Computer Aided Design (CAD)

Lecture 7

- Matlab debugging
- Structure in Matlab

Dr.Eng. Basem ElHalawany

Schedule (Draft)

Topics	Estimated Duration (# Lectures)
Introduction	1
Introduction to Matlab Environment	1
Matlab Programing (m-files) (1)	5 **(5/5)**
Modeling using Matlab Simulink Tool	4
Communication Systems Simulation (Applications)	3
Midterm	8th Week
Introduction to FPGA + Review on Digital Logic/Circuits	2
VHDL Modeling Language	4
VHDL Application	2
Introduction to OPNET Network Simulator	3
Course Closeout / Feedback/ project (s) Delivery	1

The Lecture is based on :

A. Matlab by Example: Programming Basics, Munther Gdeisat

8 Matlab Debugging, Profiling, and Code Indentation

The debugging process is the procedure of finding bugs and errors and fixing them.

8.1.1 Syntax and Runtime Errors

Syntax Errors

- These errors mainly occur as a result of the misspelling of variable or function names or from missing quotes or parentheses

```
>> x = x*(1 + 2*x));         % extra parenthesis
>> x = 1; x = x(x + 1)       % The multiplication sign is missing
>> y = `hello                % missing quote
>> z = 1; disp(Z);           % Matlab is case sensitive, so Z is not the
                             % same as z.
```

- When you type a Matlab command in the Command Window, Matlab checks for syntax errors before running the command.
  - ✓ If the command passes the syntax error check, then Matlab executes this command;
  - ✓ Otherwise, Matlab displays a message reporting that there is a syntax error

- Suppose that you attempt to run a Matlab script file that contains syntax errors.
- Matlab does not run this file and responds by reporting that the file contains syntax errors.

Runtime Errors

- Runtime errors are found by Matlab during the execution of a program,
- They are generally more difficult to fix than simple syntax errors.
- The ability to fix run-time errors is something that improves with experience

- Let us try to write a Matlab program that calculates the absolute values of a vector.

```
clear; clc; close all
x = -10:10;
for k = 0:length(x)
   if (x(k) > 0)
      x(k) = -x(k);
   end
end
disp(x)
```

```
for k = 1:length(x)
```

```
??? Attempted to access x(0); index must be a positive integer or logical.
Error in ==> absolute at 4
  if (x(k) > 0)
```

8.1.2 Debugging Matlab Code

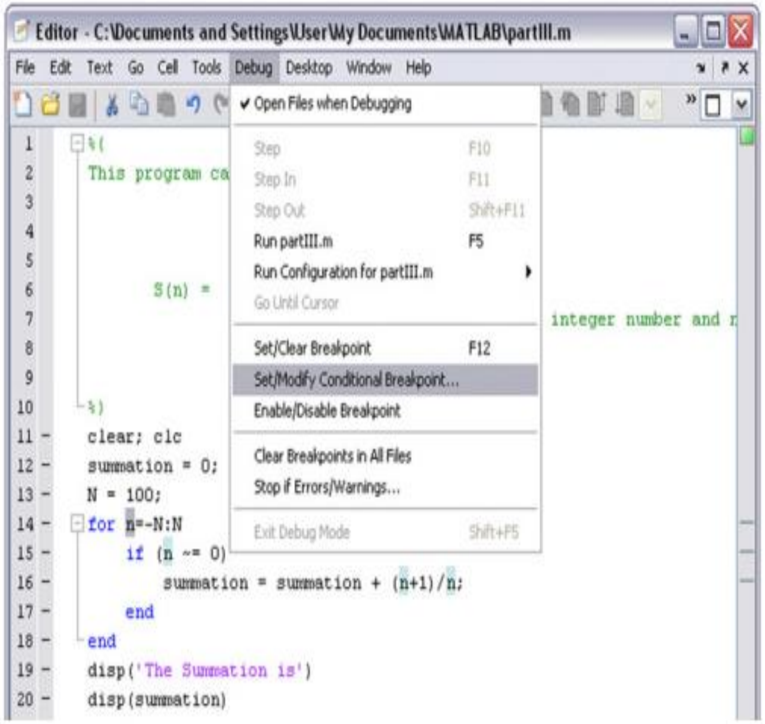
8.1.2.2 Stepping Through the Program

8.2.1.3.1 Step In Tool

8.2.1.3.2 Step Out Tool

Lesson 8.3 Advanced Matlab Debugging Tools

8.3.2 The Conditional Breakpoint Debugging Tool

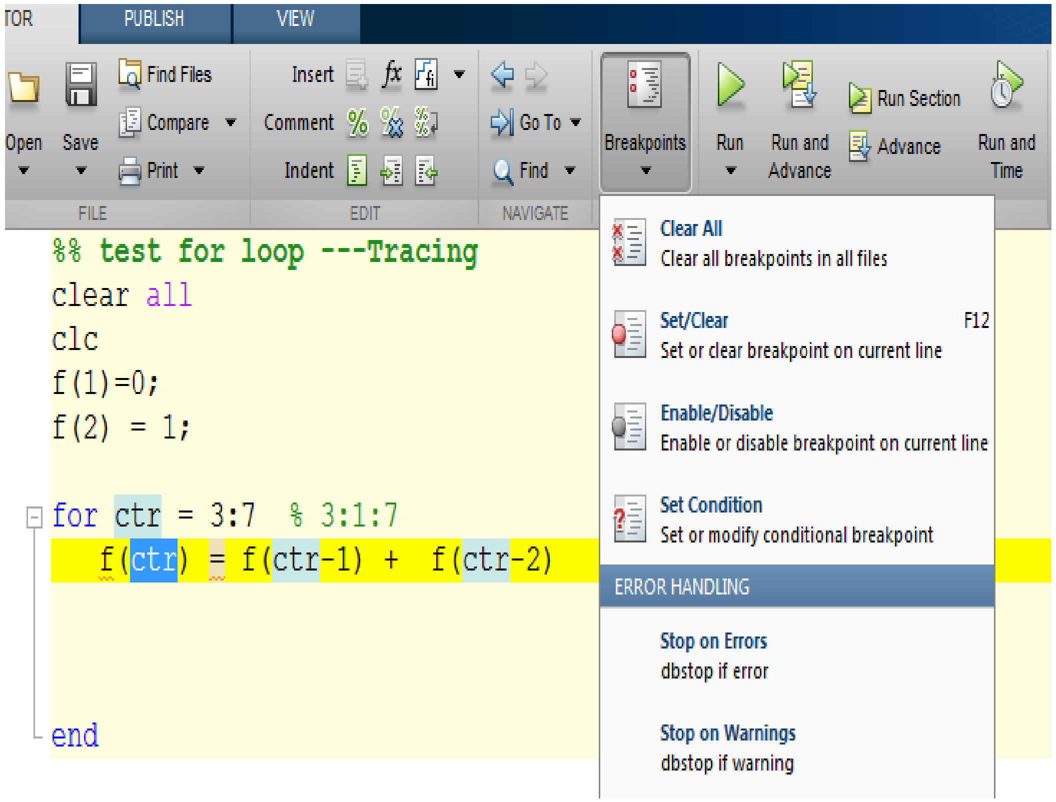


Editor - C:\Documents and Settings\User\My Documents\MATLAB\partIII.m
File Edit Text Go Cell Tools Debug Desktop Window Help
✔ Open Files when Debugging
Step F10
Step In F11
Step Out Shift+F11
Run partIII.m F5
Run Configuration for partIII.m
Go Until Cursor
Set/Clear Breakpoint F12
Set/Modify Conditional Breakpoint...
Enable/Disable Breakpoint
Clear Breakpoints in All Files
Stop if Errors/Warnings...
Exit Debug Mode Shift+F5

```
%{
This program ca

        S(n) =
                                    integer number and r

%}
clear; clc
summation = 0;
N = 100;
for n=-N:N
    if (n ~= 0)
        summation = summation + (n+1)/n;
    end
end
disp('The Summation is')
disp(summation)
```



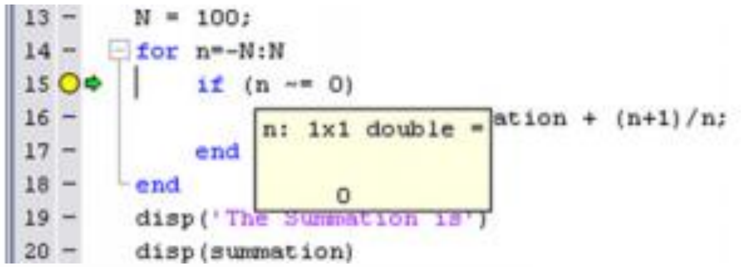
```
%% test for loop ---Tracing
clear all
clc
f(1)=0;
f(2) = 1;

for ctr = 3:7  % 3:1:7
    f(ctr) = f(ctr-1) +  f(ctr-2)



end
```

- We can use the Matlab debugging tool Set/Modify Conditional Breakpoint to check the execution of the program when n=0.

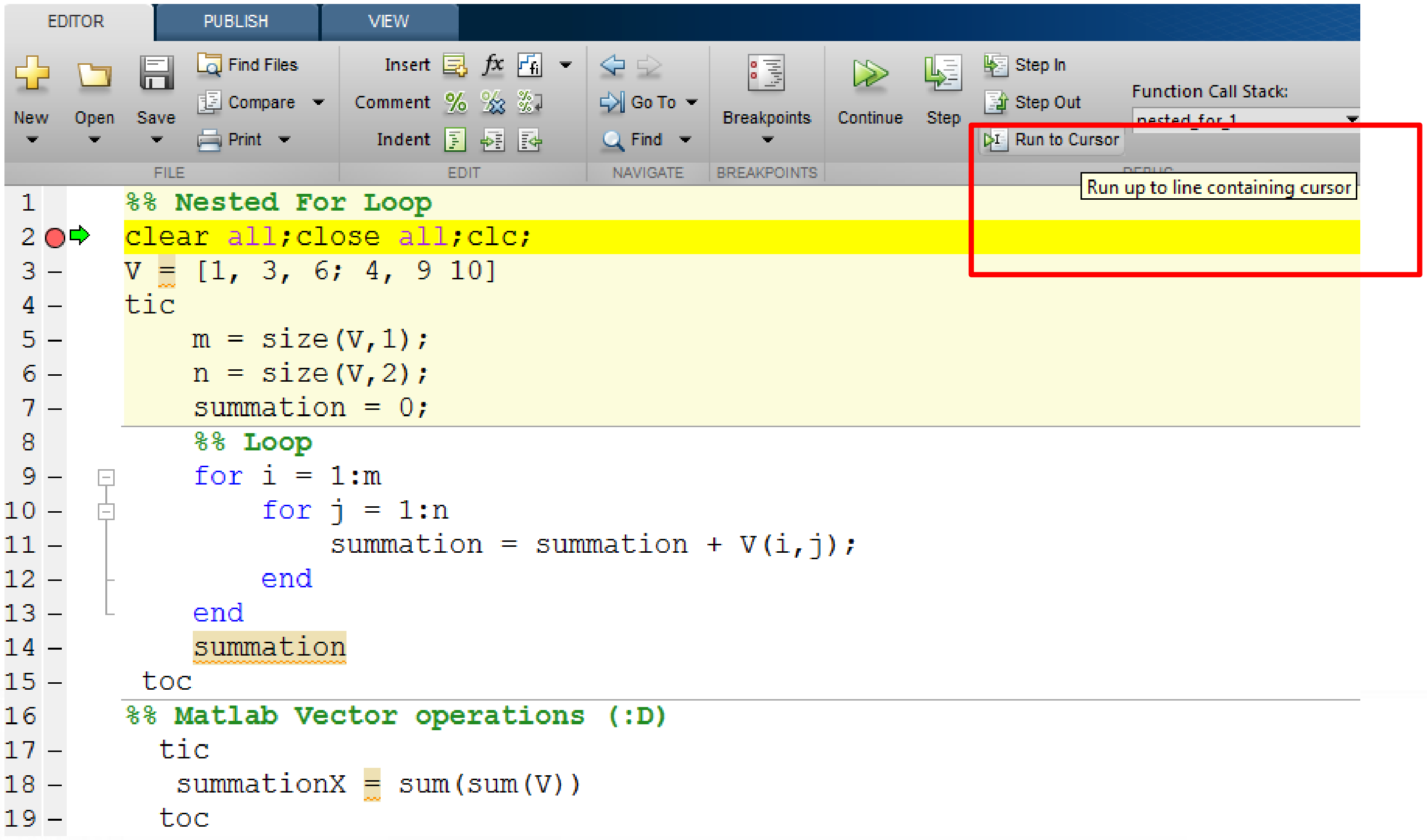


```
    N = 100;
    for n=-N:N
        if (n ~= 0)
                                  ation + (n+1)/n;

        end
    end
    disp('The Summation is')
    disp(summation)
```

8.3.3 The Go Until Cursor Debugging Tool

- Suppose that we would like to run the entire for loop.
- To do this, left-click on the code in line 14.
- Go to the Menu ! Debug ! Go Until Cursor



```
1   %% Nested For Loop
2   clear all;close all;clc;
3   V = [1, 3, 6; 4, 9 10]
4   tic
5       m = size(V,1);
6       n = size(V,2);
7       summation = 0;
8       %% Loop
9       for i = 1:m
10          for j = 1:n
11              summation = summation + V(i,j);
12          end
13      end
14      summation
15   toc
16  %% Matlab Vector operations (:D)
17    tic
18     summationX = sum(sum(V))
19    toc
```

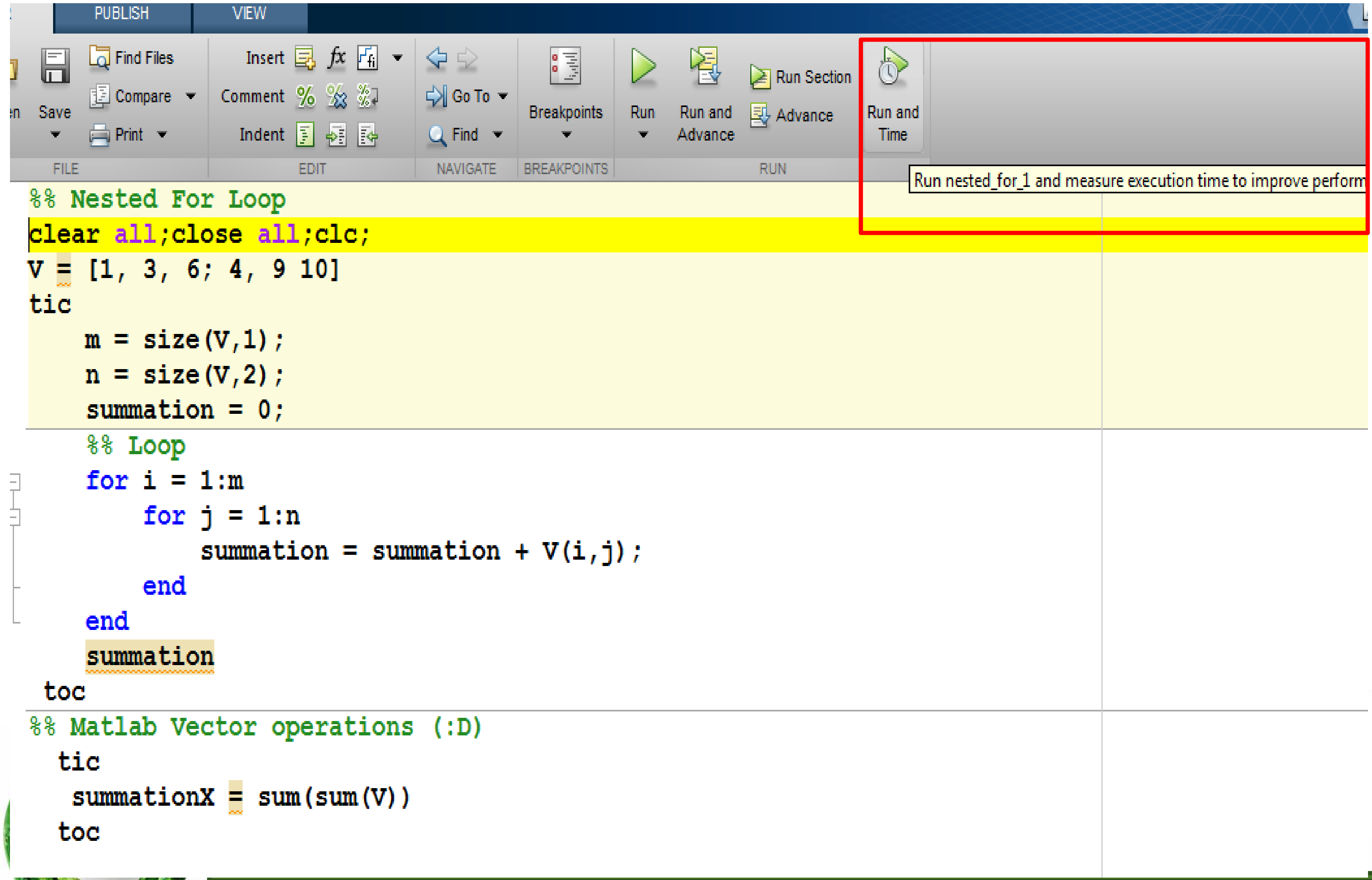
Lesson 8.4 The Matlab Profiler Tool

- It is very important to make sure that this code uses the lowest possible resources of a computer in terms of memory usage and computational power.
- We will discuss the use of Matlab tools for determining which particular sections in the code consume excessive amounts of the total execution time.
- This enables the possibility of optimizing these specific code sections.

- ✓ A Matlab program consists of a script M-file that may call a number of different Matlab functions.
- ✓ We want to find out the time that is required to execute the script file and all the functions that it calls.

Matlab contains a tool that provides us with this information, called the **Profiler**

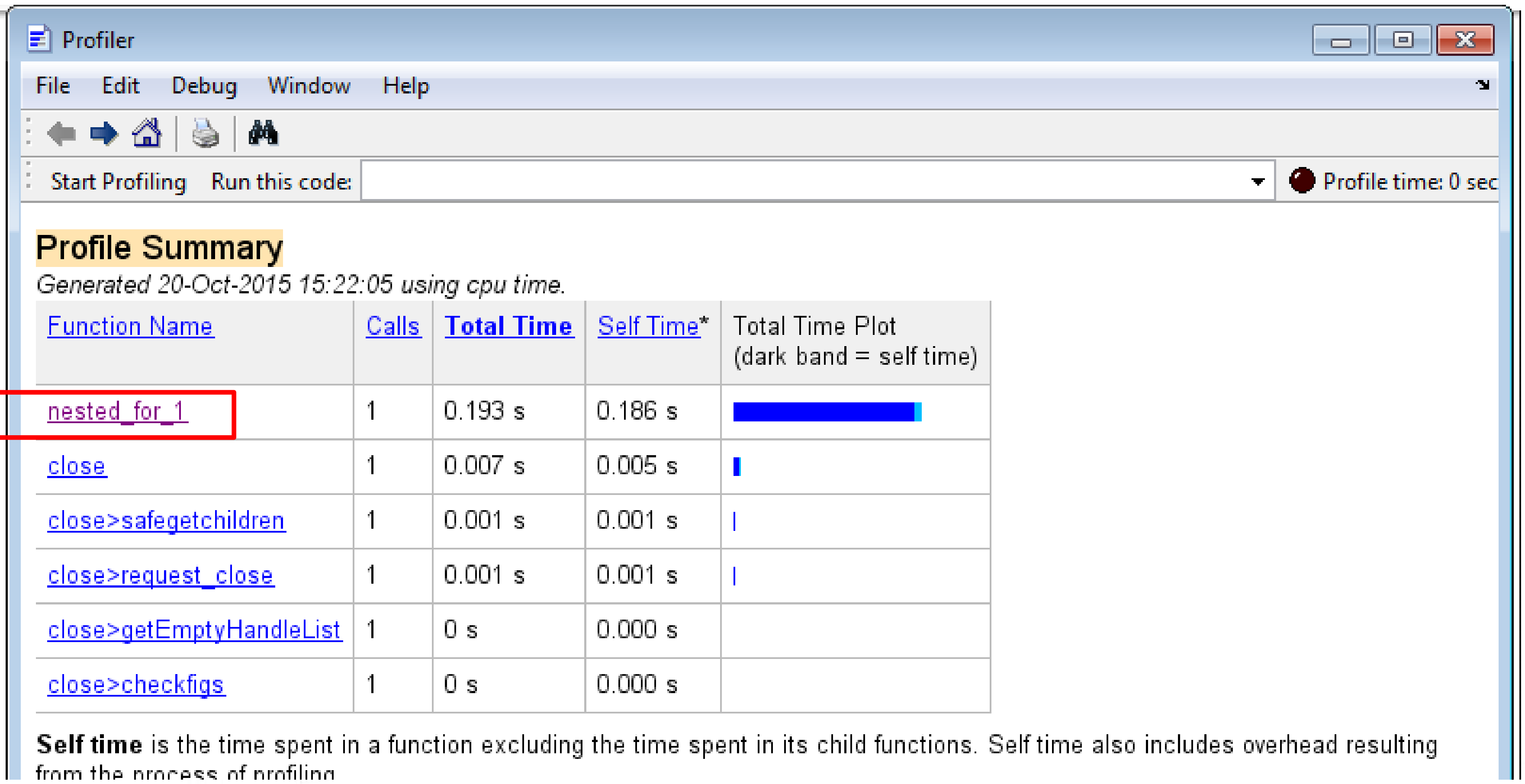
8.4.1.1 Launching the Matlab Profiler



Run nested_for_1 and measure execution time to improve perform

```
%% Nested For Loop
clear all;close all;clc;
V = [1, 3, 6; 4, 9 10]
tic
    m = size(V,1);
    n = size(V,2);
    summation = 0;
    %% Loop
    for i = 1:m
        for j = 1:n
            summation = summation + V(i,j);
        end
    end
    summation
 toc
%% Matlab Vector operations (:D)
  tic
   summationX = sum(sum(V))
  toc
```

8.4.1.1 Launching the Matlab Profiler



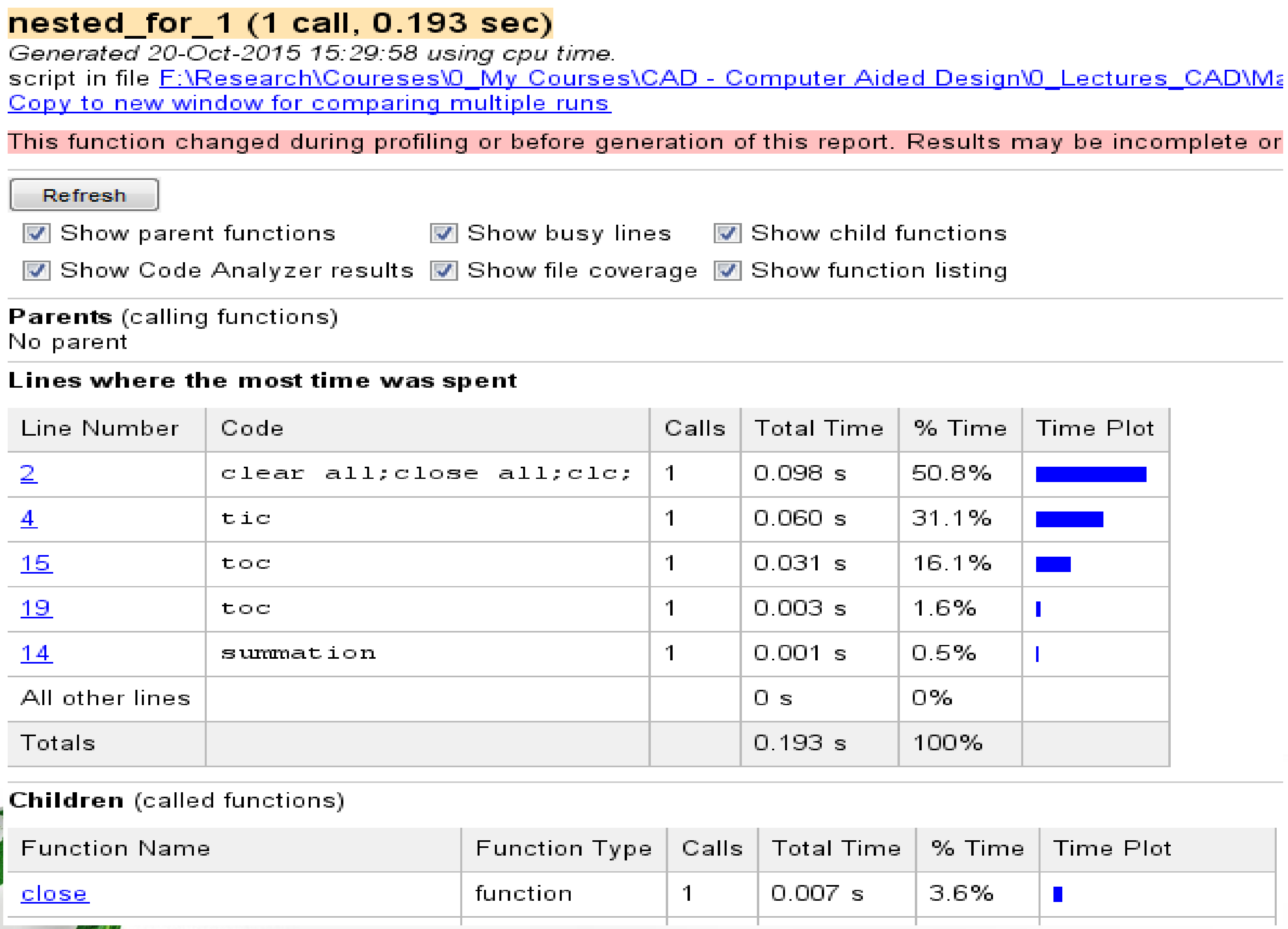
Profile Summary

Generated 20-Oct-2015 15:22:05 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
nested_for_1	1	0.193 s	0.186 s	
close	1	0.007 s	0.005 s	
close>safegetchildren	1	0.001 s	0.001 s	
close>request_close	1	0.001 s	0.001 s	
close>getEmptyHandleList	1	0 s	0.000 s	
close>checkfigs	1	0 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling

8.4.1.1 Launching the Matlab Profiler



nested_for_1 (1 call, 0.193 sec)

Generated 20-Oct-2015 15:29:58 using cpu time.

script in file F:\Research\Coureses\0_My Courses\CAD - Computer Aided Design\0_Lectures_CAD\Ma

Copy to new window for comparing multiple runs

This function changed during profiling or before generation of this report. Results may be incomplete or

Refresh

✓ Show parent functions ✓ Show busy lines ✓ Show child functions

✓ Show Code Analyzer results ✓ Show file coverage ✓ Show function listing

Parents (calling functions)

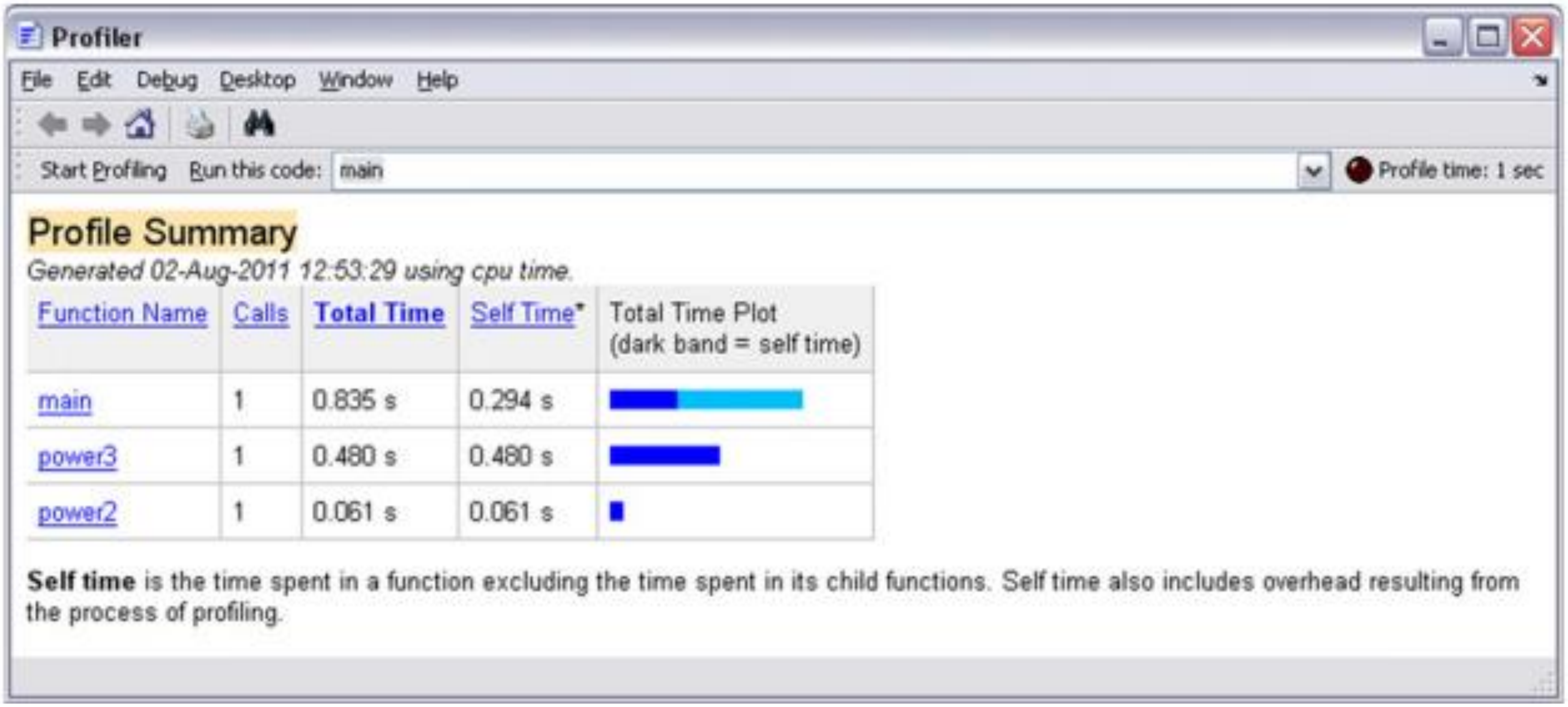
No parent

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
2	`clear all;close all;clc;`	1	0.098 s	50.8%	▬▬▬▬▬
4	`tic`	1	0.060 s	31.1%	▬▬▬
15	`toc`	1	0.031 s	16.1%	▬
19	`toc`	1	0.003 s	1.6%	▮
14	`summation`	1	0.001 s	0.5%	▮
All other lines			0 s	0%	
Totals			0.193 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
close	function	1	0.007 s	3.6%	▮


Profiler
File Edit Debug Desktop Window Help
Start Profiling Run this code: main
Profile time: 1 sec

Profile Summary

Generated 02-Aug-2011 12:53:29 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
main	1	0.835 s	0.294 s	
power3	1	0.480 s	0.480 s	
power2	1	0.061 s	0.061 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

The time required to execute the entire main program is 0.835 seconds.

The time required to execute the main program, but excluding the functions, is 0.294 seconds and this is indicated by the color 🟦.
The time required to call the functions themselves is 0.5410 seconds ($0.835 - 0.294 = 0.5410$) and this is indicated by the color 🟦.
The time required to call the function `power3` is 0.480 seconds.
The time required to call the function `power2` is 0.061 seconds.

9 Structures in Matlab

- You can use the vectors and arrays to save a collection of identical classes.
- Matlab support "Structure" type to support the collection of identical classes

```
circle.radius =   5;
circle.center =   [1,2];
circle.color =    'red';
```

- To display the contents of the circle variable, type :

```
>>circle
```

Matlab responds with

```
circle =
            radius:5
            center:[1 2]
            color:'red'
```

```
>>whos circle
```

Matlab responds with

```
Name     Size   Bytes   Class    Attributes
circle   1×1    402     struct
```

Matlab requires more memory to save a structure variable than would be the case for saving multiple individual variables.

Lesson 9.2 A Vector of Structures

- You can use Matlab to create a vector of structures.
- Let us explain this concept to you by way of an example.

Example 1

Three students study in a college. The following table shows their ages and marks in three subjects. Use a vector of structures to represent the students' information as given.

Student Name	Age	Math Mark	Physics Mark	English Mark
Alex	25	70	55	58
John	23	77	90	75
Mike	24	80	64	87

Answer

```
student(1).name = 'Alex';
student(1).age = 25;
student(1).math = 70;
student(1).physics = 55;
student(1).English = 58;
student(2).name = 'John';
student(2).age = 23;
student(2).math = 77;
student(2).physics = 90;
student(2).English = 75;
```

```
student(3).name = 'Mike';
student(3).age = 24;
student(3).math = 80;
student(3).physics = 64;
student(3).English = 87;
```

To display the fields of the vector, type at the Matlab **Command Prompt**

```
>>student
```

Matlab responds with

```
student =
1 × 3 struct array with fields:
 name
 age
 math
 physics
 English
```

- To display the contents of the first structure in the vector, type

```
>>student(1)
```

Matlab responds with

```
ans =
 name: 'Alex'
 age: 25
 math: 70
 physics: 55
 English: 58
```

- To display Alex’s mark in mathematics:

```
>>student(1).math
```

Matlab responds with

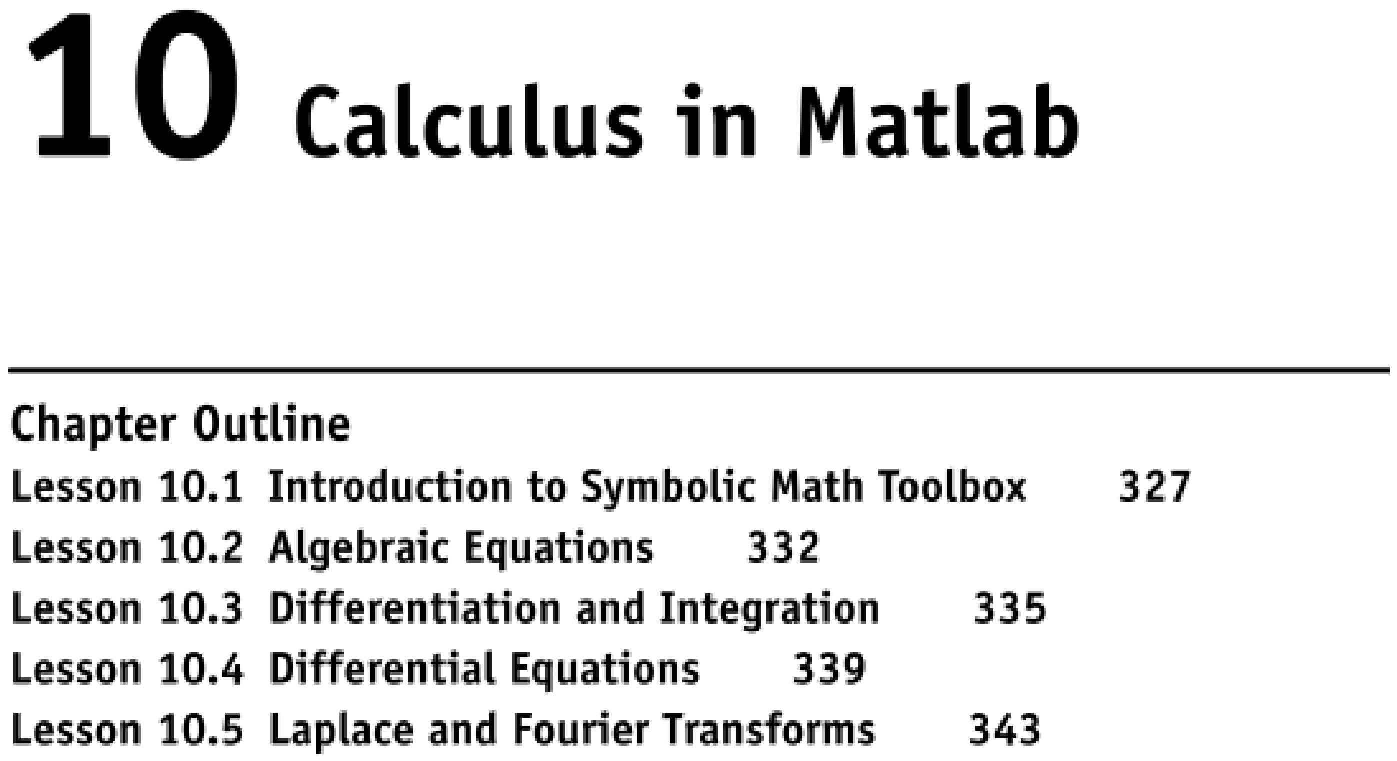
```
ans =
    70
```

- To display the mathematics marks for all the students in the vector, type

```
>>a=[student.math]
```

Matlab responds with

```
a =
       70    77    80
```



10 Calculus in Matlab

Chapter Outline

Self-Study