Numerical Solution to a system of non-linear equation

Oves Badami

September 28, 2025

1 Introduction

We have so far discussed the problem of solving equations with one variable. However, please remember that our final goal is to solve the differential equation (Poisson's equation). Let us go ahead and see the discretization process first

2 Discretization

The Poissons equation is given as

$$\frac{\partial}{\partial x}\varepsilon\frac{\partial V}{\partial x} = -\rho = -q(N_D - N_A + p - n) \tag{1}$$

where n and p are the electron and hole concentration, N_D and N_A are the doping concentration. By converting it into the difference form we can write it as (we have assumed that the mesh spacing and material is same everywhere)

$$\frac{V_{i+1} - 2V_i + V_{i-1}}{h^2} = -\frac{q}{\varepsilon_i}(N_{D,i} - N_{A,i} + p_i - n_i) \tag{2}$$

where the subscript now shows that these quantities are located at the node i. So what has happened in that in the above process is that in the differential equations we had variables V, n, p, N_D, N_A that were valid at all the location, x. But now these quantities are valid only at the select location, x_i. And now the single differential equation is converted in to a system of difference equation each valid at the particular point. Thus we now get a system of equations that are coupled to each other. And they must be solved simultaneously as change in the value of V_i would also affect the neighbouring nodes. Also note that the system of equations are non linear because n_i and p_i are non-linearly dependent on the electrostatic potential at those nodes (V_i) Of course to solve it we will require BCs as well (more about it later.

Here n and p are the electron and hole concentrations, given by

$$n = N_c\mathcal{F}_{1/2}(\frac{E_F - E_C}{kBT}) \tag{3}$$

$$p = N_v\mathcal{F}_{1/2}(\frac{E_V - E_F}{kBT}) \tag{4}$$

where $\mathcal{F}_{1/2}(x) = \frac{2}{\sqrt{\pi}}\int_0^\infty \frac{\sqrt{y}}{1+exp(y-x)}dy$ Since we are interested primarily in the differences $E_C - E_F$ or $E_V - E_F$ we can set the $E_F = 0$ or any other energy level.

3 Generalization of the Newton-Raphson Method

We have looked so far that the Newton-Raphson method for equation with one variable is derived essentially from the Taylor's series and by truncating it at the first order. This results in the new root as

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)} \tag{5}$$

To extend the work to multi variable systems we will simply write the Taylor's series for multi-variable, k, system

$$F(x_1^{n+1}, x_2^{n+1}, x_3^{n+1}\cdots, x_k^{n+1}) = F(x_1^n, x_2^n, x_3^n\cdots, x_k^n) + \Sigma_i\frac{\partial F(x_1^n, x_2^n, x_3^n\cdots, x_k^n)}{\partial x_i}(x_i^{n+1} - x_i^n) + \cdots \tag{6}$$

Since there are k variables, we will have k equations. Let us represent each equation as $F_i(x_1^{n+1}, x_2^{n+1}, x_3^{n+1}\cdots, x_k^{n+1})$

$$F_i(\mathbf{x}^{n+1}) = F_i(\mathbf{x}^n) + \nabla F_i \overline{\Delta x} \tag{7}$$

Writing the above equation for all the k variables. We can then write the equation more comprehensively as

$$F(\mathbf{x}^{n+1}) = F(\mathbf{x}^n) + \nabla F \overline{\Delta x} \tag{8}$$

Similar to the Newton Raphson method we $F_i(\mathbf{x}^{n+1}) = 0$. Thus we can write it as

$$\overline{\Delta x}^{n+1} = -J^{-1} F(\mathbf{x}^n) \tag{9}$$

where J is referred to as the Jacobian, $\overline{\Delta x}^{n+1}$ is the update vector, $F(\mathbf{x}^n)$ is the residual vector. This is now a system of equation that needs to be solved. Note that Jacobian is of size $k \times k$. To calculate the new more accurate root values

$$x^{n+1} = x^n + \overline{\Delta x}^{n+1} \tag{10}$$

At the very first iteration, we need to make a guess x^0 and then progress ahead with the iteration untill the update Δx becomes small.

Note that the Jacobian is of the form $\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix}$

4 Boundary conditions

To solve any differential equation we need to give boundary condition. In our case it would be

$$V_1 = V_L \tag{11}$$
$$V_k = V_R \tag{12}$$

where $V_{L/R}$ are the applied bias. But how can we apply it as we the equation that has to be solved is of the form $A\Delta x = b$. So what should be the BC for this. The answer is that since these are boundary nodes, the potential at each of these nodes is fixed so the the update is zero. so the BC at node 1 and node k should be

$$\Delta x_{1/k} = 0 \tag{13}$$

So how do we account for it. The answer is use their values in the initial guess.

5 Methods to solve the system of equations

Now that we have made a system of equations it is necessary to see how can we solve them. For this purpose consider a set of generic equations

$$Ax = b \tag{14}$$

where is a coefficient matrix of dimension N× N is the unknown and b is the RHS vector. Let us represent the elements of matrix A as a_{ij} where i refers to the row index and j is the column index. The underlying process is to convert/transform the matrix A into an identity matrix and apply the same transformation to the vector b. The result of the application of the transformation on b is the result.

5.1 Introduction

To start with let us say we want to solve the following system of equations

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

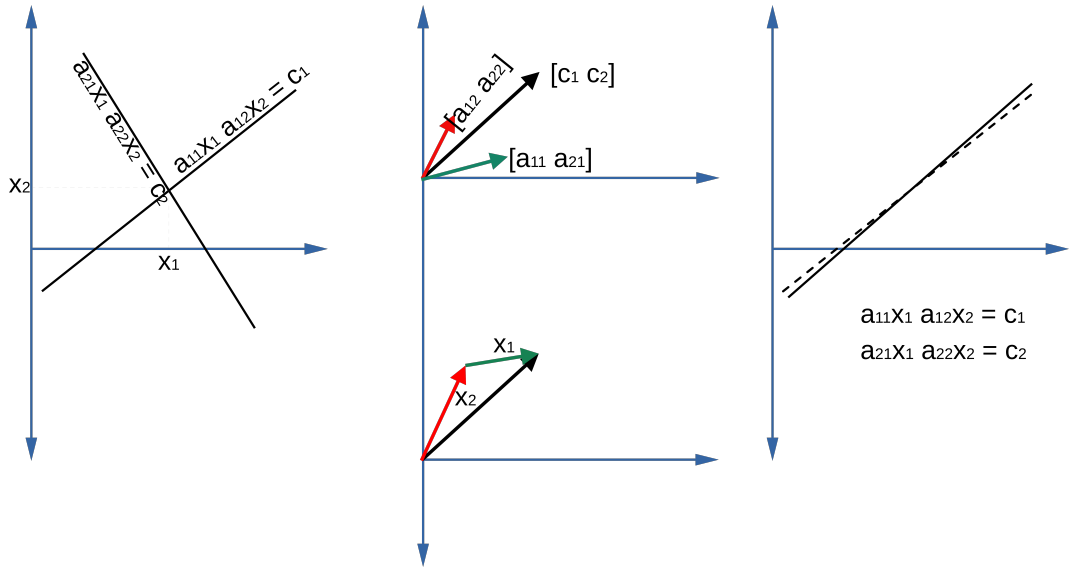
You can write both the equation for x_2 and solve for x_1 and then using the value x_1 calculate the the value of x_2 by substituting in one of the above equations. Geometrically interpretation of solving the above system of equations means that you plot them and

Just a side note what I can also write it as

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Another very interesting geometrical interpreation is to write the above equation as

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$



This can be interpreted as a linear combination of vector $[a_{11}\ a_{21}]$ and $[a_{21}\ a_{22}]$ that results in the third vector $[b_1\ b_2]$ and $[x_1\ x_2]$ are the weights. Now think about

Please remember that above are a system of linear equations. For a system of non-linear equations we have to linearize it ... This is exactly what Newton Raphson method does. It linearizes the non-linear equation.

As we have see that the solution to a system of equation is the one where both the lines intersect. However, in the third figure it is not very obvious or clear as to where they intersect. Such a system of equations are called as illconditioned. These are very sensitive to the round off errors. What they do is that a small change in the data can result in a very large change in the solution. Let us have a look at an example The solution to the following system of equations

$$\begin{aligned} 0.9999x_1 - 1.0001x_2 &= 1 \\ x_1 - x_2 &= 1 \end{aligned}$$

is $x_1 = 0.5$, $x_2 = -0.5$

Now let us perturb the equations a bit

$$\begin{aligned} 0.9999x_1 - 1.0001x_2 &= 1 \\ x_1 - x_2 &= 1 + \delta \end{aligned}$$

$x_1 = 0.5 + 5000.5\delta$ and $x_2 = -0.5 + 4999.5\delta$ This is because both the equations have very similar slope. The large change in the solution occurs because of the similar slope

5.2 Gauss Elimination

Consider a system of linear equations having N variables

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots a_{2N}x_N &= b_2 \\ &\vdots \\ a_{i1}x_1 + a_{i2}x_2 + \cdots a_{iN}x_N &= b_i \\ &\vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots a_{NN}x_N &= b_N \end{aligned}$$

The standard process is to eliminate the variables. For this purpose we divide the first equation by a_{11} and multiply it with a_{21} and subtract from the second equation would result in

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2N} - \frac{a_{21}}{a_{11}}a_{1N}\right)x_N = b_2 - \frac{a_{21}}{a_{11}}b_1$$

To simplify the notations we can write it as

$$a'_{22}x_{22} + a'_{23}x_{23} + \cdots a'_{2N}x_N = b'_2$$

Similarly for the third equation we must multiply the first equation with $\frac{a_{31}}{a_{11}}$ and subtract it from the third equation which would result in

$$a'_{32}x_2 + a'_{33}x_3 + \cdots a'_{3N}x_N = b'_3$$

where $b_3' = b_3 - \frac{a_{31}}{a_{11}} b_1$, $a_{32}' = a_{32} - a_{12}\frac{a_{31}}{a_{11}}$ and $a_{3N}' = a_{3N} - a_{1N}\frac{a_{31}}{a_{11}}$. For a generic equation i, we will get $b_i' = b_i - \frac{a_{i1}}{a_{11}} b_1$ and $a_{ij}' = a_{ij} - a_{1j}\frac{a_{i1}}{a_{11}}$ So the new set of equations are

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots a_{1N}x_N &= b_1' \\ a_{22}'x_2 + \cdots a_{2N}'x_N &= b_2' \\ &\vdots \\ a_{i2}'x_2 + \cdots a_{iN}'x_N &= b_i' \\ &\vdots \\ a_{N2}'x_2 + \cdots a_{NN}'x_N &= b_N' \end{aligned}$$

Please note that the $'$ doesn't represent differentiation The next task is to eliminate the x_2 from the eq.3 to eq.N using the above. This time the factor algorithm will be $\frac{a_{i2}'}{a_{22}'}$. Thus to a_{ij}' will be modified as $a_{ij}'' = a_{ij}' - a_{2j}'\frac{a_{i2}'}{a_{22}'}$. Repeating this process again and again ... N times we get an upper triangular

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots a_{1N}x_N &= b_1' \\ a_{22}'x_2 + \cdots a_{2N}'x_N &= b_2' \\ &\vdots \\ a_{33}''x_3 + \cdots + a_{3N}''x_N &= b_3'' \\ a_{NN}''^{\cdots}x_N &= b_N''^{\cdots} \end{aligned}$$

This process is called as the Forward Elimination. Thus the modified set of equations can be written as

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}' & a_{23}' \\ 0 & 0 & a_{33}'' \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2' \\ b_3'' \end{bmatrix}$$

The next step is called as the backward elimination. This nothing but now starting from the back and then calculating the values of x's

$$\begin{aligned} x_3 &= \frac{b_3''}{a_{33}''} \\ x_2 &= \frac{1}{a_{22}'}\left(b_2' - a_{22}'x_2\right) \\ x_1 &= \frac{1}{a_{11}}\left(b_1 - a_{13}x_3 - a_{12}x_2\right) \end{aligned}$$

In general we can write the backward substitution as

$$\begin{aligned} x_N &= \frac{b_N^{N-1}}{a_{NN}^{N-1}} \\ x_i &= \frac{b_i^{i-1} - \Sigma_{j=i+1}^{N} a_{ij}^{i-1} x_j}{a_{ii}^{i-1}} \qquad i = N-1, \cdots, 1 \end{aligned}$$

In simplified notations

$$\begin{aligned} x_N &= \frac{b_N}{a_{NN}} \\ x_i &= \frac{b_i - \Sigma_{j=i+1}^{N} a_{ij} x_j}{a_{ii}} \qquad i = N-1, \cdots, 1 \end{aligned}$$

Pseudocode

```
FOR r = 1:1:N-1
  FOR r' = r+1:1:N
    F = a_{r'r}/a_{rr}
    FOR c = r+1:1:N
      a_{r',c} = a_{r',c} - F a_{r,c}
    END
  b_{r'} = b_{r'} - F b_r
  END
END
```

$x_N = b_N/a_{NN}$
FOR r = N-1:-1:1
  $D = b_r$
  FOR c = r+1,N
    $D = D - a_{r,c}x_c$
  END
  $x_r = D/a_{r,r}$
END

5.2.1 Counting FLOPS

Let us count the number of Additions/Subtractions in the forward elimination

$$\Sigma_{r=1}^{n-1}\Sigma_{r'=r+1}^{n}1 + \Sigma_{r=1}^{n-1}\Sigma_{r'=r+1}^{n}\Sigma_{c=r+1}^{n}1$$
$$\Sigma_{r=1}^{n-1}(n-r) + \Sigma_{r=1}^{n-1}\Sigma_{r'=r+1}^{n}(n-r)$$
$$\Sigma_{r=1}^{n-1}(n-r) + \Sigma_{r=1}^{n-1}(n-r)^2$$
$$\Sigma_{r=1}^{n-1}\left(n - r + n^2 - 2nr + r^2\right)$$
$$\Sigma_{r=1}^{n-1}\left(n(n+1) - r(2n+1) + r^2\right)$$
$$n(n+1)(n-1) - (2n+1)\frac{n(n-1)}{2} + \frac{n(n-1)(2n-1)}{6}$$
$$(n+1)(n^2-n) - \frac{(2n+1)(n^2-n)}{2} + \frac{(n^2-n)(2n-1)}{6}$$
$$(n^2-n)\left(n+1-n-\frac{1}{2}+\frac{n}{3}-\frac{1}{6}\right)$$
$$(n^2-n)\left(\frac{n}{3}+\frac{1}{2}-\frac{1}{6}\right)$$
$$(n^2-n)\left(\frac{n}{3}+\frac{1}{3}\right) = \frac{n^3}{3}+\frac{n^2}{3}-\frac{n^2}{3}-\frac{n}{3}$$
$$\frac{n^3}{3} + O(n)$$

the number of Multiplications/Division in the forward elimination

$$\Sigma_{r=1}^{n-1}\Sigma_{r'=r+1}^{n}2 + \Sigma_{r=1}^{n-1}\Sigma_{r'=r+1}^{n}\Sigma_{c=r+1}^{n}1$$
$$2\Sigma_{r=1}^{n-1}(n-r) + \Sigma_{r=1}^{n-1}(n-r)^2$$
$$\Sigma_{r=1}^{n-1}\left(2n - 2r + n^2 + r^2 - 2nr\right)$$
$$\Sigma_{r=1}^{n-1}\left(n^2 + 2n - 2r(n+1) + r^2\right)$$
$$n(n+2)(n-1) - 2(n+1)\frac{n(n-1)}{2} + \frac{n(n-1)(2n-1)}{6}$$
$$(n^2-n)\left(n+2-n-1+\frac{n}{3}-\frac{1}{6}\right)$$
$$(n^2-n)\left(\frac{n}{3}+1-\frac{1}{6}\right)$$
$$(n^2-n)\left(\frac{n}{3}+\frac{5}{6}\right)$$
$$\frac{n^3}{3}+\frac{5n^2}{6}-\frac{n^2}{3}-\frac{5n}{6} = \frac{n^3}{3}+\frac{n^2}{2}-\frac{5n}{6}$$
$$\frac{n^3}{3} + O(n^2)$$

Thus the total number of FLOPS is

$$\frac{2n^3}{3} + O(n^2) \approx \frac{2n^3}{3}$$

TABLE 9.1 Number of flops for Gauss elimination.

n	Elimination	Back Substitution	Total Flops	$2n^3/3$	Percent Due to Elimination
10	705	100	805	667	87.58%
100	671550	10000	681550	666667	98.53%
1000	6.67×10^8	1×10^6	6.68×10^8	6.67×10^8	99.85%

Figure 1: Taken from book by Chapra

The FLOPs count for addition/subtraction for backward elimination

$$\begin{aligned}
&\Sigma_{r=n-1}^{1}\Sigma_{c=r+1}^{n}1 \\
&\Sigma_{r=n-1}^{1}(n-r) \\
&\Sigma_{j=1}^{n-1}j \\
&\frac{(n)(n-1)}{2} \qquad = \frac{n^2}{2} + O(n)
\end{aligned}$$

In going from equation 2 to 3 in the above we have substituted $r = n - j$
The FLOPs count for multiplication/division for backward elimination

$$\begin{aligned}
&\Sigma_{r=n-1}^{1}\Sigma_{c=r+1}^{n}1 + \Sigma_{r=n-1}^{1}1 + 1 \\
&\Sigma_{r=n-1}^{1}(n+1-r) + 1 \\
&\Sigma_{j=1}^{n-1}(j+1) + 1 \\
&\frac{n(n-1)}{2} + n - 1 + 1 \\
&\frac{n(n+1)}{2} \qquad = \frac{n^2}{2} + O(n)
\end{aligned}$$

So the total number of FLOPS are

$$\begin{aligned}
&\frac{2n^3}{3} + O(n^2) + n^2 + O(n) \\
&\frac{2n^3}{3} + O(n^2) \approx \frac{2n^3}{3}
\end{aligned}$$

5.2.2 Pivoting

The diagonal elements are called as pivots and they play a very significant role. For example if the pivot element is very small compared t the rest of the elements then the round off error can become significant thereby leading to incorrect solutions. An extreme case is that of the pivot element being zero. In this case it is easy to see that the Gauss elimination as discussed fails.

Consider the following example

$$\begin{aligned}
0.0004x_1 + 1.402x_2 &= 1.406 \\
0.4003x_1 - 1.502x_2 &= 2.501
\end{aligned}$$

and let us say we can represent numbers 4 digits. Multiplying the first equation with $0.4003/0.0004 = 1001$ and subtracting it from the second equation gives

$$-1405x_2 = -1404 \qquad x_2 = 1404/1405 = 0.9993$$

To calculate the x_1 substituting x_2 in the equation 1 we get $x_1 = 12.5$ The reason for this is that a a_{11} being very small number leads to large round off errors.

The solution to this issue is called as pivoting. Pivoting involves determining the row that has the largest coefficients in the column of the current pivot and then switching that row with the current row. This process is called as partial pivoting. The "complete pivoting" is where we also search the columns in addition to the rows as well. However, complete pivoting is rarely used because of significant increase in the computational burden which doesn't give the equivalent improvement in the performance. The computational burden increases first of all because of the search and secondly due to the column change the order of the variables also changes and this

results in additional headache of maintaining the order of variables. To illustrate the case of partial Pivoting consider the following example

$$\begin{aligned} 8x_2 + 2x_3 &= -7 \\ 3x_1 + 5x_2 + 2x_3 &= 8 \\ 6x_1 + 2x_2 + 8x_3 &= 26 \end{aligned}$$

As you can see that in the first equation the coefficient of x_1 is 0 which is naturally lesser than other coeffcients. So we go ahead and shuffle the equation as

$$\begin{aligned} 6x_1 + 2x_2 + 8x_3 &= 26 \\ 3x_1 + 5x_2 + 2x_3 &= 8 \\ 8x_2 + 2x_3 &= -7 \end{aligned}$$

Performing the row operations results in

$$\begin{aligned} 6x_1 + 2x_2 + 8x_3 &= 26 \\ 4x_2 - 2x_3 &= -5 \\ 8x_2 + 2x_3 &= -7 \end{aligned}$$

Now since the 8¿4 we again repivot the equations which results in

$$\begin{aligned} 6x_1 + 2x_2 + 8x_3 &= 26 \\ 8x_2 + 2x_3 &= -7 \\ 4x_2 - 2x_3 &= -5 \end{aligned}$$

Performing the row operations results in

$$\begin{aligned} 6x_1 + 2x_2 + 8x_3 &= 26 \\ 8x_2 + 2x_3 &= -7 \\ -3x_3 &= -3/2 \end{aligned}$$

Now we can proceed with the backward substitution.

5.3 Gauss-Jordan Method

Gauss-Jordan method is a variant of the Gauss-elimination but with only the difference that variables are eliminated from all the equations unlike the Gauss elimination method where only the variables from the subsequent equations are eliminated. This transforms the LHS matrix to an identity matrix unlike the upper triangular matrix in the earlier case. Consider an example below

Since the Gauss-Jordon method doesnt use the backward substitution it ends up having almost twice the number of FLOPS and that can mean a lot. Can you calculate the number of FLOPS for Gauss jordon method

$$\begin{aligned} 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4 \end{aligned}$$

$$\begin{bmatrix} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

Normalizing the equation 1

$$\begin{bmatrix} 1 & -0.0333 & -0.06667 & 2.61667 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

5.4 LU Decomposition Method – Dolittle Flavour

In this method the matrix **A** is decomposed into upper (U) and lower (L) triangular matrix.

$$\begin{aligned} Ax &= b \\ LUx &= b \\ Ld &= b \end{aligned}$$

This allows us to calculate the vector d using forward substitution (similar to backward substitution in Gauss elimination). Using the value of d we can use backward substitution to calculate the value os x

$$Ux = d$$

Then we can use backward and forward

$$A = LU$$

A 3×3 **L** and **U** matrix is

$$L = \begin{bmatrix} 1 & 0 & 0 \\ l_{11} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

The pseudo-code for the LU decomposition, forward and backward substitution is given below
DECOMPOSITION of A into L and U
FOR r = 1:N-1
  FOR r'=r+1:N
    $F = a_{r',r}/a_{r,r}$
    $a_{r',r} = F$
    FOR c = r+1:N
      $a_{r',c} = a_{r',c} - Fa_{r,c}$
    ENDFOR
  ENDFOR
ENDFOR

FORWARD SUBSTITUTION
FOR r = 2:N
  $S = b_r$
  FOR c = r+1:N
    $S = S - a_{r,c}b_c$
  ENDFOR
  $b_r = S$
ENDFOR
BACKWARD SUBSTITUTION
$x_N = b_N/a_{N,N}$
FOR r = N-1:1
  S = 0
  FOR c = r+1:N
    $S = S + a_{r,c}x_c$
  ENDFOR
  $x_r = (b_r - S)/a_{r,r}$
ENDFOR

The LU decomposition method doesn't provide an advantage over the Gauss Elimination in terms of FLOPs if there is only one RHS vector. However, for the case where there are multiple RHS vectors, we can perform elimination only once and then reuse it. Essentially we perform the computationally heavy task of elimination (FLOPS $\approx (2/3)N^3$) for the first time and then perform relatively lighter task of substitutions (FLOPS $O(N^2)$) only for different right hand side vectors.