APPENDIX A

Syntax Quick Reference

This appendix gives examples of the most commonly used MATLAB syntax in this book (see Table A.1).

A.1 EXPRESSIONS

```
x = 2 ^ (2 * 3) / 4;
x = A \ b;              % solution of linear equations
a == 0 & b < 0          % a equals 0 AND b less than 0
a ~= 4 | b > 0          % a not equal to 4 OR b greater than 0
```

A.2 FUNCTION M-FILES

```
function y = f(x)                              % save as f.m
% comment for help

function [out1, out2] = plonk(in1, in2, in3)  % save as plonk.m
% Three input arguments, two outputs
  ...

function junk       % no input/output arguments; save as junk.m

[t, x] = ode45(@lorenz, [0 10], x0);   % function handle with @
```

A.3 GRAPHICS

```
plot(x, y), grid     % plots vector y against vector x on a grid

plot(x, y, 'b--')    % plots a blue dashed line

plot(x, y, 'go')     % plots green circles
```

Table A.1 Operator Precedence (See **Help** on `operator precedence`)

Precedence	Operators
1.	( )
2.	^ .^ ' .' (pure transpose)
3.	+ (unary plus) - (unary minus) ~ (NOT)
4.	* / \ .* ./ .\
5.	+ (addition) - (subtraction)
6.	:
7.	> < >= <= == ~=
8.	& (AND)
9.	| (OR)

```
plot(y)  % if y is a vector plots elements against row numbers
         % if y is a matrix, plots columns against row numbers

plot(x1, y1, x2, y2) % plots y1 against x1 and
                          y2 against x2 on same graph

semilogy(x, y)    % uses a log10 scale for y

polar(theta, r)   % generates a polar plot
```

A.4 IF AND SWITCH

```
if condition
  statement        % executed if condition true
end;

if condition
  statement1       % executed if condition true
else
  statement2       % executed if condition false
end;

if a == 0          % test for equality
  x = -c / b;
else
  x = -b / (2*a);
end;
```

```
if condition1      % jumps off ladder at first true condition
  statement1
elseif condition2  % elseif one word!
  statement2
elseif condition3
  statement3
...
else
  statementE
end;

if condition statement1, else statement2, end   % command line

switch lower(expr)            % expr is string or scalar
  case {'linear','bilinear'}
    disp('Method is linear')
  case 'cubic'
    disp('Method is cubic')
  case 'nearest'
    disp('Method is nearest')
  otherwise
    disp('Unknown method.')
end
```

A.5 FOR AND WHILE

```
for i = 1:n      % repeats statements n times
  statements
end;

for i = 1:3:8    % i takes values 1, 4, 7
  ...
end;

for i = 5:-2:0   % i takes values 5, 3, 1
  ...
end;

for i = v        % index i takes on each element of vector v
  statements
end;

for v = a        % index v takes on each column of matrix a
  statements
```

```
end;

for i = 1:n, statements, end      % command line version

try,
  statements,
catch,
  statements,
end

while condition    % repeats statements while condition is true
  statements
end;

while condition statements, end   % command line version
```

A.6 INPUT/OUTPUT

```
disp( x )

disp( 'Hello there' )

disp([a b])     % two scalars on one line

disp([x' y'])   % two columns (vectors x and y
                              must be same length)

disp( ['The answer is ', num2str(x)] )

fprintf( '\n' )                %  new line

fprintf( '%5.1f\n', 1.23 )     %  **1.2

fprintf( '%12.2e\n', 0.123 )   %  ***1.23e-001

fprintf( '%4.0f and %7.2f\n', 12.34, -5.6789 )
                               %  **12 and **-5.68

fprintf( 'Answers are: %g %g\n', x, y )  % matlab decides on format

fprintf( '%10s\n', str )      % left-justified string

x = input( 'Enter value of x: ' )

name = input( 'Enter your name without apostrophes: ', 's' )
```

A.7 LOAD/SAVE

```
load filename           % retrieves all variables
                          from binary file filename.mat
load x.dat              % imports matrix x from ASCII file x.dat
save filename x y z     % saves x y and z in filename.mat
save                    % saves all workspace variables
                          in matlab.mat
save filename x /ascii  % saves x in filename (as ASCII file)
```

A.8 VECTORS AND MATRICES

```
a(3,:)                  % third row

a(:,2)                  % second column

v(1:2:9)                % every second element from 1 to 9

v([2 4 5]) = [ ]        % removes second, fourth and fifth elements

v(logical([0 1 0 1 0])) % second and fourth elements only

v'                      % transpose
```