APiE Exercise – Debugging

The purpose of this example sheet is to learn to debug and optimise MATLAB code. In the following sheet you will find three codes with bugs and non-optimised style for MATLAB programming. In each case fix any bugs you find and for higher marks also increase the execution speed. In the script please list all bugs you found and optimisations made. In the case of the optimisation use `tic toc` to indicate the speed up achieved.

Exercise 1 - Bubble Sort

Let's suppose someone asks you to put $[8, 7, 9, 5, 4]$ in an ascending order.

Starting from the first number, you compare the number with the next number, and see if it is greater. If it is, you swap the numbers. You continue to do this with the second number, third number and so on until the second last number. What this does is to bubble the largest number to the end.

$[8, 7, 9, 5, 4] \rightarrow [7, 8, 9, 5, 4]$ (as $8 > 7$) $\rightarrow [7, 8, 9, 5, 4]$ (as 8 is not > 9) $\rightarrow [7, 8, 5, 9, 4]$ (as 9 is > 5) $\rightarrow [7, 8, 5, 4, 9]$(as $9 > 4$). See how the largest number is at the end.

Now repeat this. $[7, 8, 5, 4, 9] \rightarrow [7, 8, 5, 4, 9]$ (as 7 is not > 8] $\rightarrow [7, 5, 8, 4, 9]$ (as $8 > 5$) $\rightarrow [7, 5, 4, 8, 9]$ (as $8 > 4$) $\rightarrow [7, 5, 4, 8, 9]$ (as 8 is not > 9)

Now repeat this. $[7, 5, 4, 8, 9] \rightarrow [5, 7, 4, 8, 9] \rightarrow [5, 4, 7, 8, 9] \rightarrow [5, 4, 7, 8, 9] \rightarrow [5, 4, 7, 8, 9]$

Now repeat this. $[5, 4, 7, 8, 9] \rightarrow [4, 5, 7, 8, 9] \rightarrow [4, 5, 7, 8, 9] \rightarrow [4, 5, 7, 8, 9] \rightarrow [4, 5, 7, 8, 9]$

It looks like we are done. If n is the number of the numbers in the array, it takes (n-1) swaps within each of the (n-1) repetitions. So we do not have to guess how many swaps it takes or how many repetitions it takes.

To make the bubble sort efficient, we can do the following: 1) Since with each repetition the largest number bubbles up, we may need to do less swaps. For the first repetition, we will do (n-1) swaps, for the next repetition, we do the first (n-2) swaps, and so on. 2) We can also keep track of number of swaps taking place in a repetition. If no swaps take place in a repetition, no more repetitions are needed.

The question is to debug a code which uses this method to sort an array of 100 random generated numbers.

```
function Q1
tic;
A=rand(100,1);
B=BubbleSort(A);
disp(B);
toc

function B=BubbleSort(A)
% SORTING AN ARRAY OF NUMBERS IN AN ASCENDING ORDER USING BUBBLE SORT
% Example from PiE course
% Last Revised : October 18, 2010
disp('This program shows the bubble sort method')
disp('to sort an array of integers into ascending order ')
disp('It works by comparing and swapping until the list is sorted')
disp('  ')
disp('Size of given array is:')
disp(length(A))
n=length(A);
% making (n-1) passes
bound=n+1;
while(bound)
    last_swap=0;
    % comparing each number with the next and swapping
    for i=1:1:bound
        t=A(i);
        if(t>A(i+1))
            A(1)=A(i+1);
            A(i+1)=t;
            last_swap=i;
        end
    end
    bound=last_swap;
end%end while
toc;
disp('  ')
disp ('Sorting finished ...')
B=A;
end
```

Exercise 2 - Fibonacci numbers

Fibonacci number are defined by the sequence

$$F_0 = 0 \tag{1}$$
$$F_1 = 1 \tag{2}$$
$$F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 2 \tag{3}$$

The purpose of this exercise is to compute the square of 1000 Fibonacci numbers' i.e. $G_n = (F_n)^2$ and find the sum of G_n and the number of odd numbers in G_n.

Hint : The answer is: G_n has 52 odd numbers and the sum is $1.1380e^{209}$.

```
function Q2

%% calculation the Fibonacci numbers
fib(1)=1;
fib(2)=1;
for i=1:1000
    fib(i)=fib(i-1)+fib(i-2);
end

%% calculation the square of the fibonacci numbers
for i=1:100
fib2(i)=fib(i)*fib(i);
end


%% calculation the sum and the number of odd numbers
sum=0;
count=0;
for i=1:1000
    sum=fib(i);
    if (rem(fib(i),2)=1)
        count=count+1;
    end
end

disp(count);
disp(sum);

end
```

Exercise 3 - Plotting a saddle

Consider a function of two variables $f(x, y)$. A 2D saddle is a point, in mathematics, which is a minimum in one direction and a maximum in the other direction. It gets its name because it looks like a horses saddle. The simplest 2D function with such a property is $z = y^2 - x^2$, which has a saddle at $(0, 0)$. The following MATLAB code should plot a surface plot of this shape and place arrows indicating the direction on the gradient at a few points on the surface.

```
function Q3

close all


%% setup the vectors
x=linspace(-5,5,100)
y=linspace(-5,5,100)


%% compute the function
z=y*y - x*x

%% plot the surface plot
surf(x,y,z)
shading flat;
alpha(0.5);

%%Compute the gradient
[fx,fy]=gradient(z);

%% For every 10th point plot the gradient
hold on;
x=x(1:10:100)
y=y(1:1:100)
z=z(1:10:100)
fx=fx(1:10:100)
fy=fy(1:10:100)
quiver3(x,y,z,fx,fy);
```