

Information commands:

help	Lists topics on which help is available
help Bob	Lists documentation on the command Bob.m if it exists
lookfor Bob	Search all m-files for keyword "Bob"
doc Bob	Display HTML documentation on the keyword "Bob"

Workspace Commands:

who	Lists variables currently in the workspace
whos	Lists variables currently in workspace, with size
clear	Clears the workspace of all variables
clear a	Clears only variable a from workspace
clear x y z	Clears variables x,y, and z from workspace
save Bob	Saves current workspace as Bob.mat file
save Bob x y	Saves current variables x and y as file Bob.mat
load Bob	Load workspace with variables from Bob.mat

File and Directory Commands:

what	List m-files, mat-files, and mex-files in current directory
dir	Lists all files and folders in current directory
ls	Lists all files and folders in current directory.
delete Bob	Deletes file Bob
type Bob	Lists the contents of m-file Bob.m
format short	Changes to short number display format
format long	Changes to long number display format
format long e	Changes to long exponential number display format
format compact	Changes to tightly spaced format
format loose	Changes to loosely spaced format
pwd	Display name of current directory
cd	Display name of current directory
cd Bob	Changes to dir Bob in current directory
cd a:\student\Bob	Changes to dir Bob in A:\student\ directory
cd ..	Change to directory above the current one.
clc	Clears command window
path	Displays the current path definition
echo	Displays commands executing from an M-file
diary on	Appends screen output to file diary.
diary Bob	Appends screen output to file Bob.
diary off	Turns off appending of screen out.
diary	Toggle switch to turn diary on/off.

Termination commands:

Ctrl-c	Abort commands or program run back to Matlab prompt.
quit	Exits Matlab
exit	Exits Matlab

General system commands:

info	Displays general Matlab information
computer	Displays computer type
date	Displays date as text
clock	Displays time as number array [yr, mth, day, hr, min, sec]
more	Controls the paged output according to screen size
tic	Starts a stop watch
toc	Stops a stop watch

Predefined Variables:

ans	Holds the result of the last unnamed expression or calculation
eps	Smallest numerical difference between two different real numbers
inf	Value of infinity
Inf	Value of infinity
NaN	Value of 0 / 0
pi	Value of 3.14159.....
i	Value of $\sqrt{-1}$
j	Value of $\sqrt{-1}$
realmin	Smallest usable positive real number
realmax	Largest useable positive real number
nargin	Number of function input arguments
nargout	Number of function output arguments

Simple Graphics Commands:

plot(x,y)	Plots array y vs. array x
plot(x,y,'*b')	Plots array y vs. array x with blue *'s
plot(x,y,'*b',x,z,'-r')	Plots y vs. x and z vs. x on same graph
grid	Creates a grid background on graph
xlabel('x axis label')	Labels x axis
ylabel('y axis label')	Labels y axis
title('title name')	Puts title on plot
text(gx,gy,'string')	Places text 'string' at position (gx,gy) on graph
gtext('string')	Prompts user to pick mouse point to place 'string'
legend('string1','string2',...)	Creates legend of multiple plot lines on graph
axis([xmin xmax ymin ymax])	Manually sets axis limits
axis auto	Automatically sets axis limits (default)
axis square	Fixes graph in square shape
axis equal	Fixes graph to equal x and y increments
axis normal	Resets graph to default mode
whitebg('r')	Change graph background color to red
hold on	Freezes current graph for appending other plots to it.
hold off	Removes freeze to plot
shg	Shows graphic window
clg	Clears graphic window
close	Closes the graphic window

Operators and Other special characters:

=	Assignment
[]	designates matrix beginning and end
()	parenthesis, brackets subscripts, brackets arguments of functions
' '	delimiters for character string
, or space	separates elements in row of an array
; or return	separates rows of an array
...	continues command on next line
%	designates following characters as comments
:	designate a range of values in row
*	Full matrix multiplication
/	Full matrix right division
\	Full matrix left division
^	Full matrix exponentiation
.*	Array multiplication
./	Array right division
.\	Array left division
.^	Array exponentiation
+	Addition
-	Subtraction and negation
'	Transpose of a matrix
.'	Nonconjugated transpose of a matrix
(:)	single column matrix transformation
= =	"is equal to" logical comparator
~=	"is not equal to" logical comparator
<	"is less than" logical comparator
<=	"is less than or equal to" logical comparator
>	"is greater than" logical comparator
>=	"is greater than or equal to" logical comparator
~	logical "not"
&	logical "and"
	logical "or"

Order of importance of operators:

()	parenthese	Highest priority
:	colon operator	
' '	transpose operators	
^ .^	exponentiation operators	
* / \ .* ./ .\	multiplication and division	
+ -	addition, subtraction, and negation	
< > <= >= == ~=	logical comparators	
~	logical NOT	
&	logical AND	
	logical OR	Lowest priority



Trigonometric functions: (Trig function use radians, not degrees)

Page 4

cos(x)	cosine of x
sin(x)	sine of x
tan(x)	tangent of x
asin(x)	arcsine of x
acos(x)	arccosine of x
atan(x)	2 quadrant arctangent of x
atan2(x,y)	4 quadrant arctangent of x and y
sinh(x)	hyperbolic sine of x
cosh(x)	hyperbolic cosine of x
tanh(x)	hyperbolic tangent of x
asinh(x)	hyperbolic arcsine of x
acosh(x)	hyperbolic arccosine of x

Common Math Functions:

abs(x)	absolute value of x
exp(x)	exponential operation, e^x
imag(s)	computes imaginary part of complex value s
log(x)	natural logarithm
log10(x)	base 10 logarithm
real(x)	compute real part of complex number
rem(x,y)	computes remainder of division for x/y
sign(x)	generates -1, 0, or 1 based on sign of x
sqrt(x)	square root of x
ceil(x)	round x toward +infinity
fix(x)	round x toward zero
floor(x)	round x toward -infinity
round(x)	round x to nearest integer

Statistical Functions: (A refers to a 2D matrix, v refers to a 1D array)

max(A)	returns row vector with max of each column of matrix A
max(v)	returns scalar value with maximum of vector v
min(A)	returns row vector with min of each column of matrix A
min(v)	returns scalar value with minimum of vector v
mean(A)	returns row vector with mean of each column of matrix A
mean(v)	returns scalar value with mean of vector v
median(A)	returns row vector with median of each column of matrix A
median(v)	returns scalar value with median of vector v
sort(A)	sorts each column of A in ascending order
sort(v)	sorts vector v in ascending order
sum(A)	returns row vector with sum of each column of matrix A
sum(v)	returns scalar value with sum of vector v
prod(A)	returns row vector with product of each column of matrix A
prod(v)	returns scalar value with product of vector v
cumprod(A)	determines cumulative product matrix
cumsum(A)	determines cumulative sum matrix

Logical functions: (refers to V is a vector, A is a matrix, t is a text string)

all(V) Returns 1 if all elements of vector V are nonzero, 0 if not all zero
 any(V) Returns 1 if any element of vector V is nonzero, 0 if none zero
 find(V) Returns vector of indices of nonzero elements
 all(A) For each column of matrix A, returns 1 if all the elements in each column are nonzero, 0 if not.
 any(A) For each column in matrix A, returns 1 if any of the elements in each column are nonzero, 0 if none are nonzero.
 find(A) Returns a column vector of indices of nonzero elements
 exist('t') Checks if string t is either a variable, function, or file. Brings back 0 if it does not exist, 1 if a variable, 2 if an m-file, 3 if a mex-file, 4 if Simulink, or 5 if a built-in Matlab function.
 finite(A) Returns matrix of size A with 1s for finite values, 0s for Inf or NaN
 isinf(A) Returns matrix of size A with 1s for Inf or -Inf values, 0s if not
 isempty(A) Returns 1 for an empty matrix, [], 0 is not empty
 ishold Returns 1 if 'hold plot' is ON
 isletter(t) Returns 1s where t is letters, 0s where t is not alphabetic
 isnan(A) Returns matrix of size A with 1's for NaNs, 0 where values are not NaN
 isreal(A) Returns 1 if all elements are real, 0 is any is not real
 isstr(x) Returns 1 if x is character string, 0 is not a string
 logical(A) converts a numerical matrix to a logical matrix where 0 is false, ~0 is true

Utility matrices and matrix functions: (m,n are integers, v is vector, A is a matrix)

linspace(x,y,m) linearly spaced vector from x to y with m points
 logspace(x,y,m) logarithmic spaced vector from x to y with m points
 eye(m,n) returns an m*n identity matrix
 magic(m) returns an m*m magic square with integers from 1 to m^2
 zeros(m,n) returns an m*n matrix of zeros
 ones(m,n) returns an m*n matrix of ones
 rand(m,n) returns an m*n matrix of random numbers between 0 and 1
 randn(m,n) returns an m*n matrix of normally distributed random numbers
 diag(v) generates a diagonal matrix with v as the diagonal
 diag(A) extracts the diagonal of matrix A as a vector
 length(V) returns length of vector V
 size(A) returns size of matrix A as vector [#of row, # of col]
 rot90(A) rotates matrix A by 90 degrees
 fliplr(A) flips matrix A from left to right
 flipud(A) flips matrix A from up to down
 tril(A) extracts lower triangular part of A
 triu(A) extracts upper triangular part of A
 rank(A) determine the rank of matrix A
 rref(A) returns reduced row echelon form of matrix A
 det(A) returns the determinant of square matrix A
 inv(A) returns the inverse of a square, nonsingular matrix A
 norm(A) returns norm of a matrix A
 norm(v) returns norm of a vector v
 cond(A) returns condition number of matrix A
 [L,U,P]=lu(A) extract the LU decomposition and permutation matrix P
 [V,D]=eig(A) finds eigenvectors, V and eigenvalues D of matrix A

Character String Functions:

where S is a string, n is an integer, x is a real number, A is a matrix

abs(S)	converts characters in string S to their ASCII numeric values
blanks(n)	creates n blank characters
char(n)	converts ASCII numerical values (integer matrix) to character arrays.
deblank(S)	removes the trailing blanks from a string S
eval(S)	executes the string S as a Matlab command
findstr('b',S)	finds the specified substring 'b' in the given string S
int2str(n)	converts integer matrix n to strings
ischar(x)	true (=1) if x is a character array
isletter(x)	true (=1) if x is alphabetic characters
isstring(x)	true (=1) if x is a string
lower(S)	converts any upper case letters in the string S to lower case
mat2str(A)	converts matrix A to a string
num2str(x)	converts number x to string characters
strcmp(S,T)	compares two strings S and T, returns 1 if same
strncmp(S,T,n)	compares first n characters in given strings S and T
strcat(S,T)	concatenates strings S and T horizontally, ignoring trailing blanks
strvcat(S,T)	concatenates strings S and T vertically, ignoring empty strings
upper(S)	converts lower case letters in the string S to upper case.

Output to Screen Statements:

disp('char')	display the character string 'char'
disp(S)	display the contents of character string S
disp(A)	display the contents of matrix A
disp([x, y, z])	display the contents of variables x, y, and z
disp(['string1', num2str(a), 'string2'])	display a string combined with numerical value
fprintf('%f \n', m)	prints values of matrix, m with line feed
fprintf('%i', n)	prints values of integer n
fprintf('%f %f %f \n', x, y, z)	print out values of x, y, and z on same line
fprintf('The string is %s \n', S)	print out string S in position %s.

Input from Keyboard

x=input('enter value for x: ')	prompts the user to input a value for variable x
S=input('enter a string', 's')	prompts the user to input a string to variable S
k=menu('title', 'button1', 'button2', 'button3')	creates menu with 3 button, puts integer in k
menu('message', 'Continue')	creates a button which requires user to click to continue

Format field characters for fprintf and fscanf commands:

Conversion Characters		Escape Characters	
%f	fixed point notation	\n	new line
%8.4f	fixed point with 8 space min, 4 dec. places	\t	horizontal tab
%-f	fixed point notation, left justified	\b	backspace
%e	exponential notation	\r	carriage return
%12.4e	exp. notation with 12 space min, 4 dec. places	\f	form feed
%g	either fixed point or exp. notation.	\\	backslash
%i	integer notation		
%3i	integer notation with 3 spaces min		
%s	character string notation		

if-Statement Structure:

```

if (a<0)      % general structure of if-statement
    x = 1      % calculation when if true
end %if      % closing statement of if statement

```

. logical comparators .

< > <= >= == ~=

if-elseif-else Statment Structure:

```

if (a<0)      % general structure of if-statement
    x=1        % calculation for if true
elseif(a>0)   % elseif general statement
    x=2        % calculation for elseif true
else          % else statement
    x=3        % calculation when no if or elseif is true
end %if      % closing statement of if statement

```

. logical operators .

~ NOT

& AND

| OR

xor(A,B) Exclusive OR

while Loop Structure:

```

while (k>0)   % enter loop when while statement is true
    k=k-1     % calculation when while is true
end %while   % closing statement of while statement

```

for Loop Structure:

```

for k=1:n     % enter for loop, where k =first:increment:last
    x=x+5     % calculation
    if (x>100) % secondary condition to exit loop
        break % send program to statement following end%for
    end %if   % closing statement of if
end %for     % closing statement of for

```

switch-case-otherwise Statement Structure:

```

switch flag % set switch flag where flag is a number (usually integer)
    case value1 % use this case if flag == value1
        x=x+1 % calculations for case 1
    case value2 % use this case if flag == value2
        x = x+2 % calculations for case 2
    case value3 % use this case if flag == value3
        x = x+3 % calculations for case 3
    otherwise % use this case if flag ~= any value#
        x = 0 % calculations for case otherwise
end %case % closing statement for switch

```

Other loop and if related commands:

```

break % breaks out of inner most for or while loop, goes to line following end%loop
return % returns control to calling program
keyboard % active pause, allows checking of variable values, to continue type return
error('display message') % displays user defined error message and stops program execution
pause % pauses program until user hits Enter
pause(5) % pauses program for 5 seconds

```

File Access Commands:

`fid = fopen('filename.ext','permission')` Opens file with name filename.ext ,
`fclose(fid)` Close file.
`frewind(fid)` Reset file pointer to beginning of file.
`A=fscanf(fid,'%f')` Read floating point numbers from file storing values
into column matrix A until all numbers have been read.
`B=fscanf(fid,'%f', 6)` Read 6 floating point numbers from file
storing values into column matrix B
`[A,count]=fscanf(fid,'%f',[2,inf])` Read floating point numbers from file storing them in 2xN
matrix A until all values have been read.
`fprintf(fid,'%f \n', B)` Write all values of B matrix into the file, one item per line.
`fprintf(fid,'%f %f %f \n', a,b,c)` Write values a, b, and c into the file on same line.

permissions:
'w' write file
'r' read file
'a' append file

Example of using `fscanf` to read in values of x and y from data file: Bob.dat on A drive.

```
fidBob=fopen('A:Bob.dat','r')
A=fscanf(fidBob,'%f%f',[2,5])
x = A(1,1:end)
y = A(2,1:end)
fclose(fidBob)
```

Bob.dat

1	5
2	10
3	15
4	25
5	20

this gives `x = [1 2 3 4 5]` and `y = [5 10 15 25 20]`

Example used to read long one dimensional array of unknown size in file: data.txt

```
fidA=fopen('data.txt','r')
[A,count]=fscanf(fidA,'%f')
fclose(fidA)
```

The data gets stored in the column vector A and the variable count is the number of data values read.

Example used to write the values of x and y arrays into a file called MyData.txt

```
x = [1 2 3 4 5]
y = [5 10 35 50 100]
fidMy=fopen('MyData.txt','w')
fprintf(fidMy,'MyData.txt starts here:')
for i=1:length(x)
    fprintf(fidMy,' %8.4f %f \n', x(i), y(i))
end %for
fclose(fidMy)
```

MyData.txt starts here:

1.0000	5.0000
2.0000	10.0000
3.0000	35.0000
4.0000	50.0000
5.0000	100.0000

When completed the written file will look like

ODE solving:

```
[time,solution] = ode23( 'function_name', tspan, z0 )
```

or

```
[time,solution] = ode45( 'function_name', tspan, z0 )
```

where tspan = [t0 tf]

z0 = [x0, y0, . . . list of initial conditions]

Signal processing Commands:

polyval

poly

roots

conv

deconv

tf

ss

bode

evalfr

Other 2-dimensional graphs: