Authors:
Camilo Soto (cesotor@eafit.edu.co)
Abraham Lora (amlorav@eafit.edu.co)
Sebastian Escobar (sescobarg6@eafit.edu.co)

User Manual
NUMETRIFY

---

1 Introduction

Numetrify is an interactive web application designed to facilitate the implementation and visualization of various numerical methods for solving equations and systems of equations. Aimed at students and researchers, Numetrify provides an intuitive interface for inputting matrices and vectors, allowing users to easily manipulate and modify their data. The application leverages advanced graphical and tabular representations to display results, making complex numerical computations more accessible and understandable. Whether you are working on academic studies or conducting research projects, Numetrify serves as a powerful tool to apply numerical methods effectively, enhancing both learning and productivity in the field of numerical analysis :

1.1 Roots Methods

Bisection, False Rule, Fixed Point, Incremental Search, Multiple Roots, Newton-Raphson, Secant.

1.2 Matrix Methods

Cholesky, Crout, Doolittle, Simple Gaussian Elimination, Pivot Gaussian Elimination, LU Gaussian Elimination, Jacobi, Gauss-Seidel.

1.3 Enter Data

To enter data on root methods or matrix methods like a function the user should do at this form:

- Sum: `a + b`
- Subtraction: `a - b`
- Multiplication: `a * b`
- Division: `a / b`
- Exponential: `e^a`
- Power: `a^b`
- Square Root: `sqrt(a)`

1.4 Troubleshooting

1.4.1 Error Entering Function

- Possible Cause: The function definition does not follow the correct format.
- Solution: Review the function syntax and ensure you use valid operators and the variable name `x`.

1.4.2 Error Entering Values

- Possible Cause: The entered value is not numeric.
- Solution: Ensure the value is numeric.

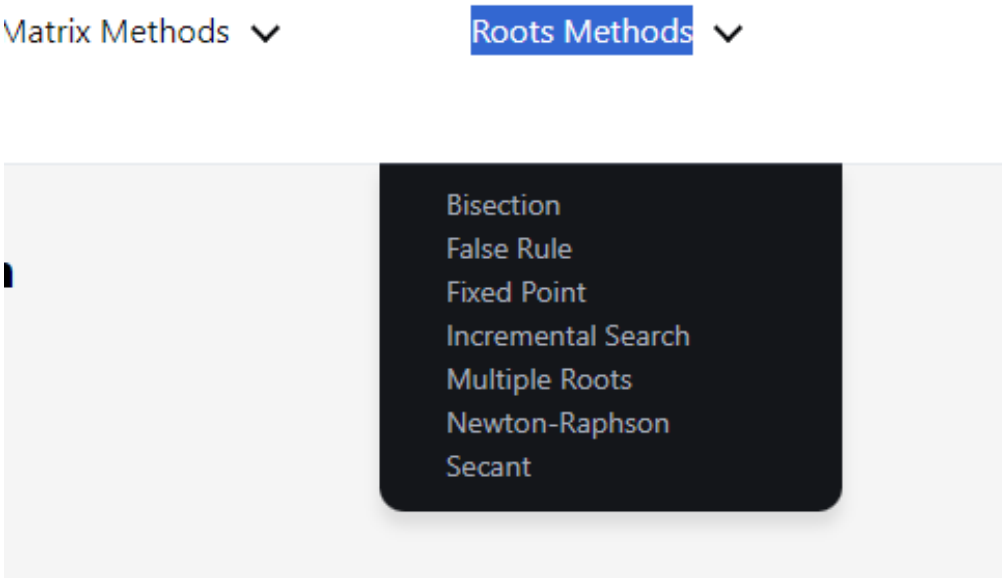
2 Menu

2.1 Roots Methods:

By clicking on this menu, we find the 7 Roots Methods, select by clicking on the desired one. After choosing a method, you will insert data entries that share the root methods which are:

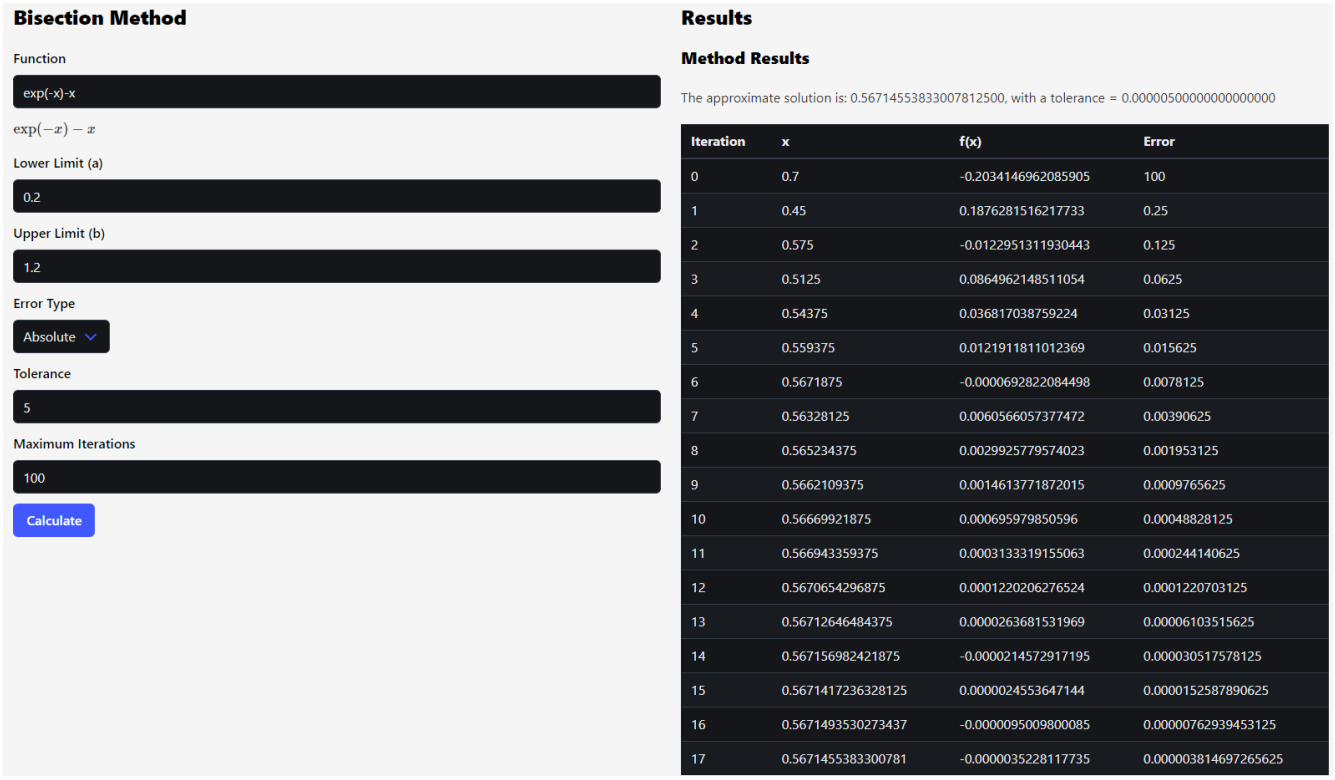
- To enter a function it must be done as indicated above and it must only be with the variable x or numerical values.
- The user must choose the type of precision and type of error, to do so they must click where the option is and select the one they prefer.
- You must also indicate the tolerance and the maximum number of iterations. The only thing you have to enter is a number in each space.

After inserting all the data on the right side of the page of each method it will show you the result with its points and the graph



2.1.1 Bisection

For bisection, the lower bound and the upper bound are added to which a numerical value must be entered. And here are the examples:



Bisection Method

Function: exp(-x)-x

$\exp(-x) - x$

Lower Limit (a): 0.2

Upper Limit (b): 1.2

Error Type: Absolute

Tolerance: 5

Maximum Iterations: 100

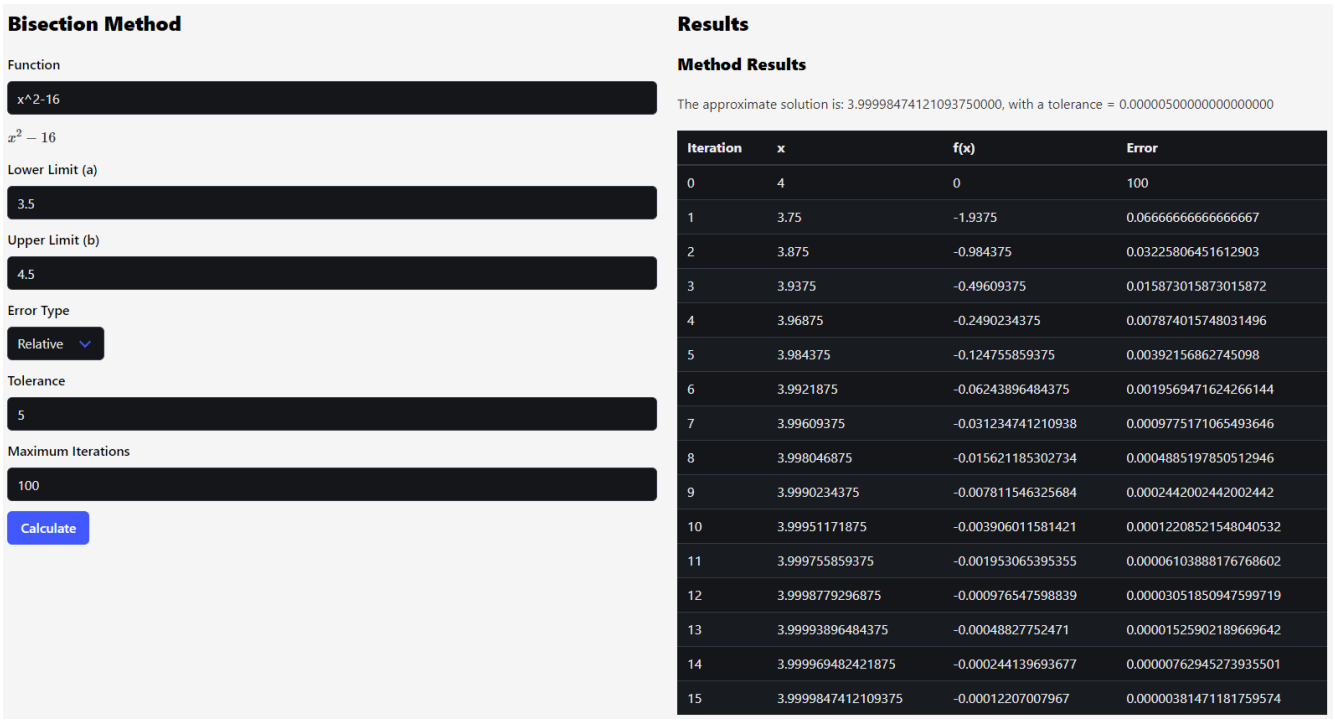
Calculate

Results

Method Results

The approximate solution is: 0.56714553833007812500, with a tolerance = 0.00000500000000000000

Iteration	x	f(x)	Error
0	0.7	-0.2034146962085905	100
1	0.45	0.1876281516217733	0.25
2	0.575	-0.0122951311930443	0.125
3	0.5125	0.0864962148511054	0.0625
4	0.54375	0.036817038759224	0.03125
5	0.559375	0.0121911811012369	0.015625
6	0.5671875	-0.0000692822084498	0.0078125
7	0.56328125	0.0060566057377472	0.00390625
8	0.565234375	0.0029925779574023	0.001953125
9	0.5662109375	0.0014613771872015	0.0009765625
10	0.56669921875	0.000695979850596	0.00048828125
11	0.566943359375	0.0003133319155063	0.000244140625
12	0.5670654296875	0.0001220206276524	0.0001220703125
13	0.56712646484375	0.0000263681531969	0.00006103515625
14	0.567156982421875	-0.0000214572917195	0.000030517578125
15	0.5671417236328125	0.0000024553647144	0.0000152587890625
16	0.5671493530273437	-0.0000095009800085	0.00000762939453125
17	0.5671455383300781	-0.0000035228117735	0.000003814697265625



Bisection Method

Function: x^2-16

$x^2 - 16$

Lower Limit (a): 3.5

Upper Limit (b): 4.5

Error Type: Relative

Tolerance: 5

Maximum Iterations: 100

Calculate

Results

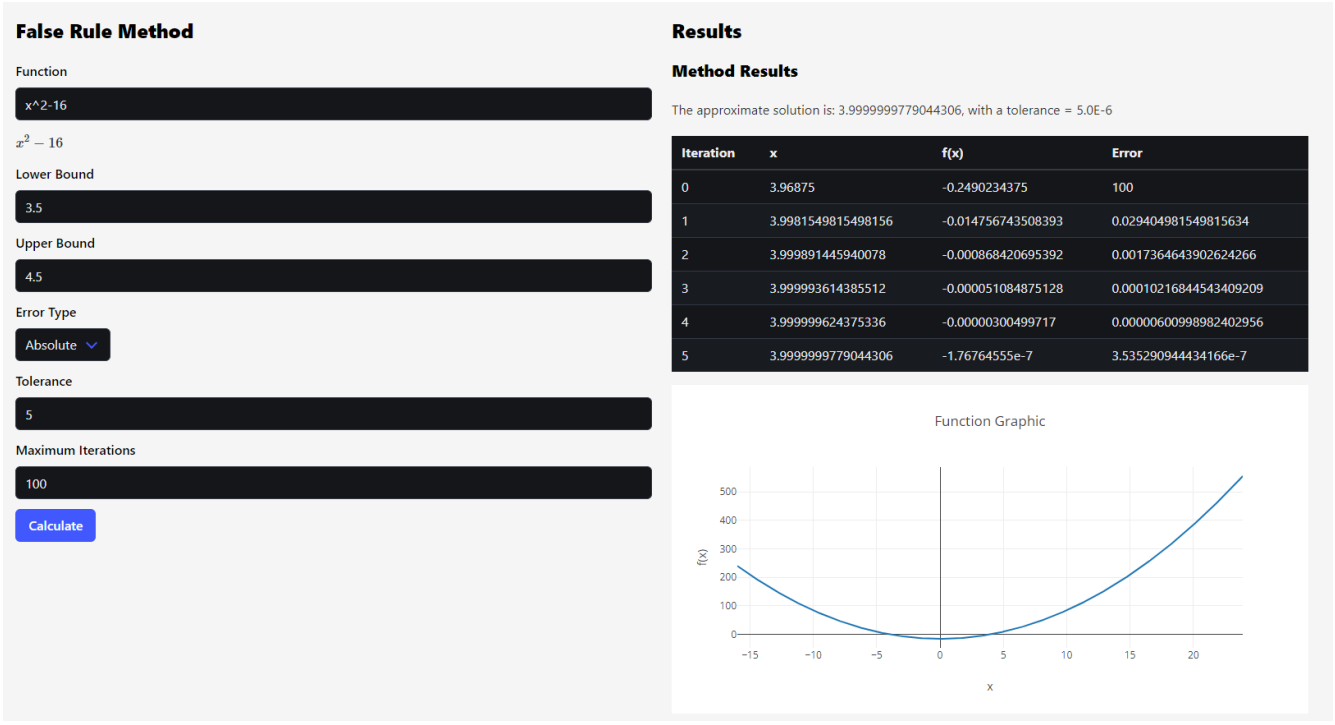
Method Results

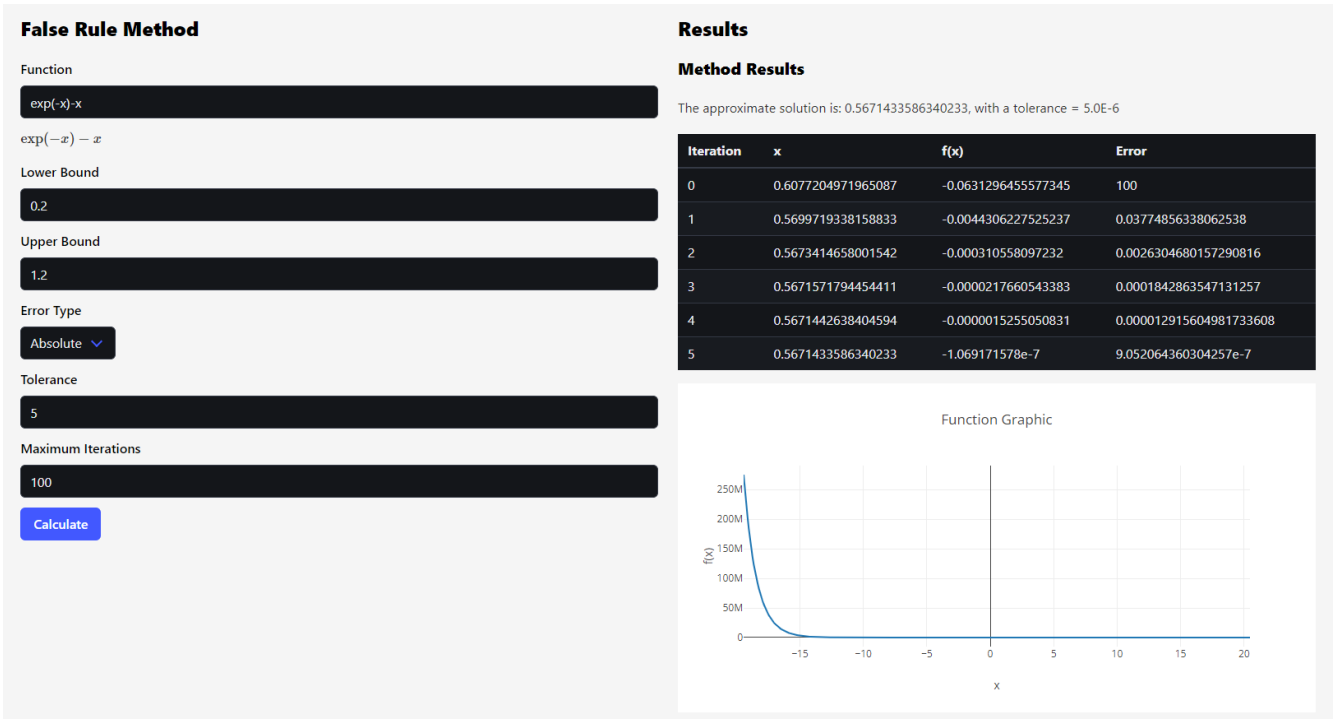
The approximate solution is: 3.99998474121093750000, with a tolerance = 0.00000500000000000000

Iteration	x	f(x)	Error
0	4	0	100
1	3.75	-1.9375	0.06666666666666667
2	3.875	-0.984375	0.03225806451612903
3	3.9375	-0.49609375	0.015873015873015872
4	3.96875	-0.2490234375	0.007874015748031496
5	3.984375	-0.124755859375	0.00392156862745098
6	3.9921875	-0.06243896484375	0.001956947162426614
7	3.99609375	-0.031234741210938	0.0009775171065493646
8	3.998046875	-0.015621185302734	0.000488519785051294
9	3.9990234375	-0.007811546325684	0.0002442002442002442
10	3.99951171875	-0.003906011581421	0.0001220852154804053
11	3.999755859375	-0.001953065395355	0.00006103888176768602
12	3.9998779296875	-0.000976547598839	0.00003051850947599719
13	3.99993896484375	-0.00048827752471	0.00001525902189669642
14	3.999969482421875	-0.000244139693677	0.00000762945273935501
15	3.9999847412109375	-0.00012207007967	0.00000381471181759574

2.1.2 False Rule

For false rule, the lower bound and the upper bound are added to which a numerical value must be entered. And here are the examples:





2.1.3 Fixed Point

For fixed point, a function g(x) must be added which must follow the same instructions as all functions. An initial guess must also be entered which must be numerical values. And here are the examples:

Fixed Point Method

Function

x^2-4

$x^2 - 4$

g(x) Function

x - ((x^2-4)/(2*x))

$x - \left(\frac{(x^2 - 4)}{(2 \cdot x)}\right)$

Initial Guess

3.5

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

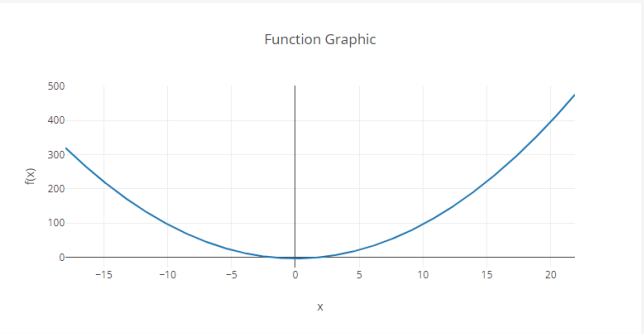
Calculate

Results

Method Results

2.0 is a root of f(x)

Iteration	x	f(x)	Error
0	3.5	8.25	100
1	2.3214285714285716	1.389030612244898	1.1785714285714284
2	2.0222527472527476	0.0895061737712854	0.299175824175824
3	2.000122433944265	0.000489750767131	0.022130313308482474
4	2.0000000037472883	1.4989153e-8	0.00012243019697688595
5	2	0	3.747288257471837e-9



Fixed Point Method

Function

cos(x)-x

$\cos(x) - x$

g(x) Function

cos(x)

$\cos(x)$

Initial Guess

0.5

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

Calculate

Results

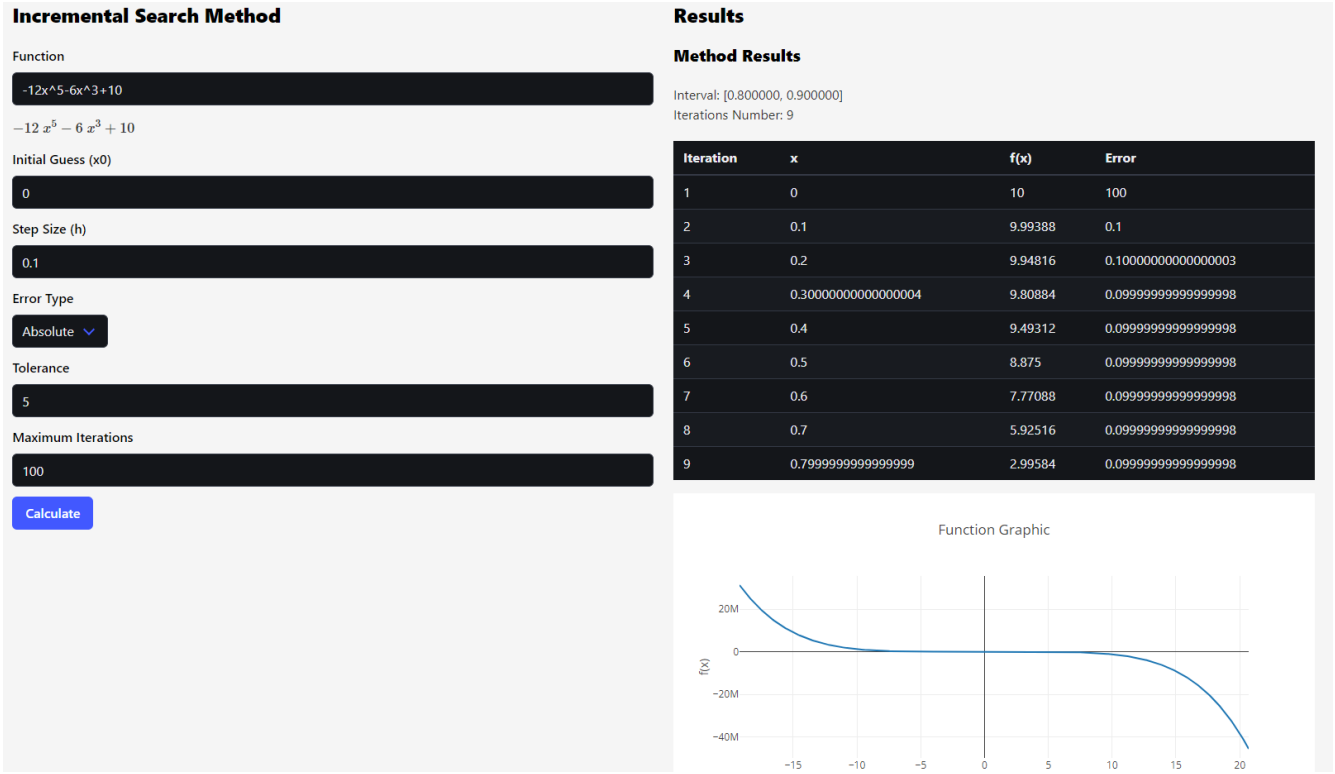
Method Results

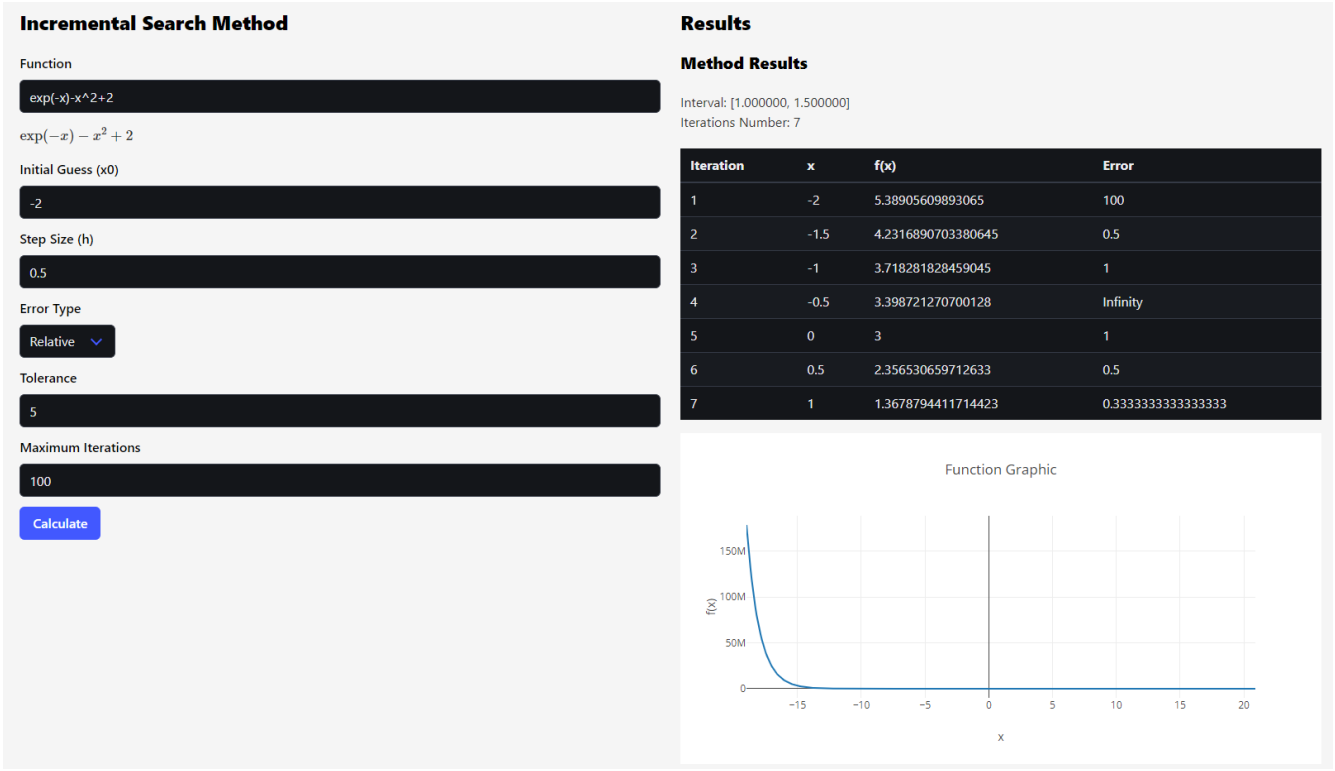
The approximate solution is: 0.7390836261034798, with a tolerance = 5.0E-6

Iteration	x	f(x)	Error
0	0.5	0.3775825618903728	100
1	0.8775825618903728	-0.2385700677251136	0.37758256189037276
2	0.6390124941652592	0.1636726065170757	0.23857006772511358
3	0.8026851006823349	-0.1079070738943287	0.1636726065170757
4	0.6947780267880062	0.0734178044940099	0.10790707389432874
5	0.7681958312820161	-0.0490303853395971	0.07341780449400992
6	0.719165445942419	0.033190313479108	0.04903038533959703
7	0.752355759421527	-0.0222746962837037	0.03319031347910795
8	0.7300810631378233	0.0150392782136168	0.02227469628370371
9	0.7451203413514401	-0.010114032336597	0.015039278213616836
10	0.7350063090148431	0.0068202136284028	0.010114032336597023
11	0.7418265226432459	-0.0045907972010145	0.006820213628402794
12	0.7372357254422314	0.0030939264360318	0.00459079720101 4518
13	0.7403296518782632	-0.0020834135460297	0.003093926436031791
14	0.7382462383322335	0.0014037244374277	0.0020834135460297
15	0.7396499627696612	-0.0009454234126779	0.0014037244374277558
16	0.7387045393569833	0.0006369129242268	0.0009454234126778926
17	0.7393414522812101	-0.000429002949107	0.0006369129242267402

- **Incremental Search:**

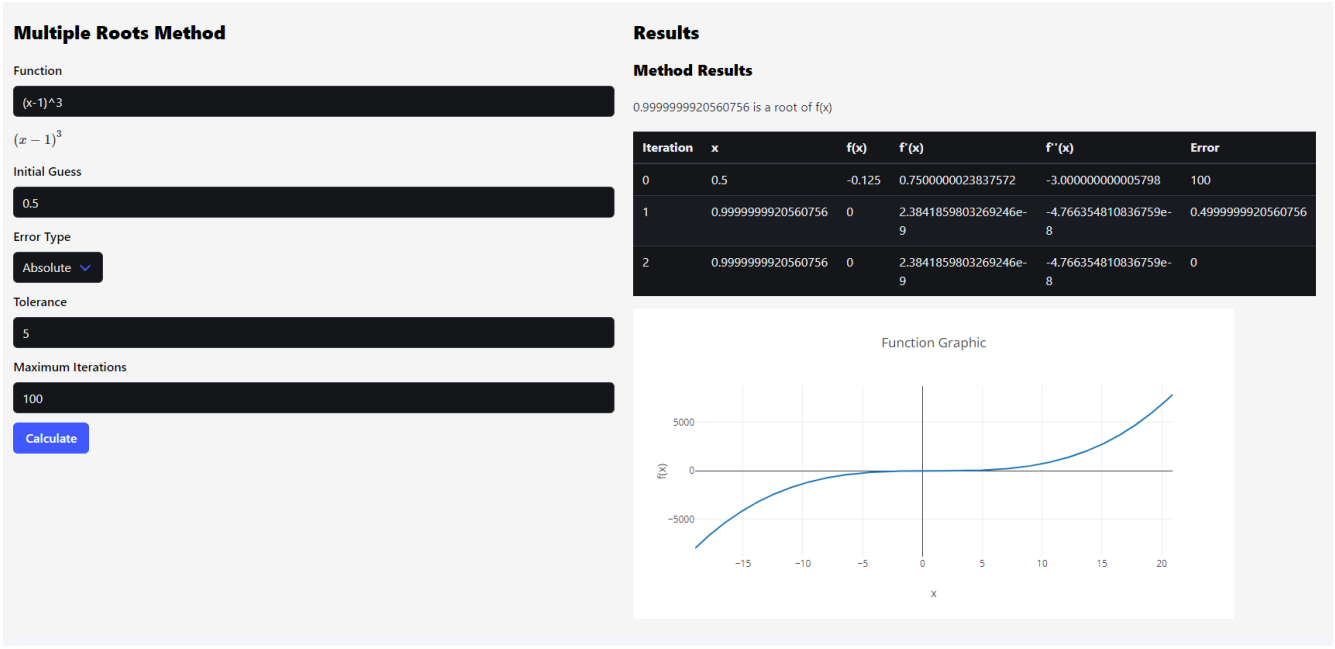
For incremental search, an initial guess and the step size must also be entered which must be numerical values. And here are the examples:

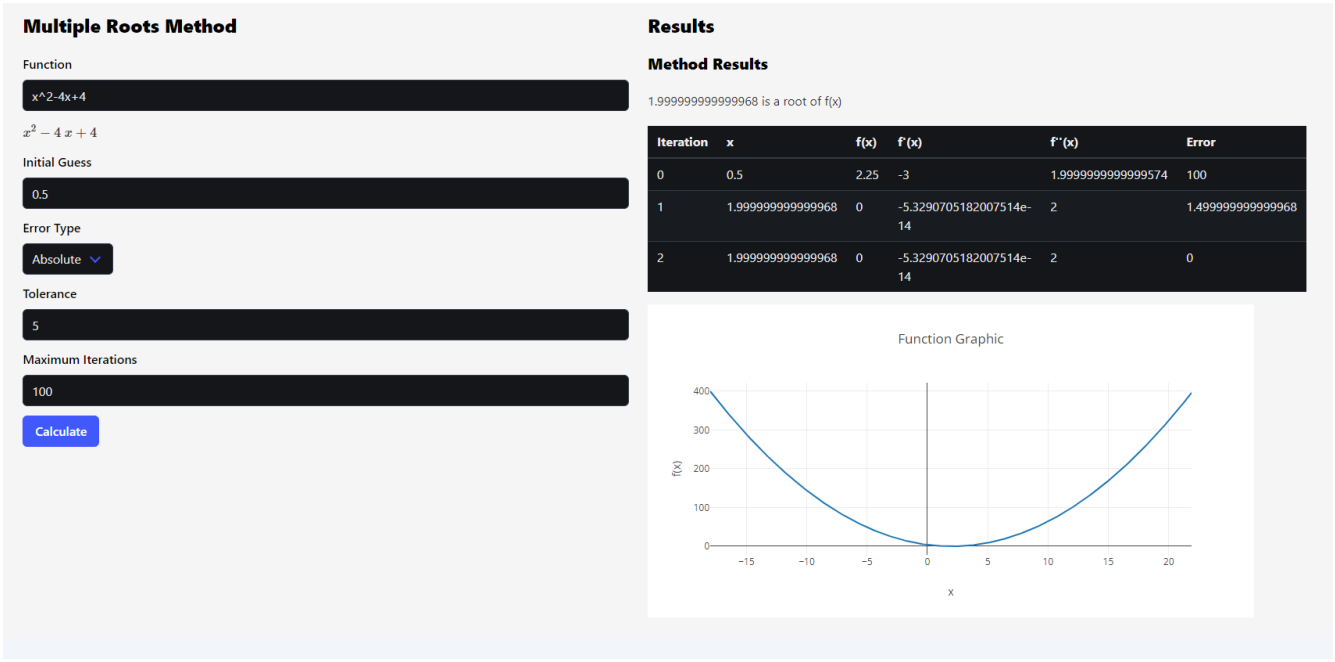




- **Multiple Roots:**

For multiple roots, an initial guess must be entered which must be numerical values. And here are the examples:





- **Newton-Raphson:**

For Newton-Raphson, an initial guess must be entered which must be numerical values. And here are the examples:

Newton-Raphson Method

Function

x^2-2

$x^2 - 2$

Initial Guess

1

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

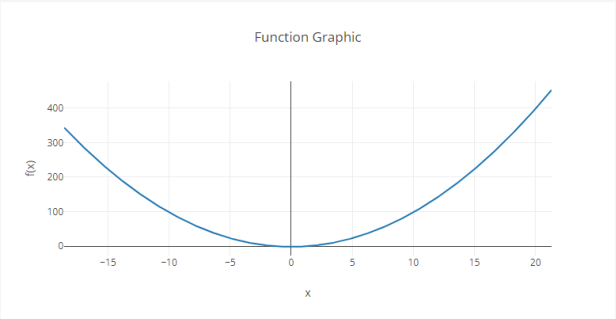
Calculate

Results

Method Results

The approximate solution is: 1.4142135623747667, with a tolerance = 5.0E-6

Iteration	x	f(x)	f'(x)	Error
0	1	-1	2.0000000999775835	100
1	1.4999999750056054	0.249999925016817	3.000000050046303	0.49999997500560545
2	1.4166666680568412	0.006944448383272	2.8333334400020007	0.08333330694876429
3	1.4142156863667839	0.0000060075658737	2.8284314750000017	0.002450981690057308
4	1.4142135623747667	4.728e-12	2.82842723	0.000002123992017155629 2



Newton-Raphson Method

Function

exp(x)-4x

$\exp(x) - 4\,x$

Initial Guess

1

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

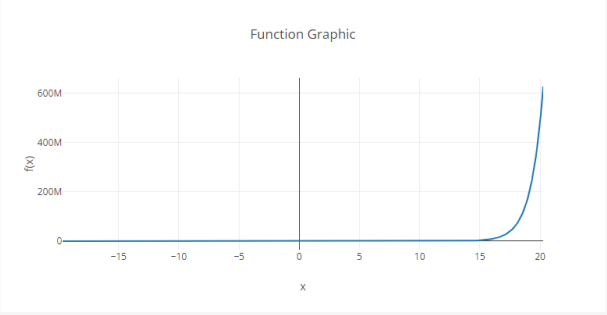
Calculate

Results

Method Results

0.35740295618138906 is a root of f(x)

Iteration	x	f(x)	f'(x)	Error
0	1	-1.281718171540955	-1.281718031709289	100
1	-1.0909705738804121e-7	1.0000003272911782	-3.000000059483199	1.0000001090970574
2	0.333333326724089	0.0622791089657901	-2.604387513982509	0.3333334358211464
3	0.35724647660344616	0.0004022307585394	-2.5706117929999657	0.023913149879357165
4	0.35740294937760625	1.74883625e-8	-2.5703881130000004	0.00015647277416008798
5	0.35740295618138906	0	-2.570388109	6.803782814923664 5e-9



- **Secant:**

For Secant, 2 initial guess must be entered which must be numerical values. And here are the examples:

Secant Method

Function

x^2-7x^2+14x-6

$x^2 - 7\,x^2 + 14\,x - 6$

Initial Guess 1

0.5

Initial Guess 2

1.5

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

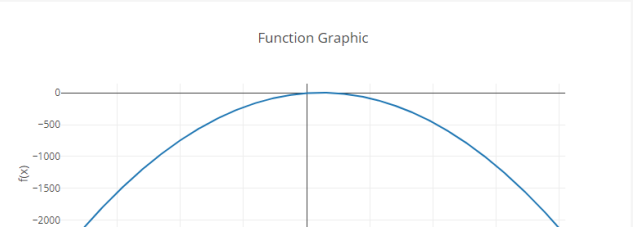
Calculate

Results

Method Results

0.5657414540893531 is a root of f(x)

Iteration	x	f(x)	Error
0	0.5	-0.5	100
1	1.5	1.5	1
2	0.75	1.125	0.75
3	-1.5	-40.5	2.25
4	0.689189189189189	0.798758217677135	2.189189189189189
5	0.6468481375358166	0.545398847300104	0.04234105165337243
6	0.5557018707493997	-0.073001224434699	0.09114626678641691
7	0.566461554297615	0.005189605182991	0.010759683548215326
8	0.5657474231164755	0.000043043053063	0.000714131181139499
9	0.5657414505107734	-2.58053765e-8	0.00000597260570212032 6
10	0.5657414540893531	0	3.578579654828218e-9



Secant Method

Function

x^3-2x+x

$x^3 - 2\,x + x$

Initial Guess 1

-2

Initial Guess 2

-1

Error Type

Absolute

Tolerance

5

Maximum Iterations

100

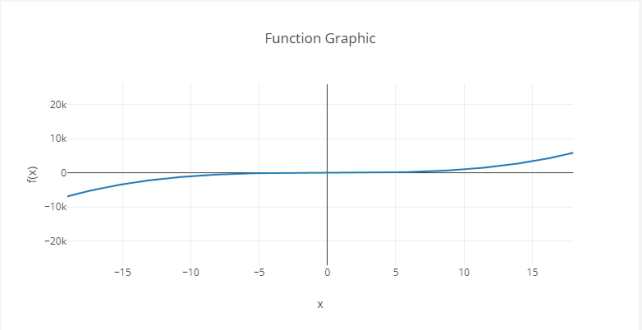
Calculate

Results

Method Results

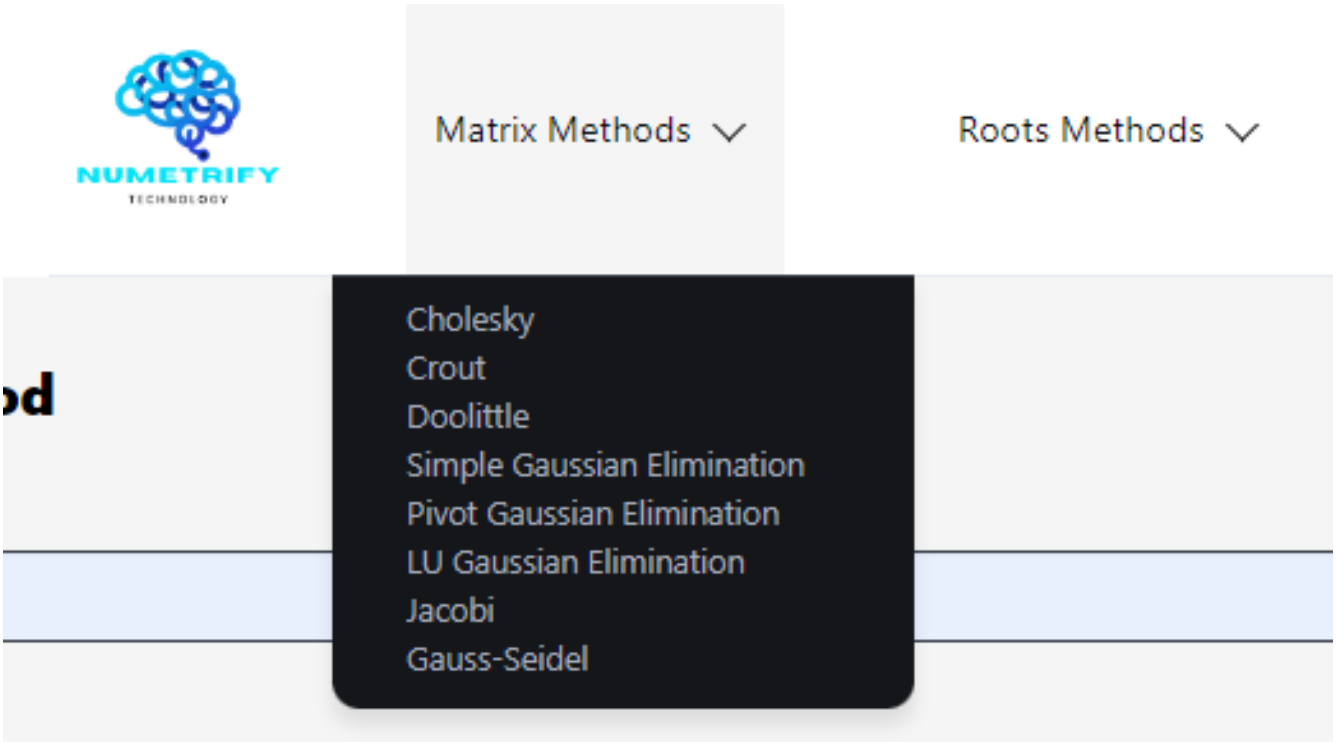
-1.0 is a root of f(x)

Iteration	x	f(x)	Error
0	-2	-6	100
1	-1	0	1



Matrix Methods:

By clicking on this menu, we find the 8 matrix methods, select by clicking on the desired one.



After selecting the matrix method, there are things in common which would be to enter the numerical data into the nxn matrix and the vector of size n and the matrix can be expanded in size with the right click but it must always be square. After that the user have to click on calculate and you can see the answer on the right side and for the methods that return LU matrices the same will be shown

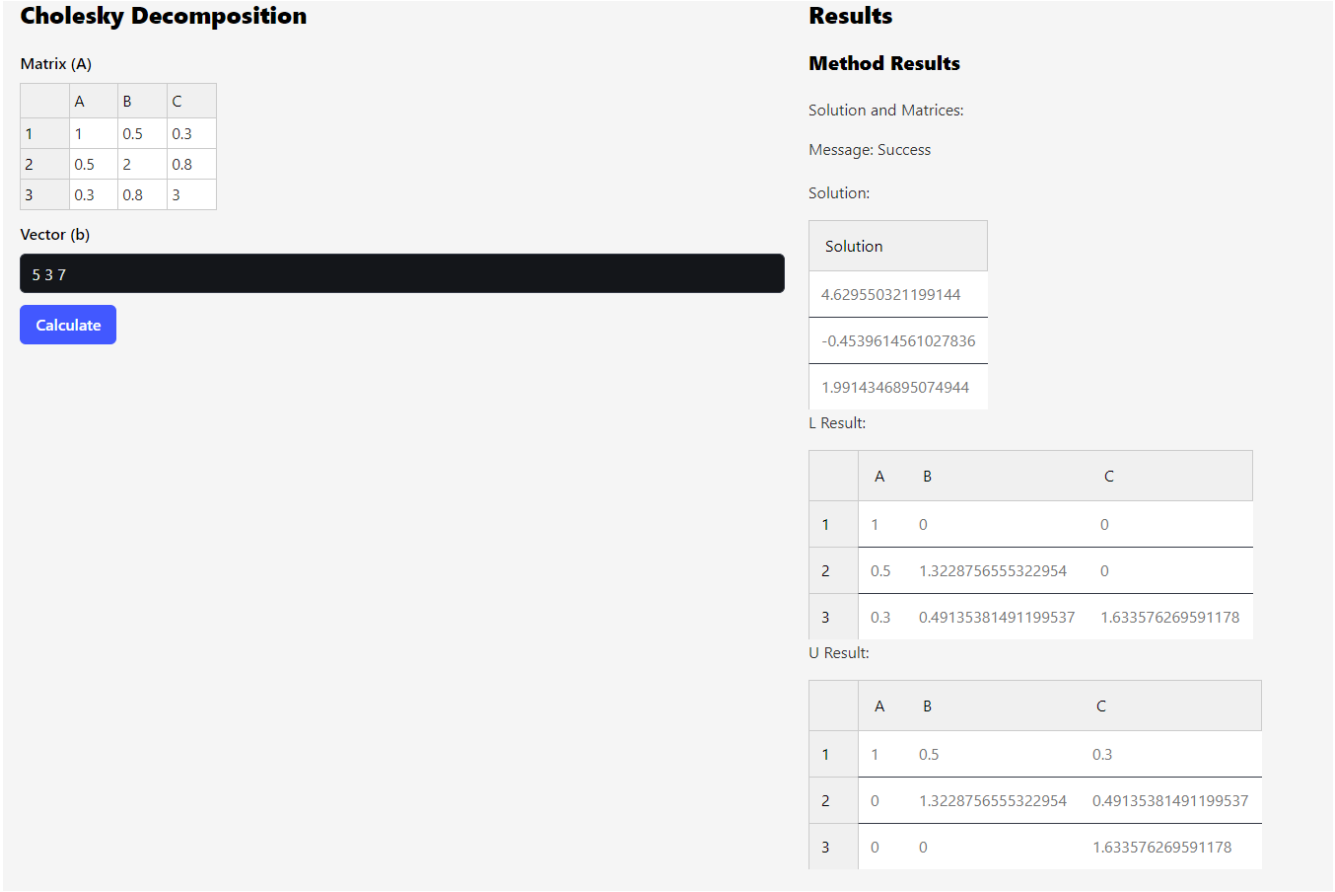
$$B = \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 2 & 0.8 \\ 0.3 & 0.8 & 3 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 5 \\ 3 \\ 7 \end{bmatrix}$$

- **Cholesky:**

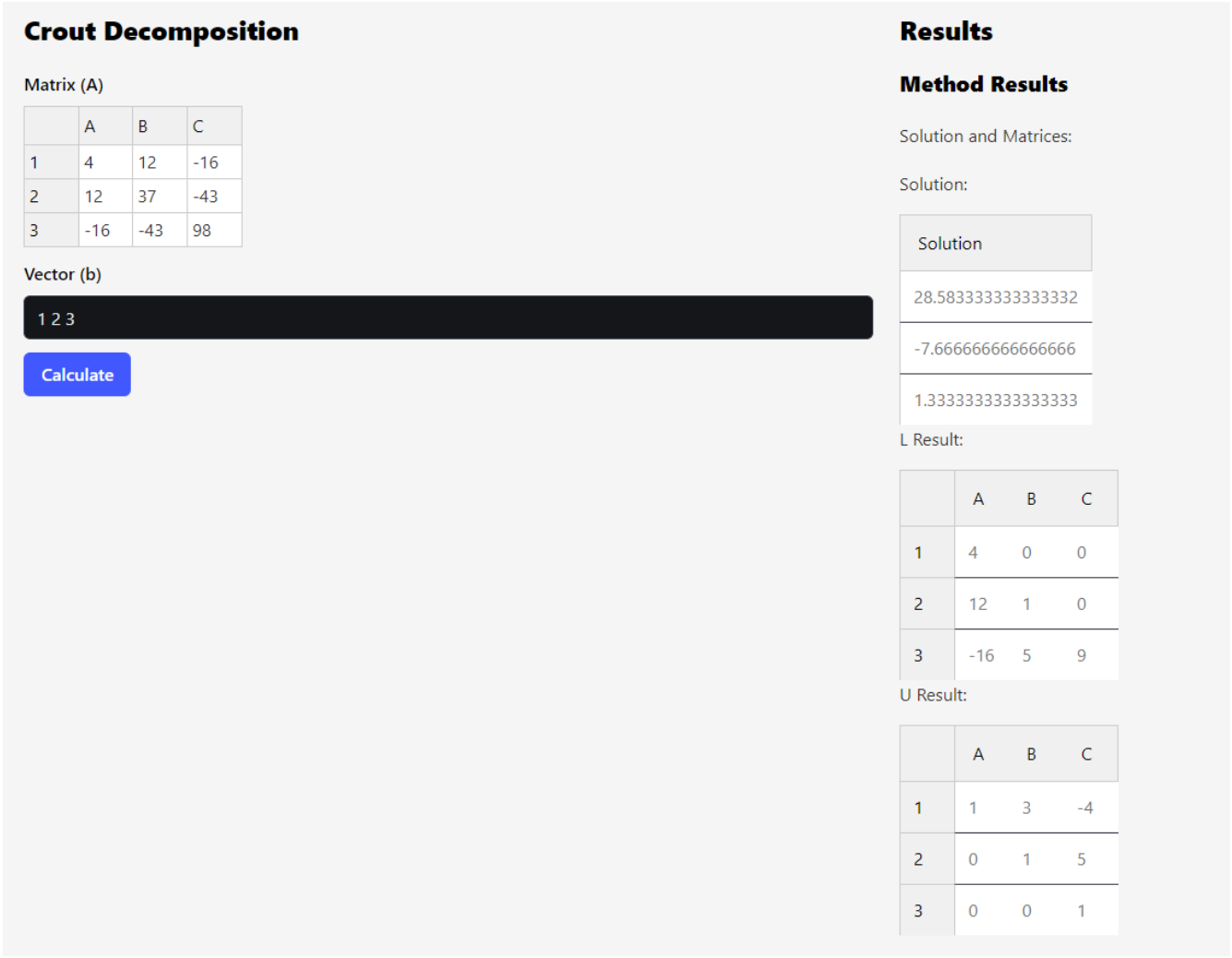
For Cholesky, only applicable to symmetric and positive definite matrices. If the matrix does not meet

these properties, the Cholesky method cannot be used. And here is the example:



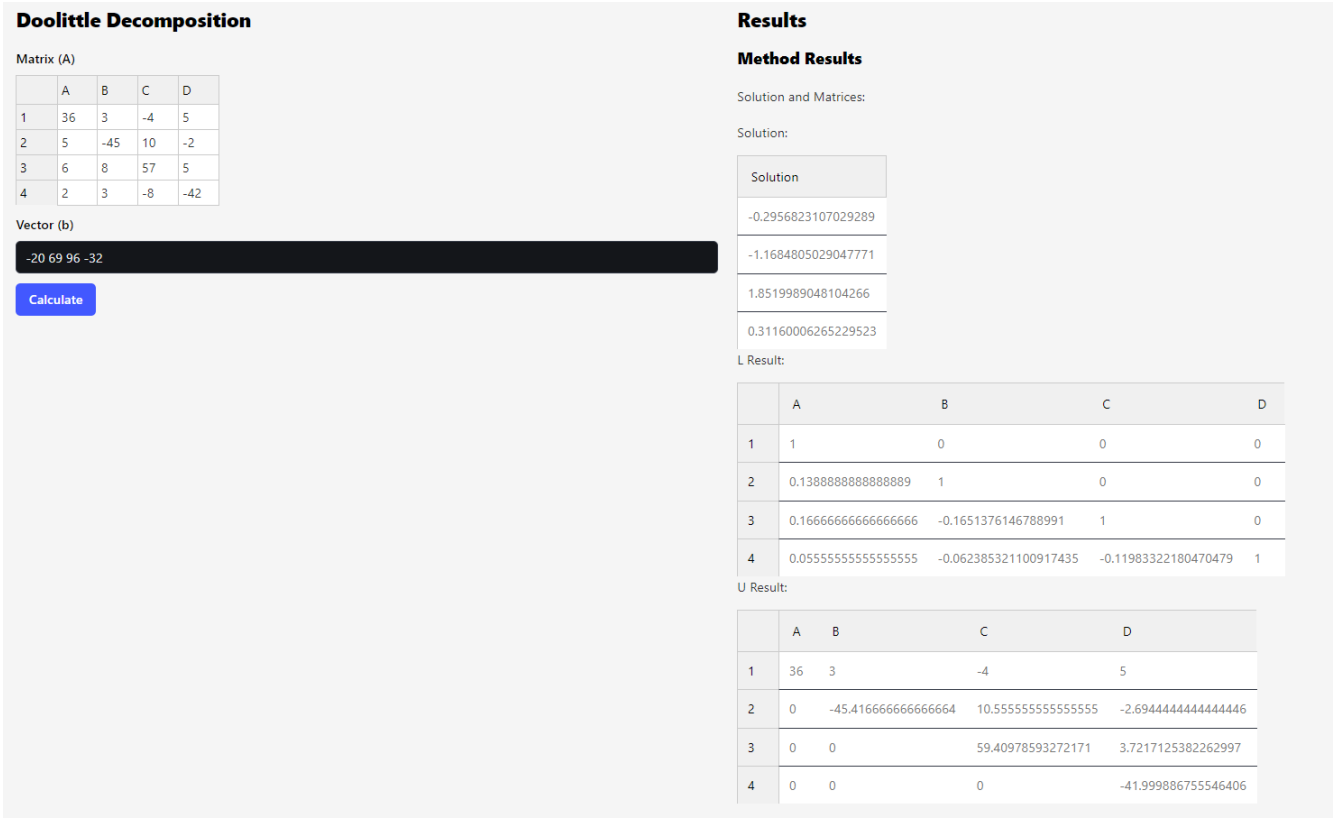
- **Crout:**

For Crout, cannot handle singular matrices (determinant zero) or matrices close to singularity. And here is the example:



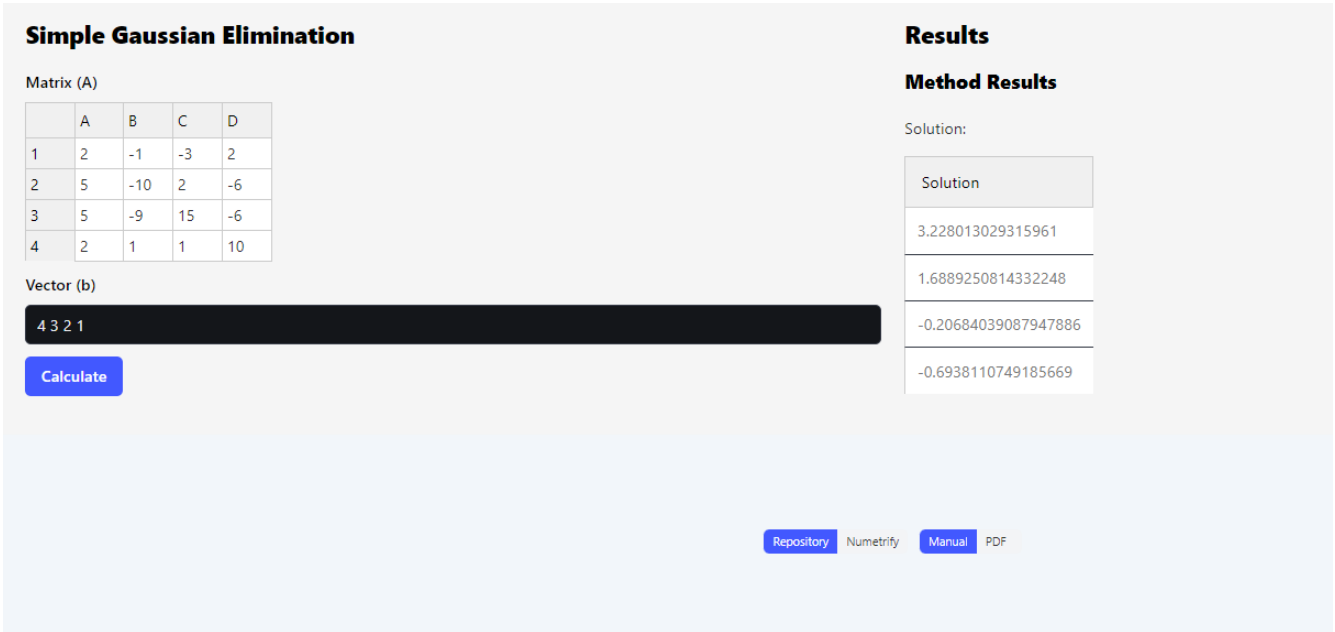
- **Doolittle:**

For Doolittle, similar to the Crout method, it does not handle singular or nearly singular matrices well. And here is the example:



- **Simple Gussian Elimination:**

For Simple Gussian Elimination, without pivoting, it can be numerically unstable and fail for singular or nearly singular matrices. And here are the examples:

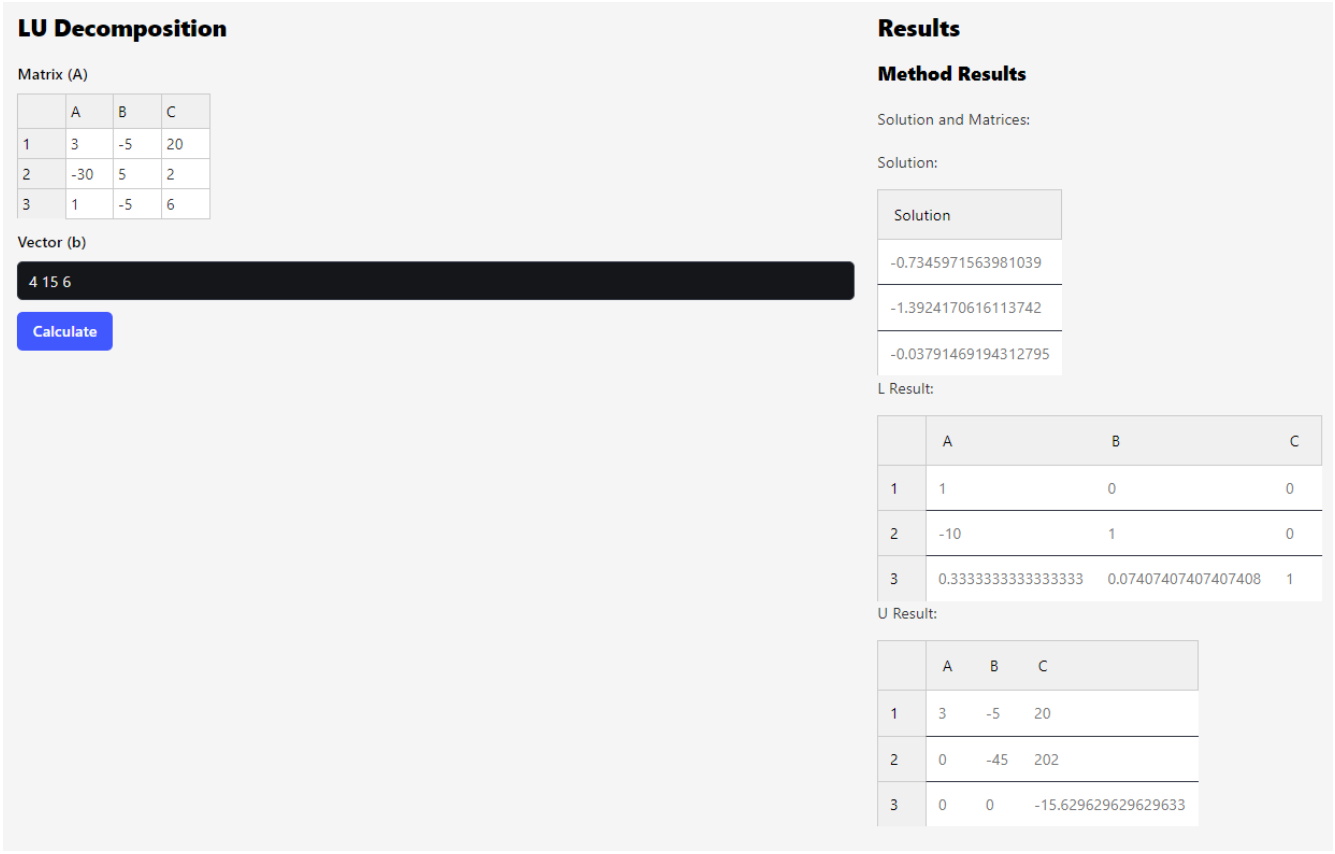


- **Pivot Gussian Elimination:**

For Pivot Gussian Elimination, although it improves stability by including pivoting, it can still be inefficient for large and sparse matrices. And here are the examples:

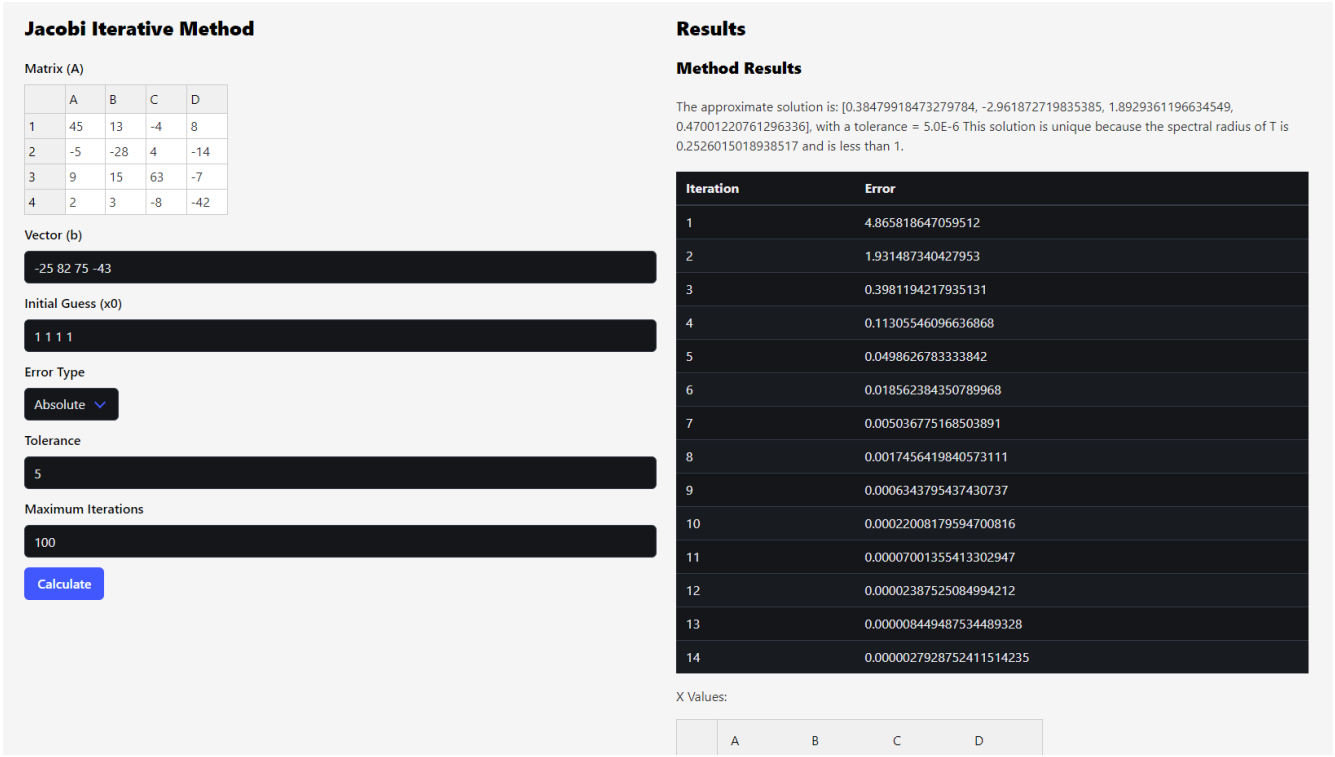
- **LU Gaussian Elimination:**

For LU Gaussian Elimination, similar to LU methods (Crout and Doolittle), it suffers with singular or nearly singular matrices. And here are the examples:



- **Jacobi:**

For Jacobi, convergence is not guaranteed for all matrices. Specifically, it requires the matrix to be strictly diagonally dominant or symmetric and positive definite to ensure convergence. And on this method the user has to enter additional information for this method, which would be the type of error, he must click and select the desired option and he must enter numerical values in the initial guess, the tolerance and the maximum number of iterations. Here is the example:



- **Gauss-Seidel:**

For Gauss-Seidel, Similar to the Jacobi method, it requires a strictly diagonally dominant matrix or a symmetric and positive definite matrix to ensure convergence. And on this method the user has to enter additional information for this method, which would be the type of error, he must click and select the desired option and he must enter numerical values in the initial guess, the tolerance and the maximum number of iterations. Here is the example: