ or <code>divide(a,b)</code>
gnuplot	$a / b$
IDL	$a / b$
Power, $a^b$	
MATLAB	$a .^ b$
R	$a ^ b$
Python	$a ** b$ <code>power(a,b)</code> <code>pow(a,b)</code>
gnuplot	$a ** b$
IDL	$a ^ b$
Axiom	$a**b$
Maxima	$a^b$ or $a**b$
bc	$a ^ b$
Remainder	
MATLAB	<code>rem(a,b)</code>
Scilab	<code>modulo(a,b)</code>
R	$a \% b$
Python	$a \% b$ <code>remainder(a,b)</code> <code>fmod(a,b)</code>
gnuplot	$a \% b$
IDL	$a \text{ MOD } b$
Axiom	<code>rem(a,b)</code>
Maxima	<code>mod(a,b)</code>
Maple	$a \text{ mod } b$
Mathematica	<code>Mod[a,b]</code>
MuPAD	$a \text{ mod } b$
Derive	<code>MOD(a,b)</code>
bc	$a \% b$
Integer division	
R	$a \%\% b$
bc	$a / b$
Increment, return new value	
Octave	<code>++a</code>
IDL	<code>++a</code> or <code>a+=1</code>
bc	<code>++a</code>
Increment, return old value	
Octave	<code>a++</code>
IDL	<code>a++</code>
bc	<code>a++</code>
In place operation to save array creation overhead	
Python	<code>a+=b</code> or <code>add(a,b,a)</code>
Octave	<code>a+=1</code>
IDL	<code>a+=1</code>
bc	<code>a+=b</code>
Factorial, $n!$	
MATLAB	<code>factorial(a)</code>
R	<code>factorial(a)</code>
Axiom	<code>factorial(a)</code>
Maxima	$a!$
Maple	$a!$

## 2.2 Relational operators

Equal	
<b>Generic</b>	<code>a == b</code>
MATLAB	<code>a == b</code>
R	<code>a == b</code>
Python	<code>a == b</code> or <code>equal(a,b)</code>
IDL	<code>a eq b</code>
bc	<code>a == b</code>
Less than	
<b>Generic</b>	<code>a &lt; b</code>
MATLAB	<code>a &lt; b</code>
R	<code>a &lt; b</code>
Python	<code>a &lt; b</code> or <code>less(a,b)</code>
IDL	<code>a lt b</code>
bc	<code>a &lt; b</code>
Greater than	
<b>Generic</b>	<code>a &gt; b</code>
MATLAB	<code>a &gt; b</code>
R	<code>a &gt; b</code>
Python	<code>a &gt; b</code> or <code>greater(a,b)</code>
IDL	<code>a gt b</code>
bc	<code>a &gt; b</code>
Less than or equal	
<b>Generic</b>	<code>a &lt;= b</code>
MATLAB	<code>a &lt;= b</code>
R	<code>a &lt;= b</code>
Python	<code>a &lt;= b</code> or <code>less_equal(a,b)</code>
IDL	<code>a le b</code>
bc	<code>a &lt;= b</code>
Greater than or equal	
<b>Generic</b>	<code>a &gt;= b</code>
MATLAB	<code>a &gt;= b</code>
R	<code>a &gt;= b</code>
Python	<code>a &gt;= b</code> or <code>greater_equal(a,b)</code>
IDL	<code>a ge b</code>
bc	<code>a &gt;= b</code>
Not Equal	
MATLAB	<code>a ~= b</code>
Scilab	<code>a ~= b</code> or <code>a &lt;&gt; b</code>
R	<code>a != b</code>
Python	<code>a != b</code> or <code>not_equal(a,b)</code>
IDL	<code>a ne b</code>
bc	<code>a != b</code>

## 2.3 Logical operators

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

	True if all elements are nonzero
MATLAB	<code>all(a)</code>
Scilab	<code>and(a)</code>

## 2.4 root and logarithm

	Square root	
<b>Generic</b>	<code>sqrt(a)</code>	
MATLAB	<code>sqrt(a)</code>	
R	<code>sqrt(a)</code>	
Python	<code>math.sqrt(a)</code>	
gnuplot	<code>sqrt(a)</code>	$\sqrt{a}$
IDL	<code>sqrt(a)</code>	
Axiom	<code>sqrt(a)</code>	
Maxima	<code>sqrt(a)</code>	
Maple	<code>sqrt(a)</code>	
Mathematica	<code>Sqrt[a]</code>	
bc	<code>sqrt(a)</code>	
	Logarithm, base $e$ (natural)	
<b>Generic</b>	<code>log(a)</code>	
MATLAB	<code>log(a)</code>	
R	<code>log(a)</code>	
Python	<code>math.log(a)</code>	
gnuplot	<code>log(a)</code>	
IDL	<code>alog(a)</code>	$\ln a = \log_e a$
Axiom	<code>log(a)</code>	
Maxima	<code>log(a)</code>	
Maple	<code>log(a)</code>	
Mathematica	<code>Log[a]</code>	
MuPAD	<code>ln(a)</code>	
bc	<code>l(a)</code>	
	Logarithm, base 10	
<b>Generic</b>	<code>log10(a)</code>	
MATLAB	<code>log10(a)</code>	
R	<code>log10(a)</code>	$\log_{10} a$
Python	<code>math.log10(a)</code>	
gnuplot	<code>log10(a)</code>	
IDL	<code>alog10(a)</code>	
	Logarithm, base 2 (binary)	
MATLAB	<code>log2(a)</code>	$\log_2 a$
R	<code>log2(a)</code>	
Python	<code>math.log(a, 2)</code>	
	Exponential function	
<b>Generic</b>	<code>exp(a)</code>	
MATLAB	<code>exp(a)</code>	
R	<code>exp(a)</code>	$e^a$
Python	<code>math.exp(a)</code>	
gnuplot	<code>exp(a)</code>	
IDL	<code>exp(a)</code>	
bc	<code>e(a)</code>	

## 2.5 Round off

	Round
MATLAB	<code>round(a)</code>
R	<code>round(a)</code>
Python	<code>round(a)</code> or <code>math.round(a)</code>
IDL	<code>round(a)</code>
	Round up
MATLAB	<code>ceil(a)</code>
R	<code>ceil(a)</code>
Python	<code>ceil(a)</code>
gnuplot	<code>ceil(a)</code>
IDL	<code>ceil(a)</code>
	Round down
MATLAB	<code>floor(a)</code>
R	<code>floor(a)</code>
Python	<code>floor(a)</code>
gnuplot	<code>floor(a)</code>
IDL	<code>floor(a)</code>
	Round towards zero
MATLAB	<code>fix(a)</code>
Python	<code>fix(a)</code>

## 2.6 Mathematical constants

	$\pi = 3.141592$
MATLAB	pi
Scilab	%pi
R	pi
Python	math.pi
IDL	!pi
Axiom	%pi
Maxima	%pi
Maple	Pi
Mathematica	Pi
MuPAD	PI
	$e = 2.718281$
MATLAB	exp(1)
Scilab	%e
R	exp(1)
Python	math.e or math.exp(1)
gnuplot	exp(1)
IDL	exp(1)
Axiom	%e
Maxima	%e
Maple	exp(1)
Mathematica	E
MuPAD	E

### 2.6.1 Missing values; IEEE-754 floating point status flags

	Not a Number
MATLAB	NaN
Python	nan
Scilab	%nan
	Infinity, $\infty$
MATLAB	Inf
Scilab	%inf
Python	inf
	Infinity, $+\infty$
Python	plus_inf
Axiom	%plusInfinity
	Infinity, $-\infty$
Python	minus_inf
Axiom	%minusInfinity
	Plus zero, $+0$
Python	plus_zero
	Minus zero, $-0$
Python	minus_zero

## 2.7 Complex numbers

	Imaginary unit	
MATLAB	i	
Scilab	%i	
R	1i	
Python	z = 1j	$i = \sqrt{-1}$
gnuplot	{0,1}	
IDL	complex(0,1)	
Axiom	%i	
Maxima	%i	
	A complex number, $3 + 4i$	
MATLAB	z = 3+4i	
Scilab	z = 3+4*i	
R	z <- 3+4i	
Python	z = 3+4j or z = complex(3,4)	
gnuplot	{3,4}	
IDL	z = complex(3,4)	
Axiom	3+4*i	
Maxima	3+4*i	
	Absolute value (modulus)	
<b>Generic</b>	abs(z)	
MATLAB	abs(z)	
R	abs(3+4i) or Mod(3+4i)	
Python	abs(3+4j)	
gnuplot	abs({3,4})	
IDL	abs(z)	
Maxima	abs(z);	
	Real part	
<b>Generic</b>	real(z)	
MATLAB	real(z)	
R	Re(3+4i)	
Python	z.real	
gnuplot	real({3,4})	
IDL	real_part(z)	
Maxima	realpart(z)	

	Imaginary part
<b>Generic</b>	<code>imag(z)</code>
MATLAB	<code>imag(z)</code>
R	<code>Im(3+4i)</code>
Python	<code>z.imag</code>
IDL	<code>imaginary(z)</code>
gnuplot	<code>imag({3,4})</code>
Maxima	<code>imagpart(z)</code>
	Argument
MATLAB	<code>arg(z)</code>
R	<code>Arg(3+4i)</code>
gnuplot	<code>arg({3,4})</code>
	Complex conjugate
<b>Generic</b>	<code>conj(z)</code>
MATLAB	<code>conj(z)</code>
R	<code>Conj(3+4i)</code>
Python	<code>z.conj(); z.conjugate()</code>
IDL	<code>conj(z)</code>

## 2.8 Trigonometry

	Sine
<b>Generic</b>	<code>sin(a)</code>
	Cosine
<b>Generic</b>	<code>cos(a)</code>
	Tangent
<b>Generic</b>	<code>tan(a)</code>
	Arcsine
<b>Generic</b>	<code>asin(a)</code> or <code>arcsin(a)</code>
	Arccosine
<b>Generic</b>	<code>acos(a)</code> or <code>arccos(a)</code>
	Arctangent
<b>Generic</b>	<code>atan(a)</code> or <code>arctan(a)</code>
	Arctangent, <code>arctan(b/a)</code>
MATLAB	<code>atan(a,b)</code>
R	<code>atan2(b,a)</code>
Python	<code>atan2(b,a)</code>
	Hyperbolic sine
<b>Generic</b>	<code>sinh(a)</code>
	Hyperbolic cosine
<b>Generic</b>	<code>cosh(a)</code>
	Hyperbolic tangent
<b>Generic</b>	<code>tanh(a)</code>
	Hypotenuse; Euclidean distance
Python	<code>hypot(x,y)</code> $\sqrt{x^2 + y^2}$

## 2.9 Generate random numbers

	Uniform distribution
MATLAB	<code>rand(1,10)</code>
Scilab	<code>rand(1,10,'uniform')</code>
R	<code>runif(10)</code>
Python	<code>random.random((10,))</code> <code>random.uniform((10,))</code>
IDL	<code>randomu(seed, 10)</code>
	Uniform: Numbers between 2 and 7
MATLAB	<code>2+5*rand(1,10)</code>
Scilab	<code>2+5*rand(1,10,'uniform')</code>
R	<code>runif(10, min=2, max=7)</code>
Python	<code>random.uniform(2,7,(10,))</code>
IDL	<code>2+5*randomu(seed, 10)</code>
	Uniform: 6,6 array
MATLAB	<code>rand(6)</code>
Scilab	<code>rand(6,6,'uniform')</code>
R	<code>matrix(runif(36),6)</code>
Python	<code>random.uniform(0,1,(6,6))</code>
IDL	<code>randomu(seed, [6,6])</code>
	Normal distribution
MATLAB	<code>randn(1,10)</code>
Scilab	<code>rand(1,10,'normal')</code>
R	<code>rnorm(10)</code>
Python	<code>random.standard_normal((10,))</code>
IDL	<code>randomn(seed, 10)</code>

## 3 Vectors

	Row vector, $1 \times n$ -matrix
MATLAB	<code>a=[2 3 4 5];</code>
R	<code>a &lt;- c(2,3,4,5)</code>
Python	<code>a=array([2,3,4,5])</code>
IDL	<code>a = [2, 3, 4, 5]</code>
	Column vector, $m \times 1$ -matrix
MATLAB	<code>adash=[2 3 4 5]';</code>
R	<code>adash &lt;- t(c(2,3,4,5))</code>
Python	<code>array([2,3,4,5])[:,NewAxis]</code> <code>array([2,3,4,5]).reshape(-1,1)</code> <code>r_[1:10,'c']</code>
IDL	<code>transpose([2,3,4,5])</code>

### 3.1 Sequences

	<code>1,2,3, ... ,10</code>
MATLAB	<code>1:10</code>
R	<code>seq(10) or 1:10</code>
Python	<code>arange(1,11, dtype=Float)</code> <code>range(1,11)</code>
IDL	<code>indgen(10)+1</code> <code>dindgen(10)+1</code>
	<code>0.0,1.0,2.0, ... ,9.0</code>
MATLAB	<code>0:9</code>
R	<code>seq(0,length=10)</code>
Python	<code>arange(10.)</code>
IDL	<code>dindgen(10)</code>
	<code>1,4,7,10</code>
MATLAB	<code>1:3:10</code>
R	<code>seq(1,10,by=3)</code>
Python	<code>arange(1,11,3)</code>
IDL	<code>indgen(4)*3+1</code>
	<code>10,9,8, ... ,1</code>
MATLAB	<code>10:-1:1</code>
R	<code>seq(10,1) or 10:1</code>
Python	<code>arange(10,0,-1)</code>
	<code>10,7,4,1</code>
MATLAB	<code>10:-3:1</code>
R	<code>seq(from=10,to=1,by=-3)</code>
Python	<code>arange(10,0,-3)</code>
	Linearly spaced vector of $n=7$ points
MATLAB	<code>linspace(1,10,7)</code>
R	<code>seq(1,10,length=7)</code>
Python	<code>linspace(1,10,7)</code>
	Reverse
MATLAB	<code>reverse(a)</code>
Scilab	<code>a(\$:-1:1)</code>
R	<code>rev(a)</code>
Python	<code>a[::-1] or</code>
IDL	<code>reverse(a)</code>
	Set all values to same scalar value
MATLAB	<code>a(:) = 3</code>
Python	<code>a.fill(3), a[:] = 3</code>

### 3.2 Concatenation (vectors)

	Concatenate two vectors
MATLAB	<code>[a a]</code>
R	<code>c(a,a)</code>
Python	<code>concatenate((a,a))</code>
IDL	<code>[a,a] or rebin(a,2,size(a))</code>
MATLAB	<code>[1:4 a]</code>
R	<code>c(1:4,a)</code>
Python	<code>concatenate((range(1,5),a), axis=1)</code>
IDL	<code>[indgen(3)+1,a]</code>

### 3.3 Repeating

	<code>1 2 3, 1 2 3</code>
MATLAB	<code>[a a]</code>
R	<code>rep(a,times=2)</code>
Python	<code>concatenate((a,a))</code>
	<code>1 1 1, 2 2 2, 3 3 3</code>
R	<code>rep(a,each=3)</code>
Python	<code>a.repeat(3) or</code>



	1, 2 2, 3 3 3
R	<code>rep(a,a)</code>
Python	<code>a.repeat(a)</code> or

### 3.4 Miss those elements out

	miss the first element
MATLAB	<code>a(2:end)</code>
Scilab	<code>a(2:\$)</code>
R	<code>a[-1]</code>
Python	<code>a[1:]</code>
	miss the tenth element
MATLAB	<code>a([1:9])</code>
R	<code>a[-10]</code>
	miss 1,4,7, ...
R	<code>a[-seq(1,50,3)]</code>
	last element
MATLAB	<code>a(end)</code>
Scilab	<code>a(\$)</code>
Python	<code>a[-1]</code>
	last two elements
MATLAB	<code>a(end-1:end)</code>
Python	<code>a[-2:]</code>

### 3.5 Maximum and minimum

	pairwise max
MATLAB	<code>max(a,b)</code>
Python	<code>maximum(a,b)</code>
R	<code>pmax(a,b)</code>
	max of all values in two vectors
MATLAB	<code>max([a b])</code>
Python	<code>concatenate((a,b)).max()</code>
R	<code>max(a,b)</code>
MATLAB	<code>[v,i] = max(a)</code>
Python	<code>v,i = a.max(0),a.argmax(0)</code>
R	<code>v &lt;- max(a) ; i &lt;- which.max(a)</code>

### 3.6 Vector multiplication

	Multiply two vectors
MATLAB	<code>a.*a</code>
R	<code>a*a</code>
Python	<code>a*a</code>
	Vector cross product, $u \times v$
IDL	<code>crossp(u,v)</code>
Mathematica	<code>cross(u,v)</code>
	Vector dot product, $u \cdot v$
MATLAB	<code>dot(u,v)</code>
Python	<code>dot(u,v)</code>

## 4 Matrices

	Define a matrix
MATLAB	<code>a = [2 3;4 5]</code>
R	<code>rbind(c(2,3),c(4,5))</code>
	<code>array(c(2,3,4,5), dim=c(2,2))</code>
Python	<code>a = array([[2,3],[4,5]])</code>
IDL	<code>a = [[2,3],[4,5]]</code>
Axiom	<code>a := matrix [[2,3],[4,5]]</code>
Maxima	<code>matrix([2,3],[4,5])</code>
Maple	<code>matrix([2,3],[4,5])</code>
Mathematica	<code>{{2,3},{4,5}}</code>
Derive	<code>[[2,3],[4,5]]</code>

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

#### 4.1 Concatenation (matrices); rbind and cbind

	Bind rows
MATLAB	<code>[a ; b]</code>
R	<code>rbind(a,b)</code>
Python	<code>concatenate((a,b), axis=0)</code> <code>vstack((a,b))</code>
	Bind columns
MATLAB	<code>[a , b]</code>
R	<code>cbind(a,b)</code>
Python	<code>concatenate((a,b), axis=1)</code> <code>hstack((a,b))</code>

Bind slices (three-way arrays)	
Python	<code>concatenate((a,b), axis=2)</code> <code>dstack((a,b))</code>
Concatenate matrices into one vector	
MATLAB	<code>[a(:), b(:)]</code>
Python	<code>concatenate((a,b), axis=None)</code>
Bind rows (from vectors)	
MATLAB	<code>[1:4 ; 1:4]</code>
R	<code>rbind(1:4,1:4)</code>
Python	<code>concatenate((r_[1:5],r_[1:5])).reshape(2,-1)</code> <code>vstack((r_[1:5],r_[1:5]))</code>
	<code>[,1] [,2] [,3] [,4]</code>
	<code>[1,] 1 2 3 4</code>
	<code>[2,] 1 2 3 4</code>
Bind columns (from vectors)	
MATLAB	<code>[1:4 ; 1:4]'</code>
R	<code>cbind(1:4,1:4)</code>
	<code>[,1] [,2]</code>
	<code>[1,] 1 1</code>
	<code>[2,] 2 2</code>
	<code>[3,] 3 3</code>
	<code>[4,] 4 4</code>

## 4.2 Array creation

0 filled array	
MATLAB	<code>zeros(3,5)</code>
R	<code>matrix(0,3,5)</code> or <code>array(0,c(3,5))</code>
Python	<code>zeros((3,5),Float)</code>
IDL	<code>dblarr(3,5)</code>
	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
0 filled array of integers	
Python	<code>zeros((3,5))</code>
IDL	<code>intarr(3,5)</code>
1 filled array	
MATLAB	<code>ones(3,5)</code>
R	<code>matrix(1,3,5)</code> or <code>array(1,c(3,5))</code>
Python	<code>ones((3,5),Float)</code>
IDL	<code>dblarr(3,5)+1</code>
	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
Any number filled array	
MATLAB	<code>ones(3,5)*9</code>
R	<code>matrix(9,3,5)</code> or <code>array(9,c(3,5))</code>
Python	
IDL	<code>intarr(3,5)+9</code>
	$\begin{bmatrix} 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \\ 9 & 9 & 9 & 9 & 9 \end{bmatrix}$
Identity matrix	
MATLAB	<code>eye(3)</code>
Scilab	<code>eye(3,3)</code>
R	<code>diag(1,3)</code>
Python	<code>identity(3)</code>
IDL	<code>identity(3)</code>
	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Diagonal	
MATLAB	<code>diag([4 5 6])</code>
R	<code>diag(c(4,5,6))</code>
Python	<code>diag((4,5,6))</code>
IDL	<code>diag_matrix([4,5,6])</code>
Axiom	<code>diagonalMatrix([4,5,6])</code>
	$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 6 \end{bmatrix}$
Magic squares; Lo Shu	
MATLAB	<code>magic(3)</code>
Scilab	<code>testmatrix('magi',3)</code>
	$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$
Empty array	
Python	<code>a = empty((3,3))</code>

## 4.3 Reshape and flatten matrices

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

Flatten to vector (by columns)	
MATLAB	<code>a(:)</code>
R	<code>as.vector(a)</code>
Python	<code>a.flatten(1)</code>
$\left[ \begin{array}{cccccc} 1 & 4 & 2 & 5 & 3 & 6 \end{array} \right]$	
Flatten upper triangle (by columns)	
MATLAB	<code>vech(a)</code>
R	<code>a[row(a) &lt;= col(a)]</code>

## 4.4 Shared data (slicing)

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

## 4.5 Indexing and accessing elements (Python: slicing)

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

## 4.6 Assignment

MATLAB	<code>a(:,1) = 99</code>
R	<code>a[,1] &lt;- 99</code>
Python	<code>a[:,0] = 99</code>

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

## 4.7 Transpose and inverse

Transpose	
MATLAB	<code>a'</code>
R	<code>t(a)</code>
Python	<code>a.conj().transpose()</code>
IDL	<code>transpose(a)</code>
Maxima	<code>transpose(a);</code>
Non-conjugate transpose	
MATLAB	<code>a.' or transpose(a)</code>
Python	<code>a.transpose()</code>
Determinant	
MATLAB	<code>det(a)</code>
R	<code>det(a)</code>
Python	<code>linalg.det(a) or determinant(a)</code>
IDL	<code>determ(a)</code>
Axiom	<code>determinant a</code>
Maxima	<code>determinant(a);</code>
Inverse	
MATLAB	<code>inv(a)</code>
R	<code>solve(a)</code>
Python	<code>linalg.inv(a) or inverse(a)</code>
IDL	<code>invert(a)</code>
Axiom	<code>inverse a</code>
Maxima	<code>invert(a),detout;</code>
Pseudo-inverse	
MATLAB	<code>pinv(a)</code>
R	<code>ginv(a)</code>
Python	<code>linalg.pinv(a)</code>
Norms	
MATLAB	<code>norm(a)</code>
Python	<code>norm(a)</code>
Eigenvalues	
MATLAB	<code>eig(a)</code>
Scilab	<code>spec(a)</code>
R	<code>eigen(a)\$values</code>
Python	<code>linalg.eig(a)[0]</code> <code>eigenvalues(a)</code>
IDL	<code>hqr(elmhes(a))</code>
Mathematica	<code>Eigenvalues[matrix]</code>
Axiom	<code>eigenvalues a</code>
Singular values	
MATLAB	<code>svd(a)</code>
R	<code>svd(a)\$d</code>
Python	<code>linalg.svd(a)</code> <code>singular_value_decomposition(a)</code>
IDL	<code>svdc,A,w,U,V</code>
Mathematica	<code>SingularValueDecomposition[m]</code>
Cholesky factorization	
MATLAB	<code>chol(a)</code>
Python	<code>linalg.cholesky(a)</code>
Eigenvectors	
MATLAB	<code>[v,1] = eig(a)</code>
Scilab	<code>[v,1] = spec(a)</code>
R	<code>eigen(a)\$vectors</code>
Python	<code>linalg.eig(a)[1]</code> <code>eigenvectors(a)</code>
Axiom	<code>eigenvectors a</code>
Rank	
MATLAB	<code>rank(a)</code>
R	<code>rank(a)</code>
Python	<code>rank(a)</code>

## 4.8 Sum

	Sum of each column
MATLAB	<code>sum(a)</code>
Scilab	<code>sum(a,'c')</code>
R	<code>apply(a,2,sum)</code>
Python	<code>a.sum(axis=0)</code>
IDL	<code>total(a,2)</code>
	Sum of each row
MATLAB	<code>sum(a')</code>
Scilab	<code>sum(a,'r')</code>
R	<code>apply(a,1,sum)</code>
Python	<code>a.sum(axis=1)</code>
IDL	<code>total(a,1)</code>
	Sum of all elements
MATLAB	<code>sum(sum(a))</code>
Scilab	<code>sum(a)</code>
R	<code>sum(a)</code>
Python	<code>a.sum()</code>
IDL	<code>total(a)</code>
	Sum along diagonal
Python	<code>a.trace(offset=0)</code>
	Cumulative sum (columns)
MATLAB	<code>cumsum(a)</code>
R	<code>apply(a,2,cumsum)</code>
Python	<code>a.cumsum(axis=0)</code>

## 4.9 Sorting

	Example data	
MATLAB	<code>a = [ 4 3 2 ; 2 8 6 ; 1 4 7 ]</code>	$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 8 & 6 \\ 1 & 4 & 7 \end{bmatrix}$
Python	<code>a = array([[4,3,2],[2,8,6],[1,4,7]])</code>	
	Flat and sorted	
MATLAB	<code>sort(a(:))</code>	$\begin{bmatrix} 1 & 2 & 2 \\ 3 & 4 & 4 \\ 6 & 7 & 8 \end{bmatrix}$
Scilab	<code>s=sort(a(:)); s(\$:-1:1)</code>	
R	<code>t(sort(a))</code>	
Python	<code>a.ravel().sort()</code> or	
	Sort each column	
MATLAB	<code>sort(a)</code>	$\begin{bmatrix} 1 & 3 & 2 \\ 2 & 4 & 6 \\ 4 & 8 & 7 \end{bmatrix}$
Scilab	<code>s=sort(a,'r'); s(\$:-1:1,:)</code>	
R	<code>apply(a,2,sort)</code>	
Python	<code>a.sort(axis=0)</code> or <code>msort(a)</code>	
IDL	<code>sort(a)</code>	
	Sort each row	
MATLAB	<code>sort(a')</code>	$\begin{bmatrix} 2 & 3 & 4 \\ 2 & 6 & 8 \\ 1 & 4 & 7 \end{bmatrix}$
Scilab	<code>s=sort(a,'c'); s(:, \$:-1:1)</code>	
R	<code>t(apply(a,1,sort))</code>	
Python	<code>a.sort(axis=1)</code>	
	Sort rows (by first row)	
MATLAB	<code>sortrows(a,1)</code>	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 8 & 6 \\ 4 & 3 & 2 \end{bmatrix}$
Python	<code>a[a[:,0].argsort(),:]</code>	
	Sort, return indices	
R	<code>order(a)</code>	
Python	<code>a.ravel().argsort()</code>	
	Sort each column, return indices	
Python	<code>a.argsort(axis=0)</code>	
	Sort each row, return indices	
Python	<code>a.argsort(axis=1)</code>	

## 4.10 Maximum and minimum

	max in each column
MATLAB	<code>max(a)</code>
Scilab	<code>max(a,'c')</code>
R	<code>apply(a,2,max)</code>
Python	<code>a.max(0)</code> or <code>amax(a [,axis=0])</code>
IDL	<code>max(a,DIMENSION=2)</code>
	max in each row
MATLAB	<code>max(a')</code>
Scilab	<code>max(a,'r')</code>
R	<code>apply(a,1,max)</code>
Python	<code>a.max(1)</code> or <code>amax(a, axis=1)</code>
IDL	<code>max(a,DIMENSION=1)</code>
	max in array
MATLAB	<code>max(max(a))</code>
Scilab	<code>max(a)</code>
R	<code>max(a)</code>
Python	<code>a.max()</code> or
IDL	<code>max(a)</code>

	return indices, i
MATLAB	<code>[v i] = max(a)</code>
Scilab	<code>[v,i] = max(a,'c')</code>
R	<code>i &lt;- apply(a,1,which.max)</code>
	pairwise max
MATLAB	<code>max(b,c)</code>
Python	<code>maximum(b,c)</code>
R	<code>pmax(b,c)</code>
MATLAB	<code>cummax(a)</code>
R	<code>apply(a,2,cummax)</code>
	max-to-min range
Python	<code>a.ptp(); a.ptp(0)</code>

## 4.11 Matrix manipulation

	Flip left-right
MATLAB	<code>fliplr(a)</code>
Scilab	<code>a(:, \$:-1:1)</code> or <code>mtlb_fliplr(a)</code>
R	<code>a[,4:1]</code>
Python	<code>fliplr(a)</code> or <code>a[:,::-1]</code>
IDL	<code>reverse(a)</code>
	Flip up-down
MATLAB	<code>flipud(a)</code>
Scilab	<code>a(\$:-1:1,:)</code>
R	<code>a[3:1,]</code>
Python	<code>flipud(a)</code> or <code>a[::-1,]</code>
IDL	<code>reverse(a,2)</code>
	Rotate 90 degrees
MATLAB	<code>rot90(a)</code>
Scilab	<code>---</code>
Python	<code>rot90(a)</code>
IDL	<code>rotate(a,1)</code>
	Repeat matrix: [ a a a ; a a a ]
MATLAB	<code>repmat(a,2,3)</code>
Scilab	<code>mtlb_repmat(a,2,3)</code>
Octave	<code>kron(ones(2,3),a)</code>
Python	<code>kron(ones((2,3)),a)</code>
R	<code>kronecker(matrix(1,2,3),a)</code>
	Triangular, upper
MATLAB	<code>triu(a)</code>
R	<code>a[lower.tri(a)] &lt;- 0</code>
Python	<code>triu(a)</code>
Mathematica	<code>UpperDiagonalMatrix[f, n]</code>
	Triangular, lower
MATLAB	<code>tril(a)</code>
R	<code>a[upper.tri(a)] &lt;- 0</code>
Python	<code>tril(a)</code>

## 4.12 Equivalents to "size"

	Matrix dimensions
MATLAB	<code>size(a)</code>
R	<code>dim(a)</code>
Python	<code>a.shape</code> or <code>a.getshape()</code>
IDL	<code>size(a)</code>
	Number of columns
MATLAB	<code>size(a,2)</code> or <code>length(a)</code>
R	<code>ncol(a)</code>
Python	<code>a.shape[1]</code> or <code>size(a, axis=1)</code>
IDL	<code>s=size(a) &amp; s[1]</code>
Axiom	<code>ncols(m)</code>
Maxima	<code>mat_ncols(m)</code>
Maple	<code>linalg[coldim](m)</code>
Mathematica	<code>Dimensions[m][[2]]</code>
Derive	<code>DIMENSION(m SUB 1)</code>
	Number of elements
MATLAB	<code>length(a(:))</code>
Scilab	<code>length(a)</code>
R	<code>prod(dim(a))</code>
Python	<code>a.size</code> or <code>size(a[, axis=None])</code>
IDL	<code>n_elements(a)</code>
	Number of dimensions
MATLAB	<code>ndims(a)</code>
Python	<code>a.ndim</code>
	Number of bytes used in memory
R	<code>object.size(a)</code>
Python	<code>a.nbytes</code>

## 4.13 Matrix- and elementwise- multiplication

Elementwise operations		
MATLAB	<code>a .* b</code>	$\begin{bmatrix} 1 & 5 \\ 9 & 16 \end{bmatrix}$
R	<code>a * b</code>	
Python	<code>a * b</code> or <code>multiply(a,b)</code>	
Matrix product (dot product)		
MATLAB	<code>a * b</code>	$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$
R	<code>a %% b</code>	
Python	<code>matrixmultiply(a,b)</code>	
IDL	<code>a # b</code> or <code>b ## a</code>	
Axiom	<code>a*b</code>	
Maxima	<code>a.b</code>	
Maple	<code>evalm(a &amp;* b)</code>	
Mathematica	<code>a.b</code>	
Inner matrix vector multiplication $a \cdot b'$		
Python	<code>inner(a,b)</code> or	$\begin{bmatrix} 5 & 11 \\ 11 & 25 \end{bmatrix}$
IDL	<code>transpose(a) # b</code>	
Outer product		
R	<code>outer(a,b)</code> or <code>a %o% b</code>	$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 4 & 8 & 12 & 16 \end{bmatrix}$
Python	<code>outer(a,b)</code> or	
IDL	<code>a # b</code>	
Cross product		
R	<code>crossprod(a,b)</code> or <code>t(a) %% b</code>	$\begin{bmatrix} 10 & 14 \\ 14 & 20 \end{bmatrix}$
Kronecker product		
MATLAB	<code>kron(a,b)</code>	$\begin{bmatrix} 1 & 2 & 2 & 4 \\ 3 & 4 & 6 & 8 \\ 3 & 6 & 4 & 8 \\ 9 & 12 & 12 & 16 \end{bmatrix}$
Scilab	<code>kron(a,b)</code> or <code>a .* b</code>	
R	<code>kronecker(a,b)</code>	
Python	<code>kron(a,b)</code>	
Matrix division, $b \cdot a^{-1}$		
MATLAB	<code>a / b</code>	
Left matrix division, $b^{-1} \cdot a$ (solve linear equations)		
MATLAB	<code>a \ b</code>	$Ax = b$
Scilab	<code>linsolve(a,b)</code>	
R	<code>solve(a,b)</code>	
Python	<code>linalg.solve(a,b)</code> <code>solve_linear_equations(a,b)</code>	
IDL	<code>cramer(a,b)</code>	
Vector dot product		
Python	<code>vdot(a,b)</code>	
Cross product		
Python	<code>cross(a,b)</code>	

## 4.14 Find; conditional indexing

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

	Replace values
Python	<code>a.put(2,indices)</code>

## 5 Multi-way arrays

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

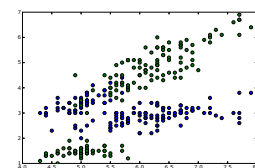
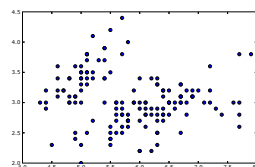
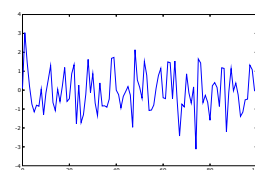
## 6 File input and output

	Reading from a file (2d)
MATLAB	<code>f = load('data.txt')</code>
R	<code>f &lt;- read.table("data.txt")</code>
Python	<code>f = fromfile("data.txt")</code>
IDL	<code>f = load("data.txt")</code>
IDL	<code>read()</code>
	Reading from a file (2d)
MATLAB	<code>f = load('data.txt')</code>
R	<code>f &lt;- read.table("data.txt")</code>
Python	<code>f = load("data.txt")</code>
IDL	<code>read()</code>
	Reading from a CSV file (2d)
MATLAB	<code>x = dlmread('data.csv', ',');</code>
R	<code>f &lt;- read.table(file="data.csv", sep=";")</code>
Python	<code>f = load('data.csv', delimiter=';')</code>
gnuplot	<code>set datafile separator ";"</code>
IDL	<code>x = read_ascii(data_start=1, delimiter=',')</code>
	Writing to a file (2d)
MATLAB	<code>save -ascii data.txt f</code>
R	<code>write(f, file="data.txt")</code>
Python	<code>save('data.csv', f, fmt='%.6f', delimiter=';')</code>
	Writing to a file (1d)
Python	<code>f.tofile(file='data.csv', format='%.6f', sep=';')</code>
	Reading from a file (1d)
Python	<code>f = fromfile(file='data.csv', sep=';')</code>

## 7 Plotting

### 7.1 Basic x-y plots

	1d line plot
MATLAB	<code>plot(a)</code>
R	<code>plot(a, type="l")</code>
Python	<code>plot(a)</code>
IDL	<code>plot, a</code>
	2d scatter plot
MATLAB	<code>plot(x(:,1), x(:,2), 'o')</code>
R	<code>plot(x[,1], x[,2])</code>
Python	<code>plot(x[:,0], x[:,1], 'o')</code>
IDL	<code>plot, x(1,*), x(2,*)</code>
	Two graphs in one plot
MATLAB	<code>plot(x1,y1, x2,y2)</code>
Python	<code>plot(x1,y1, 'bo', x2,y2, 'go')</code>





Overplotting: Add new plots to current	
MATLAB	<code>plot(x1,y1)</code> <code>hold on</code>
R	<code>plot(x2,y2)</code> <code>plot(x1,y1)</code>
Python	<code>matplot(x2,y2,add=T)</code> <code>plot(x1,y1,'o')</code> <code>plot(x2,y2,'o')</code>
IDL	<code>show()</code> # as normal <code>plot, x1, y1</code> <code>oplot, x2, y2</code>
subplots	
MATLAB	<code>subplot(211)</code>
Python	<code>subplot(211)</code>
IDL	<code>!p.multi(0,2,1)</code>
Plotting symbols and color	
MATLAB	<code>plot(x,y,'ro-')</code>
R	<code>plot(x,y,type="b",col="red")</code>
Python	<code>plot(x,y,'ro-')</code>
IDL	<code>plot, x,y, line=1, psym=-1</code>

### 7.1.1 Axes and titles

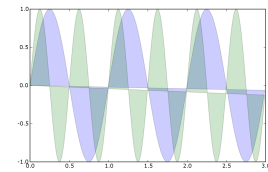
Turn on grid lines	
MATLAB	<code>grid on</code>
R	<code>grid()</code>
Python	<code>grid()</code>
1:1 aspect ratio	
MATLAB	<code>axis equal</code>
Octave	<code>axis('equal')</code> <code>replot</code>
R	<code>plot(c(1:10,10:1), asp=1)</code>
Python	<code>figure(figsize=(6,6))</code>
gnuplot	<code>set size ratio -1</code>
Set axes manually	
MATLAB	<code>axis([ 0 10 0 5 ])</code>
Scilab	<code>plot('axis',[ 0 10 0 5 ])</code>
R	<code>plot(x,y, xlim=c(0,10), ylim=c(0,5))</code>
gnuplot	<code>set xrange [0:10]</code> <code>set yrange [0:5]</code>
Python	<code>axis([ 0, 10, 0, 5 ])</code>
IDL	<code>plot, x(1,*), x(2,*),</code> <code>xran=[0,10], yran=[0,5]</code>
Axis labels and titles	
MATLAB	<code>title('title')</code> <code>xlabel('x-axis')</code> <code>ylabel('y-axis')</code>
R	<code>plot(1:10, main="title",</code> <code>xlab="x-axis", ylab="y-axis")</code>
IDL	<code>plot, x,y, title='title',</code> <code>xtitle='x-axis', ytitle='y-axis'</code>
Insert text	
Python	<code>text(2,25,'hello')</code>
IDL	<code>xyouts, 2,25, 'hello'</code>

### 7.1.2 Log plots

logarithmic y-axis	
MATLAB	<code>semilogy(a)</code>
R	<code>plot(x,y, log="y")</code>
Python	<code>semilogy(a)</code>
IDL	<code>plot, x,y, /YLOG or plot_io, x,y</code>
logarithmic x-axis	
MATLAB	<code>semilogx(a)</code>
R	<code>plot(x,y, log="x")</code>
Python	<code>semilogx(a)</code>
IDL	<code>plot, x,y, /XLOG or plot_oi, x,y</code>
logarithmic x and y axes	
MATLAB	<code>loglog(a)</code>
R	<code>plot(x,y, log="xy")</code>
Python	<code>loglog(a)</code>
IDL	<code>plot_oo, x,y</code>

### 7.1.3 Filled plots and bar plots

	Filled plot
MATLAB	<code>fill(t,s,'b', t,c,'g')</code>
Octave	<code>% fill has a bug?</code>
R	<code>plot(t,s, type="n", xlab="", ylab="")</code> <code>polygon(t,s, col="lightblue")</code> <code>polygon(t,c, col="lightgreen")</code>
Python	<code>fill(t,s,'b', t,c,'g', alpha=0.2)</code>
gnuplot	<code>set xrange [0:3]</code> <code>set samples 3/.01</code> <code>plot sin(2*pi*x) with filledcurves,\</code> <code>sin(4*pi*x) with filledcurves</code>
	Stem-and-Leaf plot
R	<code>stem(x[,3])</code>



```

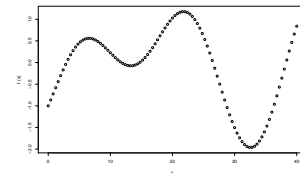
5  5
6  71
7  033
8  00113345567889
9  0133566677788
10 32674

```

### 7.1.4 Functions

	Defining functions
MATLAB	<code>f = inline('sin(x/3) - cos(x/5)')</code>
Scilab	<code>deff('y = f(x)', 'y = sin(x/3) - cos(x/5)')</code>
R	<code>f &lt;- function(x) sin(x/3) - cos(x/5)</code>
	Plot a function for given range
MATLAB	<code>ezplot(f,[0,40])</code>
Octave	<code>% no ezplot</code>
Scilab	<code>fplot2d([0:.5:40],f)</code>
R	<code>plot(f, xlim=c(0,40), type='p')</code>
Python	<code>x = arrayrange(0,40,.5)</code> <code>y = sin(x/3) - cos(x/5)</code> <code>plot(x,y, 'o')</code>
gnuplot	<code>set xrange [0,40]</code> <code>plot sin(x/3) - cos(x/5) with points</code>
Axiom	<code>draw( sin(x/3) - cos(x/5), x=0..40 )</code>

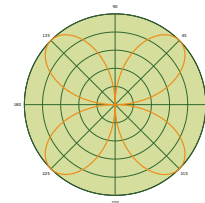
$$f(x) = \sin\left(\frac{x}{3}\right) - \cos\left(\frac{x}{5}\right)$$



## 7.2 Polar plots

MATLAB	<code>theta = 0:.001:2*pi;</code> <code>r = sin(2*theta);</code>
Python	<code>theta = arange(0,2*pi,0.001)</code> <code>r = sin(2*theta)</code>
MATLAB	<code>polar(theta, rho)</code>
Scilab	<code>polarplot(theta, rho)</code>
Python	<code>polar(theta, rho)</code>
gnuplot	<code>set polar</code> <code>plot sin(2*t)</code>

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

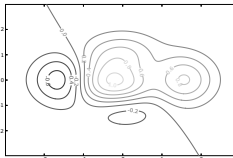
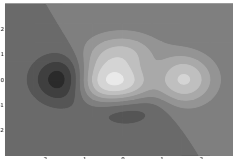

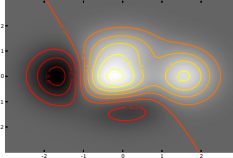


## 7.3 Histogram plots

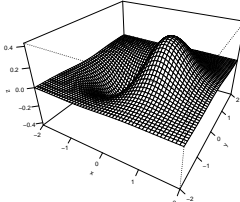
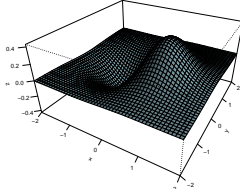
MATLAB	<code>hist(randn(1000,1))</code>
R	<code>hist(rnorm(1000))</code>
IDL	<code>plot, histogram(randomn(5,1000))</code>
MATLAB	<code>hist(randn(1000,1), -4:4)</code>
R	<code>hist(rnorm(1000), breaks= -4:4)</code>
R	<code>hist(rnorm(1000), breaks=c(seq(-5,0,0.25), seq(0.5,5,0.5)), freq=F)</code>
MATLAB	<code>plot(sort(a))</code>
R	<code>plot(apply(a,1,sort),type="l")</code>

## 7.4 3d data

### 7.4.1 Contour and image plots

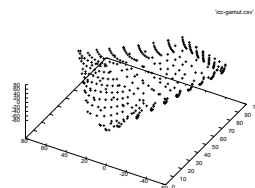
Contour plot			
MATLAB	contour(z)		
R	contour(z)		
Python	levels, colls = contour(Z, V, origin='lower', extent=(-3,3,-3,3)) clabel(colls, levels, inline=1, fmt='%1.1f', fontsize=10)		
IDL	contour, z		
Filled contour plot			
MATLAB	contourf(z); colormap(gray)		
R	filled.contour(x,y,z, nlevels=7, color=gray.colors)		
Python	contourf(Z, V, cmap=cm.gray, origin='lower', extent=(-3,3,-3,3))		
IDL	contour, z, nlevels=7, /fill contour, z, nlevels=7, /overplot, /downhill		
Plot image data			
MATLAB	image(z) colormap(gray)		
R	image(z, col=gray.colors(256))		
Python	im = imshow(Z, interpolation='bilinear', origin='lower', extent=(-3,3,-3,3))		
IDL	tv, z loadct,0		
Image with contours			
Python	# imshow() and contour() as above		
Direction field vectors			
MATLAB	quiver()		
Scilab	champ()		
Python	quiver()		

### 7.4.2 Perspective plots of surfaces over the x-y plane

MATLAB	<code>n=-2:.1:2; [x,y] = meshgrid(n,n); z=x.*exp(-x.^2-y.^2);</code>	$f(x,y) = xe^{-x^2-y^2}$
Python	<code>n=arrayrange(-2,2,.1) [x,y] = meshgrid(n,n) z = x*power(math.e,-x**2-y**2)</code>	
R	<code>f &lt;- function(x,y) x*exp(-x^2-y^2) n &lt;- seq(-2,2, length=40) z &lt;- outer(n,n,f)</code>	
	<code>z &lt;- outer(n,n,f)</code>	
Mesh plot		
MATLAB	<code>mesh(z)</code>	
R	<code>persp(x,y,z, theta=30, phi=30, expand=0.6, ticktype='detailed')</code>	
Python	<code>surface, z</code>	
IDL	<code>loadct,3</code>	
Surface plot		
MATLAB	<code>surf(x,y,z) or surfl(x,y,z)</code>	
Octave	<code>% no surfl()</code>	
R	<code>persp(x,y,z, theta=30, phi=30, expand=0.6, col='lightblue', shade=0.75, ltheta=120, ticktype='detailed')</code>	
IDL	<code>shade_surf, z loadct,3</code>	

### 7.4.3 Scatter (cloud) plots

	3d scatter plot
MATLAB	<code>plot3(x,y,z,'k')</code>
R	<code>cloud(z~x*y)</code>
gnuplot	<code>splot 'icc-gamut.csv'</code>



## 7.5 Save plot to a graphics file

	PostScript
MATLAB	<code>plot(1:10)</code>
Octave	<code>print -depsc2 foo.eps</code> <code>gset output "foo.eps"</code> <code>gset terminal postscript eps</code> <code>plot(1:10)</code>
R	<code>postscript(file="foo.eps")</code> <code>plot(1:10)</code> <code>dev.off()</code>
Python	<code>savefig('foo.eps')</code>
gnuplot	<code>set terminal postscript enhanced eps color</code> <code>set output 'foo.eps'</code> <code>plot 1:10</code>
IDL	<code>set_plot,'PS'</code> <code>device, file='foo.eps', /land</code> <code>plot x,y</code> <code>device,/close &amp; set_plot,'win'</code>
	PDF
Python	<code>savefig('foo.pdf')</code>
R	<code>pdf(file='foo.pdf')</code>
MATLAB	
	SVG (vector graphics for www)
Python	<code>savefig('foo.svg')</code>
R	<code>devSVG(file='foo.svg')</code>
gnuplot	<code>set terminal svg</code> <code>set output 'foo.svg'</code>
MATLAB	
	PNG (raster graphics)
MATLAB	<code>print -dpng foo.png</code>
Python	<code>savefig('foo.png')</code>
R	<code>png(filename = "Rplot%03d.png"</code> <code>set terminal png medium</code> <code>set output 'foo.png'</code>
gnuplot	
	Output TeX/LaTeX math
Axiom	<code>outputAsTex(e)</code>
Maxima	<code>tex(e);</code>
Maple	<code>latex(e);</code>
Mathematica	<code>TeXForm[e]</code>
MuPAD	<code>generate::TeX(e);</code>

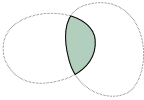
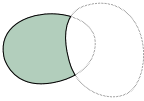
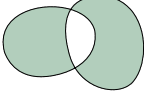
## 8 Data analysis

### 8.1 Set membership operators

	Create sets
MATLAB	<code>a = [ 1 2 2 5 2 ];</code> <code>b = [ 2 3 4 ];</code>
R	<code>a &lt;- c(1,2,2,5,2)</code> <code>b &lt;- c(2,3,4)</code>
Python	<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>
	Set unique
MATLAB	<code>unique(a)</code>
R	<code>unique(a)</code>
Python	<code>unique1d(a)</code> <code>unique(a)</code> <code>set(a)</code>
Maxima	<code>setify(a)</code>
	Set union
MATLAB	<code>union(a,b)</code>
R	<code>union(a,b)</code>
Python	<code>union1d(a,b)</code> <code>a.union(b)</code>
Maxima	<code>union(a,b)</code>

$\begin{bmatrix} 1 & 2 & 5 \end{bmatrix}$



Set intersection		
MATLAB	<code>intersect(a,b)</code>	
R	<code>intersect(a,b)</code>	
Python	<code>intersect1d(a, a.intersection(b))</code>	
Set difference		
MATLAB	<code>setdiff(a,b)</code>	
R	<code>setdiff(a,b)</code>	
Python	<code>setdiff1d(a,b)</code>	
Maxima	<code>a.difference(b)</code> <code>setdifference(a,b)</code> <code>complement(b,a)</code>	
Set exclusion		
MATLAB	<code>setxor(a,b)</code>	
R	<code>setdiff(union(a,b), intersect(a,b))</code>	
Python	<code>setxor1d(a,b)</code> <code>a.symmetric_difference(b)</code>	
True for set member		
MATLAB	<code>ismember(2,a)</code>	
R	<code>is.element(2,a)</code> or <code>2 %in% a</code>	
Python	<code>2 in a</code> <code>setmember1d(2,a)</code> <code>contains(a,2)</code>	

## 8.2 Statistics

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

## 8.3 Interpolation and regression

Straight line fit	
MATLAB	<code>z = polyval(polyfit(x,y,1),x)</code> <code>plot(x,y,'o', x,z,'-')</code>
R	<code>z &lt;- lm(y~x)</code> <code>plot(x,y)</code> <code>abline(z)</code>
Python	<code>(a,b) = polyfit(x,y,1)</code> <code>plot(x,y,'o', x,a*x+b,'-')</code>
IDL	<code>poly_fit(x,y,1)</code>
Linear least squares $y = ax + b$	
MATLAB	<code>a = x\y</code>
R	<code>solve(a,b)</code>
Python	<code>linalg.lstsq(x,y)</code> <code>(a,b) = linear_least_squares(x,y)[0]</code>
Polynomial fit	
MATLAB	<code>polyfit(x,y,3)</code>
Python	<code>polyfit(x,y,3)</code>

## 8.4 Non-linear methods

### 8.4.1 Polynomials, root finding

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

### 8.4.2 Differential equations

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

## 8.5 Fourier analysis

	Fast fourier transform
<b>Generic</b>	<code>fft(a)</code>
MATLAB	<code>fft(a)</code>
R	<code>fft(a)</code>
Python	<code>fft(a)</code> or <code>fft(a)</code>
IDL	<code>fft(a)</code>
	Inverse fourier transform
MATLAB	<code>ifft(a)</code>
R	<code>fft(a, inverse=TRUE)</code>
Python	<code>ifft(a)</code> or <code>inverse_fft(a)</code>
IDL	<code>fft(a),/inverse</code>
	Linear convolution
Python	<code>convolve(x,y)</code>
IDL	<code>convol()</code>

## 9 Symbolic algebra; calculus

	Decimal output
Axiom	<code>numeric %</code>
Maxima	<code>%,numer;</code>
	Simplification
Axiom	<code>simplify(e)</code> or <code>normalize(e)</code>
Maxima	<code>ratsimp(e)</code> or <code>radcan(e)</code>
Maple	<code>simplify(e)</code>
Mathematica	<code>Simplify[e]</code> or <code>FullSimplify[e]</code>
MuPAD	<code>simplify(e)</code> or <code>normal(e)</code>
REDUCE	<code>e</code>
Derive	<code>e</code>
	Expand
	Rectangular form
Axiom	<code>rectform e</code>
	Factorization
MATLAB	<code>factor()</code>
Axiom	<code>factor()</code>
	Integration of functions
Axiom	<code>integrate(f(x), x=0..1)</code>
Maxima	<code>integrate(f(x), x, 0, 1)</code>
Maple	<code>int(f(x), x=0..1)</code>
MuPAD	<code>int(f(x), x=0..1)</code>
Mathematica	<code>Integrate[f[x], {x,0,1}]</code>
	Differentiation
Axiom	<code>differentiate(%,x)</code>
Maxima	<code>diff(%,x)</code>
	Taylor/Laurent/etc. series approximation
Axiom	<code>series(%,x=0)</code>
	Solve equations
Axiom	<code>solve(sys,vars)</code>

$$\int_0^1 f(x)dx$$

	Laplace transform
Axiom	laplace(e,t,s)

## 10 Programming

	Script file extension
MATLAB	.m
Scilab	.sce
R	.R
Python	.py
gnuplot	.gp or .plt
IDL	.idlbatch
Maxima	.mc or .mac
bc	bc
	Comment symbol (rest of line)
MATLAB	%
Octave	% or #
Scilab	//
R	#
Python	#
gnuplot	#
IDL	;
Axiom	--
Mathematica	(* .. *)
Maple	#
Maxima	/* .. */
MuPAD	//
Derive	".."
REDUCE	%
bc	/* .. */
	Import library functions
MATLAB	% must be in MATLABPATH
Octave	% must be in LOADPATH
Scilab	getf('foo.sci')
R	library(RSvgDevice)
Python	from pylab import *
Maxima	load(SET);
	Eval
MATLAB	string='a=234'; eval(string)
Scilab	eval() evstr() execstr()
R	string <- "a <- 234" eval(parse(text=string))
Python	string="a=234" eval(string)

### 10.1 Loops

	for-statement
MATLAB	for i=1:5; disp(i); end
R	for(i in 1:5) print(i)
Python	for i in range(1,6): print(i)
IDL	for k=1,5 do print,k
	Multiline for statements
MATLAB	for i=1:5 disp(i) disp(i*2) end
R	for(i in 1:5) { print(i) print(i*2) }
Python	for i in range(1,6): print(i) print(i*2)
IDL	for k=1,5 do begin \$ print, i &\$ print, i*2 &\$ end

### 10.2 Conditionals

	if-statement
MATLAB	if 1>0 a=100; end
R	if (1>0) a <- 100
Python	if 1>0: a=100
IDL	if 1 gt 0 then a=100
	if-else-statement
MATLAB	if 1>0 a=100; else a=0; end
IDL	if 1 gt 0 then a=100 else a=0

	Ternary operator (if?true:false)	
R	<code>ifelse(a&gt;0,a,0)</code>	
gnuplot	<code>a&gt;0?a:0</code>	$a > 0 ? a : 0$
IDL	<code>a&gt;0?a:0</code>	

## 10.3 Debugging

	Most recent evaluated expression
MATLAB	<code>ans</code>
R	<code>.Last.value</code>
Axiom	<code>%</code>
Maxima	<code>%</code>
	List variables loaded into memory
MATLAB	<code>whos</code> or <code>who</code>
R	<code>objects()</code>
IDL	<code>help</code>
	Clear variable $x$ from memory
MATLAB	<code>clear x</code> or <code>clear [all]</code>
R	<code>rm(x)</code>
Axiom	<code>)clear properties x</code>
	Print
MATLAB	<code>disp(a)</code>
R	<code>print(a)</code>
Python	<code>print a</code>
IDL	<code>print, a</code>

## 10.4 Working directory and OS

	List files in directory
MATLAB	<code>dir</code> or <code>ls</code>
R	<code>list.files()</code> or <code>dir()</code>
Python	<code>os.listdir(".")</code>
IDL	<code>dir</code>
	List script files in directory
MATLAB	<code>what</code>
R	<code>list.files(pattern=".r\$")</code>
Python	<code>grep.grep("*.py")</code>
	Displays the current working directory
MATLAB	<code>pwd</code>
R	<code>getwd()</code>
Python	<code>os.getcwd()</code>
gnuplot	<code>pwd</code>
IDL	<code>sd</code>
	Change working directory
MATLAB	<code>cd foo</code>
Scilab	<code>chdir('foo')</code>
R	<code>setwd('foo')</code>
Python	<code>os.chdir('foo')</code>
gnuplot	<code>cd 'foo'</code>
IDL	<code>cd, 'foo</code> or <code>sd, 'foo</code>
Axiom	<code>)cd "foo"</code>
	Invoke a System Command
MATLAB	<code>!notepad</code>
Scilab	<code>host('notepad')</code>
Octave	<code>system("notepad")</code>
R	<code>system("notepad")</code>
Python	<code>os.system('notepad')</code>
	<code>os.popen('notepad')</code>
gnuplot	<code>!notepad</code>
IDL	<code>spawn, 'notepad'</code>

<sup>2</sup>This document is still draft quality. Most 2d plot examples are made using Matplotlib, and 3d plots using R or Gnuplot.

<sup>3</sup>Version number 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/>; Maxima 5.9.1, <http://maxima.sf.net/>.

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

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