Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 6s (2025)

# Using GeoGebra and SageMath to Assist Engineering Students in Tracing Different Curves and Solving Numerical Methods

**Bhavna[1], Shubham Ghodke[2] ,Kamalkishor Maniyar[3]**

[1,2,3]First Year Engineering Department, Dr. D. Y. Patil Institute of Technology Pimpri, Pune-18

[1]bhavnasingh2092@gmail.com, [2] shubhamghodake000@gmail.com .[3] kkmaniyar2020@gmail.com

***Article History:***

***Received:*** *23-10-2024*

***Revised:*** *07-12-2024*

***Accepted:*** *14-12-2024*

**Abstract:**

Recently, there has been a growing emphasis on the use of technology and software in the educational process. The purpose of this article is to show readers how to use the software Geogebra and SageMath to trace the different curve and solving numerical methods. This application serves as a helpful tool for instructors and students for projects, training, lectures, and other purposes in addition to being a computer environment. The author discusses numerical techniques, how to make certain graphical representations in this application, and how to trace any curve (Parametric, Polar, and Cartesian) using Geogebra and SageMath.

Traditional techniques of tracing the curve and solving numerical procedures, such as the Gauss-Seidal and Gauss-Jordan methods, might be time-consuming. For this reason, tools like Geogebra and SageMath can make calculations for students, teachers, engineers, and other professionals go more quickly.

In an effort to add to the specialized literature, the author developed fresh examples of abstract usage of GeoGebra and SageMath. The article notes the need for the education system to integrate blended learning and teaching methods, which combine computer programs with traditional teaching methods to improve the teaching of mathematics. It recommends using GeoGebra, SageMath programs for assisted instruction in the teaching and learning of mathematics, specifically for cases of numerical methods and Tracing the curve, respectively.

***Keywords***: GeoGebra, SageMath, Tracing Curve, Numerical Methods.

## 1. Introduction

Engineering is defined as "the application of mathematics and sciences to the construction and design of projects for societal use". Mathematics teaches rational thinking and provides tools for conducting analyses to learn about systems. Traditional curve-tracing and numerical procedure-solving approaches, such as the Gauss-Seidal and Gauss-Jordan methods, may be time-consuming. As a result, Geogebra and SageMath will accelerate computations for students, instructors, engineers, and other professionals.

The article recommends the use of GeoGebra, SageMath, and other related programs for assisted instruction in the teaching and learning of mathematical concepts, for specific cases of numerical methods and traces, respectively; and notes the need for the education system to integrate interconnected teaching and learning methods, in which software programs are used in conjunction with traditional teaching methods, to enhance the teaching process for mathematics.

**Comparison between GeoGebra and SageMath:**

Although GeoGebra and SageMath share many functions, their main differences are in the use cases and target audiences. Due to its emphasis on dynamic visuals and user-friendly interface, GeoGebra is frequently used in educational settings. It's very helpful for interactively teaching and understanding mathematical topics. SageMath is preferred by professionals, academics, and individuals who need a more comprehensive computational toolset due to its open-source nature and wider mathematical coverage.

In conclusion, GeoGebra and SageMath are useful mathematical tools that meet the demands and preferences of many mathematical communities. While SageMath provides an open-source platform that is more adaptable and appropriate for sophisticated mathematical computations and research, GeoGebra shines in interactive instruction and visualization.

## 2. Objectives

**Rapid Visualization**:

- Without the need for tedious computations, students may easily visualize curves and geometric relationships by swiftly creating and manipulating function graphs with **GeoGebra.**
- Compared to conventional approaches, SageMath allows for the quick execution of complex computations, saving time while solving problems.

**Interactive Learning**:

- By experimenting with parameters and seeing instant outcomes, students can learn via exploration and discovery more quickly.

**Automated Calculations**:

- SageMath automates repetitive calculations, allowing students to focus on understanding concepts rather than getting bogged down in arithmetic.

**Streamlined Problem-Solving**:

- Both tools offer built-in functions for numerical methods, making it easier to implement algorithms like Newton’s method, integration, and differential equations.

**Precision of Calculations**:

- **SageMath** supports high-precision arithmetic, reducing rounding errors common in numerical methods and ensuring that results are reliable.

**Graphical Accuracy**:

- **GeoGebra** provides high-quality graphing capabilities, enabling precise tracing of curves and geometric figures, which helps in visualizing mathematical concepts accurately.

## 3. Methods

**GeoGebra:**

GeoGebra is a dynamic mathematics program that integrates algebra, spreadsheets, graphing, calculus, geometry, and other subjects in an engaging and integrated manner. It was created to offer a flexible platform that would enable experts, educators, and students to investigate and illustrate mathematical ideas in an interesting and dynamic way.

The name, GeoGebra, is a play on the words "geometry" and "algebra," signifying its main goal of uniting these two essential areas of mathematics. By letting users make things, work with algebraic formulas, and see the dynamic interactions between various mathematical objects, the program is intended to support experiential learning and mathematical inquiry.

Due to its ability to support active learning, visualization, and the growth of mathematical reasoning, GeoGebra is extensively utilized in education at all levels, from elementary schools to colleges. While students can individually investigate mathematical subjects via practical exercises and experimentation, teachers frequently utilize GeoGebra to construct interactive courses and presentations.

Tracing curves with the traditional approach takes a long time since there are many rules to follow, such as guaranteeing symmetry, identifying if the curve passes through the origin, determining the region of absence, asymptotes, and points of intersection, among others.

To further understand, we may trace any curve using the free and open-source GeoGebra program.

**Example1: Trace the Curve**

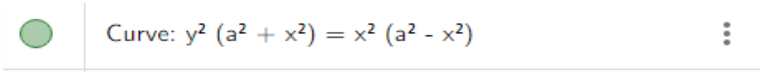
$$Curve: y^2\,(a^2 + x^2) = x^2\,(a^2 - x^2)$$

- Firstly declare the variable ‘a’.
- Write the equation of Curve which we have to trace.
- Draw a Tangent.

**GeoGebra Code:**

- Firstly declare the variable ‘a’.


- Write the equation of Curve which we have to trace.



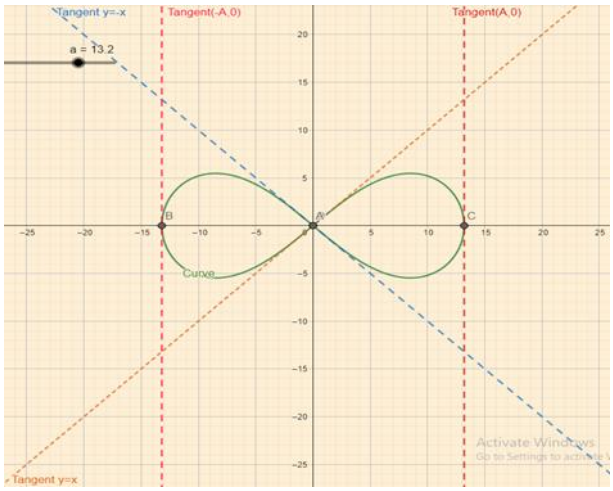



Figure 1:Plotting the Curve $y^2\ (a^2 + x^2) = x^2\ (a^2 - x^2)$

- Draw a tangents


Tracing the curve using SAGEMATH

```
In [3]: var('x,y')
        f(x,y)=y^2*(4+x^2)-x^2*(4-x^2)
```

```
In [4]: pretty_print(f)
```

$$(x, y) \mapsto \left(x^2 - 4\right)x^2 + \left(x^2 + 4\right)y^2$$

```
In [155]: p=implicit_plot(f,(-2,2),(-2,2),color='green',axes_labels=['$x$', '$y$'],title="Cartesian Curve")
```

```
In [149]: pq=point2d([(2,0),(-2,0),(0,0)])                                    #Points of intersectons with Coordinate ax
          es
          p1=plot(x,(x,-3,3),figsize=8,color='blue',linestyle='--')                    #tangent at Origin
          p2=plot(-x,(x,-3,3),figsize=8,color='orange',linestyle='--')                 #tangent at Origin
          p3=implicit_plot(x==2,(x,-3,3),(y,-3,3),figsize=8,color='red',linestyle=':')  #tangent parallel to Y axis
          p4=implicit_plot(x==-2,(x,-3,3),(y,-3,3),figsize=8,color='gray',linestyle=':') #tangent parallel to Y axis
          p5=plot(0,(x,-3,3),figsize=8,color='black')                                  #X-axis
          p6=implicit_plot(x==0,(x,-3,3),(y,-3,3),figsize=8,color='black')              #Y-axis
```

```
In [150]: botleft = dict(horizontal_alignment='left', vertical_alignment='bottom')
          botright = dict(horizontal_alignment='right', vertical_alignment='bottom')
```

```
In [151]: tp = text('x = 2',(1.8, 2.7), **botright)          # Adding lables to Curves
          tp += text('x = -2',(-1.1, 2.7), **botright)
          tp += text('y=x',(2.8, 2.1), **botright)
          tp += text('y=-x',(-2.5, 2.2), **botright)
          tp += text('(-2,0)',(-2, 0), **botright)
          tp += text('(2,0)',(2, 0), **botright)
          tp += text('(0,0)',(0,0), **botright)
```

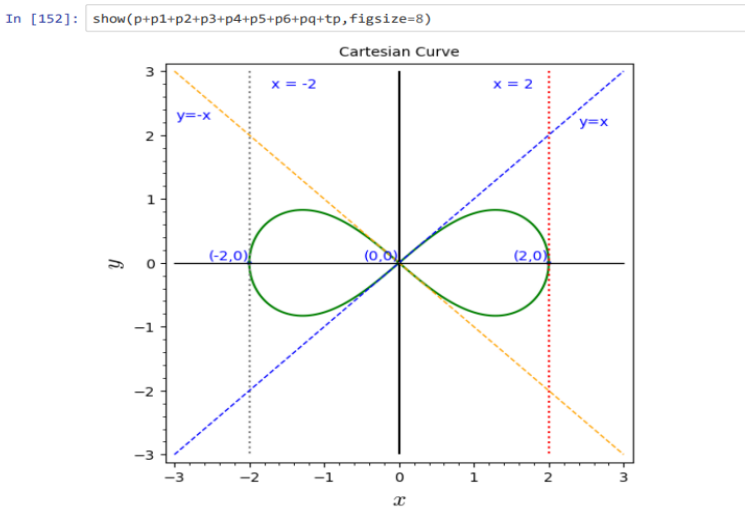



Figure 2:Plotting the Curve $y^2\,(a^2+x^2)=x^2\,(a^2-x^2)$ using Sagemath

**Example 2: Trace the Curve2**

$$y^2\,(a+x)=x^2\,(a-x)$$

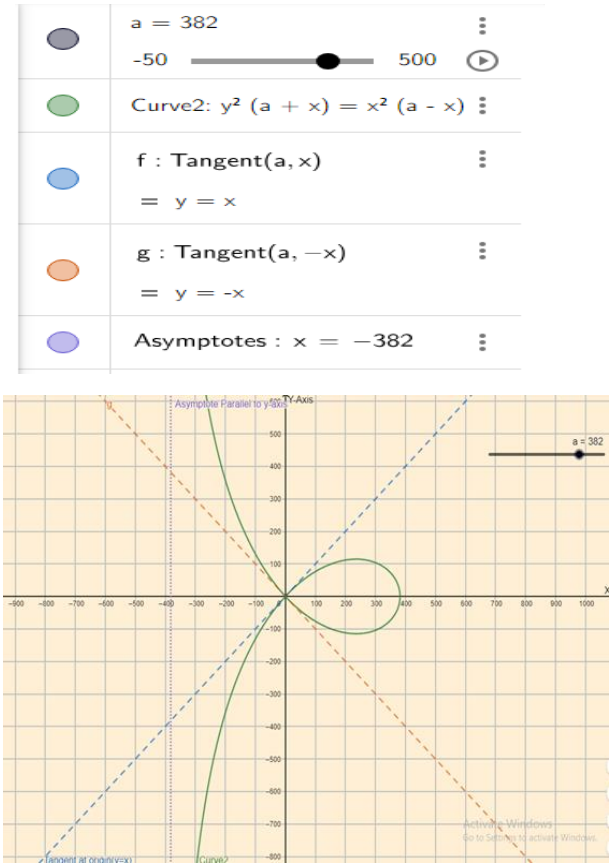



Figure 2:Plotting the $Curve2$: $y^2\,(a+x)=x^2\,(a-x)$

**SageMath:**

A powerful and free mathematics software program, SageMath—often just called Sage—offers a thorough environment for mathematical investigation, study, and calculation. Mathematical topics

covered by SageMath include algebra, calculus, number theory, combinatorics, and more. It was created as an open-source, cost-effective substitute for proprietary mathematical software.

Researchers and professionals working in mathematics, physics, and related subjects are especially fond of SageMath. It offers a strong, flexible, and cost-free substitute for those who need sophisticated computational instruments for their tasks. The software is a great resource for mathematical study and teaching because of its dedication to transparency, cooperation, and adaptability.

Gauss Jacobi and Gauss Siedel methods are numerical mehods for solving system of linear equations. This methods are particularly useful for solving systems of linear equations when the matrix is sparse or when direct methods like Gaussian elimination are computationally expensive. Still for larger systems manual computation of iterations can be very difficult task.for ease of computations in this article we are using sagemath which is open source mathematical software and easily available on any device.

**Guass Jacobi method**

We write the system as $AX = B$

$$Let\ A = L + D + U \qquad \ldots (1)$$

Where L is strictly lower triangular matrix, D is Diagonal matrix and U is strictly upper triangular matrix

Then $AX = B\ reduces\ to\ (L + D + U)X = B\ which\ can\ be\ written\ as\ DX = -(L + U)X + B.$

$If\ D\ is\ invertiable\ then$

$$X = -D^{-1}(L + U)X + D^{-1}B$$

$$Hence\ the\ iterative\ sequence\ is\ given\ by$$

$$X^{(k+1)} = -D^{-1}(L + U)X^{(k)} + D^{-1}B \ldots\ldots (2)$$

$$Thus\ the\ iteration\ matrix\ H = -D^{-1}(L + U)\ and\ C = D^{-1}B \ldots.. (3)$$

**Guass Seidal method**

$$Let\ A = L + D + U$$

$where\ D\ is\ the\ Digonal\ part\ , L$ is strictly lower triangular matrix

and U is strictly upper triangular matrix . Then $\mathrm{AX} = \mathrm{B}$ reduces to

$(\mathrm{L} + \mathrm{D} + \mathrm{U})\mathrm{X} = \mathrm{B}$ which can be written as $(\mathrm{L} + \mathrm{D})\mathrm{X} = -\mathrm{UX} + \mathrm{B}.$

If $\mathrm{L} + \mathrm{D}$ is invertiabe . Then we have

$$\mathrm{X} = -(\mathrm{L} + \mathrm{D})^{-1}UX + (\mathrm{L} + \mathrm{D})^{-1}\mathrm{B}$$

$Hence\ the\ iterative\ sequence\ is\ given\ by$

$$\mathrm{X}^{(\mathrm{k}+1)} = -(\mathrm{L} + \mathrm{D})^{-1}UX^{(k)} + (\mathrm{L} + \mathrm{D})^{-1}\mathrm{B} \ldots . (4)$$

*Thus the iteration matrix* $H == -(\mathrm{L}+\mathrm{D})^{-1}U$ *and* $C = (\mathrm{L}+\mathrm{D})^{-1}\mathrm{B}$ …. (5)

**Example 1) Solve the system of linear equations by using Guass Jacobi method and Gauss seidel methods $5x-2y+3z=-1$ ; $-3x+9y+z=2$ ; $2x-y-7z=3$ , with initial approximation as $x=0, y=0, z=0$.**

**Solution:**

**By Gauss Jacobi Method:**

Firstly define function Gauss_Jacobi which takes inputes as coefficient matrix A ,constant matrix B ,initial approximation column matrix x0, no of iteration N, and tolerance limit.

```
def Gauss_Jacobi(A,B,x0,N,tol=1e-8):
```

Initializing matrices H, C, L, D, U and column matrices x , xold

```
n=A.ncols()
H=zero_matrix(RR,n,n) ##initatializing matrices  H,C,L,D,U
C=zero_matrix(RR,n,n)
L=zero_matrix(RR,n,n)
D=zero_matrix(RR,n,n)
U=zero_matrix(RR,n,n)
x=zero_matrix(RR,n,1)
xold=zero_matrix(RR,n,1)
```

Separate matrix A as mentioned in equation (1),and calculate $D^{-1}$ ,H, C as mentioned in equation (2)

```
for i in range(0,n):       #Separating matrix A as A=L+U+D
    for j in range(0,n):
        if (i==j):
            D[i,j]=A[i,j]
        elif(i<j):
            U[i,j]=A[i,j]
        else:
            L[i,j]=A[i,j]
D1=D.inverse()            #Applying formulae to Create matrices D1,H,C
H=-D1*L-D1*U
C=D1*B
xold=x0
pretty_print("H= ",H)
pretty_print("C=",C)
pretty_print("D1=",D1)
```

Lastly we apply iteration scheme as mentioned in equation (2)

```
for i in range(1,N):   #Applying Gauss Jacobi iteration scheme
    x=H*xold+C
    err = norm(x-xold) # error term to compare with tolerance
    if(err<tol):
        break
    xold=x
    pretty_print('The {0} iteration is {1}'.format(i,x))
```

Now we give input as mentioned in example with the initial approximation and tolerance limit 0.0001.

```
A=matrix(RR,[[5,-2,3],[-3,9,1],[2,-1,-7]])
B=matrix(RR,[[-1],[2],[3]])
x0=matrix(RR,[[0],[0],[0]])

Gauss_Jacobi(A,B,x0,10,tol=0.0001)
```

**Output:**

$$H=\begin{pmatrix} 0.000000000000000 & 0.400000000000000 & -0.600000000000000 \\ 0.333333333333333 & 0.000000000000000 & -0.111111111111111 \\ 0.285714285714286 & -0.142857142857143 & 0.000000000000000 \end{pmatrix}$$

$$C=\begin{pmatrix} -0.200000000000000 \\ 0.222222222222222 \\ -0.428571428571429 \end{pmatrix}$$

$$D1=\begin{pmatrix} 0.200000000000000 & 0.000000000000000 & 0.000000000000000 \\ 0.000000000000000 & 0.111111111111111 & 0.000000000000000 \\ -0.000000000000000 & -0.000000000000000 & -0.142857142857143 \end{pmatrix}$$

The 1 iteration is [-0.200000000000000] [ 0.222222222222222] [-0.428571428571429]

The 2 iteration is [ 0.146031746031746] [ 0.203174603174603] [-0.517460317460317]

The 3 iteration is [ 0.191746031746032] [ 0.328395061728395] [-0.415873015873016]

The 4 iteration is [ 0.180881834215168] [ 0.332345679012346] [-0.420700428319476]

The 5 iteration is [ 0.185358528596624] [ 0.329260658996109] [-0.424368858654573]

The 6 iteration is [ 0.186325578791187] [ 0.331160493827161] [-0.422649085971837]

The 7 iteration is [ 0.186053649113967] [ 0.331291758038378] [-0.422644190892112]

The 8 iteration is [ 0.186103217750618] [ 0.331200570914890] [-0.422740637115778]

**Hence the solution is $x = 0.1861, y = 0.3312, z = -0.4227$**

**By Gauss Seidel Method: Function:**

```
def Gauss_Seidel(A,B,x0,N,tol=1e-8):
    n=A.ncols()
    H=zero_matrix(RR,n,n)  ##initatializing matrices  H,C,L,D,U
    C=zero_matrix(RR,n,n)
    L=zero_matrix(RR,n,n)
    D=zero_matrix(RR,n,n)
    U=zero_matrix(RR,n,n)
    x=zero_matrix(RR,n,1)
    xold=zero_matrix(RR,n,1)
    for i in range(0,n):    #Separating matrix A as A=L+U+D
        for j in range(0,n):
            if (i==j):
                D[i,j]=A[i,j]
            elif(i<j):
                U[i,j]=A[i,j]
            else:
                L[i,j]=A[i,j]
    H=-(D+L).inverse()*U          #Applying formulae to Create matrices H,C
    C=(D+L).inverse()*B
    xold=x0
    pretty_print("H= ",H)
    pretty_print("C=",C)
    for i in range(1,N): #Applying Gauss Seidel iteration scheme
        x=H*xold+C
        err = norm(x-xold) # error term to compare with tolerance
        if(err<tol):
            break
        xold=x
        pretty_print('The {0} iteration is {1}'.format(i,x))
    return
```

**Input:**

```
A=matrix(RR,[[5,-2,3],[-3,9,1],[2,-1,-7]])
B=matrix(RR,[[-1],[2],[3]])
x0=matrix(RR,[[0],[0],[0]])

Gauss_Seidel(A,B,x0,11,tol=0.0001)
```

**Output:**

$$H=\begin{pmatrix} 0.000000000000000 & 0.400000000000000 & -0.600000000000000 \\ 0.000000000000000 & 0.133333333333333 & -0.311111111111111 \\ 0.000000000000000 & 0.0952380952380952 & -0.126984126984127 \end{pmatrix}$$

$$C=\begin{pmatrix} -0.200000000000000 \\ 0.155555555555556 \\ -0.507936507936508 \end{pmatrix}$$

The 1 iteration is [-0.200000000000000] [ 0.155555555555556] [-0.507936507936508]

The 2 iteration is [ 0.166984126984127] [ 0.334320987654321] [-0.428621819098010]

The 3 iteration is [ 0.190901486520534] [ 0.333480697628846] [-0.421668246369682]

The 4 iteration is [ 0.186393226873348] [ 0.331205325221081] [-0.422631267353484]

The 5 iteration is [ 0.186060890500522] [ 0.331201548761672] [-0.422725681108661]

## 4. Results

By leveraging the efficiency of GeoGebra and SageMath, engineering students can enhance their learning experience, leading to quicker mastery of complex concepts. The accuracy provided by these tools ensures that students can trust their results, which is crucial for their future work in engineering. Together, these tools create a powerful educational environment that fosters both understanding and application of mathematical principles.

The visualization of difficult mathematical concepts and the use of numerical techniques are two major benefits of integrating GeoGebra and SageMath into engineering education. With the help of GeoGebra's dynamic geometry features, students may change and study curves with ease, developing a greater grasp of their characteristics.

In the meantime, SageMath offers a strong foundation for carrying out numerical analysis and symbolic computations, empowering students to efficiently solve equations and simulate real-world issues. Using these tools, students can investigate the connections between various mathematical entities, see how their calculations are shown, and get practical experiences addressing engineering challenges. In addition to enhancing their analytical abilities, this combination gets them ready for real-world work in the future.

## References

[1] http://www.jstor.org/stable/10.5951/mathteacher.106.3.0228
[2] https://www.geogebra.org/.
[3] https://issuu.com/ijasrweditor/docs/calculations_ofsometrigonometricinterpolationpolyn