Anonymous FUNCTIONS (AF)

Anonymous Functions (AF) give you a quick means of creating simple functions without having to create M−files each time.

Anonymous Functions

- Anonymous Functions (AF) give you a quick means of creating simple functions without having to create M−files each time.
- You can construct an anonymous function either in the Command Window or in the Editor.
- AF is a function that is not stored in a program file
- AF is associated with a variable whose data type is function_handle (also called *function pointer* in other languages)
- AF can accept inputs and return outputs, just as standard functions do.
- AF can contain only a **single executable statement**.

Syntax

HIYM:
How many results can AF handle?

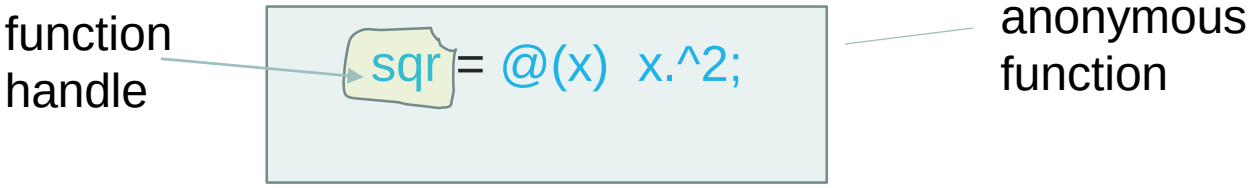
- The syntax for creating an anonymous function (AF) is

fhandle = @(arglist) expr

  - expr represents the body of the function.
    - The code that performs the main task your function is to accomplish.
    - This consists of any **single**, valid MATLAB expression which may contain calls to other functions
  - arglist is a comma−separated list of input arguments to be passed to the function. ( ) parentheses are required to place the arglist
  - The above two components are similar to the body and input argument list components of any function.
  - The @ sign is the MATLAB operator that constructs a function handle.
  - fhandle is the function handle name. Use valid MATLAB names
    - Creating a function handle for an anonymous function gives you a means of invoking the function.
    - It is also useful when you want to pass your anonymous function in a call to some other function.

Example

Create an anonymous function whose objective is to find the square of an array:



- Variable **sqr** is the function handle.
- The @ operator creates the handle, and the parentheses () immediately after the @ operator serves to include the function input arguments.
- The statement x.^2 is the function only statement and computes the square of the x-array.
- The results are stored in cells whose address are located through the function handle.

This anonymous function accepts a single input x array, and implicitly returns an output, which is an array the same size as x that contains the squared values.

Comparison

Anonymous Function

```
sqr = @(x) x.^2;
```

- no ‘function’ keyword
- no bla, bla, bla
- no termination ‘end’
- no auxiliary variable ‘R’ to store result
- Written inline within the program
- sqr is a function handle and through it the anonymous function can be called.

Standard Function

```
function [R]=sqr(x)
    % bla, bla, bla
    R=x.^2;
end
```

- Written as independent program
- sqr is a standard function name

Function Call

To find the square of a particular value (5) just pass the value to the function handle, just as you would pass an input argument to a standard function:

Example-1

```
>> sqr=@(x) x.^2          %definition

>> x=5;
>> a = sqr(x)
>> a =
          25
```

Example-2

```
>> sqr=@(x) x.^2          %definition

>> a = sqr(5)
>> a =
          25
```

Compare Standard to Anonymous Functions

AF Advantages: their simplicity as they can be written in line with the programs without having to write other independent program.

AF Disadvantages: the function body must contain ONE single argument, therefore, works for simple tasks.

Anonymous:

- Call the anonymous function
  - y=sqr(x)
- sqr is a function handle and through it the anonymous function can be called.

Standard:

- Call the standard function
  - y=sqr(x)
- sqr is a standard function

Is there any difference in syntax in the function call?

quad function

- Quadrature is a numerical method used to find the area under the graph of a function, that is, to compute a definite integral.

$$q = \int_a^b f(x)dx$$

- q = quad(fun,a,b) tries to approximate the integral of function fun from a to b to within an error of 1e-6 using recursive adaptive Simpson quadrature.
- fun is a function handle.
- Limits a and b must be finite. The function y = fun(x) should accept a vector argument x and return a vector result y, the integrand evaluated at each element of x.

function handles as input of standard functions

Function handles can be passed in within the argument list to other standard functions.

The code shown here passes the sqr function handle to the MATLAB **quad** function to compute its integral from zero to one:

```
>> quad(sqr, 0, 1)

ans =

    0.3333
```

$$I = \int_0^1 x^2 dx$$

- quad is a library function. *Quadrature* is a numerical method used to find the area under the graph of a function, that is, to compute a definite integral

function handles as input of standard functions

The code shown here passes the sqr function handle to the MATLAB **quad** function to compute its integral from one to two:

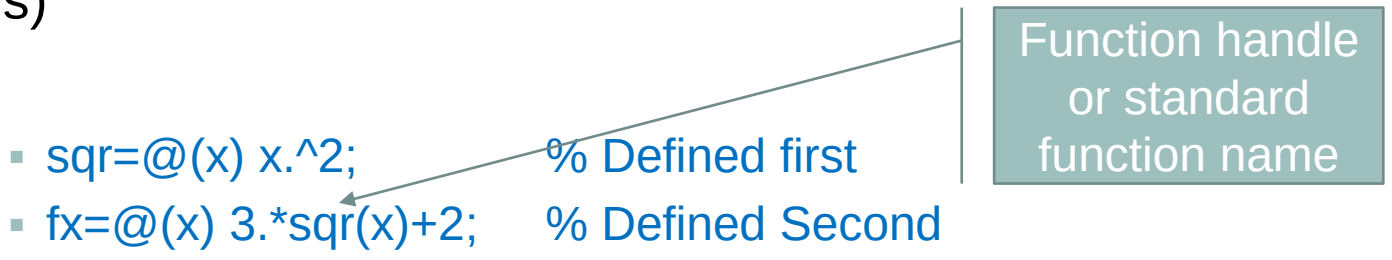
- % Integrates x^2 in the range x=[1,2];
  - fun=@(x) x.^2; % x must be a vector
  - a=1; b=2; % x =[a,b]
  - q = quad(fun,a,b) % a and b are both arguments for fun
- % this is like x=[a:h:b], the step 'h' is decide by quad

$$I=\int_1^2 x^2 dx$$

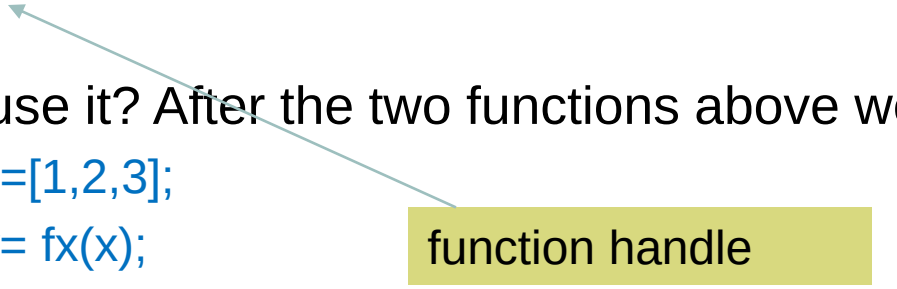
quad is a library function. *Quadrature* is a numerical method used to find the area under the graph of a function, that is, to compute a definite integral

function handles within Anonymous Functions

- We want to compute fx=3x^2+2 with and Anonymous Function. The computation of x^2 will be computed also with another Anonymous Function.
- This is equivalent to an AF within another AF (nested anonymous functions)



- How to use it? After the two functions above were defined:
  - x=[1,2,3];
  - y= fx(x);



Two-Input Example

There is not restrictions as the number of arguments in the Anonymous Functions. The following AF uses two input arguments, x and y. The example assumes that variables A and B are already defined:

- sumAxBy = @(x, y) (A*x + B*y);

To call this function, assigning 5 to x and 7 to y, for example, type:

- >> sumAxBy(5, 7)

or, more appropriate:

- >>x=5; y=7;
- >>sumAxBy(x,y)

No Input Arguments

For anonymous functions that do not take any input arguments, construct the function using empty parentheses for the input argument list:

- t = @() datestr(now);

Also use empty parentheses when invoking (i.e., calling) the function:

- t() <Enter>
- ans =
- 04−Sep−2003 10:17:59

- You must include the parentheses. If you type the function handle name with no parentheses, MATLAB just identifies the handle; it does not execute the related function:
- t <Enter>
- t =
- @() datestr(now)

“By-passing” the one-line restriction of AF

- We want to write a function to find the minimum and maximum of a set of numbers and store the results in an array.
- Minimum and Maximum Example

```
min_and_max = @(x) [min(x), max(x)];
```

- Call:

```
min_and_max([3 4 1 6 2])
ans =
    1     6
```

Need Brackets: Numbers represent the argument x (one argument, i.e., an array)

“By-passing” the one-line restriction of AF

- We want to write a function to find the minimum and maximum of a set of numbers and store the results in an array.
- Minimum and Maximum Example

```
min_and_max = @(x) [min(x), max(x)];
```

- Alternative Call:

```
x=[3 4 1 6 2];
min_and_max(x)
ans =
     1     6
```

Example

Use the anonymous function in a vectorized program, e.g., Trapezoidal Integration of (x^2) in the range x=[0,2]:

```
% Trapezoidal Program
clc, clear
n=100; a=0; b=2.0; h=(b-a)/n;
 f=@(x) x.^2;   %creates the AF

    x=[a:h:b];
    c=ones(1,n+1); c(2:1:n)=2;
    t=c.*f(x);     % call the anonymous function
    s=sum(t);

I=(1/2)*h*s;
% please add the output statement
```

- How many values in x-array?
- How many values after the call of f(x)?
- How many values in t-array?
- How many values in s?
- How many values in I?

User Input a Function Handle

% How a USER can input a function handle?

% Integral Program

```
clc, clear, close
fprintf('Enter anonymous function whose integral you are searching for as \n');
f = input('e.g. f(x)=x.^2 you enter @(x)x.^2 \n');
```

OUTPUT

Enter anonymous function whose root you are searching for as

e.g. If the function is f(x)=x.^3 you enter @(x)x.^3

@(x)x.^2 <Enter>

user input + enter

% After this input f has been defined as a function handler

User Input a Function Handle, Example

Use the anonymous function in a vectorized program, e.g., Trapezoidal Integration of (x^2) in the range x=[0,2]:

BEFORE:

```
% Trapezoidal Program
clc, clear
n=100; a=0; b=2.0; h=(b-a)/n;
 f=@(x) x.^2;   %creates the AF


    x=[a:h:b];
    c=ones(1,n+1); c(2:1:n)=2;
    t=c.*f(x);     % call the anonymous function
    s=sum(t);


I=(1/2)*h*s;
% please add the output statement
```

AFTER

```
% Trapezoidal Program
clc, clear
n=100; a=0; b=2.0; h=(b-a)/n;
fprintf('Enter function whose integral you are computing \n');
f = input(‘ as AF, e.g. f(x)=x.^2 you enter @(x)x.^2 \n');
    x=[a:h:b];
    c=ones(1,n+1); c(2:1:n)=2;
    t=c.*f(x);     % call the anonymous function
    s=sum(t);


I=(1/2)*h*s;
% please add the output statement
```

Exercises:

1. Construct an Anonymous Function to convert degree Farhenheit to Celsius and use it within a program to compute a Table of degrees Celsius vs Farhenheit in the range F=[-50:10:200]

$$C = (5/9)\,(F - 32)$$

2. Construct an Anonymous Function to compute the integrand **F** of the integral below in the range x=[1:0.1:2].

$$I = \int_{a=1}^{b=2} \left| \frac{\cos(x)}{x^3} \right|^2 \left(\frac{\ln(x)}{3} \right)^{(1/2)} e^{-x}\, dx$$

3. Construct an Anonymous Function to compute the function below. Use the AF within a program to find F(x) in the range x=[0:0.1:3]. Print the results as a Table v vs F(x)

$$f(x) = x^3 - x - 2$$

Additional Exercises

- “Practice makes perfect”

1. Identify the problems in the document “004-functions-exercises-v5.pdf” which can be solved with the anonymous function syntax.
2. Solve them.

REFERENCES

- http://www.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html
- http://blogs.mathworks.com/loren/2013/01/10/introduction-to-functional-programming-with-anonymous-functions-part-1/