MONASH University

MONASH
ENGINEERING
ENG1060

FUNCTION HANDLES AND ANONYMOUS FUNCTIONS

Presented by Tony Vo

Slides by Tony Vo

GROUP OF EIGHT AUSTRALIA

FUNCTION HANDLES

- A function handle is a variable that represents a function
  - It provides a link between a variable and a function
  - Data Type: function_handle

- You can create function handles for:
  - Built-in functions
  - User-defined functions
  - Anonymous functions

- Syntax: <variable> = @<function>

FUNCTION HANDLE EXAMPLES

```
clc; clear all; close all;

%% Some data to play around with
x = linspace(0, 2*pi);  % 100 points from 0 to 2pi

my_handle = @sin;       % my_handle is now sin()
figure; plot(x, my_handle(x));
title('sin(x)');

%%
my_handle = @tan;       % my_handle changed to tan()
figure;plot(x, my_handle(x));
title('tan(x)');

%%
my_handle = @MyMaths;   % my_handle --> user function
figure; plot(x, my_handle(x));
title('MyMaths(x)');
```

```
function output = MyMaths(input)
output = input.^3 + input.^2 + input.^1;
```

ANONYMOUS FUNCTIONS

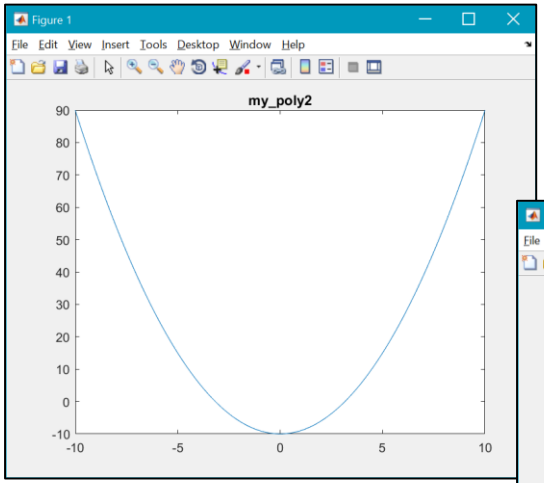
- Sometimes, you want to create a function "on-the-fly" without a function file
  - Anonymous functions can achieve this
  - Don't need to create and manage too many function files
- Anonymous functions are a single-line function
  - Syntax: fn_handle = @(inputs) <command>
  - The function can have many input arguments
  - But the function can only return one output
  - Often used to represent mathematical expressions
- When you create an anonymous function, a function handle is returned

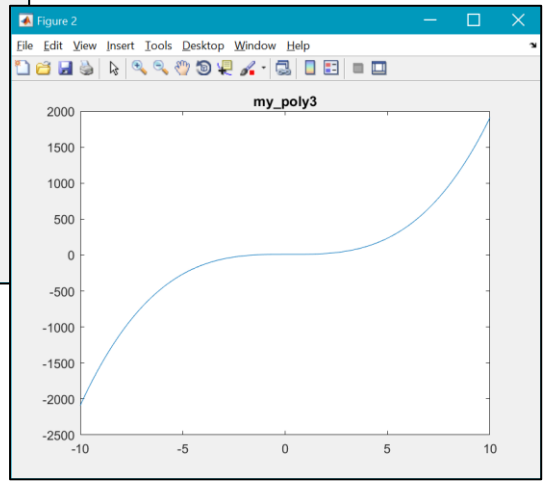
ANONYMOUS FUNCTION EXAMPLES

- Syntax: fn_handle = @(inputs) <command>

```
my_poly2 = @(x) x.^2 - 10;
my_poly3 = @(x) 2*x.^3 - my_poly2(x);
pythagoras = @(B,C) sqrt(B.^2 + C.^2);
today = @() fprintf('Today is %s\n', date());
```

```
x = linspace(-10, 10);
figure;
plot(x, my_poly2(x));
title('my\_poly2')

figure;
plot(x, my_poly3(x));
title('my\_poly3')
pause;
```





ANONYMOUS FUNCTION EXAMPLES

- Syntax: fn_handle = @(inputs) <command>

```
my_poly2 = @(x) x.^2 - 10;
my_poly3 = @(x) 2*x.^3 - my_poly2(x);
pythagoras = @(B,C) sqrt(B.^2 + C.^2);
today = @() fprintf('Today is %s\n', date());
```

```
H = pythagoras(3,4);
disp(H);
pause;

today();
```

Command Window

```
     5


Today is 23-Aug-2016
>>
```

Workspace

Name	Value
H	5
my_poly2	@(x)x.^2-10
my_poly3	@(x)2*x.^3-my_poly2(x)
pythagoras	@(B,C)sqrt(B.^2+C.^2)
today	@()fprintf('Today is %s\n',date())
x	*1x100 double*

SINE TAYLOR USING ANONYMOUS FUNCTIONS

- Replace the SineTaylor function with an anonymous function
  - Recall, the SineTaylor function only had one line of coding for its output

SineTaylor.m

```
function taylor_result = SineTaylor(x,n)
taylor_result = (-1)^n / factorial(2*n + 1) * x.^(2*n + 1);
```

```
% comparing entire 360 degrees of sine values
x = -pi:0.1:pi;

% prompt user for maximum n
n_max = input('Enter the maximum n: ');

% initialising the sin_approx variable
sin_approx = 0;

% Summing Taylor series terms up to n = n_max
for n = 0:n_max
    sin_approx = sin_approx + SineTaylor(x, n);
end
```

```
% comparing entire 360 degrees of sine values
x = -pi:0.1:pi;

% prompt user for maximum n
n_max = input('Enter the maximum n: ');

% initialising the sin_approx variable
sin_approx = 0;

% Defining Anonymous Function for Sine Taylor term
st_fn = @(x,n) (-1)^n / factorial(2*n + 1) * x.^(2*n+1);

for n = 0:n_max
    sin_approx = sin_approx + st_fn(x, n);
end
```

FUNCTION HANDLES AS FUNCTION INPUT

- A function handle is basically a pointer (alias) to the actual function
  - This means we can now pass functions as input arguments to other functions!

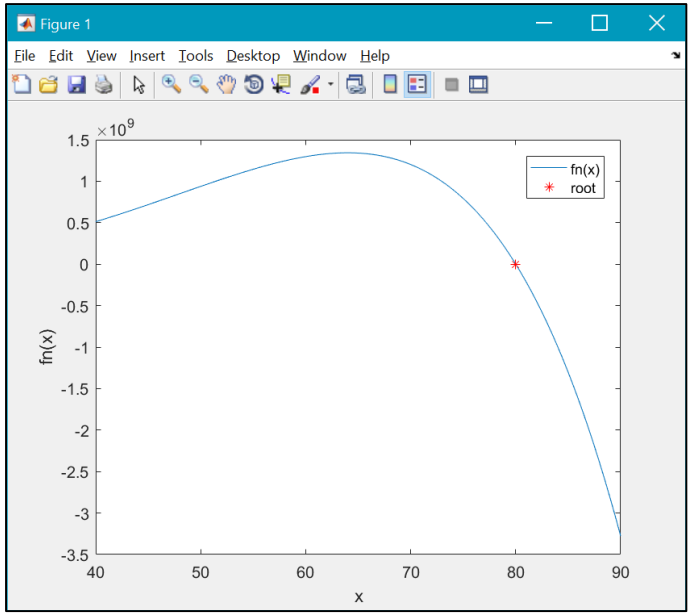
- E.g. Use MATLAB's fzero() to find where a function crosses the x axis (y=0)

```
% anonymous function
fn = @(x) -5*x.^5 + 400*x.^4 + 3*x.^3 + 20*x.^2 - x + 5;
x0 = fzero(fn, 75);

% plotting
x = linspace(40, 90);
plot(x, fn(x), '-', x0, fn(x0), 'r*');
xlabel('x');
ylabel('fn(x)')
legend('fn(x)','root')
```

FUNCTION HANDLES AS FUNCTION INPUT

- E.g. Use MATLAB's fzero() to find where a function crosses the x axis (y=0)

```
% anonymous function
fn = @(x) -5*x.^5 + 400*x.^4 + 3*x.^3 + 20*x.^2 - x + 5;
x0 = fzero(fn, 75);

% plotting
x = linspace(40, 90);
plot(x, fn(x), '-', x0, fn(x0), 'r*');
xlabel('x');
ylabel('fn(x)')
legend('fn(x)','root')
```



SUMMARY

- Function handles
- Anonymous functions

- Can an anonymous function be referenced inside another anonymous function?