ALGO TESTER AND GRAPH PLOTTER

Project Report

An *Algo Tester and Graph Plotter* is a software developed using **Java Swing** technology to demonstrate the actual execution of various algorithms used in computer graphics and numerical methods. The application provides graphical representation of the execution of C.G. algorithms related to circle drawing, line drawing, various transformations

Another section of Numerical Method related algorithms includes, Interpolation and Extrapolation, Root Finding Algorithm and Simultaneous Linear Equation solving algorithms.

Developer:
Prakhar Gupta

Table of Content

Purpose and Reason

An *algo tester and graph plotter* is a software developed to demonstrate the actual execution of various algorithms used in computer graphics and numerical methods for individuals related to computer science and mathematics field. The application provides graphical representation of the execution of algorithms related to circle drawing, line drawing, various transformations (e.g., translation, rotation, scaling, and shearing of an object) and polygon clipping.

In this application, another section of numerical method related algorithm is also included which includes, interpolation and extrapolation, root finding algorithm and simultaneous linear equation solving algorithm. The system is flexible to be used and reduces run time error as much as possible. The graphical section of this application is designed to cater the need of the teachers to explain their students that how the graphical algorithms executes and work. And other section, is designed to cater the need of researchers and scientists.

This project provide graphical user interface and developed using java programming language. This project is user-friendly and requires minimum human intervention. Individuals just have to enter the required data in required field.

Objective:

Objective about this project is to:

1. To represent real time execution of graphical algorithms on the user's screen.
2. Real time plotting of graph of polynomial equation and transformaions.
3. To integrate various graphical and mathematical algorithms into one system.
4. To make system easy to use with increased accuracy and effectiveness.
5. To validate the input given by the user.
6. To provide better interface to the user so that he/she doesn't face any difficulty to understand the working of system.

Proposed System

This application has two different sections to tackle the problem of two different subjects. From which one is *Numerical Methods* related problems like finding root of equation, interpolation and extrapolation, and simultaneous linear equation. For these three sub-problems the first section is further divide into three sub-sections.

Another interesting section is *Computer Graphics*, in which problems related to circle drawing, line drawing, transformations and algebraic and transcendental function graph drawing using graphical approach is resolved.

The purpose of this project is to automate the manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements. The required software and hardware are easily available and easy to work with.

The proposed system provides a better GUI application which efficiently and effectively deals with above described problems

Here, for the ease of the user, integration of all the numerical method related problem under one section and all graphical problems into another section called computer graphics.

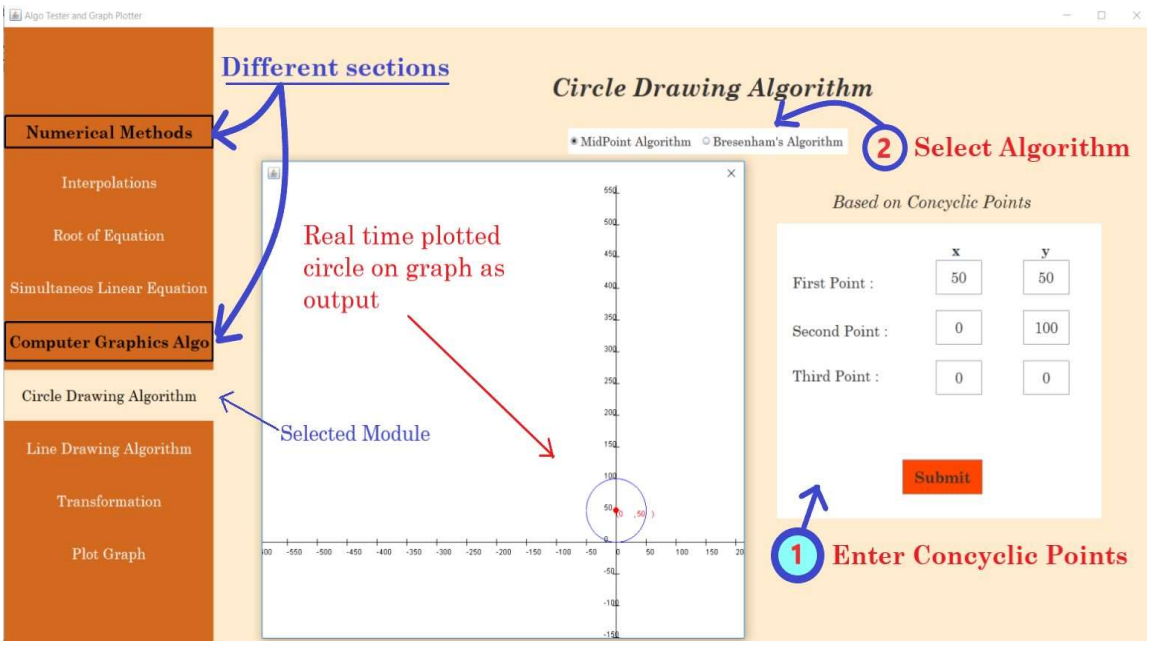
The proposed system is flexible and user friendly. The end-user who is related to mathematics or computer science field can operate this calculator very easily.

The end users of this application may be scientists, researchers, CS or math students, computer science teachers. In short, anybody related to CS or mathematics field.

In this system user only has to come with their specific problem and feed data accordingly into appropriate section and this will generate solution to that problem.

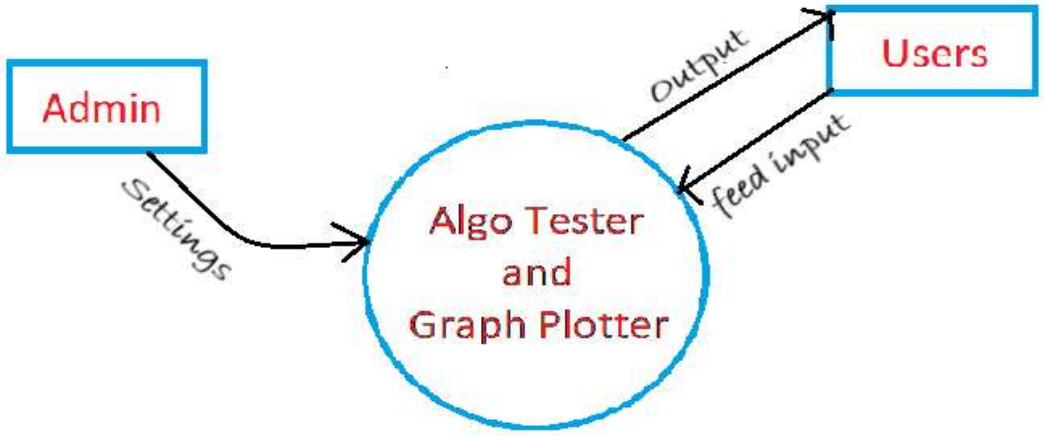
For instance, user want to draw circle and he has three points lie on a circle. Then user has to input those points into circle drawing sub-section under computer graphics section and select algorithm either midpoint circle drawing or bresenham's algorithm and he is done. The output is runtime circle drawing on graph. This is as much more easy as its seems.

For the convenience of users, this application also validates the input of user, if user enters any incorrect data, by giving appropriate warning message.

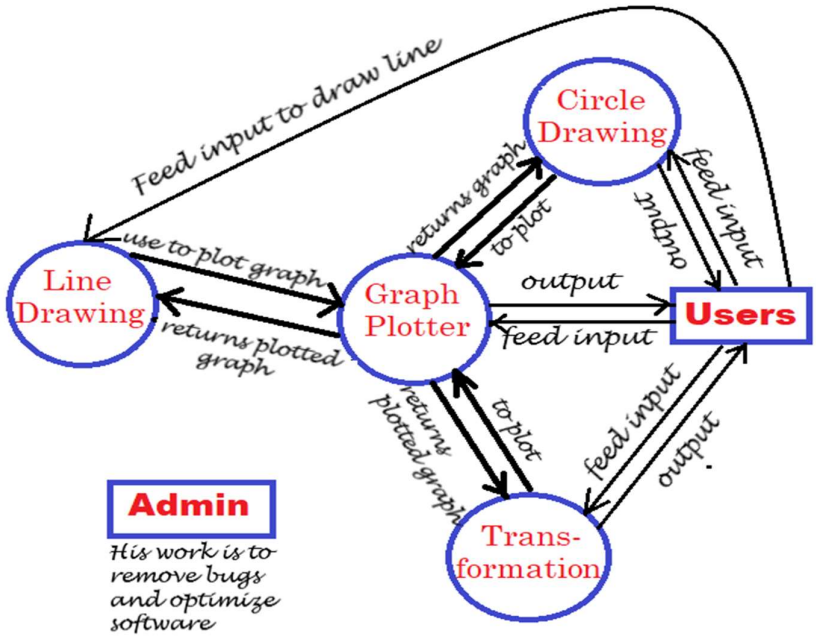


Screen Shot of Application

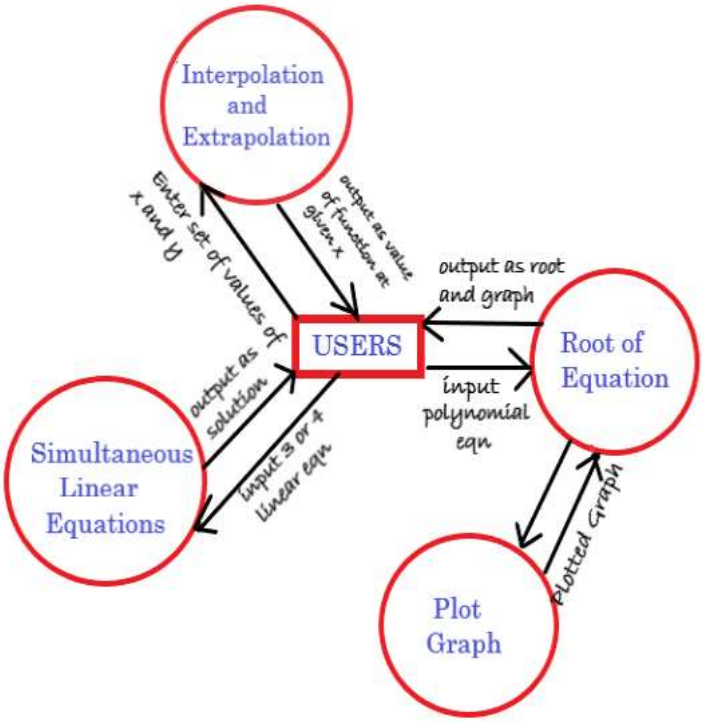
DFD of Proposed System



Level 0 DFD



Level 1: Dfd of Graphical module



Level 1 DFD: Numerical Method

Requirement Analysis:

To fulfill the task written in the requirement specification document, it is required to find the answer of following questions:

- What is algebraic equation?
- What is Interpolation and Extrapolation?
- What is Simultaneous Linear Equation?
- What is transformation and how it work?
- What is circle and what is its properties? How we can find center and radius with given con cyclic points?
- Properties of line and algorithms to draw lines.
- What is their properties?
- Which methodology should be adopted?
- How those methods can be implemented?

To answer these questions, it is required understand following topics:

Numerical Method Module

1.Algebraic Equation

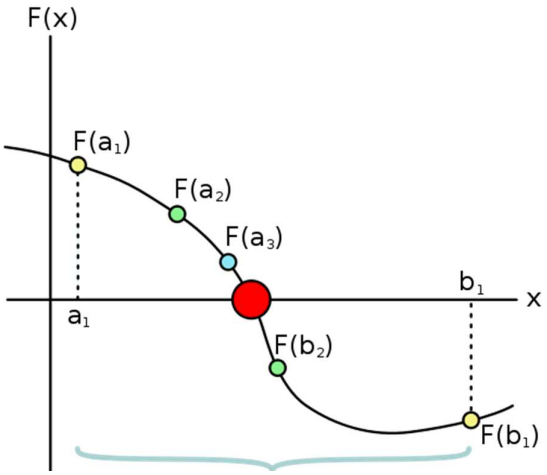
An expression in the form

$$P(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n$$

where a's are constants ($a_0 \neq 0$) and n is a positive integer, is called a *polynomial* in x of degree n. The polynomial P(x)=0 is called an *algebraic equation* of degree n.

The value α of x which satisfies P(x) = 0 ...(1) is called a **root** P(x) = 0. Geometrically, a root of (1) is that value of x where the graph of y = P(x) crosses the x-axis.

he process of finding roots of an Equation is known as the solution of that equation.



Bisection Method:

In mathematics, the **bisection method** is a root-finding method that applies to any continuous functions for which one knows two values with opposite signs. The method consists of repeatedly bisecting the interval defined by these values and then selecting the subinterval in which the function changes sign, and therefore must contain a root.

The input for the method is a continuous function *f*, an interval [*a*, *b*], and the function values *f*(*a*) and *f*(*b*). The function values are of opposite sign (there is at least one zero crossing within the interval). Each iteration performs these steps:

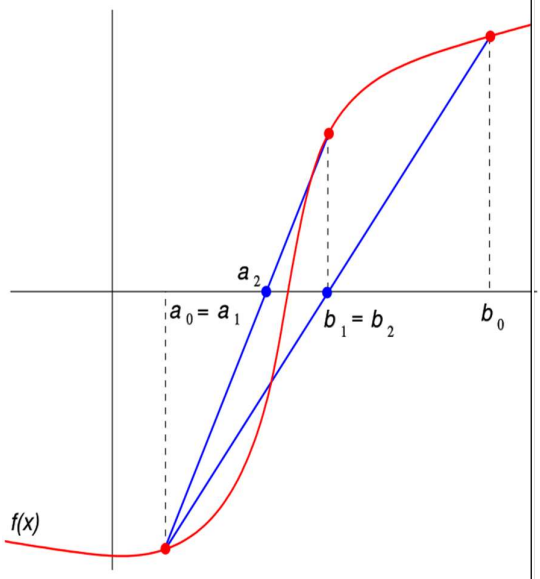
1. Calculate *c*, the midpoint of the interval, $c = (a + b)/2$.
2. Calculate the function value at the midpoint, *f*(*c*).
3. If convergence is satisfactory (that is, $(c - a)$ is sufficiently small, or $|f(c)|$ is sufficiently small), return *c* and stop iterating.
4. Examine the sign of *f*(*c*) and replace either (*a*, *f*(*a*)) or (*b*, *f*(*b*)) with (*c*, *f*(*c*)) so that there is a zero crossing within the new interval.

Algorithm:

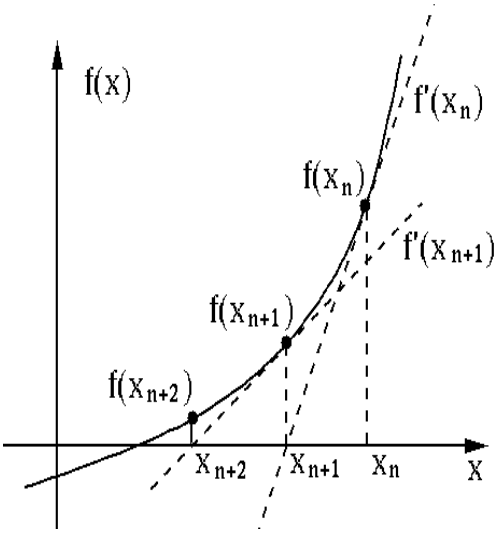
```
INPUT: Function f,
       endpoint values a, b,
       tolerance TOL,
       maximum iterations NMAX
CONDITIONS: a < b,
            either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
OUTPUT: value which differs from a root of f(x) = 0 by less than TOL

N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
  c ← (a + b)/2 # new midpoint
  If f(c) = 0 or (b − a)/2 < TOL then # solution found
    Output(c)
    Stop
  EndIf
  N ← N + 1 # increment step counter
  If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```

Regular Falsi Method:

The **Regular Falsi Method** calculates the new solution estimate as the x-intercept of the line segment joining the endpoints of the function on the current bracketing interval. Essentially, the root is being approximated by replacing the actual function by a line segment on the bracketing interval and then using the classical double false position formula on that line segment



More precisely, suppose that in the k-th iteration the bracketing interval is (a_k, b_k). Construct the line through the points $(a_k, f(a_k))$ and $(b_k, f(b_k))$, as illustrated. This line is a secant or chord of the graph of the function f. In point-slope form, its equation is given by



$$y - f(b_k) = \frac{f(b_k) - f(a_k)}{b_k - a_k}(x - b_k).$$

Now choose c_k to be the x-intercept of this line, that is, the value of x for which $y = 0$, and substitute these values to obtain

$$f(b_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(c_k - b_k) = 0.$$

Solving this equation for c_k gives:

$$c_k = b_k - f(b_k)\frac{b_k - a_k}{f(b_k) - f(a_k)} = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$$

Newton Raphson Method:

The **Newton–Raphson method** (also known as Newton's method) is a way to quickly find a good approximation for the root of a real-valued function f(x) = 0. It uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it.

Suppose you need to find the root of a continuous, differentiable function f(x), and you know the root you are looking for is near the point $x = x_0$. Then Newton's method tells us that a better approximation for the root is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

This process may be repeated as many times as necessary to get the desired accuracy. In general, for any x-value x_n, next value is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Note: the term "near" is used loosely because it does not need a precise definition in this context. However, x_0 should be closer to the root you need than to any other root (if the function has multiple roots).

2.Simultaneous Linear Equation

Solving of a system of linear algebraic equations appears frequently in many engineering problems. Most of numerical techniques which deals with partial differential equations, represent the governing equations of physical phenomena in the form of a system of linear algebraic equations.

Suppose we have given following equations:

$$a_1 x_1 + b_1 x_2 + c_1 x_3 = d_1$$
$$a_2 x_1 + b_2 x_2 + c_2 x_3 = d_2$$
$$a_3 x_1 + b_3 x_2 + c_3 x_3 = d_3$$

Above equation are Simultaneous Linear Equation. In this we have to find the value of x_1, x_2, x_3.

Gauss Elimination Method

Gauss elimination technique is a well-known numerical method which is employed in many scientific problems.
Consider an arbitrary system of linear algebraic equations as follows:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2$$
.
.
.
$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n$$

where x_i are unknowns and a_{ij} are coefficients of unknowns and c_i are equations' constants. This system of algebraic equation can be written in the matrix form as follows:
[A]{x}={C}
Where [A] is the matrix of coefficient and {x} is the vector of unknowns and {C} is the vector of constants.

Gauss elimination method eliminate unknowns' coefficients of the equations one by one. Therefore the matrix of coefficients of the system of linear equations is transformed to an upper triangular matrix. The last transformed equation has only one unknown which can be determined easily. This evaluated unknown can be used in the upper equation for determining the next unknown and so on. Finally the system of linear equations can be solved by back substitution of evaluated unknowns.

3.Interpolation and Extrapolation

Suppose we have given the following values of y = f(x) for a set of values of x:

X:	x_0	x_1	x_2	x_3 …….x_n
Y:	y_0	y_1	y_2	y_3 …….y_n

Then the process of finding the value of y corresponding to any value of $x = x_i$ between x_0 and x_n is called *Interpolation*. Thus the *interpolation is the technique of estimating the value of a function for any intermediate value of the independent variable* while the process of computing the value of the function outside the given range is called *Extrapolation*.

Newton Forward Interpolation Formula:

$$y_p = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \cdots$$

This formula is particularly useful for interpolating the values of f(x) near the beginning of the set of values given. 'h' is called the interval of difference and **p = (x – a) / h**, Here a is first term.

Example:

Input : Value of Sin 52

θ°	45°	50°	55°	60°
sin θ	0.7071	0.7660	0.8192	0.8660

Output :

	Differences			
x°	$10^4 y$	$10^4 \Delta y$	$10^4 \Delta^2 y$	$10^4 \Delta^3 y$
45°	7071	589	− 57	− 7
50°	7660	532	− 64	
55°	8192	468		
60°	8660			

Value at Sin 52 is 0.788003

Newton Backward Interpolation Formula:

$$y_p = y_n + p\nabla y_n + \frac{p(p+1)}{2!}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \cdots$$

This formula is useful when the value of f(x) is required near the end of the table. h is called the interval of difference and

u = (x – a_n) / h, Here 'a_n' is last term.

Computer Graphics Module

1.Line Generation Algorithm

A line connects two points. It is a basic element in graphics. To draw a line, you need two points between which you can draw a line. In the following three algorithms, we refer the one point of line as X0,Y0 and the second point of line as X1,Y1 .

DDA Algorithm

Digital Differential Analyzer DDA algorithm is the simple line generation algorithm which is explained step by step here.

Step 1 – Get the input of two end points (X0,Y0)(X0,Y0) and (X1,Y1)(X1,Y1).

Step 2 – Calculate the difference between two end points.

$$dx = X_1 - X_0$$
$$dy = Y_1 - Y_0$$

Step 3 – Based on the calculated difference in step-2, you need to identify the number of steps to put pixel. If dx > dy, then you need more steps in x coordinate; otherwise in y coordinate.

```
if (absolute(dx) > absolute(dy))
   Steps = absolute(dx);
else
   Steps = absolute(dy);
```

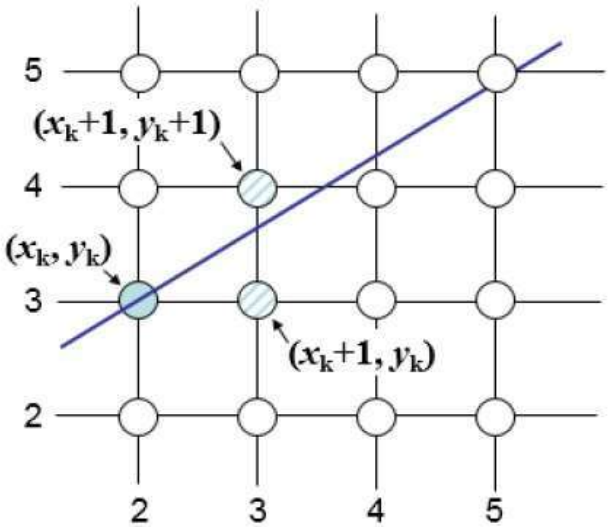
Step 4 – Calculate the increment in x coordinate and y coordinate.

```
Xincrement = dx / (float) steps;
Yincrement = dy / (float) steps;
```

Step 5 – Put the pixel by successfully incrementing x and y coordinates accordingly and complete the drawing of the line.

```
for(int v=0; v < Steps; v++)
{
   x = x + Xincrement;
   y = y + Yincrement;
   putpixel(Round(x), Round(y));
}
```

Bresenham's Line Generation



The **Bresenham algorithm** is another incremental scan conversion algorithm. The big advantage of this algorithm is that, it uses only integer calculations. Moving across the x axis in unit intervals and at each step choose between two different y coordinates.

For example, as shown in the following illustration, from position 2,32,3 you need to choose between 3,33,3 and 3,43,4. You would like the point that is closer to the original line.

Step 1 – Input the two end-points of line, storing the left end-point in (x0,y0)(x0,y0).

Step 2 – Plot the point (x0,y0)(x0,y0).

Step 3 – Calculate the constants dx, dy, 2dy, and 2dy–2dx2dy–2dx and get the first value for the decision parameter as –

$$p0=2dy–dxp0=2dy–dx$$

Step 4 – At each XkXk along the line, starting at k = 0, perform the following test –

If pkpk < 0, the next point to plot is (xk+1,yk)(xk+1,yk) and

$$pk+1=pk+2dypk+1=pk+2dy$$

Otherwise,

$$(xk,yk+1)(xk,yk+1)$$

$$pk+1=pk+2dy–2dxpk+1=pk+2dy–2dx$$

Step 5 – Repeat step 4 dx–1dx–1 times.

For m > 1, find out whether you need to increment x while incrementing y each time.

After solving, the equation for decision parameter PkPk will be very similar, just the x and y in the equation gets interchanged.

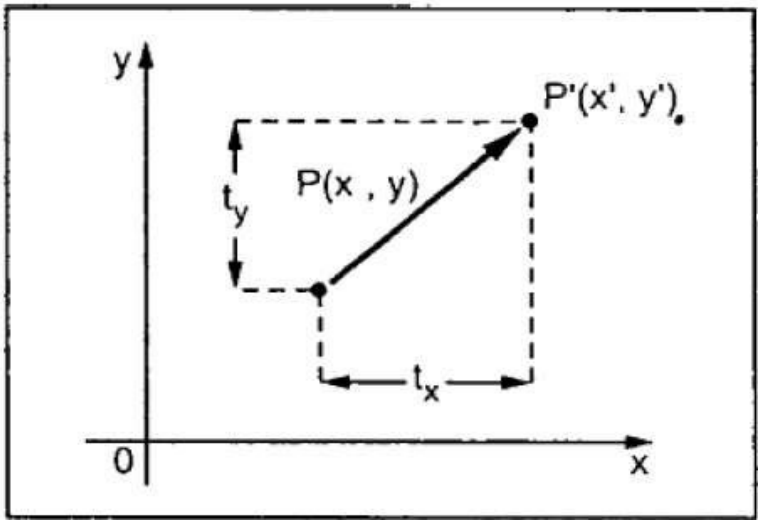
2.Transformation

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

Translation

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (t_x, t_y) to the original coordinate X,Y to get the new coordinate X′,Y′.



From the above figure, you can write that –

$$X' = X + t_x$$

$$Y' = Y + t_y$$

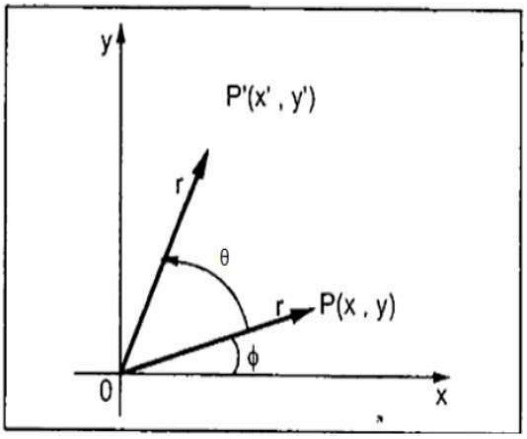
The pair (t_x, t_y) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

$$P' = P + T$$

Rotation

In rotation, we rotate the object at particular angle θ thetatheta from its origin. From the following figure, we can see that the point PX,YX,Y is located at angle ϕ from the horizontal X coordinate with distance r from the origin.

Let us suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P' X',Y'X',Y'.



Using standard trigonometric the original coordinate of point P X,Y can be represented as –

$$X = r\cos\varphi \ldots\ldots (1)$$
$$Y = r\sin\varphi \ldots\ldots (2)$$

Same way we can represent the point P' X',Y' as –

$$x' = r\cos(\varphi+\theta) = r\cos\varphi\cos\theta - r\sin\varphi\sin\theta \ldots\ldots (3)$$
$$y' = r\sin(\varphi+\theta) = r\cos\varphi\sin\theta + r\sin\varphi\cos\theta \ldots\ldots (4)$$

Substituting equation 1 & 2 in 3 & 4 respectively, we will get

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Representing the above equation in matrix form,

$$[X \quad Y]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \text{ OR}$$

$$P' = P \cdot R$$

Where R is the rotation matrix

$$R = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

The rotation angle can be positive and negative.

Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

Let us assume that the original coordinates are X,Y, the scaling factors are (S_X, S_Y), and the produced coordinates are X′,Y′. This can be mathematically represented as shown below –

$$X' = X \cdot S_X \text{ and } Y' = Y \cdot S_Y$$

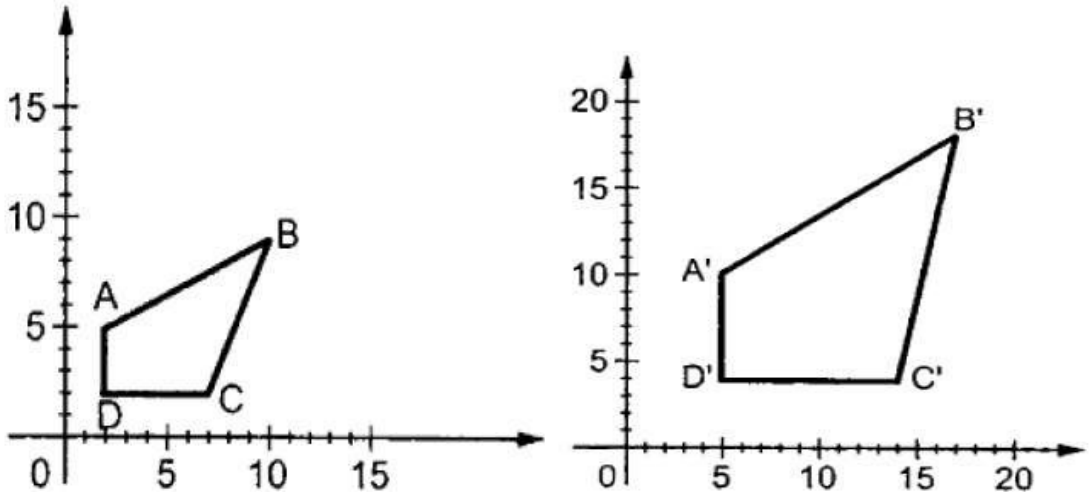
The scaling factor S_X, S_Y scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below –

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

OR

$$P' = P \cdot S$$

Where S is the scaling matrix. The scaling process is shown in the following figure.



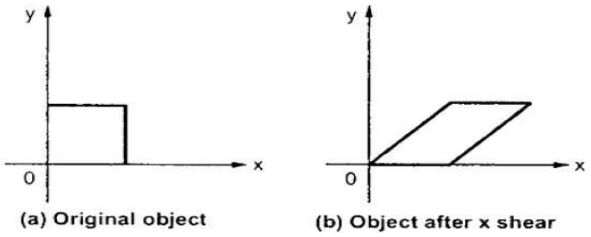
If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

Shear

A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations **X-Shear** and **Y-Shear**. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values. Shearing is also termed as **Skewing**.

X-Shear

The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left as shown in below figure.



(a) Original object (b) Object after x shear

The transformation matrix for X-Shear can be represented as –

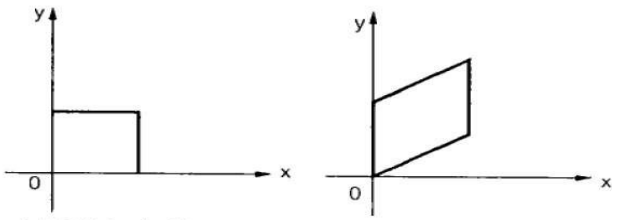
$$X_{sh} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y' = Y + Sh_y \cdot X$$

$$X' = X$$

Y-Shear

The Y-Shear preserves the X coordinates and changes the Y coordinates which causes the horizontal lines to transform into lines which slopes up or down as shown in the following figure.



(a) Original object (b) Object after y shear

The Y-Shear can be represented in matrix from as –

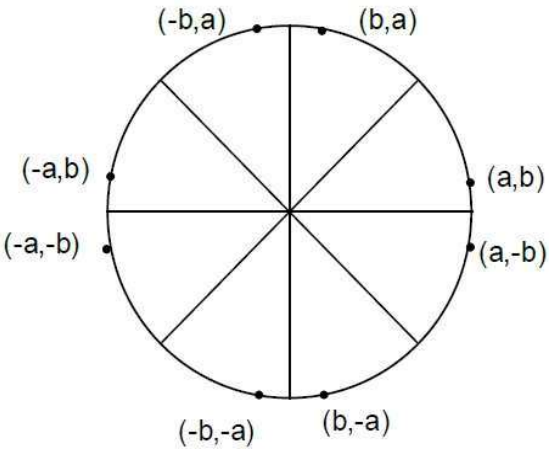
$$Y_{sh} = \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_x \cdot Y$$

$$Y' = Y$$

3.Circle Generation Algorithm

Drawing a circle on the screen is a little complex than drawing a line. There are two popular algorithms for generating a circle – **Bresenham's Algorithm** and **Midpoint Circle Algorithm**. These algorithms are based on the idea of determining the subsequent points required to draw the circle. Let us discuss the algorithms in detail –

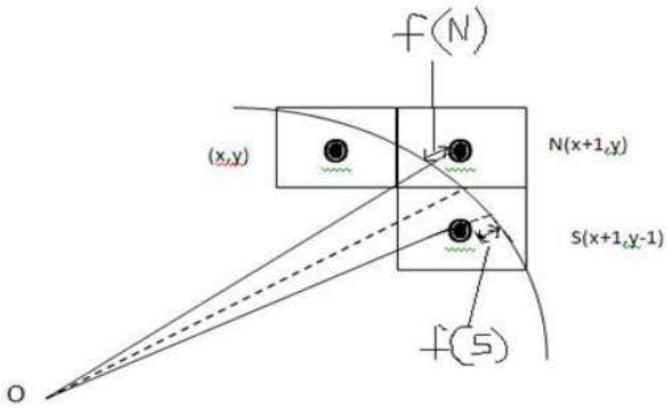


The equation of circle is $X^2+Y^2=r^2$, where r is radius.

Bresenham's Algorithm

We cannot display a continuous arc on the raster display. Instead, we have to choose the nearest pixel position to complete the arc.

From the following illustration, you can see that we have put the pixel at X,YX,Y location and now need to decide where to put the next pixel – at N (X+1,Y) or at S (X+1,Y–1).

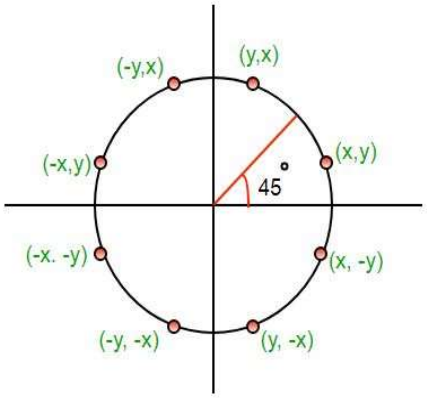


This can be decided by the decision parameter **d**.

- If d <= 0, then N(X+1,Y) is to be chosen as next pixel.
- If d > 0, then S(X+1,Y–1) is to be chosen as the next pixel.

Mid Point Algorithm

The **mid-point** circle drawing algorithm is an algorithm used to determine the points needed for rasterizing a circle.
We use the **mid-point** algorithm to calculate all the perimeter points of the circle in the **first octant** and then print them along with their mirror points in the other octants. This will work because a circle is symmetric about it's centre.



The algorithm is very similar to the Mid-Point Line Generation Algorithm. Here, only the boundary condition is different.
For any given pixel (x, y), the next pixel to be plotted is either **(x, y+1)** or **(x-1, y+1)**. This can be decided by following the steps below.

1. Find the mid-point **p** of the two possible pixels i.e (x-0.5, y+1)
2. If **p** lies inside or on the circle perimeter, we plot the pixel (x, y+1), otherwise if it's outside we plot the pixel (x-1, y+1)

4.Graph Plotter

In this module, user can draw any type of graph.

1. Sine Function,
2. Cosine Function,
3. Tangent Function,
4. Logarithmic Function,
5. Exponential Function,
6. Polynomial function

by giving function parameter.

System Design:

The aim of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.

Based on the above requirements analysis and SRS document the system design is to prepared

The system design as per the requirement analysis of this project should comprises of have three module:

This system can only be used by Scientists or mathematics field person. They can use this application easily by simply clicking on their required module and enter required information.

Root of equation Module:

To find root of equation through this application, user simply has to click on 'R*oot of equation'* button from the side panel.

Next step is to enter coefficients of the equation.

This module supports up to 3 degrees of equation solving.

If user wants to calculate root of quadratic equation, then he/she only has to put *'0'* in coefficient of x cube.

User is allowed to enter numeric values only.

Here I implemented *Bisection method, Regular Falsi Method, and Newton Raphson Method* to solve the given equation.

User also has to select method from which he wants to calculate his answer.

Interpolation Module:

Interpolation module firstly accept number of data which user want to enter. After pressing submit, entered number of text fields will appear (*'n'* text fields for both X and Y), where user have to fill data value of X and Y. Now, user has to enter value of X for which value of Y is needed. After filling all the information, press *'calculate'.*

If there is any mistake while entering data then this application give appropriate warning to the user.

This module supports max 20 data entry text fields.

In this module, I used Newton's Forward and Backward interpolation formula and my program automatically detects where to use forward Interpolation and where to use backward interpolation formula.

Simultaneous Linear Equation:

Here, this module firstly requires number of equations to solve. After entering number of equation. It displays such number of empty boxes (if user enter 3, then 12 boxes will created). After entering data, it will calculate 'n' unknowns.

User is allowed to enter numeric values only. Otherwise it will show warning.

Here I used gauss elimination method to find the values of unknowns.

Line Generation Module

Here, this module requires end points of a line to plot on graph in the format of pixel. After then user has to select algorithm either DDA Line generation or Bresenham's Line Generation algorithm.

User is allowed to enter numeric values only. Otherwise it will show warning.

Circle Generation Module

In this module, user has a choice to plot circle in two different ways:

1) Through Center and radius of circle.
2) Through concyclic points.

In first, simply user has to input center and radius of circle. And in second, user has to enter any three points that lie on circle. After choosing method, then user has to opt algorithm, either midpoint circle generation or bresenham's circle generation algorithm.

User is allowed to enter numeric values only. Otherwise it will show warning.

Tranformation Module

This is very interesting module. Here we can perform 2D transformations i.e. translation, rotation, scaling, and shearing of given polygon.

In this module, user firstly has to enter coordinates of polygon on which transformation has to perform.

Now if user wants to perform *translation* then he has to provide translation parameter (T_x , T_y). and if user want to apply another transformation on the resulting polygon of previous transformation, then user only has to click on *'Use previous values'* button and then give parameters of transformation to be performed.

User is allowed to enter numeric values only. Otherwise it will show warning.

Graph Plotter:

This is also very interesting module, here one can plot graph of any algebraic and transcendental function.

Here user can plot algebraic equation graph, sine, cosine, tangent, logarithm, exponential graph by only giving function parameter value

Methodology:

Software Engineering is a planned and systematic approach to the development of software. It is a discipline which is consist of methods, tools and techniques used for developing and maintaining software. For solving problems in a setting there must be a development strategy that encompasses the process, methods tools and generic phases used for solving problems. This development strategy is referred as *process model* or *software engineering paradigm.*

The selection of process model is done on the basis of nature of project and application, the methods and tools to be used and the control and deliverables that are required.

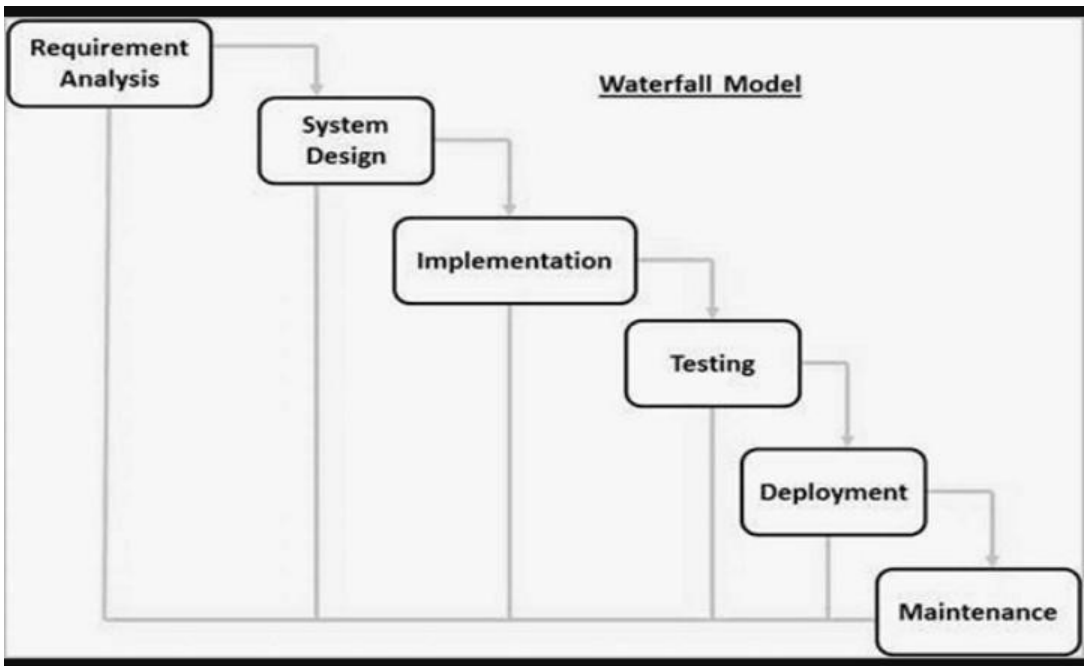
The process model which is implemented for the development of this project is *Waterfall Model.*

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



So all these features proved this methodology to be appropriate for designing this project as this method is linear or sequential based as well as output from one phase serves input for another phase also this model is easy to implement all these feature makes this model of SDLC best for development of this project.

Integration and System Testing:

Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

A) 'Root of Equation' module testing:

I tested this module using following set of values:

Sr	x^3	x^2	x	constant	Method	Result
i	1	1	1	7	Bisection	2.105
ii	1	0	-4	-9	Bisection	2.7064
iii	1	0	-2	-5	Regular Falsi	2.094
iv	1	1	0	-1	Regular Falsi	0.7548
v	3	-9	0	8	Newton Raphson	1.226
vi		0	-3	1	any	Numeric value required
vii	A	c	1	#	any	Numeric value required

This module is function well under all types of input.

B) 'Interpolation and Extrapolation' module testing:

Case1:

X:	100	150	200	250	300	350	400
Y:	10.63	13.03	15.04	16.81	18.42	19.90	21.27

i. For x = 218 , the value of function f(x)= 15.697 (Interpolation)

ii. For x = 410 , the value of function f(x)= 21.53 (Extrapolation)

Case2:

X:	3	4	5	6	7	8	9
Y:	4.8	8.4	14.5	23.6	36.2	52.8	73.9

i. For x = 1 , the value of function f(x)= 3.1 (Interpolation)

ii. For x = 10 , the value of function f(x)= 100 (Extrapolation)

Case3:

X:	20	25	30	35	40	45
Y:	354	332		260	231	204

Here a warning (value of y3 is required) is generated.

- If value other than numeric value is entered in any field, there will be generated warning (Numeric value required).

C) 'Simultaneous Linear Equation' module testing:

Case1

10x –7y +3z +5u = 6

–6x +8y –z –4u = 5

3x +y +4z +11u = 2

5x –9y –2z +4u = 7

Answer:

x = 5,

y = 4,

z = –7,

u = 1

Case2

x + 4y –z = –5

x + y –6z = –12

3x –y –z = 4

Answer:

x = 1.647,

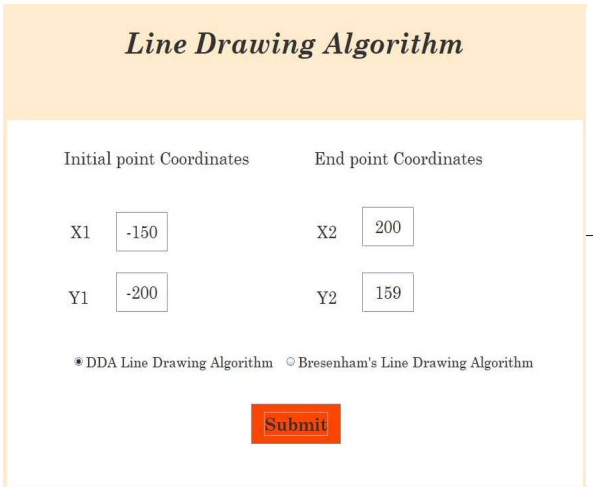
y = –1.140,

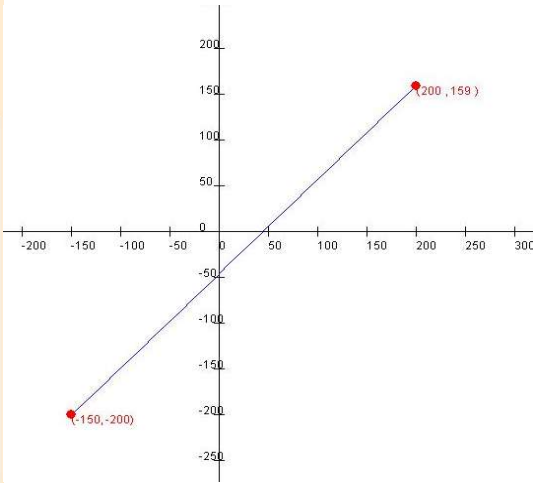
z = 2.084

❖ If value other than numeric value is entered in any field, there will be generated warning (Numeric value required).

D) Line Generation module testing:

Case 1:



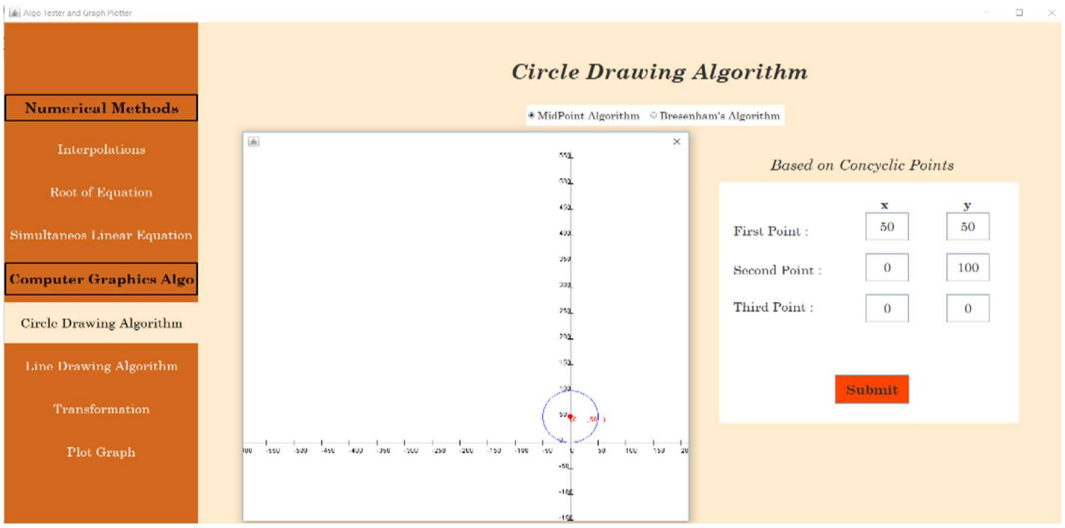


Case 2:

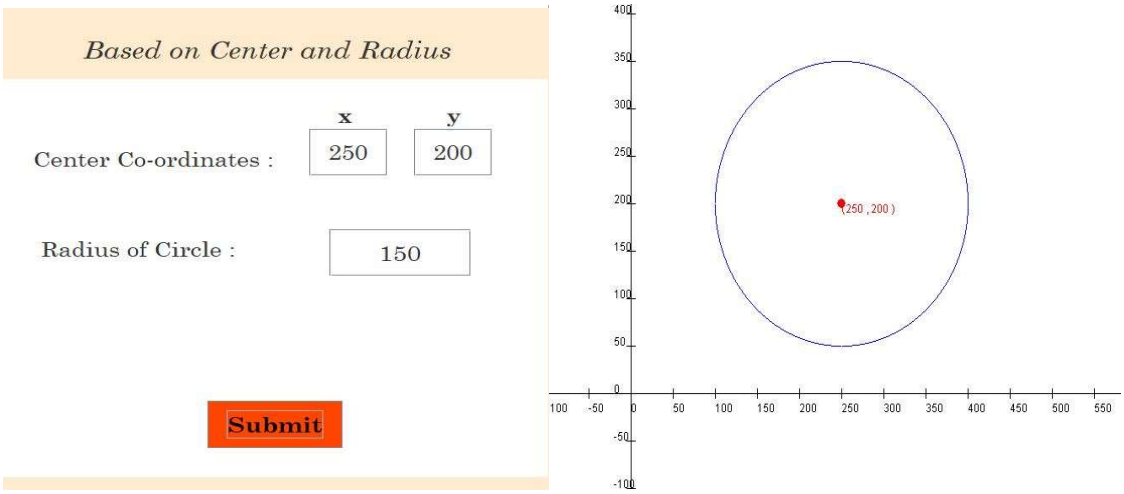
If user input any invalid figure like leave required space empty or fill it with invalid input then a pop up warning message will appear.

E) Circle generation algorithm module testing:

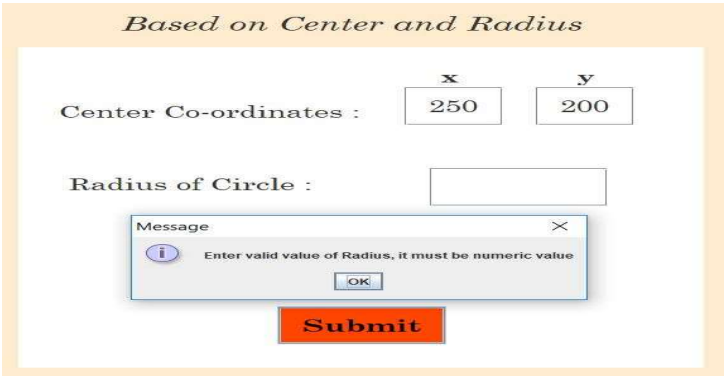
Case1:



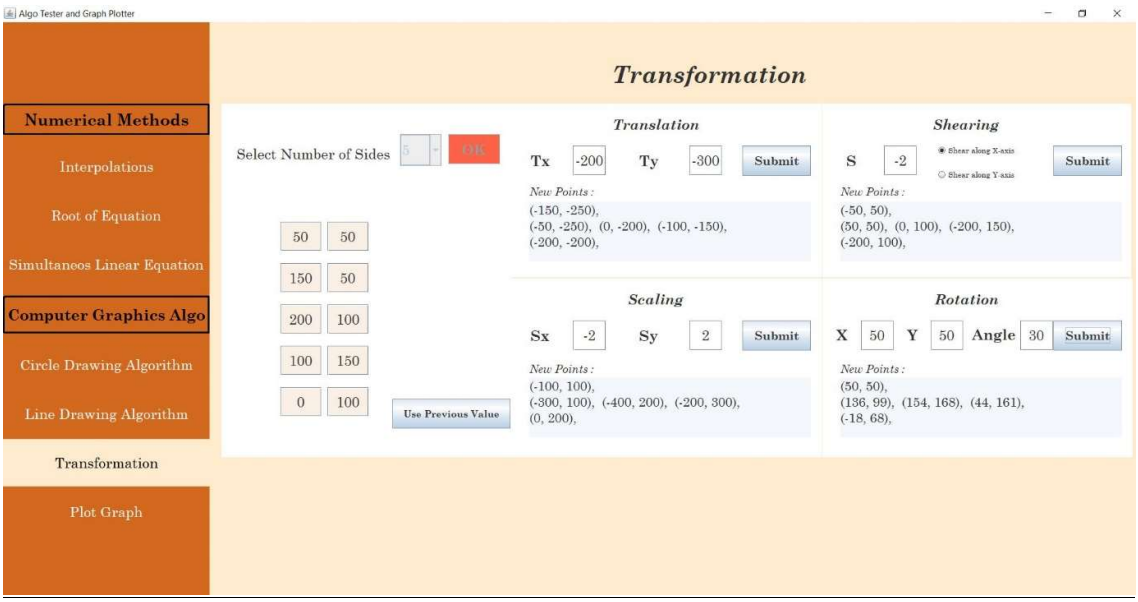
Case2:



Case 3(error case):



F) Transformation module testing:



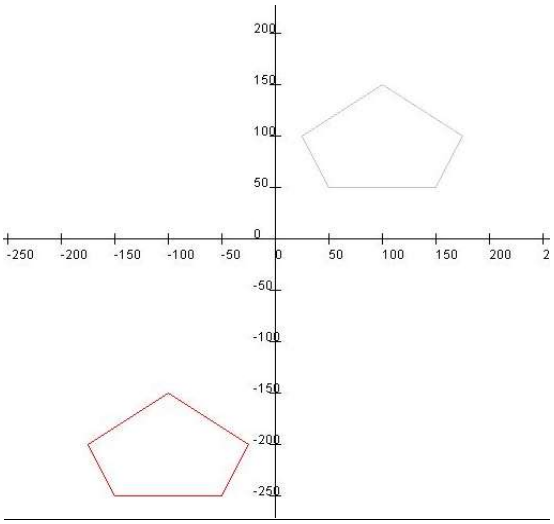


Figure 1: Translation

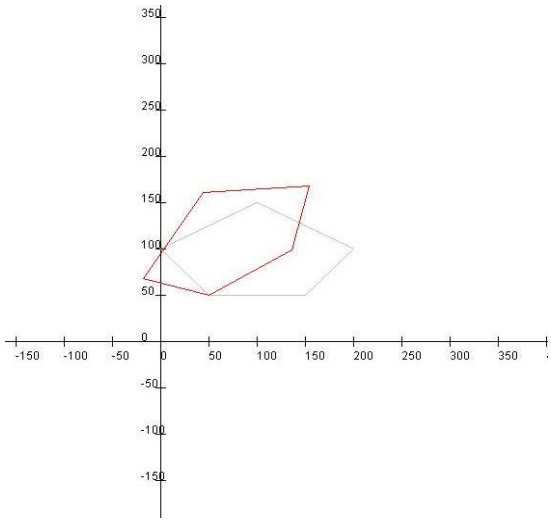


Figure 2: Rotation

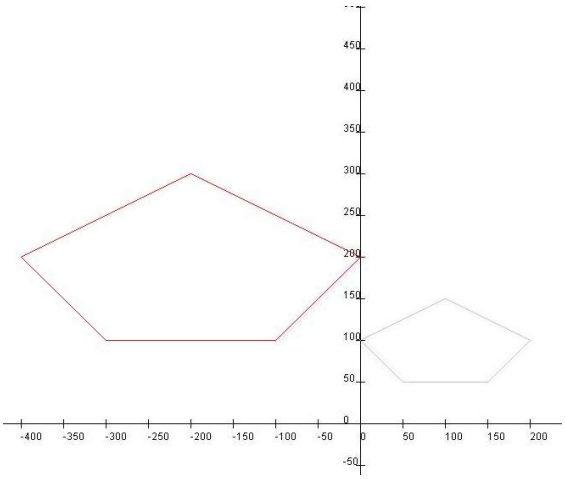


Figure 3: Scaling

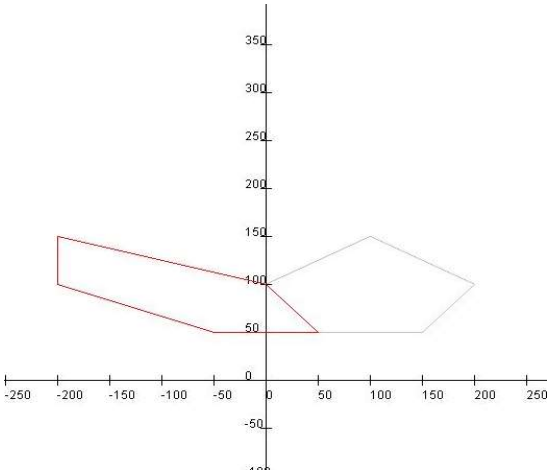


Figure 4: Shearing`

After successfully integrating and testing the software we will come to know that whether the system is fulfilling the user requirements or not that are listed in SRS documentation and if there are working properly then they are deployed for use otherwise send to next stage that is maintenance phase or stage.

Maintenance Phase

Maintenance: Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop a full software. The error encountered during testing phase of the Software Development Life Cycle re maintained or rectified in this phase by the developer and then the software is again send to testing phase to check whether the error has been rectified or not. There are basically three types of maintenance:

- **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
- **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
- **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as work on a new computer platform or with a new operating system.

In this project I have only adopted or use the corrective and perfective maintenance to rectify the error as well as in updating it as per the new requirement arises I have not come across the adaptive problem so adaptive type of maintenance is still unused or untouched in maintenance problem.

Hardware and Software Requirements:

Windows XP, Vista, Windows 7(ultimate, enterprise) – 64 bits

Java version: 1.4.0 (minimum), 5.0 or above (recommended)

Software: Eclipse Luna

Processor: 1.5 GHz or faster

Memory: 1 GB

Free Disk Space: 500MB

Hardware: Monitor, Keyboard, Mouse

Advantages of Proposed System:

- It satisfies the user requirements.
- Easy to understand by the user.
- Easy to operate
- Have a good user interface.
- Improves user's mistake while entering data.
- Integration of all the methods in one system.
- This calculator also maintains the accuracy of the output even after the decimal point.
- Expandable
- Minimum time needed for the various processing
- Greater efficiency

Future Scope:

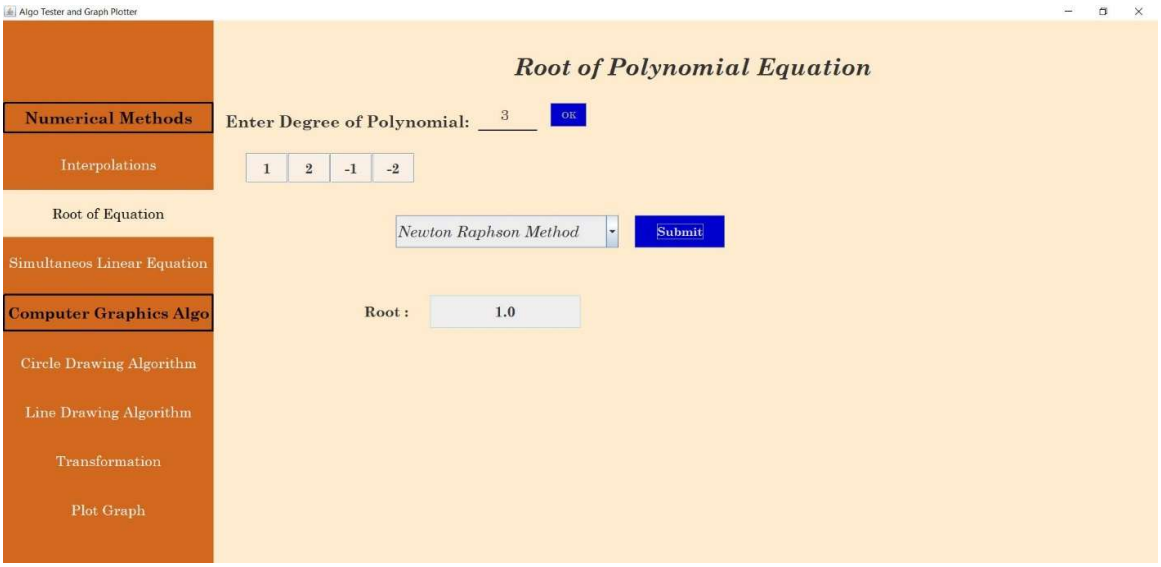
- Nothing is perfect in this world. So we are also no exception. Although I tried my best to present information effectively, yet there can be further enhancement in the application.

- We can give more advance software for this project including more facilities.

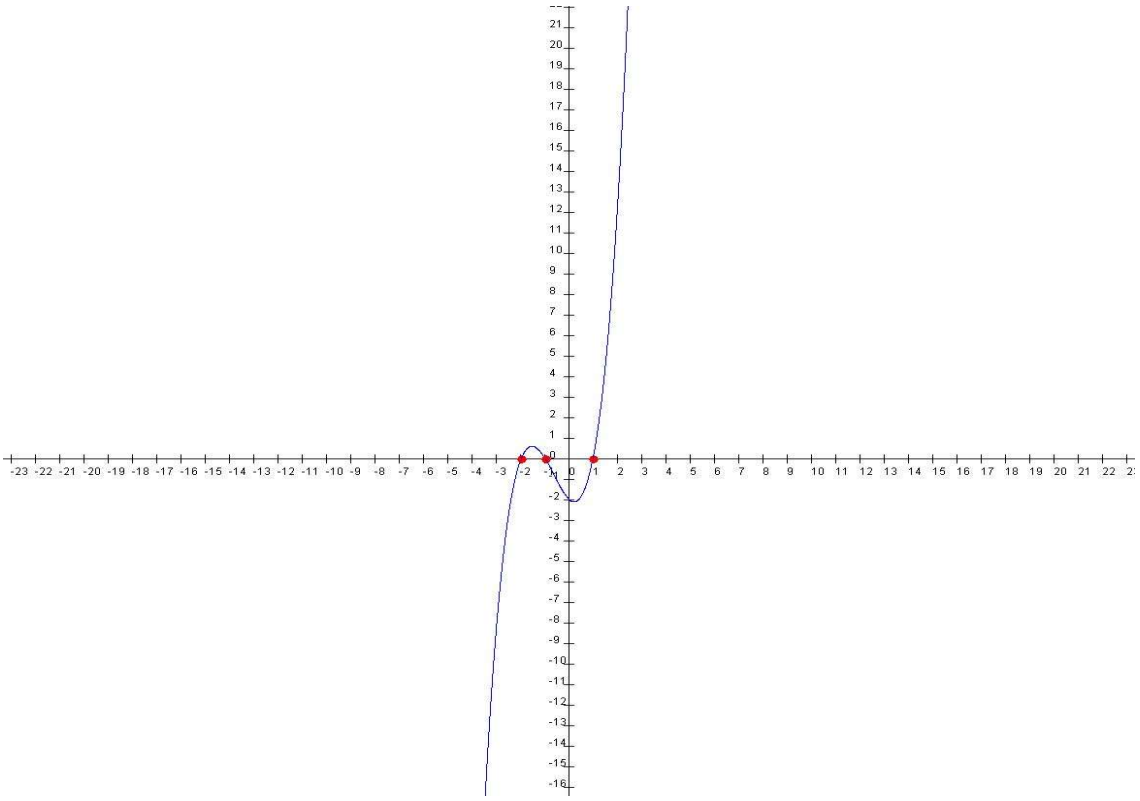
- We can host the platform on online server to make it accessible worldwide.

Screen Shot

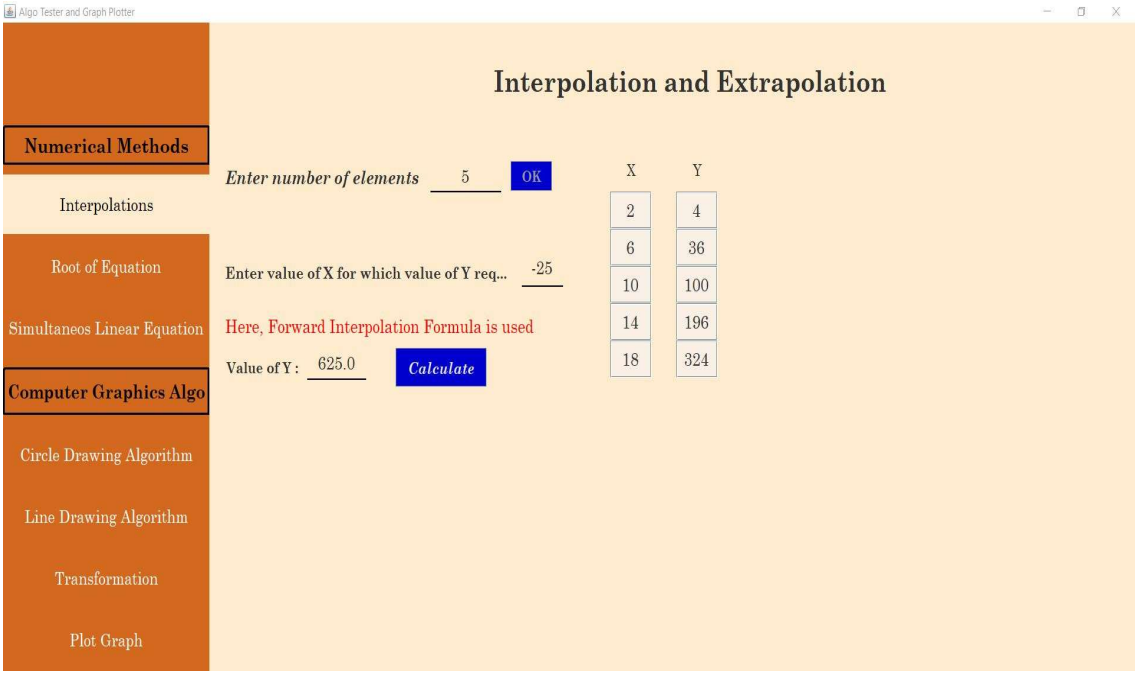
Root of Polynomial



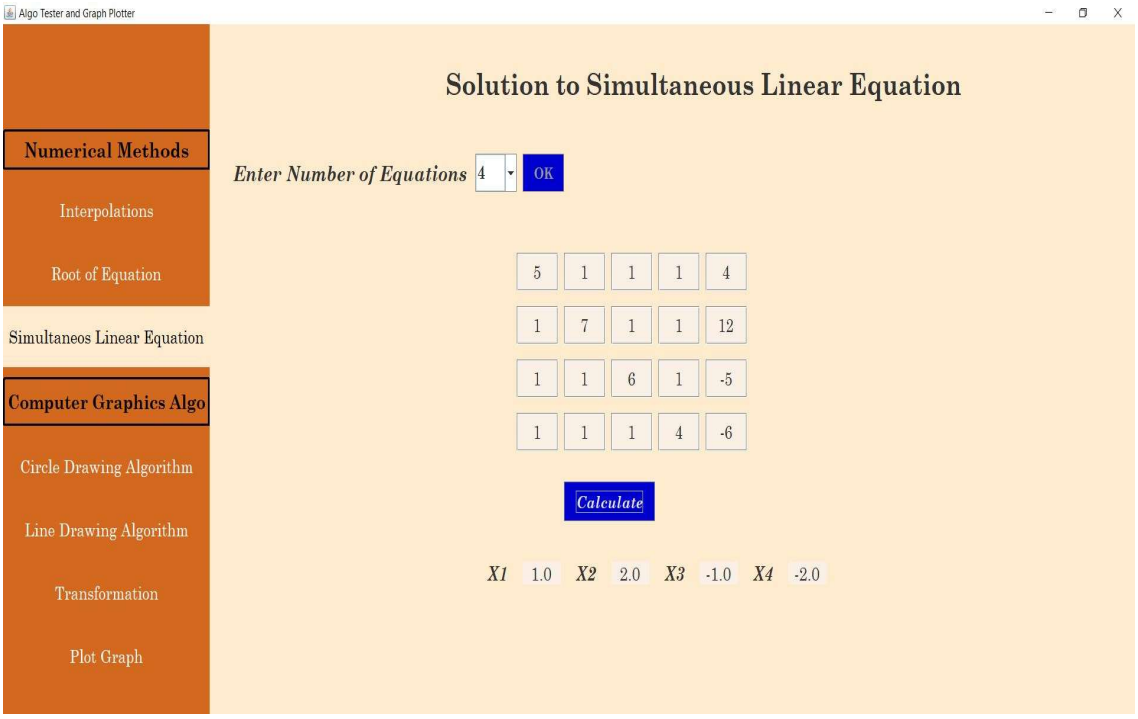
Graph of above polynomial:



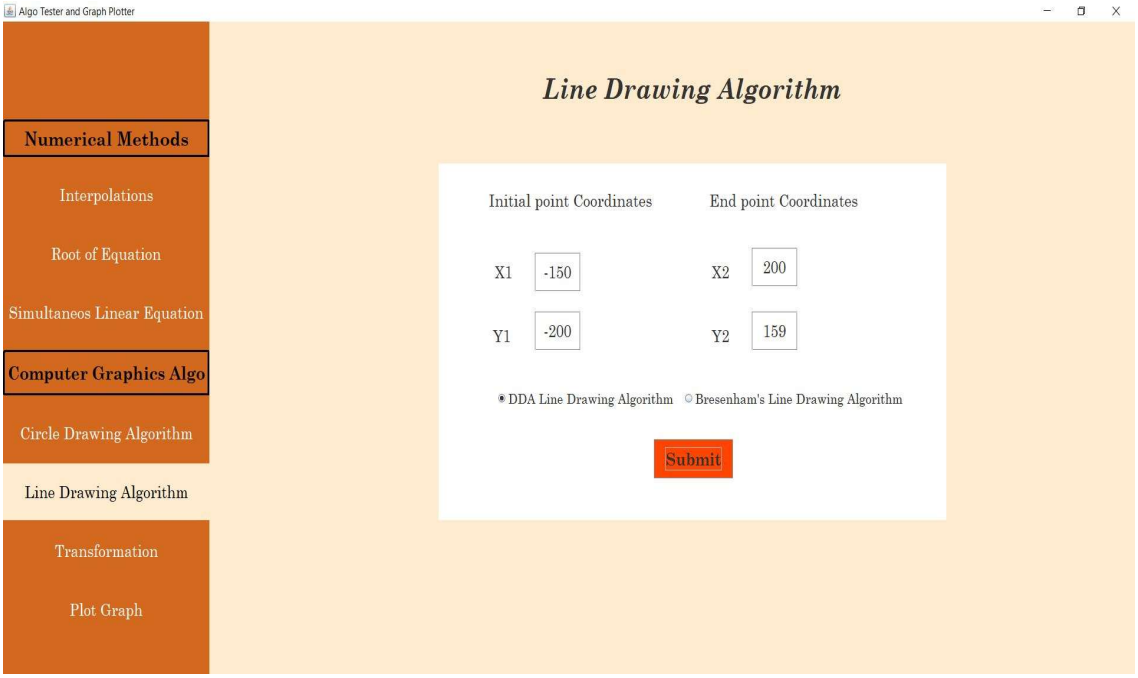
Interpolation and Extrapolation



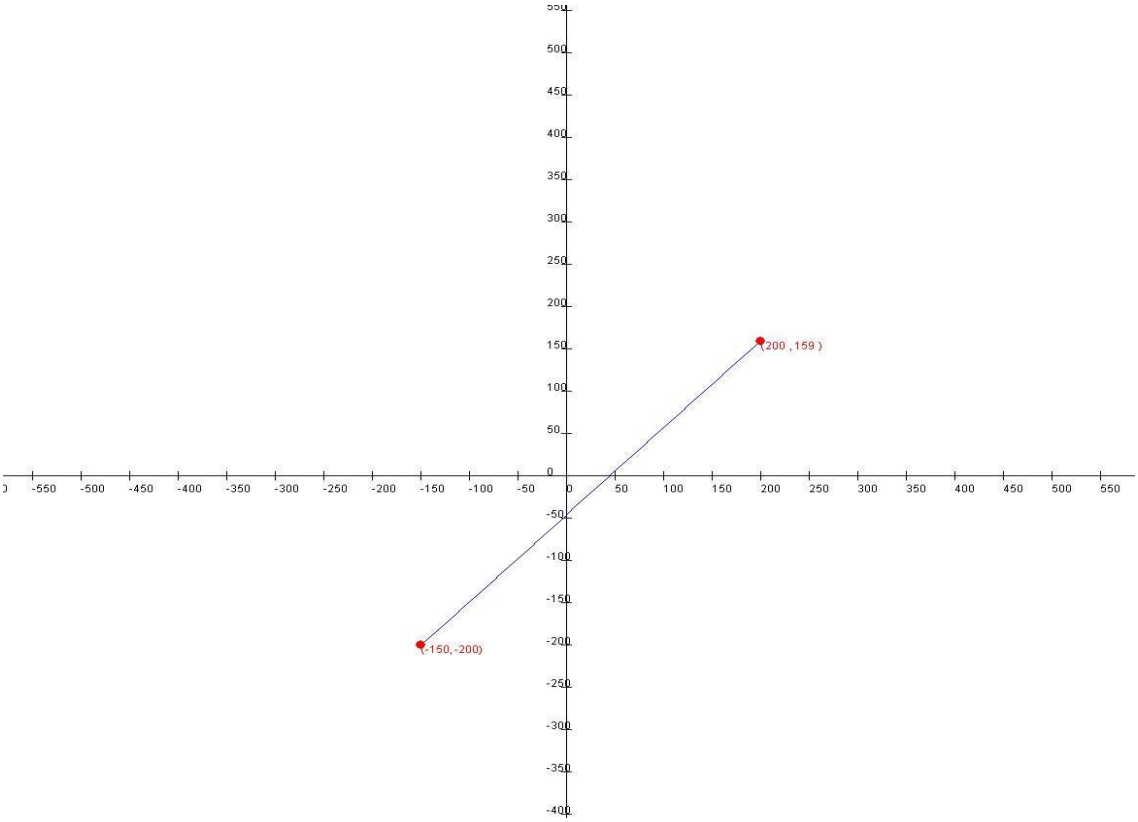
Simultaneous Linear Equation



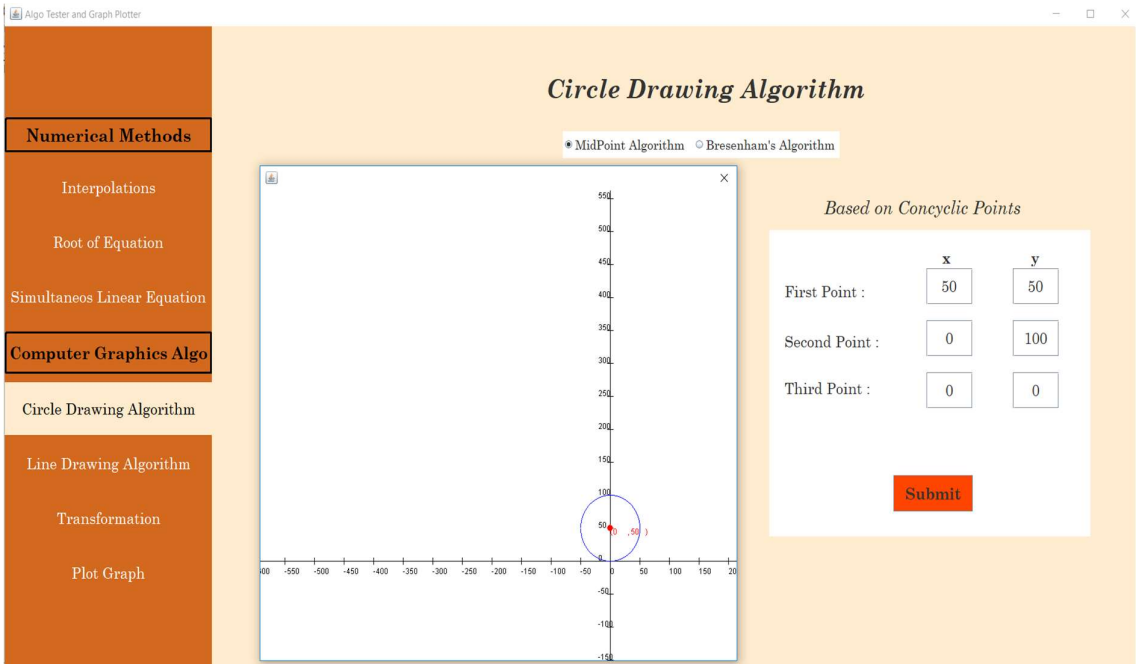
Line Generation:

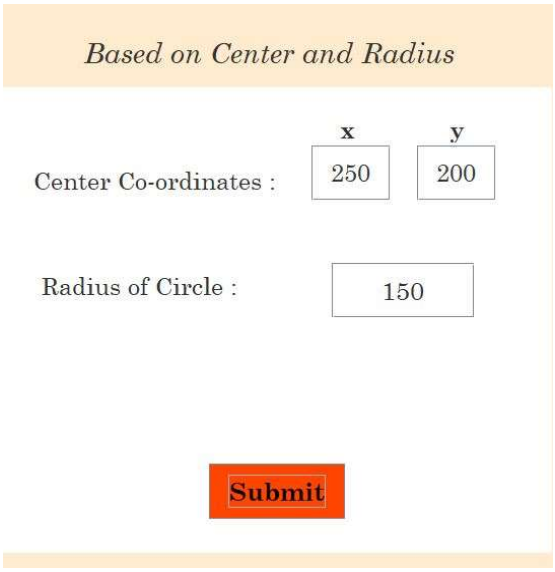


Output:

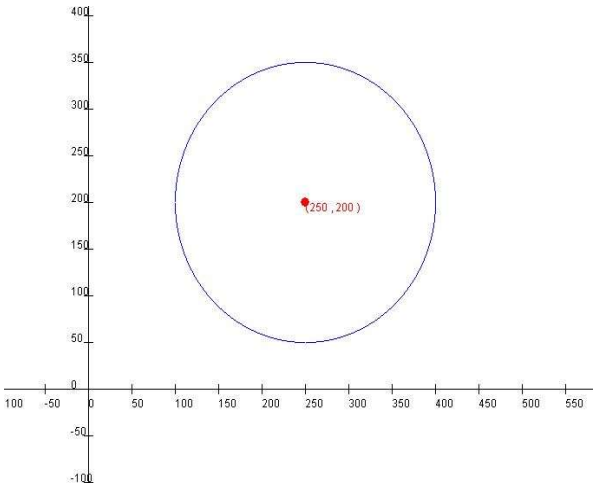


Circle Generation

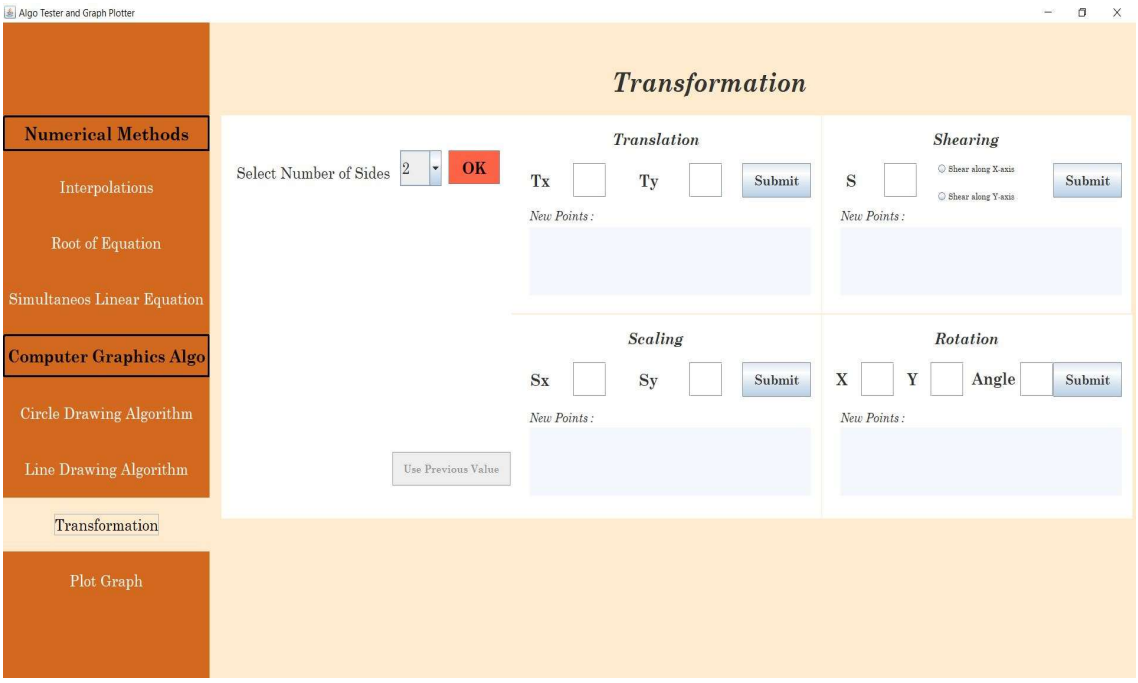


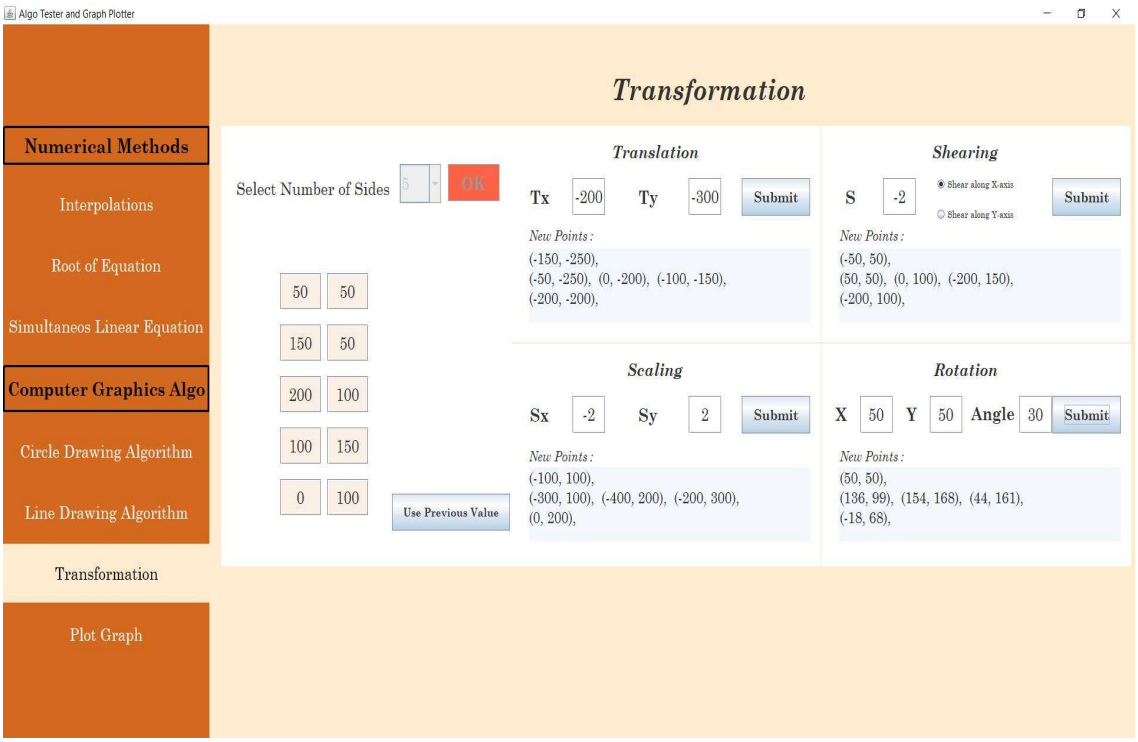


Output:

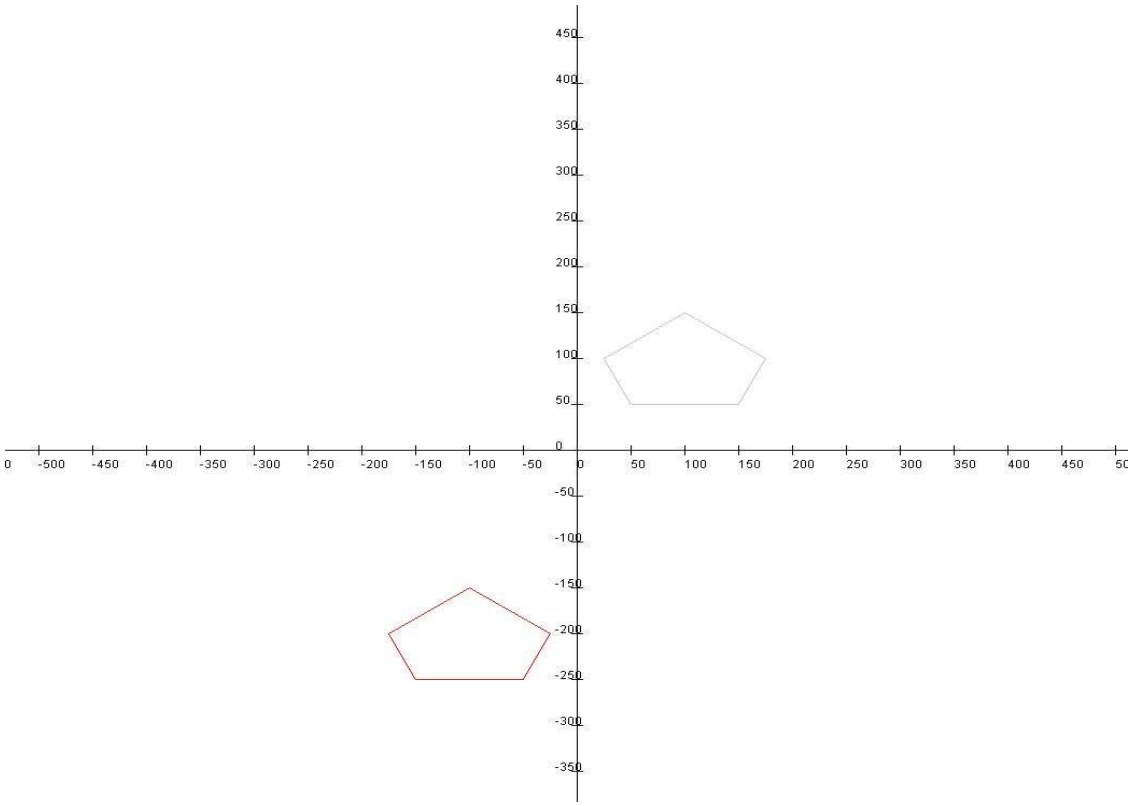


Transformation

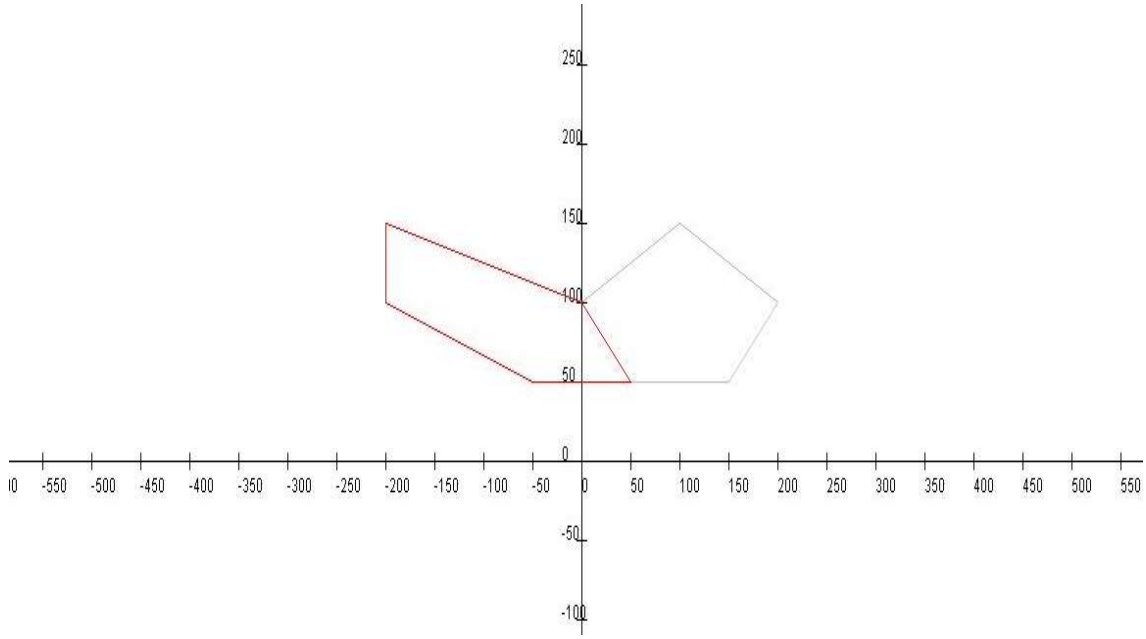




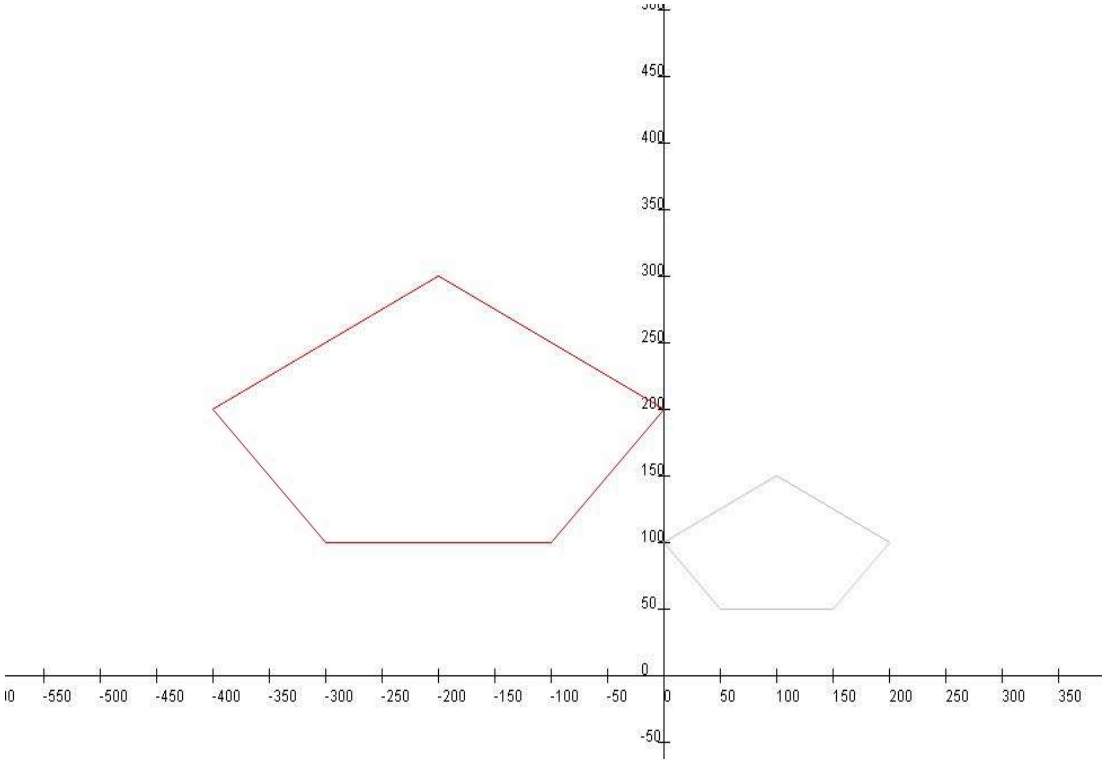
Translation



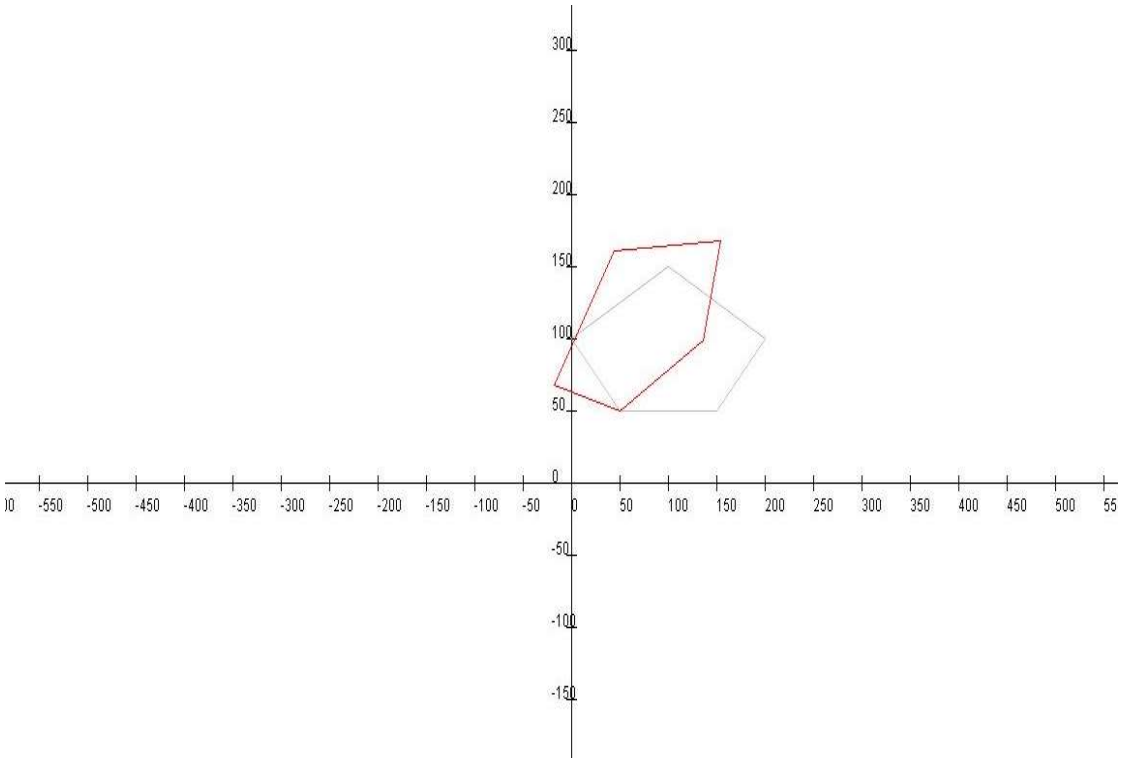
Shearing



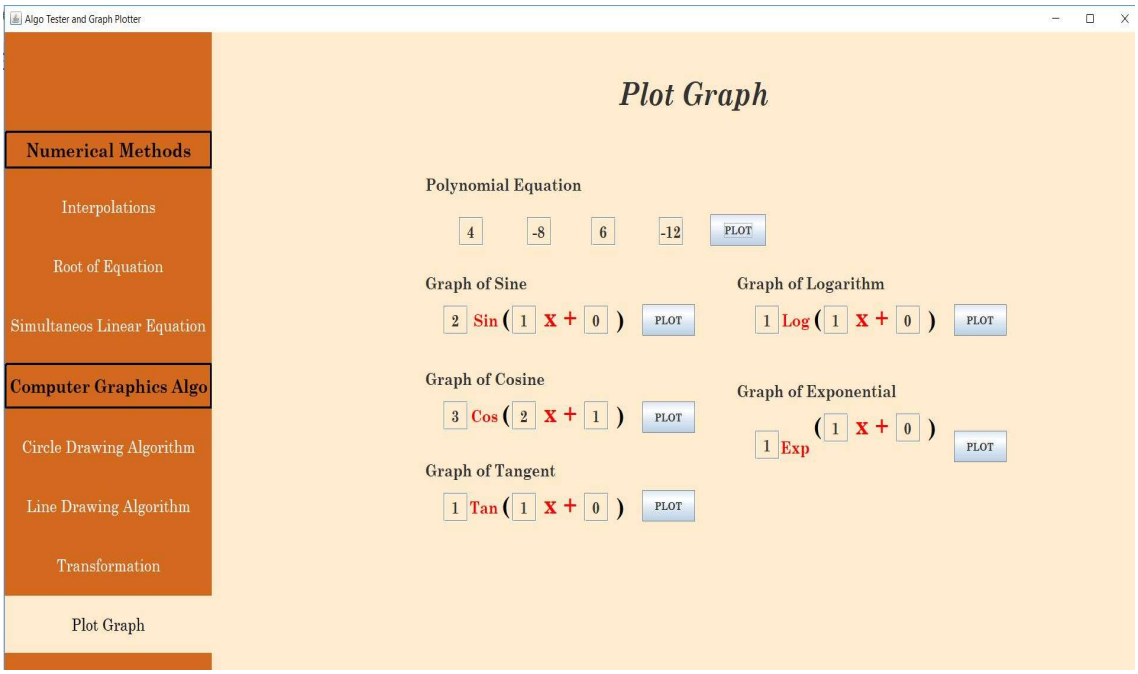
Scaling



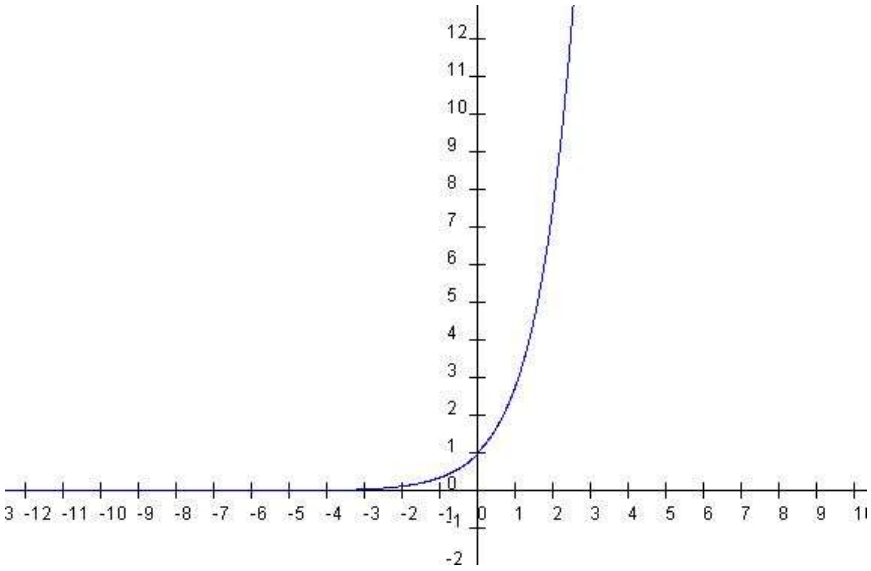
Rotation



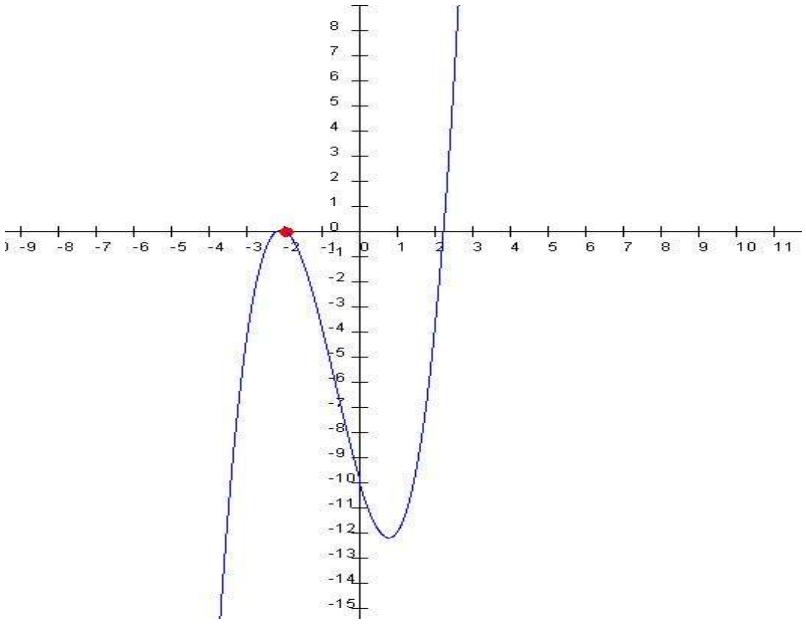
Plot Graph



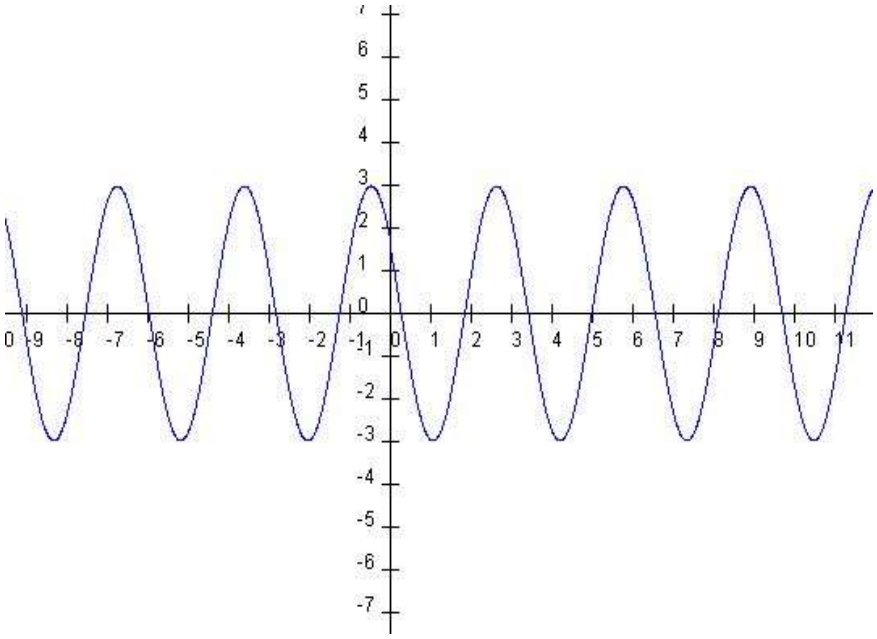
Exponential Graph:



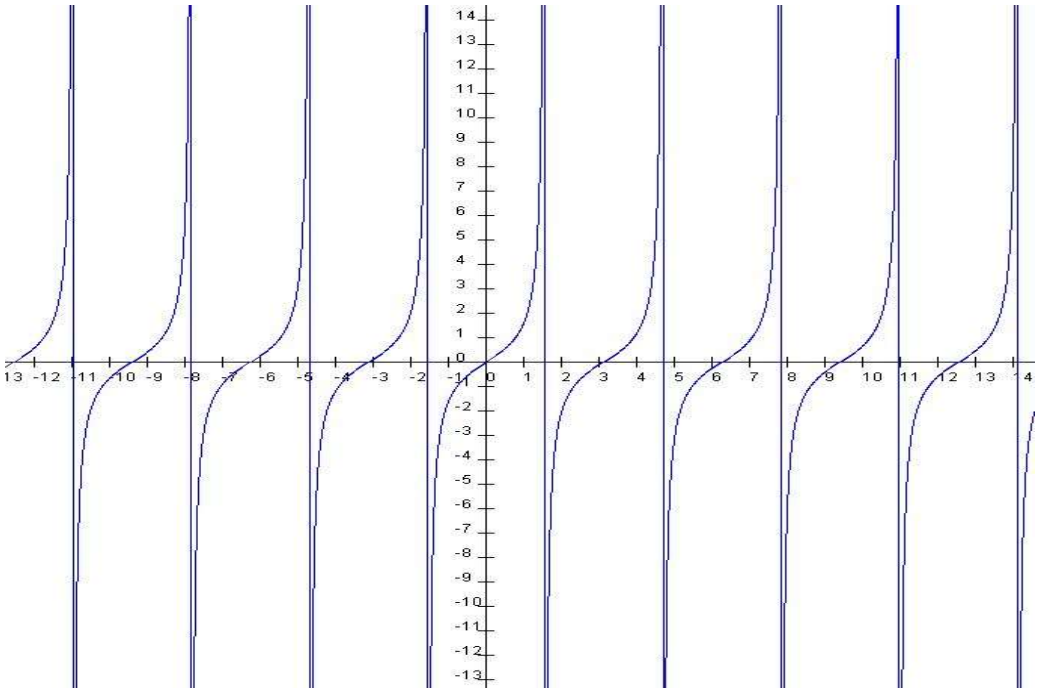
Cubic Polynomial



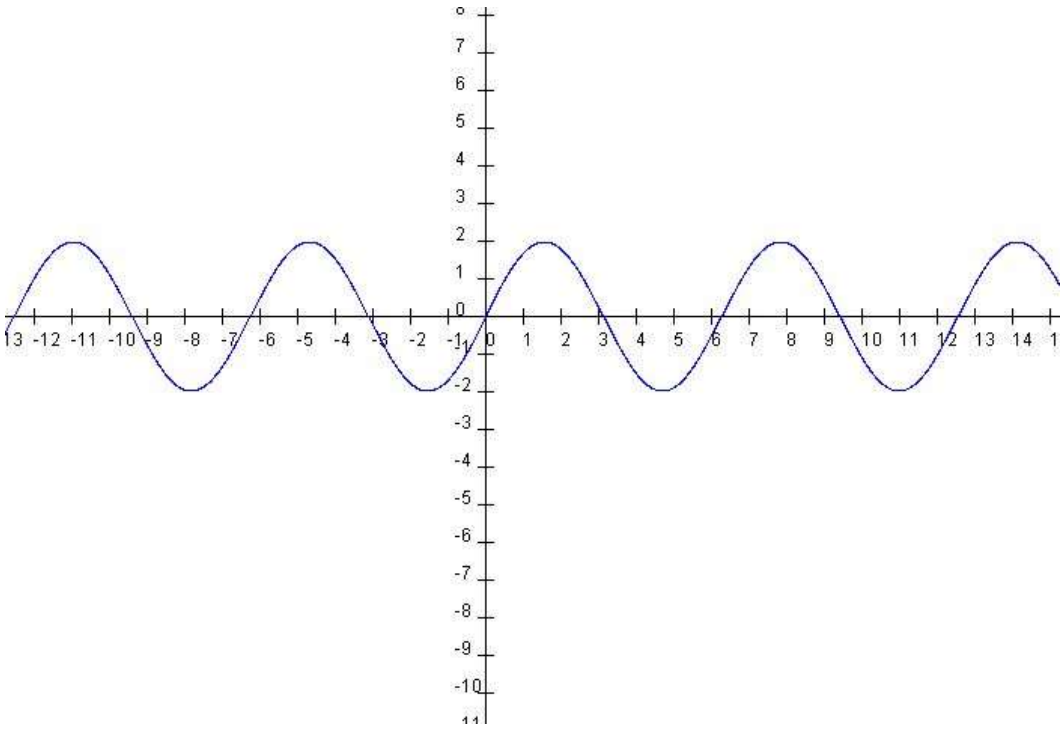
Cosine Function:



Tangent Function:



Sine Function



Logarithmic Function