Numerical Analysis

LAB MID

SUBMITTED TO:
SIR HASEEB

SUBMITTED BY:
M QASIM NAUMAN - 221345
CLASS: BSCS-5(B)

DEPARTMENT OF COMPUTER SCIENCE
AIR UNIVERSITY ISLAMABAD

Question 1:

PART 1 – Input 5 number, calculate the sum of the highest 3

```
% Part 1
numbers = input('Enter five numbers as an array in the following format [1,2,3,4,5]: ');

if length(numbers) ~= 5
    disp('Please enter exactly five numbers.');
else
    s_number = sort(numbers, 'descend');

    avg_high_3 = mean(s_number(1:3));

    disp(['The entered number are : ', num2str(numbers)]);
    disp(['Numbers sorted in descending order are : ', num2str(s_number)]);
    disp(['Avergae of the largest 3 numbers : ', num2str(avg_high_3)]);
end
```

Output:

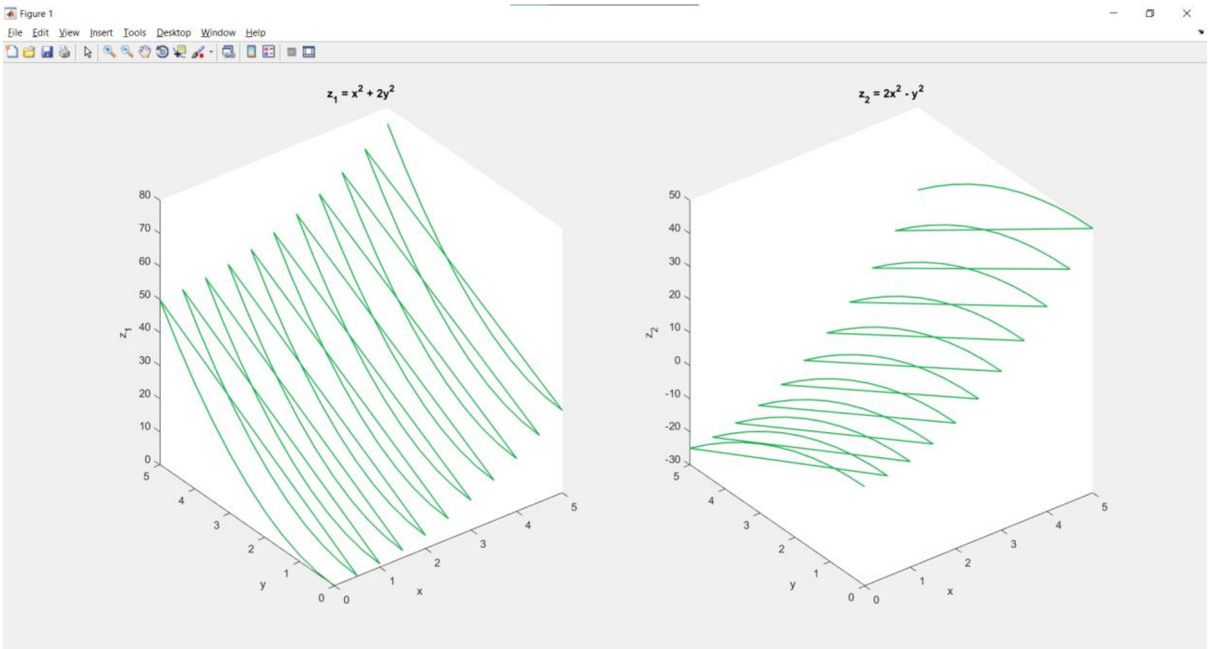
```
Enter five numbers as an array in the following format [1,2,3,4,5]: [2,4,6,1,7]
The entered number are : 2   4   6   1   7
Numbers sorted in descending order are : 7   6   4   2   1
Avergae of the largest 3 numbers : 5.6667
```

PART 2 – 3D graph of the given functions

```
% Part 2
x = 0:0.5:5;
y = 0:0.5:5;

z1_values = [];
z2_values = [];
x_values = [];
y_values = [];

for i = 1:length(x)
    for j = 1:length(y)
        x_values(end + 1) = x(i);
        y_values(end + 1) = y(j);
        z1_values(end + 1) = x(i)^2 + 2 * y(j)^2;
        z2_values(end + 1) = 2 * x(i)^2 - y(j)^2;
    end
end

figure;

subplot(1, 2, 1);
plot3(x_values, y_values, z1_values, 'LineWidth', 1, 'Color', [0.1, 0.7, 0.3]);
title('z_1 = x^2 + 2y^2');
xlabel('x');
ylabel('y');
zlabel('z_1');

subplot(1, 2, 2);
plot3(x_values, y_values, z2_values, 'LineWidth', 1, 'Color', [0.1, 0.7, 0.3]);
title('z_2 = 2x^2 - y^2');
```

```
subplot(1, 2, 2);
plot3(x_values, y_values, z2_values, 'LineWidth', 1, 'Color', [0.1, 0.7, 0.3]);
title('z_2 = 2x^2 - y^2');
xlabel('x');
ylabel('y');
zlabel('z_2');
```

Output:



Question 2:

Algorithm:

- The first step is to give the input which is always a continuous function along with the derivate of it. Lets say the function as F(x) and F'(x) as the derivate
- Next we initialize the values of x0 and x1 which can be given and if not we start it from zero
- Next we calculate the approximation using the formula
  x(n+1) = x(n) – f(x(n)/f'(x(n))
- Next we check the convergence, if | x(n+1) – x(n) | < tol then we assume the x(n+1) as the root;
- We repeat until the difference is smaller than tolerance.
- Display the output

Code to get the answer of given function

```
f = @(x) exp(x) - 3*x^2;
df = @(x) exp(x) - 6*x;

x0 = 0.5;
tol = 1e-6;
max_iter = 5;
x1=0;

for i = 1:max_iter
    x1 = x0 - (f(x0) / df(x0));

    disp(['Iteration : ',num2str(i),': x= ',num2str(x1)])

    if abs(x1 - x0) < tol
        break;
    end

    x0 = x1;
end
root=x1;
disp(['Approximate Root (Newton-Raphson): ', root]);
```

Output:

```
>> q2
Iteration : 1: x= 1.1651
Iteration : 2: x= 0.93623
Iteration : 3: x= 0.9104
Iteration : 4: x= 0.91001
Iteration : 5: x= 0.91001
Approximate Root (Newton-Raphson):
```

Question 3:

For Loop to print the sum of even number is range of 1 to 20

```
sum = 0;

for i = 1:19
    if ( mod(i, 2) == 0 )
        sum = sum+i;
    end
end

disp(['Sum of even number between 1 and 20 is : ', num2str(sum)]);
```

Output:

```
>> q3
Sum of even number between 1 and 20 is : 90
```

While loop to print the square of numbers below 50

```
square = 0;
number = 1;
while(square < 49)
    square = number*number;
    disp(['Square of ', num2str(number), ' = ', num2str(square)]);
    number = number + 1;
end
```

Output:

```
Square of 1 = 1
Square of 2 = 4
Square of 3 = 9
Square of 4 = 16
Square of 5 = 25
Square of 6 = 36
Square of 7 = 49
```

Loop breaking on a condition

```
disp('This program exits the for loop when the total sum of number becomes greater than 40');
sum1 = 0;
for i = 1:19
    if ( mod(i, 2) == 0 )
        sum1 = sum1+i;
        disp(['Sum is ', num2str(sum1)]);
    end
    if (sum1>40)
        break;
    end
end
```

Output:

```
This program exits the for loop when the total sum of number becomes greater than 40
Sum is 2
Sum is 6
Sum is 12
Sum is 20
Sum is 30
Sum is 42
```

Question 4:

Code using Newton Forward Method

```
x = [0.0 0.2 0.4 0.6 0.8];
y = [2.0 2.30 2.55 2.85 3.20];

n = length(y);

diff_table = zeros(n, n);
diff_table(:, 1) = y';

for j = 2:n
    for i = 1:n-j+1
        diff_table(i, j) = diff_table(i+1, j-1) - diff_table(i, j-1);
    end
end

disp('Forward Difference Table');
disp(diff_table);

xp = 0.5;
h = x(2) - x(1);
p = (xp - x(1)) / h;

yp = diff_table(1, 1);
term = 1;

for k = 2:n
    term = term * (p-(k-2)) / (k - 1);
    yp = yp + term * diff_table(1, k);
end

fprintf('Interpolated value at x = %.2f is y = %.4f\n', xp, yp);
```

Output:

```
>> q4
Forward Difference Table
    2.0000    0.3000   -0.0500    0.1000   -0.1000
    2.3000    0.2500    0.0500   -0.0000         0
    2.5500    0.3000    0.0500         0         0
    2.8500    0.3500         0         0         0
    3.2000         0         0         0         0

Interpolated value at x = 0.50 is y = 2.6914
```