MONASH University

MONASH
ENGINEERING
ENG1060

FUNCTIONS: MULTIPLE INPUTS

Presented by Tony Vo

Slides by Tony Vo

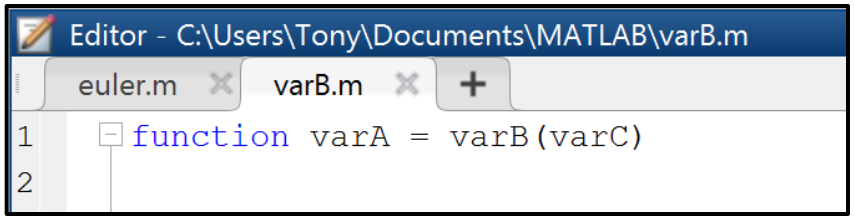
GROUP OF EIGHT AUSTRALIA

RECALL: HOW TO CREATE A FUNCTION

- Start a new m-file (script file)
  - Declare it as a function with a function header

- The function header declaration

```
function outputs = function_name(inputs)
```



CALCULATING SINE TAYLOR

MONASH University

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

```
%Created by: Tony Vo
%class example
clear all; close all; clc;

%create a vector of x values
x=-pi:0.01:pi;

%calculating  the first two terms
n=0;
sin_n0 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=1;
sin_n1 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=2;
sin_n2 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=3;
sin_n3 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=4;
sin_n4 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
```

```
%adding the terms to get the Sine approximation
sin_approx = sin_n0 + sin_n1 + sin_n2 + sin_n3 +
sin_n4;
MATLAB_sin = sin(x);

%plotting
plot(x,sin_approx,'b','linewidth',5)
hold on
plot(x,MATLAB_sin,'r','linewidth',5)

xlabel('x  (radians)')
ylabel('sin(x)')
legend('sin\_approx','MATLAB\_sin')
```

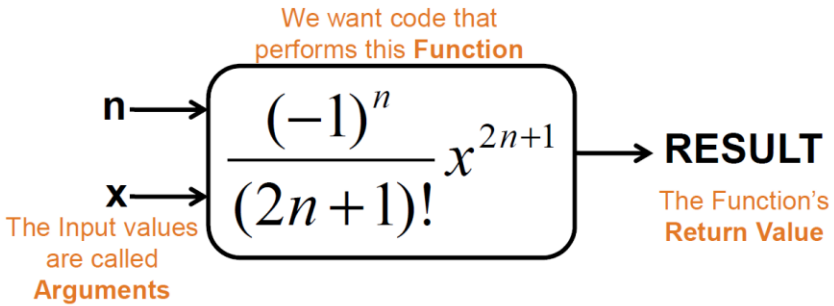
DESIGNING A SINE TAYLOR FUNCTION

MONASH University

- The maths (and code) being repeated

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

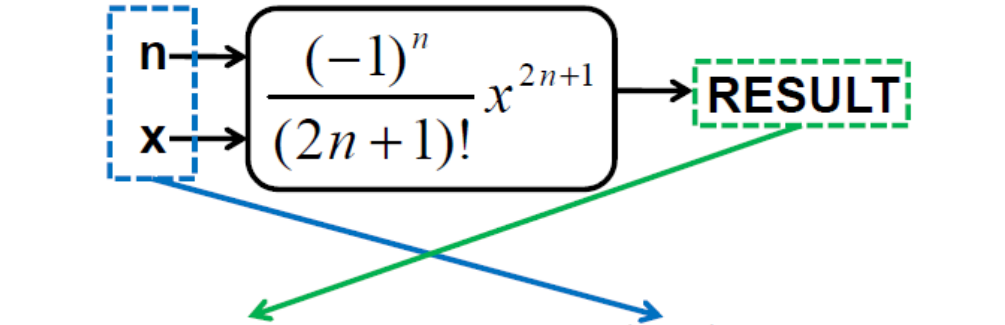
- We want code that performs this function



WRITING A SINE TAYLOR FUNCTION

- The function declaration

  function outputs = function_name(inputs)



WRITING A SINE TAYLOR FUNCTION

MONASH University

- The function declaration

  function outputs = function_name(inputs)

  - outputs = taylor_result
  - inputs = x and n
  - function name = SineTaylor

SineTaylor.m

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

- The point of a function is to determine its outputs

- What's missing from the above code?

FUNCTION DOCUMENTATION

- Include the following information AFTER the function declaration
  - Function declaration without the word "function"
  - Name, ID and date and description of what the function does
  - Description of the input argument and outputs

SineTaylor.m

```
1   function taylor_result = SineTaylor(x,n)
2   % taylor_result = SineTaylor(x,n)
3   % Written by Tony Vo
4   % Created on 12/07/2015
5   %
6   % Input Arguments
7   % ----
8   % x - input values for Taylor term (can be a vector)
9   % n - n value for Taylor term (must be a scalar)
10  % Outputs
11  % ----
12  % taylor_result - taylor series term values (can be vector)
13  taylor_result = (-1)^n / factorial(2*n + 1) * x.^(2*n + 1);
14
```

CALLING THE SINETAYLOR FUNCTION

```
%Created by: Tony Vo
%class example
clear all; close all; clc;

%create a vector of x values
x=-pi:0.01:pi;

%calculating the first two terms
n=0;
sin_n0 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=1;
sin_n1 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=2;
sin_n2 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=3;
sin_n3 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
n=4;
sin_n4 = (-1)^n/factorial(2*n+1)*x.^(2*n+1);
```

SineTaylor.m

```
1   function taylor_result = SineTaylor(x,n)
```

```
x = -pi:0.1:pi;

% n= 0
sin_n0 = SineTaylor(x,  0)
% n= 1
sin_n1 = SineTaylor(x,  1)
% n= 2
sin_n2 = SineTaylor(x,  2)
% n= 3
sin_n3 = SineTaylor(x,  3)
% n= 4
sin_n4 = SineTaylor(x,  4)
% n= 5
sin_n5 = SineTaylor(x,  5);
```

CALLING THE SINETAYLOR FUNCTION

SineTaylor.m

```
1  function taylor_result = SineTaylor(x,n)
```

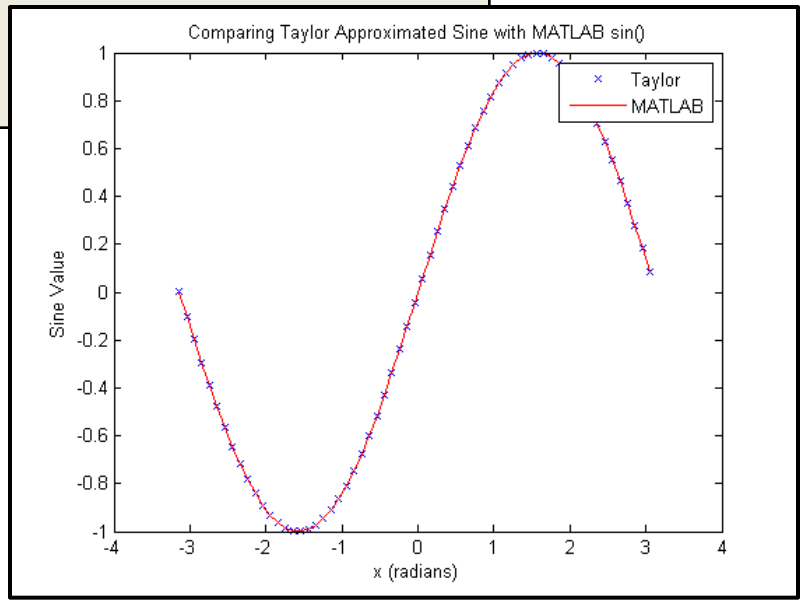
```
% Written by Tony Vo
% Created on 12/07/2015
%
% Uses SineTaylor function to generate Sine Taylor Series
% terms to approximate sine function. Approximated values
% are compared against MATLAB's sin() function using plot()

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

% Summing Taylor series terms up to n = 5
sin_approx = SineTaylor(x, 0) + SineTaylor(x, 1) + ...
SineTaylor(x, 2) + SineTaylor(x, 3) + SineTaylor(x, 4) ...
+ SineTaylor(x, 5);
```

CALLING THE SINETAYLOR FUNCTION

```
% Plotting Taylor sine against MATLAB sine and labelling plot
plot(x, sin_approx, 'bx', x, sin(x), 'r')
title('Comparing Taylor Approximated Sine with MATLAB sin()');
xlabel('x (radians)');
ylabel('Sine Value');
legend('Taylor', 'MATLAB');
```



ORDER OF INPUTS AND OUTPUTS

- Use an m-file to call functions
  - Order of inputs and outputs matter, variable names don't

SineTaylor.m

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

Command Window

```
>> a = SineTaylor(y,z)
   d = SineTaylor(w,k)
   g = SineTaylor(p,q)
   h = SineTaylor(c,m)
```

REVISIT: WHY DO WE NEED FUNCTIONS?

- A function lets your record a pattern of code and reuse it with different input arguments
  - The function uses its input arguments to produce an output
- Functions prevent you from repeating yourself
  - You save time and your code is shorter
  - Functions force you to think about structure before you code
- Functions make the code modular which is
  - More reusable
  - Easier to document and share with others

SUMMARY

- Functions can take multiple inputs
- Call functions using a separate m-file
- When calling functions
  - the order of the inputs is important
  - the variable name does not need to match

- Can inputs to functions be matrices or are they limited to scalars and vectors?