**Numerical Methods HW-1**

**REPORT**

**Nilay Altun 150110224**

**Okan Gül 150110214**

**Q1.** For the first part of this assignment we are asked to implement 3 different methods of finding roots. These are Bisection method, Secant method and Newton-Raphson method. Also with this methods we are asked to plot graphs to see the efficiency of the methods.

**Bisection Method :**

In this method we are given a function and two initial condition(upper and lower limits) to find the root of the function. While finding the root we assume that the error is smaller than the 0.01.

For finding the result every step we first find the middle point of the limits and look for some conditions according to that conditions we change the upper or lower limit to narrow down our scope.

Every division process and changing the limits is our one iteration. We are using these iteration number for plotting our graph. Also the other variable of our graph is errors of each step.

Goal error : 0.01

First lower limit : -4

First upper limit : -1

Iteration Number : 9

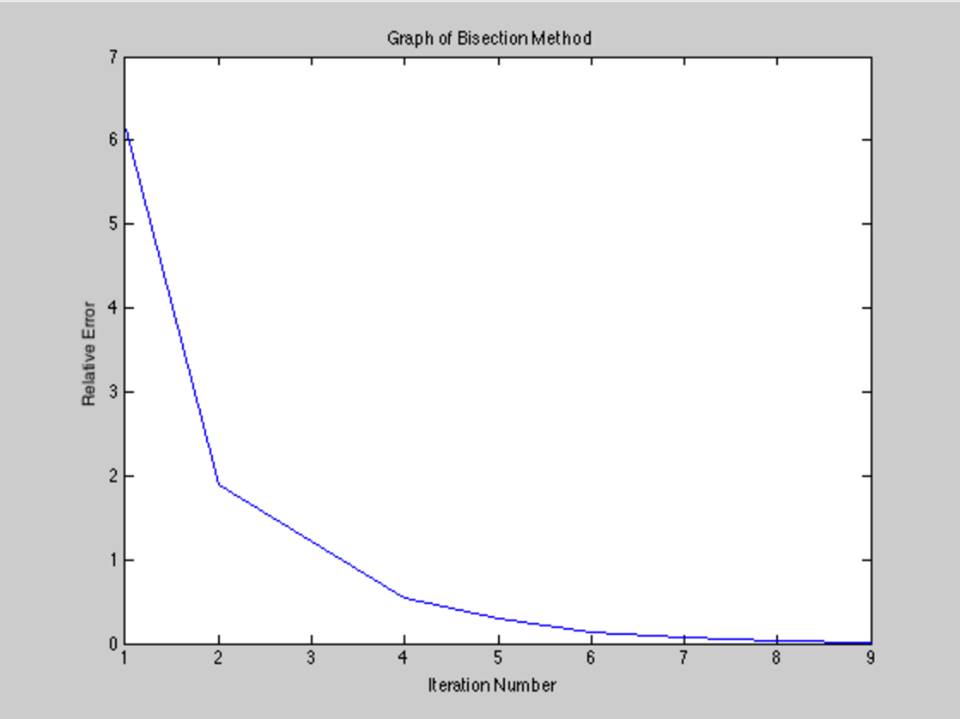
The root is : -1.9990

Final error is : 0.0176

X label of graph: İteration number , Y label of graph: Relative error. Every breaking point on the graph represents error at that iteration.

Advantage of this method, function root always converges and the root scope always increases in half.

Disadvantage of this method, the convergence is slow. If first guesses are too close to root than the convergence going to be very slow. It will effect performance.



**Secant Method:**

In this method again we are given a function and two initial guesses. With this information we are changing the gueeses by using a secant formula. Every step also we find relative error by using secant formula of error to reach our goal error of 0.01.

When we find our goal error our program breaks the for loop for printing the root. Also for plotting our graph we store relative errors of every step in a vector. With this information we calculate errors of each step.

Goal error : 0.01

First guess : -4

Second guess is : -4.1

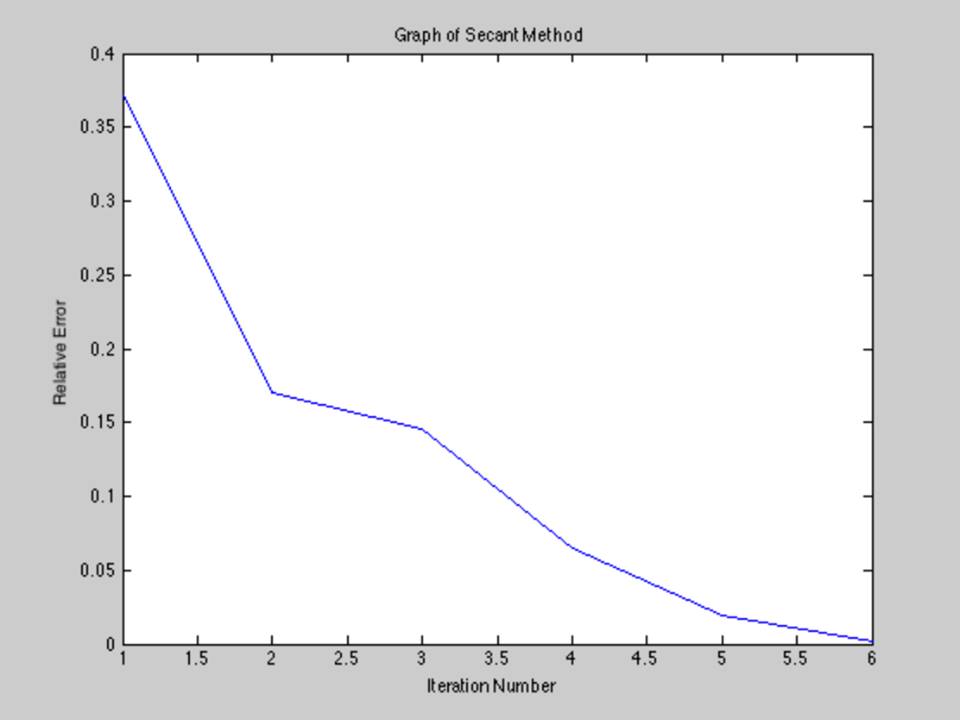
Iterations Number : 6

The root is : -2.0001

Final error is : 0.0022

X label of graph: Iteration number , Y label of graph: Relative error. Every breaking point on the graph represents error at that iteration.

Advantage of this method, we just need two guesses and don’t need to change variables every step like Bisection Method. Also we don’t need scope for root. It converges faster than Bisection Method if it converges. We can see that in the graph, iteration number is 6. So we can say that Secant Method is faster than Bisection Method.



**Newton-Raphson Method:**

In this method again we are given a specific function to work with but the difference is with the Newton- Raphson formula we need to find the derivative of the given function. To provide this information we calculate the derivative of the function.

When we find our goal error our program breaks the for loop for printing the root. Also for plotting our graph we store relative errors of every step in a vector. With this information we calculate errors of each step. Working method is similar to Secant Method but in this method we just need one condition.

Goal error : 0.01

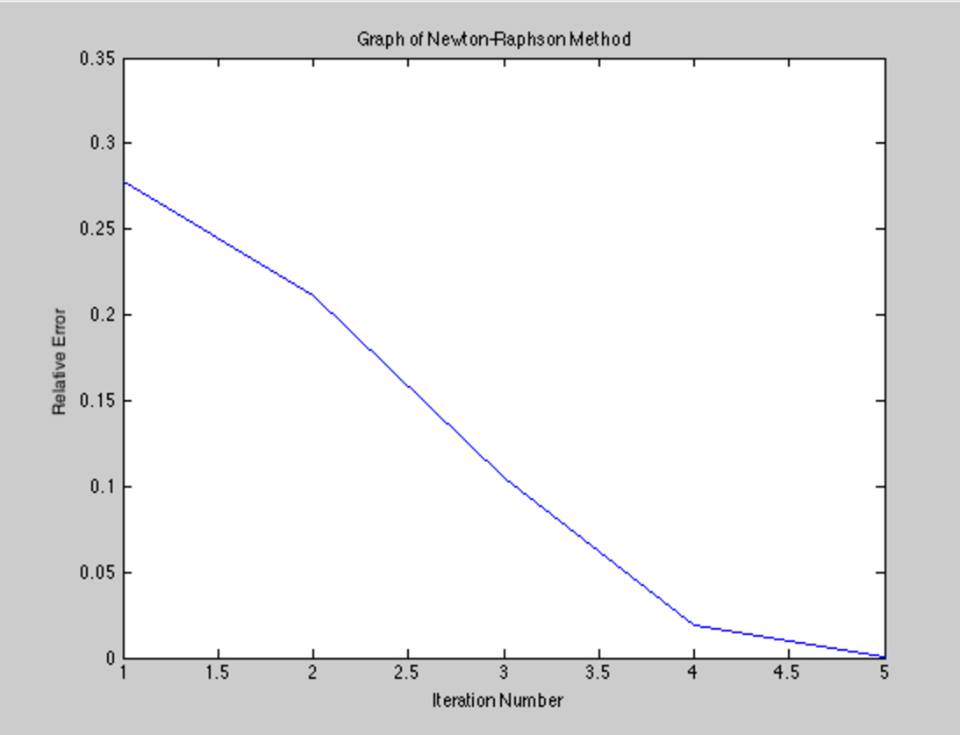
First Guess : -4

Iterations Number : 5

The root is : -2.0011

Final error is : 0.0005

Advantage of this method, like Secant Method it also converges fast. The best way of this method, it’s only gets one guess.



**Conclusion:**

Like we can see every graph on these methods, in every iteration the relative error decreases until it reaches error under 0.01.

The faster method is: Newton-Raphson method, because the iteration number is slowest and also just takes one initial guess.

**Q2.**

**La Grange Method:**

Lagrange interpolating polynomial: formula of a parabola | matematicasVisuales

First, we used f(x) = sin(x) function for selecting numbers in a range of 0,2\*pi. We kept these variables an integer values. After, replaced these integer values in above formula and kept doing same procedures.

**For 3 points -** We need to calculate a function by using this formula. This formula is second degree and need to take 3 points for calculating the function. After than that we are changing ‘’x’’ variables for plotting. We gave that initial variables by using pi values. Such as x=3\*pi/2.

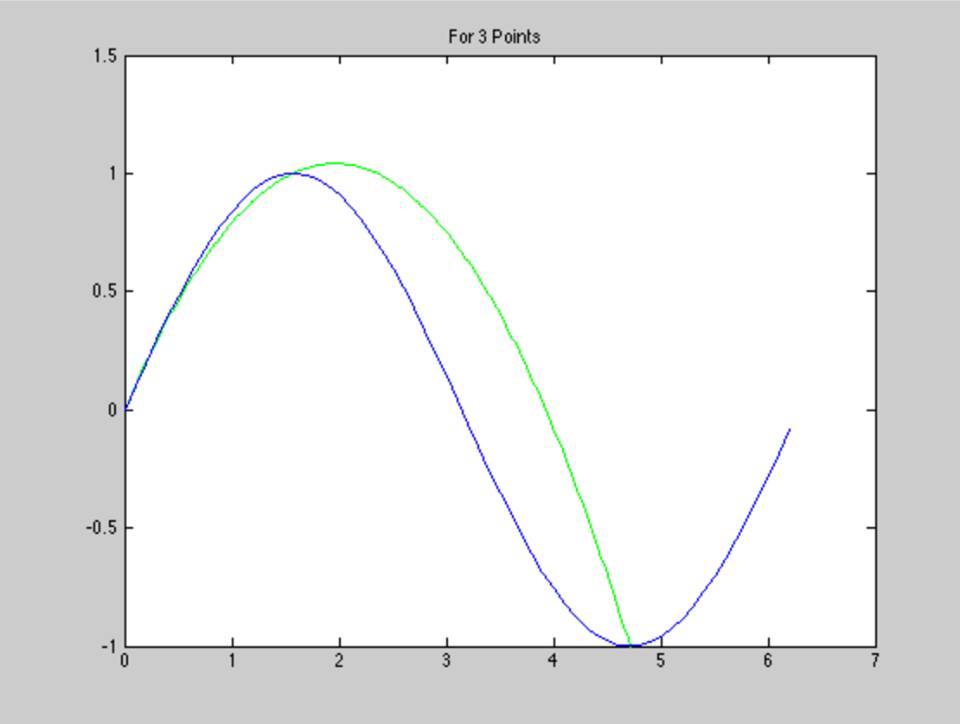
**For 5 points –** We again used the same formula for calculating the function of 5 points. If we do it another way, we need a formula for 4 degree and It will take more time to write that formula. So we grouped the 3 points in 3 way and reunite the groups for plotting the graph.

**For 9 points** – Like 5 points solving, we again grouped points 7 times by using 3 different point each time. After combining these functions and keeping the variables in an array, we found the graph.

**For 20 points –**In same way, we grouped points 18 times by using 3 different point each time. After combining these functions and keeping the variables in an array, we found the graph.

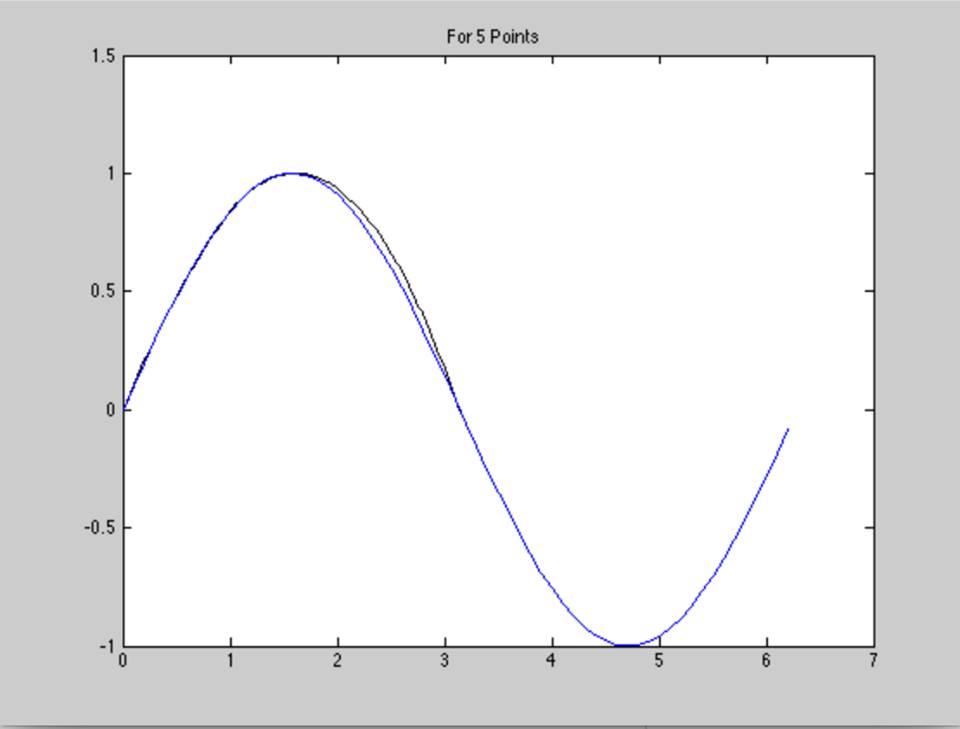
**Conclusion:**

We saw that, while increasing the numbers the function that we found, start to look like sin(x) function. In the plot that we gave 20 points, graphs are looking almost same with sin(x). We show that graphs separately to see more easily. And added the one figure all of them.

****

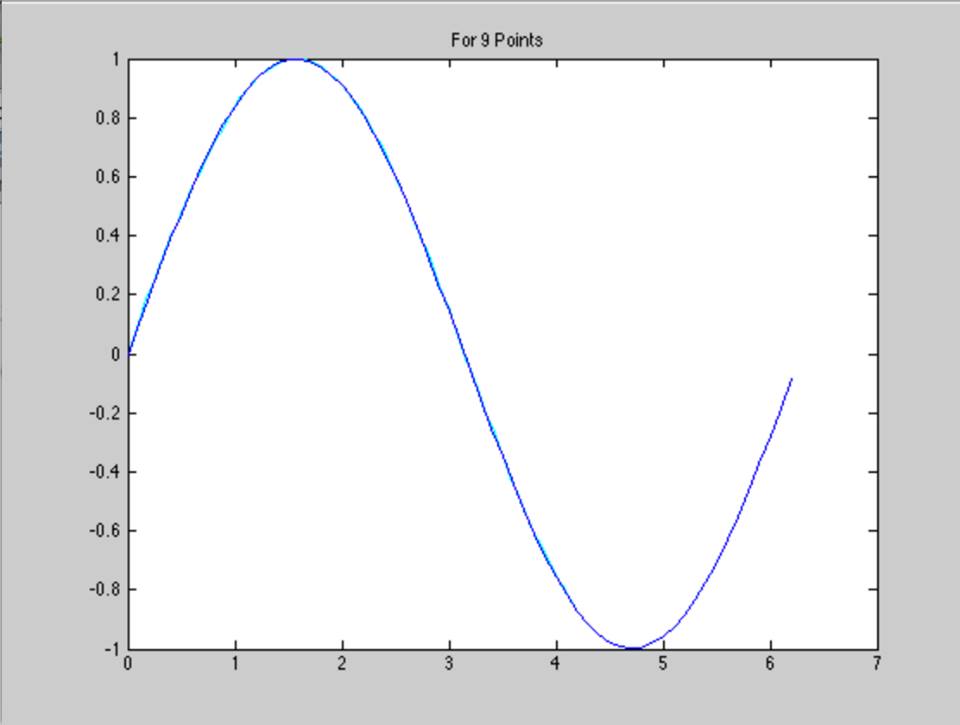
Blue-sin(x)

Green-Our function that we found in 3 points



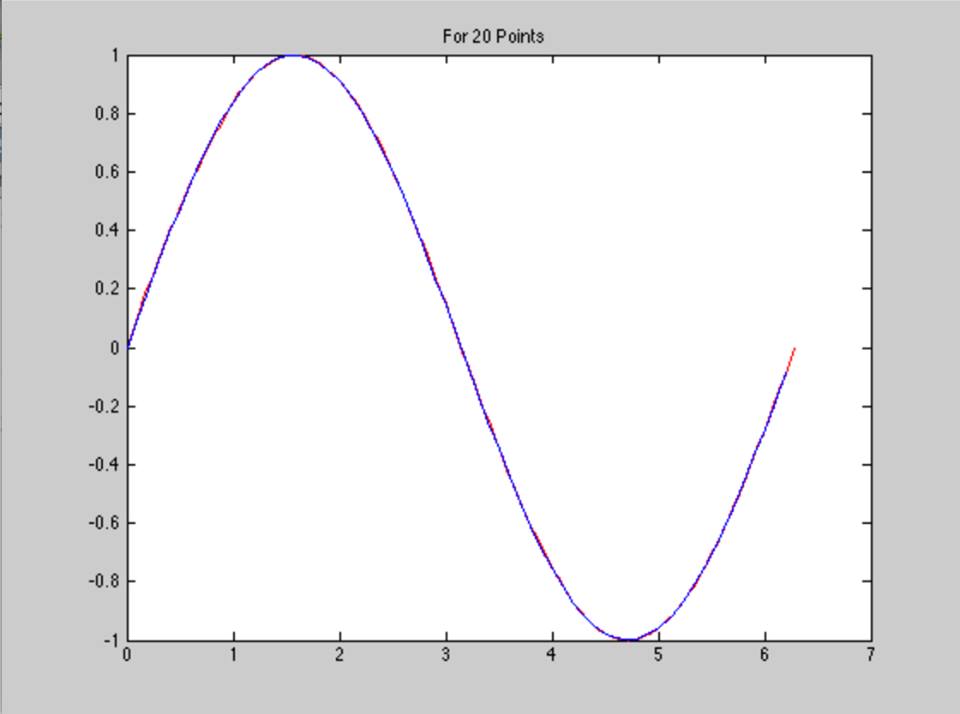
Blue-sin(x)

Black-Our function that we found in 5 points



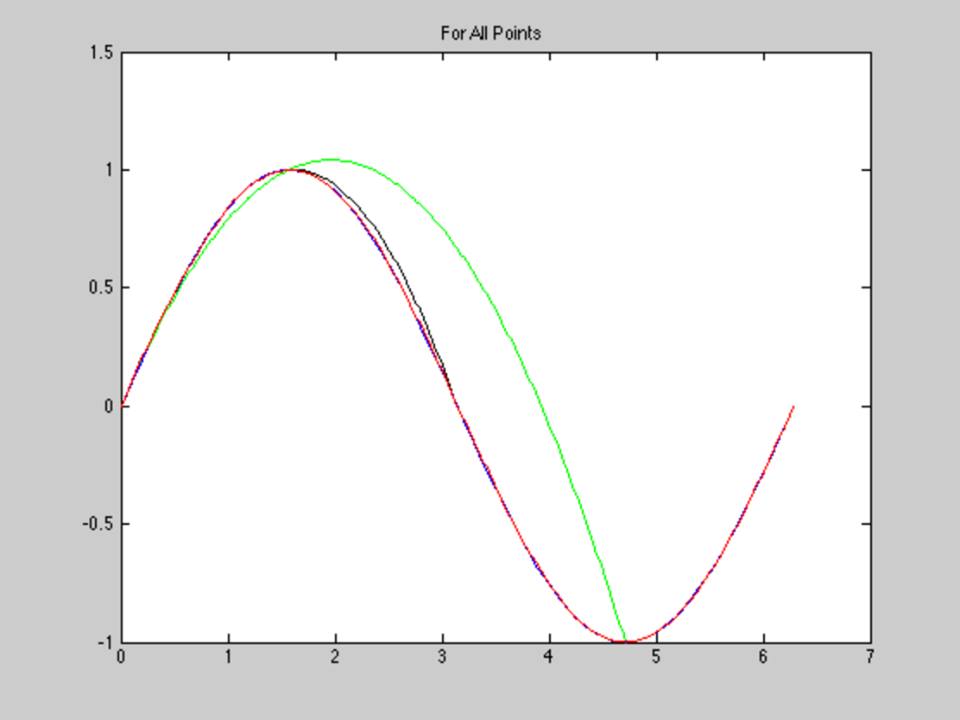
Blue-sin(x)

Cyan-Our function that we found in 9 points



Blue-sin(x)

Red-Our function that we found in 20 points



Blue-sin(x)

Green-Our function that we found in 3 points

Black-Our function that we found in 5 points

Cyan-Our function that we found in 9 points

Red-Our function that we found in 20 points