



MATLAB

for Engineering Applications
Fifth Edition

William J. Palm III

© McGraw Hill LLC. All rights reserved. No reproduction or distribution without the prior written consent of McGraw Hill LLC.

Chapter 03

Functions

© McGraw Hill LLC

2

Getting Help for Functions

You can use the `lookfor` command to find functions that are relevant to your application.

For example, type `lookfor imaginary` to get a list of the functions that deal with imaginary numbers. You will see listed:

<code>imag</code>	Complex imaginary part
<code>i</code>	Imaginary unit
<code>j</code>	Imaginary unit

Common mathematical functions

Exponential

`exp(x)` Exponential; e^x

`sqrt(x)` Square root; \sqrt{x}

Logarithmic

`log(x)` Natural logarithm; $\ln x$

`log10(x)` Common (base 10) logarithm; $\log x = \log_{10} x$

Some common mathematical functions ₁

Complex

<code>abs (x)</code>	Absolute value.
<code>angle (x)</code>	Angle of a complex number.
<code>conj (x)</code>	Complex conjugate.
<code>imag (x)</code>	Imaginary part of a complex number.
<code>real (x)</code>	Real part of a complex number.

Some common mathematical functions ₂

Numeric

<code>ceil (x)</code>	Round to nearest integer toward ∞ .
<code>fix (x)</code>	Round to nearest integer toward zero.
<code>floor (x)</code>	Round to nearest integer toward $-\infty$.
<code>round (x)</code>	Round toward nearest integer.
<code>sign (x)</code>	Signum function: +1 if $x > 0$; if $x = 0$; -1 if $x < 0$.

Operations with Complex Numbers ₁

```
>>x = -3 + 4i;  
>>y = 6 - 8i;  
>>mag_x = abs(x)  
mag_x =  
    5.0000  
>>mag_y = abs(y)  
mag_y =  
   10.0000  
>>mag_product = abs(x*y)  
    50.0000
```

© McGraw Hill LLC

7

Operations with Complex Numbers ₂

```
>>angle_x = angle(x)  
angle_x =  
    2.2143  
>>angle_y = angle(y)  
angle_y =  
   -0.9273  
>>sum_angles = angle_x + angle_y  
sum_angles =  
    1.2870  
>>angle_product = angle(x*y)  
angle_product =  
    1.2870
```

© McGraw Hill LLC

8

Operations on Arrays

MATLAB will treat a variable as an array automatically. For example, to compute the square roots of 5, 7, and 15, type

```
>>x = [5,7,15];
>>y = sqrt(x)
y =
    2.2361    2.6358    3.8730
```

Question 1

For $x = 6 - 7i$ the *angle* (x) and the *abs*(x) are

- a) -0.86 rad, 9.2 (correct)
- b) 0.86 rad, -9.2
- c) -0.86 rad, -9.2
- d) 0.86 rad, 9.2

```
>> x=6-7i;
>> angle(x)
ans =
    -0.8622
>> abs(x)
ans =
    9.2195
```

Question 2

Q2- Find the real part, and imaginary part of the $\sqrt{3 + 5i}$

- a) 3.2 , 2.3
- b) 2.1 , 1.2 (correct)
- c) 2.2 , 3.3
- d) 3.3 , 2.2

```
>> x=sqrt(3+5i)
x =
    2.1013 + 1.1897i
>> real(x)
ans =
    2.1013
>> imag(x)
ans =
    1.1897
```

Expressing Function Arguments

We can write $\sin 2$ in text, but MATLAB requires parentheses surrounding the 2 (which is called the *function argument* or *parameter*).

Thus, to evaluate $\sin 2$ in MATLAB, we type `sin(2)`. The MATLAB function name must be followed by a pair of parentheses that surround the argument.

To express in text the sine of the second element of the array x , we would type `sin(x(2))`. However, in MATLAB you cannot use square brackets or braces in this way, and you must type `sin(x(2))`.

Expressing Function Arguments ₂

To evaluate $\sin(x^2 + 5)$, you type `sin(x.^2 + 5)`.

To evaluate $\sin(\sqrt{x} + 1)$, you type `sin(sqrt(x)+1)`.

Using a function as an argument of another function is called *function composition*. Be sure to check the order of precedence and the number and placement of parentheses when typing such expressions.

Every left-facing parenthesis requires a right-facing mate. However, this condition does not guarantee that the expression is correct!

Expressing Function Arguments ₃

Another common mistake involves expressions like $\sin^2 x$, which means $(\sin x)^2$.

In MATLAB we write this expression as `(sin(x))^2`, *not* as `sin^2(x)`, `sin^2x`, `sin(x^2)`, or `sin(x)^2`!

Expressing Function Arguments ⁴

The MATLAB trigonometric functions operate in radian mode. Thus `sin(5)` computes the sine of 5 rad, not the sine of 5°.

To convert between degrees and radians, use the relation

$$q_{\text{radians}} = (\pi / 180) q_{\text{degrees}}.$$

Trigonometric Functions

<code>cos(x)</code>	Cosine; $\cos x$.
<code>cot(x)</code>	Cotangent; $\cot x$.
<code>csc(x)</code>	Cosecant; $\csc x$.
<code>sec(x)</code>	Secant; $\sec x$.
<code>sin(x)</code>	Sine; $\sin x$.
<code>tan(x)</code>	Tangent; $\tan x$.

Inverse Trigonometric Functions

$\arccos(x)$	Inverse cosine; $\arccos x$.
$\operatorname{arccot}(x)$	Inverse cotangent; $\operatorname{arccot} x$.
$\operatorname{arccsc}(x)$	Inverse cosecant; $\operatorname{arccsc} x$.
$\operatorname{arcsec}(x)$	Inverse secant; $\operatorname{arcsec} x$.
$\arcsin(x)$	Inverse sine; $\arcsin x$.
$\arctan(x)$	Inverse tangent; $\arctan x$.
$\operatorname{atan2}(y, x)$	Four-quadrant inverse tangent.

Hyperbolic Functions

$\cosh(x)$	Hyperbolic cosine
$\coth(x)$	Hyperbolic cotangent.
$\operatorname{csch}(x)$	Hyperbolic cosecant
$\operatorname{sech}(x)$	Hyperbolic secant
$\sinh(x)$	Hyperbolic sine
$\tanh(x)$	Hyperbolic tangent

Inverse Hyperbolic Functions

$\operatorname{acosh}(x)$	Inverse hyperbolic cosine
$\operatorname{acoth}(x)$	Inverse hyperbolic cotangent
$\operatorname{acsch}(x)$	Inverse hyperbolic cosecant
$\operatorname{asech}(x)$	Inverse hyperbolic secant
$\operatorname{asinh}(x)$	Inverse hyperbolic sine
$\operatorname{atanh}(x)$	Inverse hyperbolic tangent;

Question 3

For $x = 3 + 8i$ the $\operatorname{acsc}(x)$ is

- a) $-0.0409 - 0.1095i$
- b) $0.0409 + 0.1095i$
- c) $0.0409 - 0.1095i$ (correct)
- d) $-0.0409 + 0.1095i$

```
>> x=3+8*i;
>> acsc(x)

ans =

    0.0409 - 0.1095i
```

Question 4

Q4- For x in the range $0 \leq x \leq 2\pi$ use MATLAB to calculate $\tan(2x) - 2\tan x/(1 - \tan^2 x)$. From the results you can conclude that:

- a) $2\tan(2x) < \left(\frac{2\tan x}{1-\tan^2 x}\right)$
- b) $\tan(2x) > 2\left(\frac{2\tan x}{1-\tan^2 x}\right)$
- c) $\tan(2x) > \left(\frac{2\tan x}{1-\tan^2 x}\right)$
- d) $\tan(2x) = \left(\frac{2\tan x}{1-\tan^2 x}\right)$ (correct)

```
>> x=0:pi/2:2*pi;
>> y=tan(2.*x)-(2*tan(x))/(1-tan(x).^2)
```

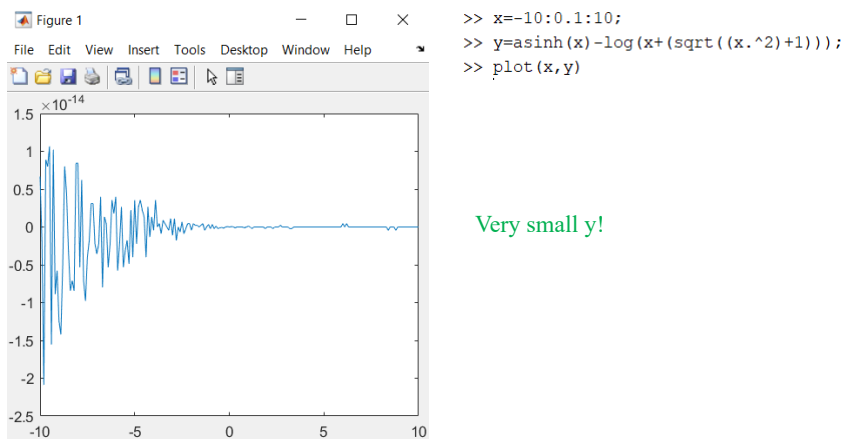
y =

```
1.0e-15 *
0.1255    0.0030   -0.1195   -0.2419   -0.3644
```

Very small y!

Question 5

Q5- For x in the range $-10 \leq x \leq 10$ use MATLAB to plot $y = \sinh^{-1} x - \ln(x + \sqrt{x^2 + 1})$. The result looks like



Very small y!

User-Defined Functions 1

To create a function file, open the Editor as described in Chapter 1, by selecting New under the HOME tab on the Toolstrip, but instead of selecting Script, select Function. The first line in a function file must begin with a function definition line that has a list of inputs and outputs. This line distinguishes a function M-file from a script M-file. Its syntax is shown in the next slide.

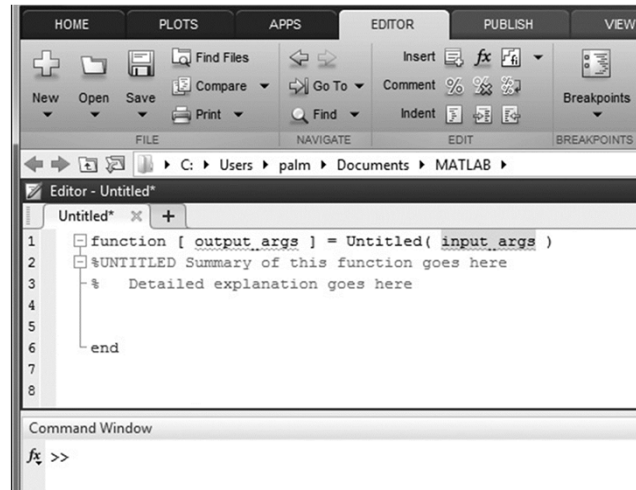
User-Defined Functions 2

The first line in a function file must begin with a *function definition line* that has a list of inputs and outputs. This line distinguishes a function M-file from a script M-file. Its syntax is as follows:

```
function [output variables] = name(input
variables)
```

Note that the output variables are enclosed in *square brackets*, while the input variables must be enclosed with *parentheses*. The function name (here, name) should be the same as the file name in which it is saved (with the .m extension).

The default Editor Window when creating a new function ¹



[Access the text alternative for slide images.](#)

© McGraw Hill LLC

Source: MATLAB

25

User-Defined Functions: Example ¹

```

function z = fun(x,y)
    u = 3*x;
    z = u + 6*y.^2;
end

```

Note the use of a semicolon at the end of the lines. This prevents the values of `u` and `z` from being displayed.

Note also the use of the array exponentiation operator (`.^`). This enables the function to accept `y` as an array.

© McGraw Hill LLC

26

User-Defined Functions: Example ₂

Call this function with its output argument:

```
>>z = fun(3,7)
```

```
z =
```

```
303
```

The function uses $x = 3$ and $y = 7$ to compute z .

© McGraw Hill LLC

27

User-Defined Functions: Example ₃

Call this function without its output argument and try to access its value. You will see an error message.

```
>>fun(3,7)
```

```
ans =
```

```
303
```

```
>>z
```

```
??? Undefined function or variable 'z'.
```

© McGraw Hill LLC

28

User-Defined Functions: Example ₄

Assign the output argument to another variable:

```
>>q = fun(3,7)
```

```
q =
```

```
    303
```

You can suppress the output by putting a semicolon after the function call.

For example, if you type `q = fun(3,7);` the value of `q` will be computed but not displayed (because of the semicolon).

© McGraw Hill LLC

29

Local Variables ₁

The variables `x` and `y` are local to the function `fun`, so unless you pass their values by naming them `x` and `y`, their values will not be available in the workspace outside the function. The variable `u` is also local to the function. For example,

```
>>x = 3; y = 7;
```

```
>>q = fun(x,y);
```

```
>>x
```

```
x =
```

```
    3
```

```
>>y
```

```
y =
```

```
    7
```

```
>>u
```

```
??? Undefined function or variable 'u'.
```

© McGraw Hill LLC

30

Argument Order

Only the order of the arguments is important, not the names of the arguments:

```
>>x = 7;y = 3;
```

```
>>z = fun(y,x)
```

```
z =
```

```
    303
```

The second line is equivalent to `z = fun(3,7)` .

© McGraw Hill LLC

31

Arrays as Inputs

You can use arrays as input arguments:

```
>>r = fun(2:4,7:9)
```

```
r =
```

```
    300    393    498
```

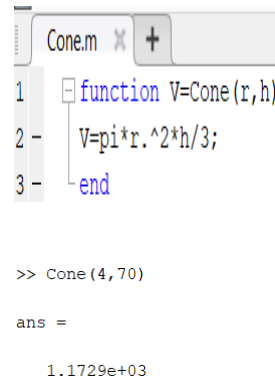
© McGraw Hill LLC

32

Question 6

Q6- Create a function called `cone` that computes the volume V of a cone whose height is h and whose radius is r . (Do not forget to check if a file already exists by that name!) The volume is given by $V = \pi r^2 h/3$. For $r=4$ and $h=70$ the result is:

- a) 3.27e+03
- b) 2.14e+02
- c) 1.17e+03 (correct)
- d) 4.22e+02



```

Cone.m
1 function V=cone(r,h)
2     V=pi*r.^2*h/3;
3 end

>> cone(4,70)

ans =

    1.1729e+03
  
```

© McGraw Hill LLC

33

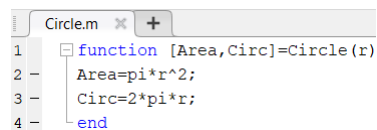
Multiple Outputs

A function may have more than one output. These are enclosed in square brackets.

For example, the function `Circle` computes the area A and circumference C of a circle, given its radius as an input argument.

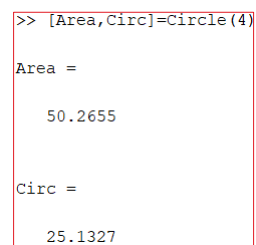
```

function [A, C] = Circle(r)
    A = pi*r.^2;
    C = 2*pi*r;
end
  
```



```

Circle.m
1 function [Area,Circ]=Circle(r)
2     Area=pi*r^2;
3     Circ=2*pi*r;
4 end
  
```



```

>> [Area,Circ]=Circle(4)

Area =

    50.2655

Circ =

    25.1327
  
```

© McGraw Hill LLC

34

Question 7

Q7- Write a function to calculate the area, $A = 2\pi rh + 2\pi r^2$, and the volume, $V = \pi r^2 h$, of the cylinder shown in Fig. 1.

A and V for $r=3$ and $h=5$ are

- a) 643.1, 545.3
- b) 231.3, 214.2
- c) 350.8, 481.6
- d) 150.8, 141.4 (correct)

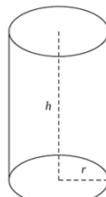


Fig. 1

```
Cylinder.m
1 function [A,V] = Cylinder(r,h)
2     A=2*pi*r*h+2*pi*r^2;
3     V=pi*r^2*h;
4     end
```

```
>> [A,V]=Cylinder(3,5)

A =

    150.7964

V =

    141.3717
```

Example of No Inputs and No Outputs

A function may have no input arguments and no output list.

For example, the function `show_date` clears all variables, clears the screen, computes and stores the date in the variable `today`, and then displays the value of `today`.

```
function show_date
    clear
    clc
    today = date
end
```

Examples of Function Definition Lines

1. One input, one output:

```
function [area_square] = square(side)
```

2. Brackets are optional for one input, one output:

```
function area_square = square(side)
```

3. Three inputs, one output:

```
function [volume_box] = box(height,width,length)
```

4. One input, two outputs:

```
function [area_circle,circumf] = circle(radius)
```

5. No named output: function sqplot(side)

Function Example 1

```
function [dist,vel] = drop(g,v0,t);
    % Computes the distance travelled and the
    % velocity of a dropped object,
    % as functions of g,
    % the initial velocity v0, and
    % the time t.
    vel = g*t + v0;
    dist = 0.5*g*t.^2 + v0*t;
end
```

Function Example 2

1. The variable names used in the function definition may, but need not, be used when the function is called:

```
>>a = 32.2;

>>initial_speed = 10;

>>time = 5;

>>[feet_dropped,speed] = . . .
drop(a,initial_speed,time)
```

Function Example 3

2. The input variables need not be assigned values outside the function prior to the function call:

```
[feet_dropped,speed] = drop(32.2,10,5)
```

3. The inputs and outputs may be arrays:

```
[feet_dropped,speed]=drop(32.2,10,0:1:5)
```

This function call produces the arrays `feet_dropped` and `speed`, each with six values corresponding to the six values of time in the array `time`.

Local Variables 2

The names of the input variables given in the function definition line are local to that function.

This means that other variable names can be used when you call the function.

All variables inside a function are erased after the function finishes executing, except when the same variable names appear in the output variable list used in the function call.

Global Variables

The `global` command declares certain variables global, and therefore their values are available to the basic workspace and to other functions that declare these variables global.

The syntax to declare the variables `a`, `x`, and `q` is

```
global a x q
```

Any assignment to those variables, in any function or in the base workspace, is available to all the other functions declaring them global.

Function Handles

You can create a function handle to any function by using the *at* sign, @, before the function name. You can then use the handle to reference the function. To create a handle to the function $y = x + 2e^{-x} - 3$, define the following function file:

```
function y = f1(x)
    y = x + 2*exp(-x) - 3;
end
```

You can pass the function as an argument to another function. For example, we can plot the function over $0 \leq x \leq 6$ as follows:

```
>>fplot(0:0.01:6,@f1)
```

Finding Zeros of a Function

You can use the `fzero` function to find the zero of a function of a single variable, which is denoted by x . One form of its syntax is

```
fzero(@function, x0)
```

where `@function` is the function handle for the function `function`, and `x0` is a user-supplied guess for the zero.

The `fzero` function returns a value of x that is near `x0`.

It identifies only points where the function crosses the x -axis, not points where the function just touches the axis.

For example, `fzero(@cos, 2)` returns the value 1.5708.

Using fzero with User-Defined Functions ₁

To use the `fzero` function to find the zeros of more complicated functions, it is more convenient to define the function in a function file.

For example, if $y = x + 2e^{-x} - 3$, define the following function file:

```
function y = f1(x)
    y = x + 2*exp(-x) - 3;
end
```

Using fzero with User-Defined Functions ₂

Plotting the function $y = x + 2e^{-x} - 3$ shows that there is a zero near $x = -0.5$ and one near $x = 3$.

To find a more precise value of the zero near $x = -0.5$, type

```
>>x = fzero(@f1,-0.5)
```

The answer is $x = -0.5831$.

Question 8

The equation $e^{-0.1x}\sin(x+2) = 0.1$ has two solutions in the interval $1 < x < 5$. The solutions are.

- a) 1.03, 4.43 (correct)
- b) 2, 6.7
- c) 3, 7.5
- d) 4, 9.4

```
Solve.m
1 function y=Solve(x)
2     y=exp(-0.1*x)*sin(x+2)-0.1;
3     end
```

```
>> fzero(@Solve,1)
```

```
ans =
```

```
1.0305
```

```
>> fzero(@Solve,2.5)
```

```
ans =
```

```
1.0305
```

```
>> fzero(@Solve,3.5)
```

```
ans =
```

```
4.4397
```

Finding the Minimum of a Function

The `fminbnd` function finds the minimum of a function of a single variable, which is denoted by `x`. One form of its syntax is

```
fminbnd(@function, x1, x2)
```

where `@function` is the function handle for the function. The `fminbnd` function returns a value of `x` that minimizes the function in the interval $x_1 \leq x \leq x_2$.

For example, `fminbnd(@cos, 0, 4)` returns the value 3.1416.

Finding the Minimum of a Function ₂

When using `fminbnd` it is more convenient to define the function in a function file. For example, if $y = 1 - xe^{-x}$, define the following function file:

```
function y = f2(x)
y = 1-x.*exp(-x);
end
```

To find the value of x that gives a minimum of y for $0 \leq x \leq 5$, type

```
>>x = fminbnd(@f2,0,5)
```

The answer is $x = 1$. To find the minimum value of y , type $y = f2(x)$. The result is $y = 0.6321$.

© McGraw Hill LLC

49

Finding the Minimum of a Function ₃

A function can have one or more *local* minima and a *global* minimum.

If the specified range of the independent variable does not enclose the global minimum, `fminbnd` will not find the global minimum.

`fminbnd` will find a minimum that occurs on a boundary.

© McGraw Hill LLC

50

Finding the Minimum of a Function ₄

To find the minimum of a function of more than one variable, use the `fminsearch` function. One form of its syntax is

```
fminsearch(@function, x0)
```

where `@function` is the function handle of the function in question. The vector `x0` is a guess that must be supplied by the user.

Finding the Minimum of a Function ₅

To minimize the function $f = xe^{-x^2-y^2}$, we first define it in an M-file, using the vector `x` whose elements are `x(1) = x` and `x(2) = y`.

```
function f = f4(x)
    f = x(1) .* exp(-x(1).^2 - x(2).^2);
end
```

Suppose we guess that the minimum is near $x = y = 0$. The session is

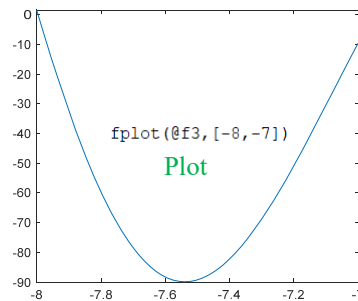
```
>>fminsearch(@f4,[0,0])
ans =
    -0.7071    0.000
```

Thus, the minimum occurs at $x = -0.7071, y = 0$.

Question 9

The function $y = 0.5 + e^{-0.6x} \sin(3x + 2)$ has one minimum point in the interval $-8 < x < -7$. Find the value of x at the minimum.

- a) -7.23
- b) -7.35
- c) -7.53 (correct)
- d) -7.78



```
f3.m
1 function y=f3(x)
2 y=0.5+exp(-0.6*x)*sin(3*x+2);
3 end
>> x=fminbnd(@f3,-8,-7)

x =
    -7.5393

>> f3(x)

ans =
   -89.8724
```

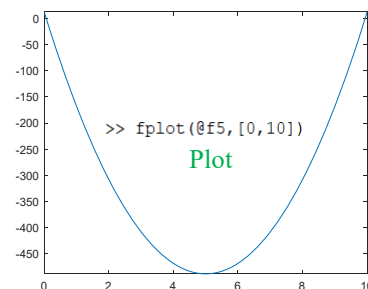
© McGraw Hill LLC

53

Question 10

Create a primary function that uses a function handle with a nested function to compute the minimum of the function $20x^2 - 200x + 12$ over the range $0 \leq x \leq 10$.

- a) 341
- b) -488 (correct)
- c) 120
- d) -14



```
f5.m
1 function y=f5(x)
2 y=20*(x.^2)-(200.*x)+12;
3 end
>> x=fminbnd(@f5,0,10)

x =
     5.0000

>> f5(x)

ans =
   -488

>> fplot(@f5,[0,10])
```

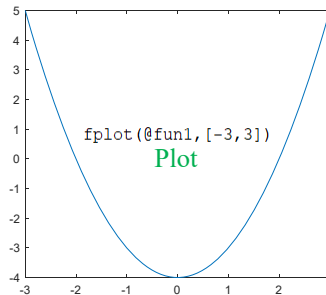
© McGraw Hill LLC

54

Question 11

Write a function to compute $y = x^2 - 4$. The values returned by `fminsearch(@fun1,-1)` and `fzero(@fun1,0)` are:

- a) $1.7\text{e-}15, 4$
- b) $6.3\text{e-}17, -4$
- c) $8.9\text{e-}16, -2$ (correct)
- d) $7.4\text{e-}14, 2$



```

fun1.m  x  +
1  function y=fun1(x)
2  y=x^2-4;
3  end
>> fminsearch(@fun1,-1)

ans =

    8.8818e-16

>> fzero(@fun1,0)

ans =

    -2

>> fplot(@fun1, [-3, 3])

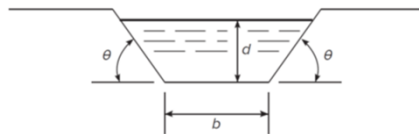
```

© McGraw Hill LLC

55

Question 12

Q12- Find the depth d and angle θ to minimize the perimeter length (L) of the channel shown below to provide an area (S) of 380 ft^2 .



$$L = \frac{S}{d} - \frac{d}{\tan \theta} + \frac{2d}{\sin \theta}$$

- a) $d = 24.7 \quad \theta = 1.24 \text{ rad}$
- b) $d = 12.6 \quad \theta = 1.12 \text{ rad}$
- c) $d = 18.9 \quad \theta = 1.41 \text{ rad}$
- d) $d = 14.8 \quad \theta = 1.04 \text{ rad}$ (correct)

```

f.m  x  +
1  function y=f(x)
2  y=(380./x(1))-(x(1)/tan(x(2)))+(2.*x(1)/sin(x(2)));
3  end
>> fminsearch(@f,[20,1])

ans =

    14.8119    1.0472

```

© McGraw Hill LLC

56

Function Handles ₂

One way to invoke, or “call,” a function into action is to use a function handle `@`. Consider the `fzero` function used with the user-defined function `fun1`, which computes $y = x^2 - 4$.

```
>>[x, value] = fzero(@fun1,[0, 3])
```

Anonymous Functions ₁

Anonymous functions enable you to create a simple function without needing to create an M-file for it. You can construct an anonymous function either at the MATLAB command line or from within another function or script. The syntax for creating an anonymous function from an expression is

```
fhandle = @(arglist) expr
```

where `arglist` is a comma-separated list of input arguments to be passed to the function, and `expr` is any single, valid MATLAB expression.

Anonymous Functions ₂

To create a simple function called `sq` to calculate the square of a number, type

```
>>sq = @(x) x.^2;
```

To improve readability, you may enclose the expression in parentheses, as `sq = @(x) (x.^2);`. To execute the function, type the name of the function handle, followed by any input arguments enclosed in parentheses. For example,

```
>>sq([5,7])
```

```
ans =
```

```
25    49
```

© McGraw Hill LLC

59

Anonymous Functions ₃

You might think that this particular anonymous function will not save you any work because typing `sq([5,7])` requires nine keystrokes, one more than is required to type `[5,7].^2`.

Here, however, the anonymous function protects you from forgetting to type the period (.) required for array exponentiation.

Anonymous functions are useful, however, for more complicated functions involving numerous keystrokes.

© McGraw Hill LLC

60

Anonymous Functions ₄

You can pass the handle of an anonymous function to other functions. For example, to find the minimum of the polynomial $4x^2 - 50x + 5$ over the interval $[-10, 10]$, you type

```
>>poly1 = @(x) 4*x.^2 - 50*x + 5;
>>fminbnd(poly1, -10, 10)

ans =

    6.2500
```

If you are not going to use that polynomial again, you can omit the handle definition line and type instead

```
>>fminbnd(@(x) 4*x.^2 - 50*x + 5, -10, 10)
```

Multiple Input Arguments ₁

You can create anonymous functions having more than one input. For example, to define the function

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

```
>>sqrtsum = @(x,y) sqrt(x.^2 + y.^2);
```

Then type

```
>>sqrtsum(3, 4)

ans =

    5
```

Multiple Input Arguments ₂

As another example, consider the function defining a plane, $z = Ax + By$. The scalar variables A and B must be assigned values before you create the function handle. For example,

```
>>A = 6; B = 4:
>>plane = @(x,y) A*x + B*y;
>>z = plane(2,8)

z =

    44
```

Question 13

Write an anonymous function to calculate $\sqrt{e^{3x} + y^3 - \sin x}$

The result for $x=4$, $y=3$ is

- a) 403.4 (correct)
- b) 340.2
- c) 278.9
- d) 169.4

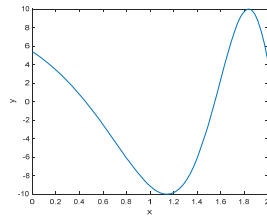
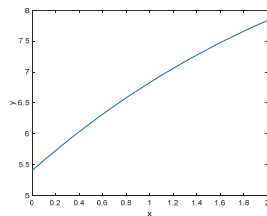
```
>> fun=@(x,y) (sqrt(exp(3*x)+y.^3-sin(x)));
>> fun(4,3)

ans =

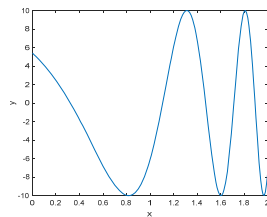
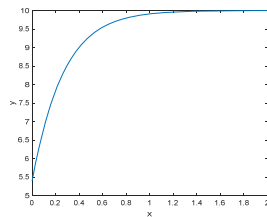
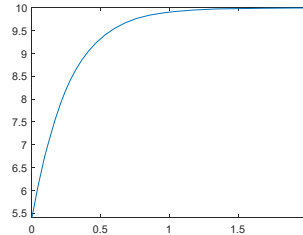
    403.4632
```


Question 14

Create an anonymous function for $10 \cos(e^{-2x})$ and use it to plot the function over the range $0 \leq x \leq 2$. Which of the following graphs represents the plot.



```
>> y=@(x) (10*cos(exp(-2*x)));  
>> fplot(y,[0,2])
```



© McGraw Hill LLC

65

Calling One Function within Another

One anonymous function can call another to implement function composition. Consider the function $5 \sin(x^3)$. It is composed of the functions $g(y) = 5 \sin(y)$ and $f(x) = x^3$. In the following session the function whose handle is `h` calls the functions whose handles are `f` and `g`.

```
>>f = @(x) x.^3;  
>>g = @(x) 5*sin(x);  
>>h = @(x) g(f(x));  
>>h(2)  
ans =  
    4.9468
```

© McGraw Hill LLC

66

Question 15

Assume $f = @(x) x.^3$; $g = @(x) 5 * \sin(x)$;

Use MATLAB to calculate $h(x, y) = g(f(x)) + 5 * y$;

The value returned by $h(6,1)$ is

- a) 9.7
- b) 14.2
- c) 8.48 (correct)
- d) 13.8

```
>> f=@(x) (x.^3);
>> g=@(x) (5*sin(x));
>> h=@(x,y) (g(f(x)))+(5*y);
>> h(6,1)
```

```
ans =

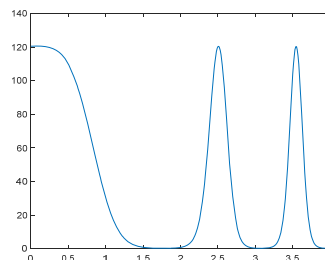
    8.4803
```

© McGraw Hill LLC

67

Question 16

Create four anonymous functions to represent the function $6e^{3 \cos x^2}$, which is composed of the functions $h(z) = 6e^z$, $g(y) = 3 \cos y$, and $f(x) = x^2$. Use the anonymous functions to plot $6e^{3 \cos x^2}$ over the range $0 \leq x \leq 4$.



```
>> h=@(x) (6*exp(x));
>> g=@(x) (3*cos(x));
>> f=@(x) (x.^2);
>> T=@(x) (h(g(f(x))))
```

```
T =

function_handle with value:
```

```
@(x) (h(g(f(x))))
```

```
>> T(2)
```

```
ans =

    0.8444
```

```
>> fplot(T,[0,4])
```

© McGraw Hill LLC

68

Variables and Anonymous Functions ₁

Variables can appear in anonymous functions in two ways:
 As variables specified in the argument list, as for example
`f = @(x) x.^3;`, and

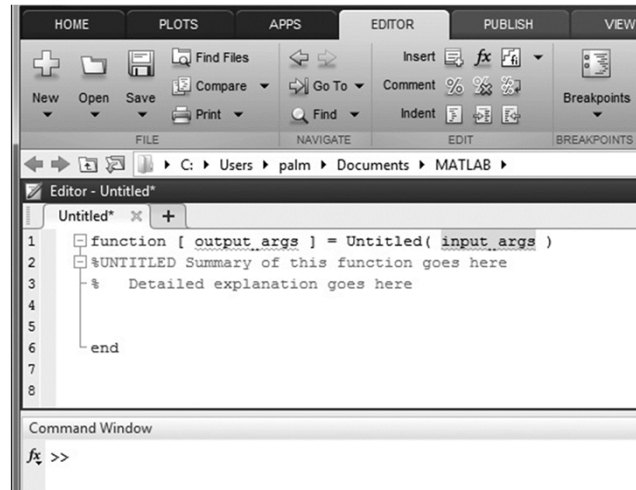
Variables and Anonymous Functions ₂

As variables specified in the body of the expression, as for example with the variables A and B in `plane = @(x,y) A*x + B*y`.

When the function is created MATLAB captures the values of these variables and retains those values for the lifetime of the function handle. If the values of A or B are changed after the handle is created, their values associated with the handle do not change.

This feature has both advantages and disadvantages, so you must keep it in mind.

The default Editor Window when creating a new function ²



[Access the text alternative for slide images.](#)

© McGraw Hill LLC

Source: MATLAB

71

Importing Spreadsheet Files

Some spreadsheet programs store data in the .wk1 format. You can use the command

`M = wklread('filename')` to import this data into MATLAB and store it in the matrix M.

The command `A = xlsread('filename')` imports the Microsoft Excel workbook file `filename.xls` into the array A. The command `[A, B] = xlsread('filename')` imports all numeric data into the array A and all text data into the cell array B.

© McGraw Hill LLC

72



Because learning changes everything.®

www.mheducation.com

© McGraw Hill LLC. All rights reserved. No reproduction or distribution without the prior written consent of McGraw Hill LLC.