

MATLAB Tutorial 04

ENME 303 Computational Methods for Engineers

Parham Oveissi

Improving MATLAB Codes

- Use ; to suppress the outputs
- Use functions for tasks that are repeated
- Make your code as modular as possible
- Use MATLAB vectorization techniques
- Use preallocation
- Use MATLAB Code Analyzer Report tool to improve your code

Let's Improve this code

```
clc; clear
num_elems = 5000;

for i = 1:num_elems
    for j = 1:num_elems

        A(i,j) = rand

        if A(i,j) > 0.5
            B(i,j) = A(i,j)
        else
            B(i,j) = -A(i,j)
        end
    end
end
```

Note:
use tic; toc; to get the
run time of your code.

Step1: Use ; to Suppress the Outputs

```
clc; clear
num_elemsnts = 5000;

for i = 1:num_elemsnts
    for j = 1:num_elemsnts

        A(i,j) = rand;

        if A(i,j) > 0.5
            B(i,j) = A(i,j);
        else
            B(i,j) = -A(i,j);
        end
    end
end
```

Step2: Use Preallocation

```
clc; clear
num_elems = 5000;

A = zeros(num_elems);
B = zeros(num_elems);

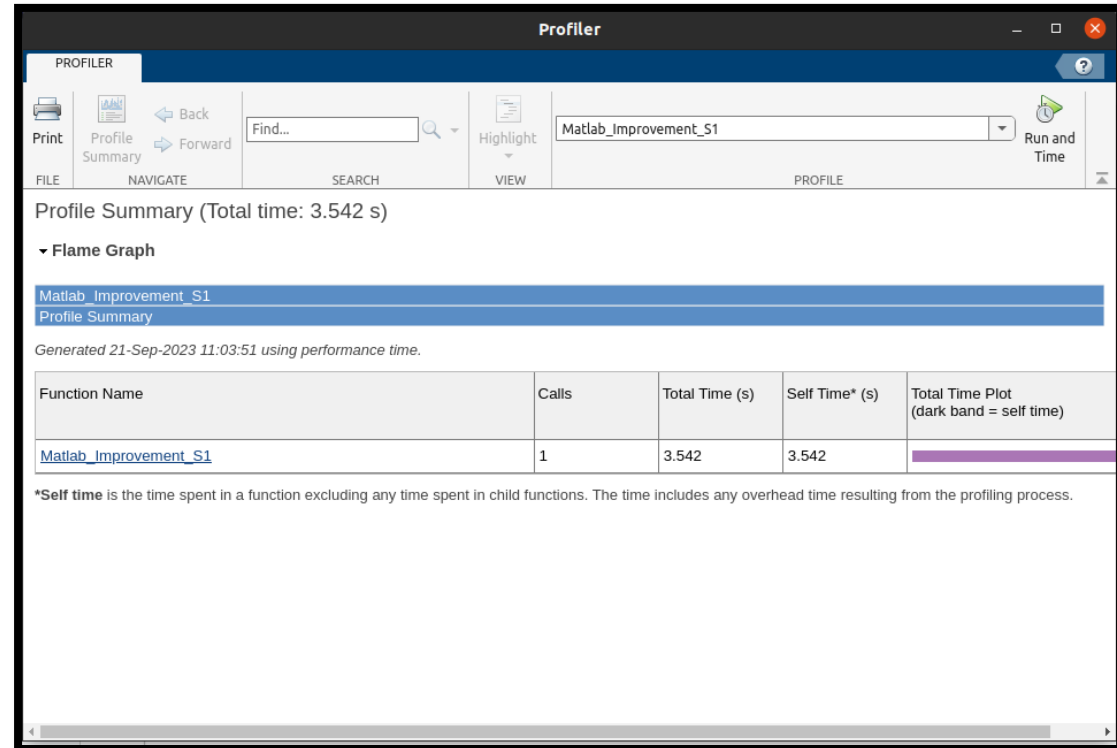
for i = 1:num_elems
    for j = 1:num_elems

        A(i,j) = rand;

        if A(i,j) > 0.5
            B(i,j) = A(i,j);
        else
            B(i,j) = -A(i,j);
        end
    end
end
```

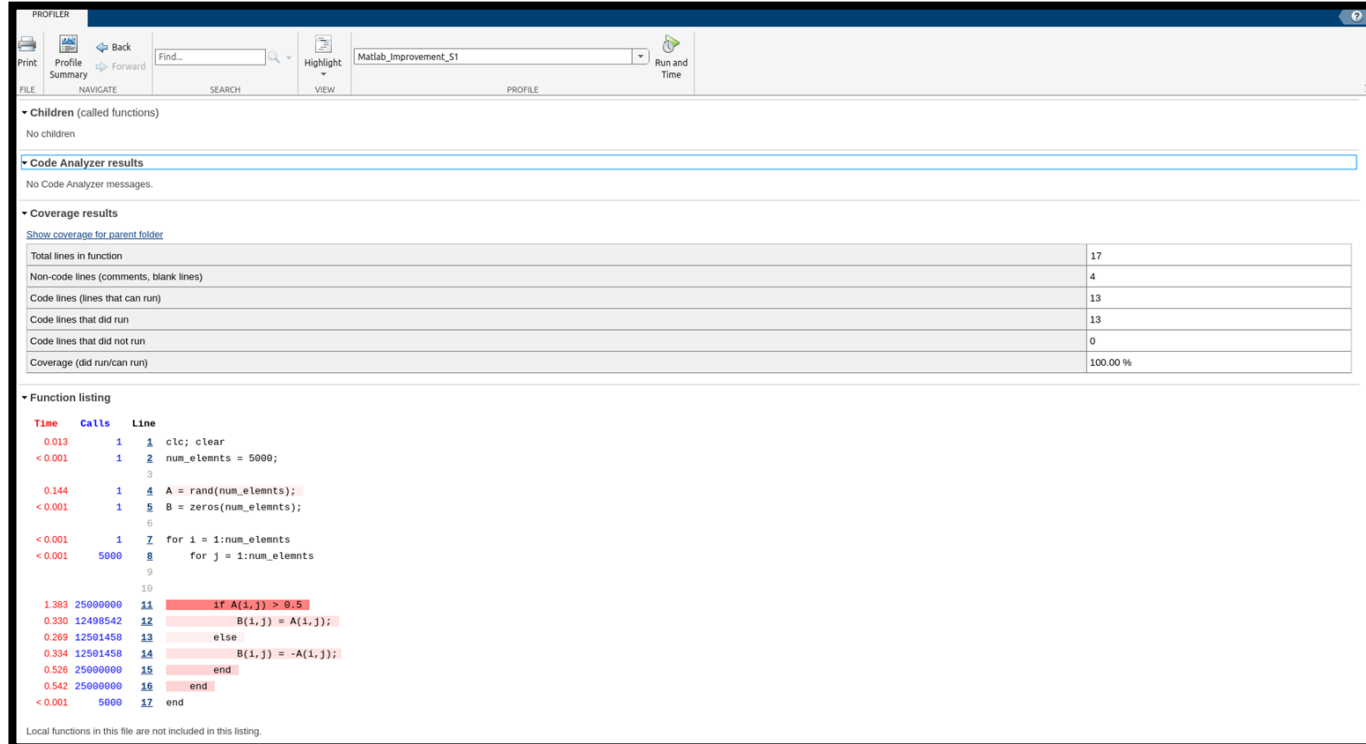
Profiler

- Use MATLAB Profiler tool to get the run time.
- Click on the name of the function to see more details about your code and the Code Analyzer tool messages.



Profiler

- Use MATLAB Profiler tool to get the run time.
- Click on the name of the function to see more details about your code and the Code Analyzer tool messages.



The screenshot shows the MATLAB Profiler interface for a function named 'Matlab_Improvement_51'. The interface includes a toolbar with options like Print, Profile, Summary, Back, Forward, Find, Highlight, and Run and Time. The main content area is divided into several sections:

- Children (called functions):** No children.
- Code Analyzer results:** No Code Analyzer messages.
- Coverage results:** A table showing coverage statistics.

Show coverage for parent folder	
Total lines in function	17
Non-code lines (comments, blank lines)	4
Code lines (lines that can run)	13
Code lines that did run	13
Code lines that did not run	0
Coverage (did run/can run)	100.00 %
- Function listing:** A table showing the execution time and call count for each line of code.

Time	Calls	Line
0.013	1	1 cld; clear
< 0.001	1	2 num_elems = 5000;
0.144	1	4 A = rand(num_elems);
< 0.001	1	5 B = zeros(num_elems);
< 0.001	1	7 for i = 1:num_elems
< 0.001	5000	8 for j = 1:num_elems
1.383	25000000	11 if A(i,j) > 0.5
0.330	12498542	12 B(i,j) = A(i,j);
0.269	12501458	13 else
0.334	12501458	14 B(i,j) = -A(i,j);
0.526	25000000	15 end
0.542	25000000	16 end
< 0.001	5000	17 end

Local functions in this file are not included in this listing.

Step3: Use MATLAB Vectorization Techniques

```
clc; clear
num_elems = 5000;

A = rand(num_elems);
B = zeros(num_elems);

for i = 1:num_elems
    for j = 1:num_elems

        if A(i,j) > 0.5
            B(i,j) = A(i,j);
        else
            B(i,j) = -A(i,j);
        end
    end
end
```


Step4: Use MATLAB Logical Indexing

```
clc; clear  
num_elems = 5000;  
  
A = rand(num_elems);  
  
B = A;  
  
index = A < 0.5;  
  
B(index) = -A(index);
```

fsolve

Nonlinear system solver

Solves a problem specified by

$$F(x) = 0$$

for x , where $F(x)$ is a function that returns a vector value.

```
clc; clear

f1 = @(x)x^2 + 3*x + 2;

x0 = 5;

options = optimoptions ('fsolve','Display','iter') ;
[x , fval] = fsolve (f1, x0, options);
```

Thanks!