User Manual Group 1.3

Group members:

Long Quoc Truong

April Hudspeth

Sarah Janssen

Latisha Konz

Rui Zhang

**To Run program**

<https://github.com/ucd-2017-csci4650-team13/Project1>

Open Matlab and navigate to scripts.

Open SuperSolver.m

Add to path when prompted

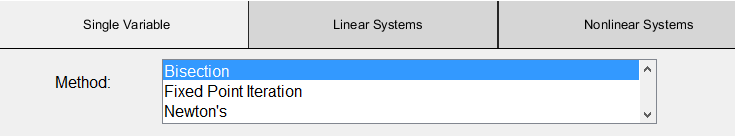
Now whenever you open Matlab, you can simply type SuperSolver and the program will appear.

Click run to open GUI

**Running Single Variable Bisection Method**

Step one:

Toggle to the single variable screen and choose Bisection from the drop down menu.



Step two:

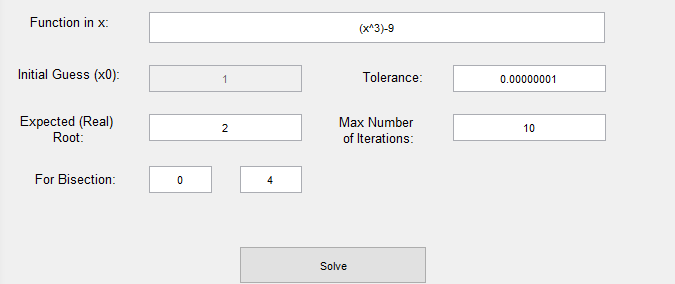
Enter a function into the box labeled “function in x”, the function should be set equal to zero. For instance if we wished to find the root of you would want to enter .

Step three:

In the expected root box you can enter a guess for the root. Keep in mind that the farther off the expected root is from the calculated root, the larger the error becomes.

Step four:

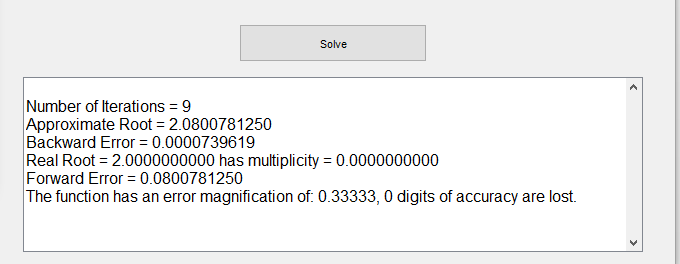
Enter an interval from a to b that you wish the program to search for a root. From our sample equation it is pretty obvious that a root will be near 2 so the interval from 0 to 4 is sure to return a result.



Step five:

Press the solve button. A new window will pop up with a graph of your function as well as a table of each of the calculated with each of its associated error.

On the home screen the print out shows the number of iterations that it took to find the approximate root, in this case it was 9, the approximate root, the backward error, real root with multiplicity, forward error, and calculation of error magnification with digits of accuracy lost.

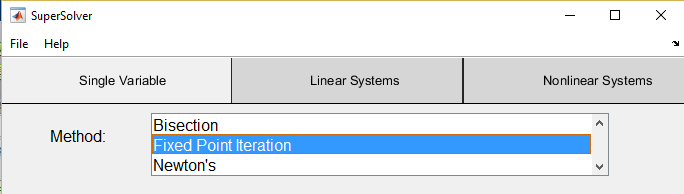


If you would like to further explore the bisection method program, attempt the same calculation with a guess that is not as close to the real root, for instance 10 and see how the error becomes larger.

**Running Single Variable Fixed Point Iteration**

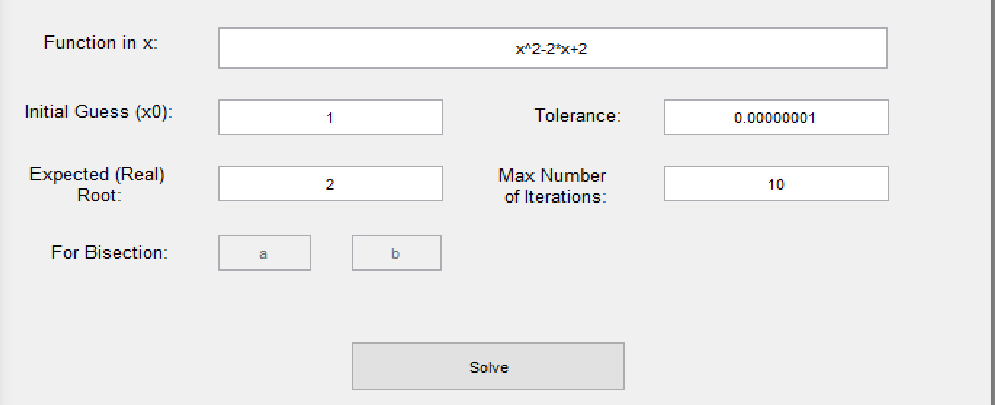
Step one:

Toggle to the Single Variable screen and choose Fixed Point Iteration from the drop down menu.



Step two:

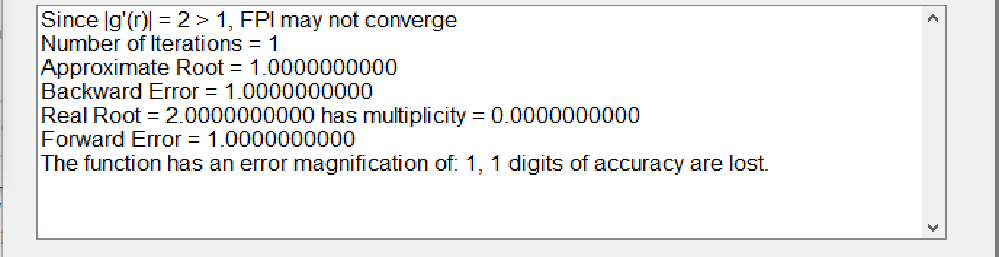
Enter a function into the box labeled “function in x”, the function should be set equal to zero. For instance if you would like to use fixed point iteration to find the roots of , you would need to enter it in like below. You can also enter in a guess. Since it is pretty clear that we will have a root at 1 and 2, lets enter in 2.



Step three:

Press solve and a new window will pop up with the solution using fixed point iteration and we can see that the roots are at 1 and 2 as expected.

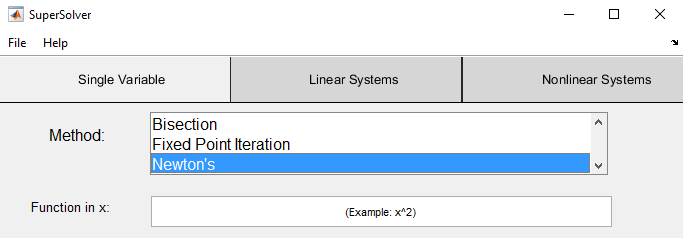
On the home screen we also have a summary of data for the equation.



**Running Single Variable Newton’s**

Step one:

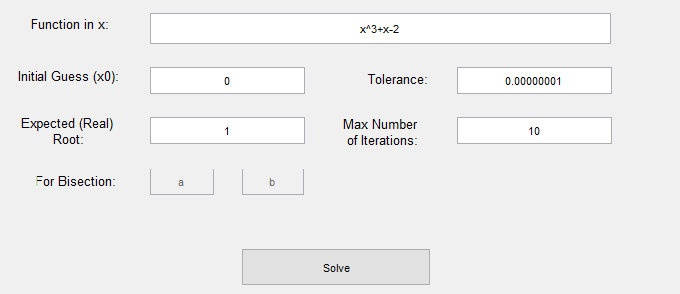
Toggle to the Single Variable screen and choose Newton’s from the drop down menu.



Step two:

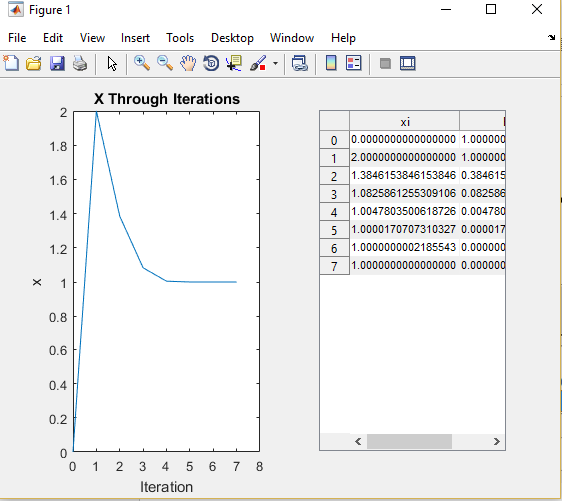
Enter in the function for which you would like to apply newton’s method. For example try to apply two steps of the Newton method with initial guess x0=0 on .

Your input should look like this:

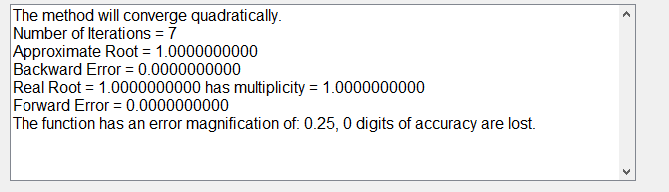


Step three:

Once you press the solve button a new window will pop up with the calculated iterations of and its associated error as well as a graph of the iterations.



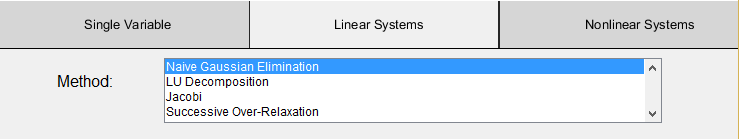
On the home screen you will also see a summary of data for the expression:



**Running Linear Systems Naïve Gaussian Elimination**

Step one:

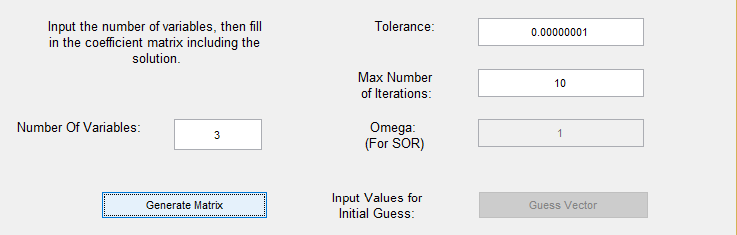
Toggle to the Linear Systems screen and choose Naïve Gaussian Elimination.



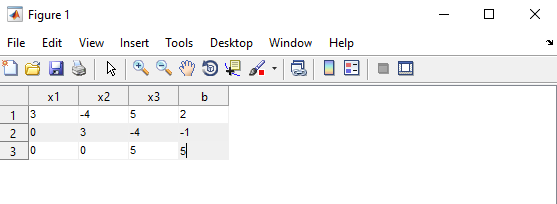
Step two:

Enter the number of variables of the system of equations you wish to solve in the box and click generate matrix. A second window will appear for you to enter your augmented matrix. For instance, if we wish to solve this system of equations:

You would enter three into the number of variables box, and click generate matrix.

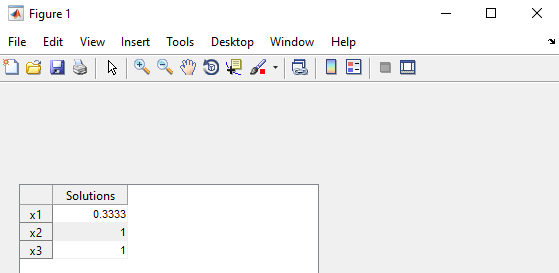


You can then enter your augmented matrix into the new screen that pops up.



Step three:

Once your values have been entered into your augmented matrix you may press the solve button. A new window will pop up with the solution to the system of equations. In this case the exact answer to the system is (1/3, 1, 1).



The answer in the box does in fact match the exact answer found for the system of equations.

On the main screen you will also see some additional information including run time, in this instance it was 0.0052449 seconds, as well as the number of operations, which was 15.

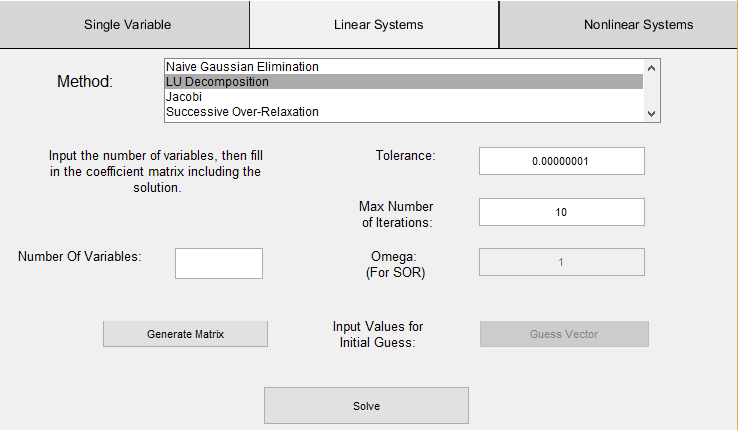
If you would like more practice with Naïve Gaussian Elimination you can try this system of equations:

The exact answer is (2, -1/2, -1), see what you get.

**Running Linear Systems LU Decomposition**

Step one:

From the Linear Systems screen, choose LU Decomposition from the drop down menu.



Step two:

In the Number of Variables box enter the number of variables for the system of equations in which you wish to factor. For instance if we wished to use LU Decomposition to factor the system:

You would enter 2 and press Generate Matrix.

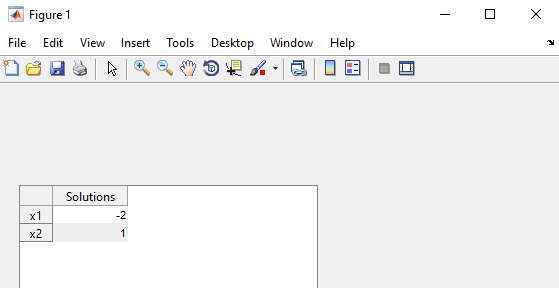
Step three:

From the new box that pops up enter the augmented matrix to be factored.



Step four:

Once all the values have been enter press the solve button. A new window will pop up with the solutions to the LU Decomposition of the system of equations.



In this instance the solution is (-2, 1) and the system can be checked by the Gaussian Elimination program.

On the home screen you will notice some additional information regarding the solving of this system. The run time was 0.0050321 seconds and it took the program 18 operations to arrive at the solution.

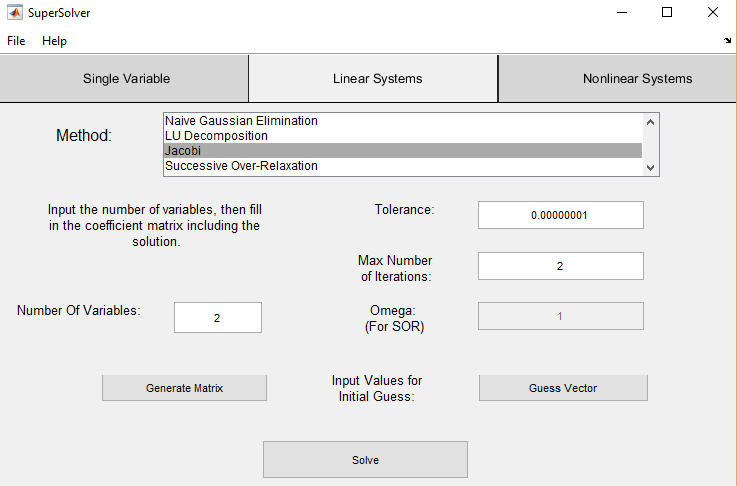
If you would like to try another system with LU Decomposition try:

The exact answer is (-1, 1) see what you get. Does it match the answer of the same system in Gaussian Elimination?

**Running Linear Systems Jacobi**

Step one:

From the Linear Systems menu choose Jacobi from the drop down menu.



Step two:

In the number of variables box enter the number of variables in the system of equations that you wish to solve, then click the Generate Matrix button. A new window will appear to enter the augmented matrix.

Step three:

Once you have entered the augmented matrix from the main screen you can enter your initial guess and the max number of iterations that you wish the program to run on your system. Your guess vector for initial values will need to have the same number of values as the number of variables in your system.

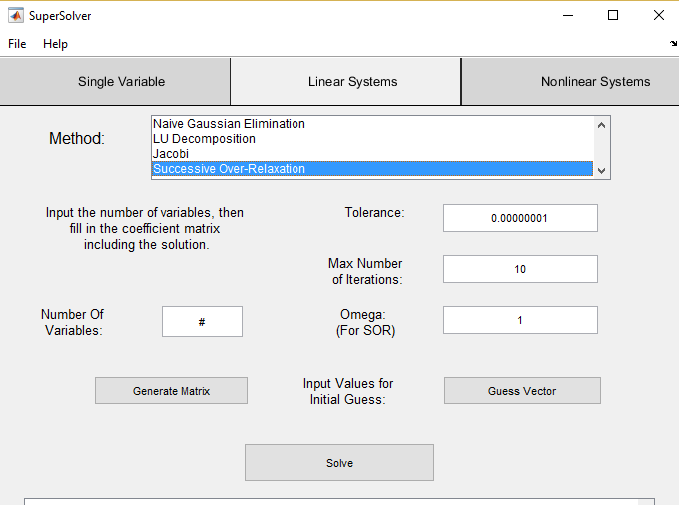
Step four:

Once all of the values have been entered you can click the solve button and a new window will pop up with the solution to the system of equations. On the home screen you will also notice some additional information such as the number of operations that it took the program to come to the solution and the run time in seconds of the program.

**Running Linear Systems Successive Over-Relaxation**

Step one:

From the Linear Systems screen choose Successive Over-Relaxation from the drop down menu.



Step two:

Fill in the number of variables for your system of equations, and click generate matrix. A new window will pop up for you to enter in your augmented matrix.

Step three:

Once you have entered in your values to your matrix you can click on the guess vector box and enter in your initial value guess. Remember that the guess vector has to have the same amount of values as the number of variables in your system.

Step four:

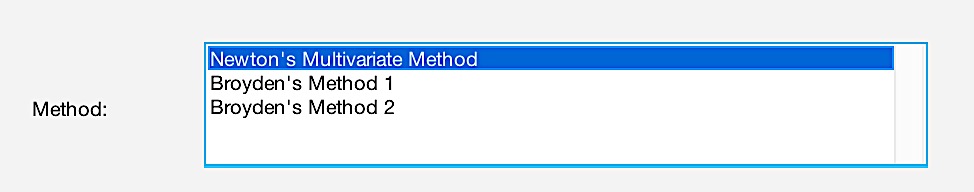
Enter omega or your relaxation parameter for the system of equations, and enter a value for the number of iterations you wish the program to calculate. Once all the values have been entered you can press calculate and a new window will pop up with the solution to the system of equations.

On the home screen additional information is provided as well such as run time in seconds and the amount of operations that the program took to come to the solution.

**Running Newton’s Multivariate Method**

Step 1:

In the list box select Newton’s Multivariate Method



Enter initial guess as a numeric array:

Example:

[1,2]



Step 2:

Enter a space-delimited list of variables that will be used for the equations.

Example:

u v

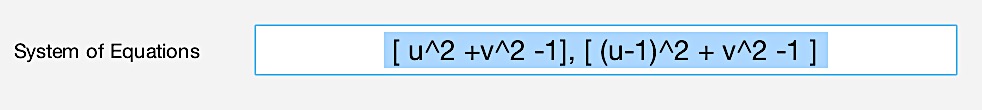


Step 3:

Enter the array of equations in the same format as the initial guess:

Example:

[u^2 + v^2], [ (u-1)^2 – 4\*v + 4]



\*\*Variables in equation must correspond to the variables listed above and in the order of appearance (‘u’ appeared first, then ‘v’ so list them in that order)

Step 4:

Enter the number of iterations you would like to run as an integer:

Example:

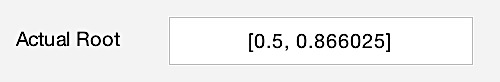
15

Step 5:

Enter the actual root/roots for error calculation as a numeric array

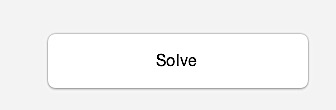
Example:

[1, 0]



Step 6:

After all the input has been entered (and Newton’s method is selected in the listbox), press “Solve” to run the program.



**Running Broyden’s Method**

Step1:

In the list box menu select either Broyden 1 or Broyden 2.

Repeat Newton’s steps 1-5

-Enter initial guess, system of equations, variables, and number of iterations in the same format as for Newton’s

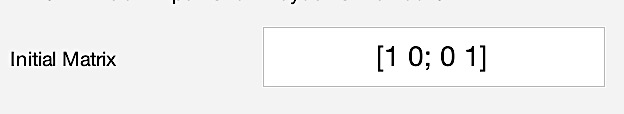
Step 2:

Enter an initial Matrix in the form [ a11 a21 ; a12 a22]

Example:

[2 4 ; 5 3] 🡪

\*\* The semicolon indicates a new row and numbers are space delimited



Step 3:

After all the text boxes have the appropriate fields (and one of Broyden’s method is selected), press the “Solve” button.

Note:

If you do not want to calculate the root, a default value will be entered. In this case, you can simply ignore the forward and backward error output.