Scientific Computing Using Python

Professor. Vivek Aggarwal and Professor. Mani Mehra

Department of Mathematics

Indian Institute of Technology, Delhi

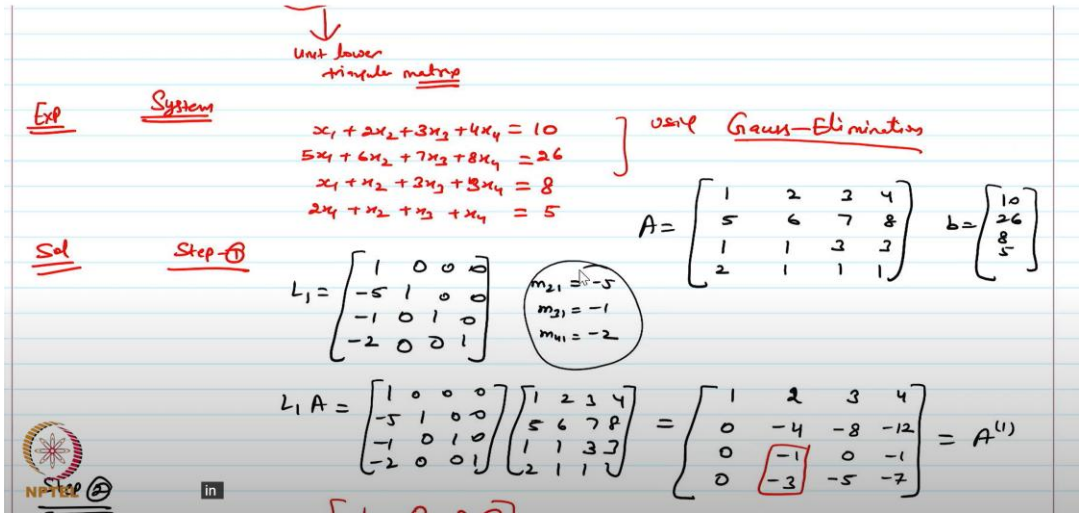
Lecture No. 14

Welcome to Scientific Computing Python. So, in the last lecture, we saw Gauss elimination and how we can drive it. So today, we will go ahead from that. Let's start again. So, in the last lecture, we calculated all this. We took an example and calculated according to that, and we came to know that our system got converted into the form of an upper triangular matrix, and we solved it very easily by backward substitution.

So now, what are the problems in this? If we do this process, then some limitations will come in it. So today, we will see what effect will these limitations have on us. So, limitations of Gauss elimination.

So now see what is there in this. Wherever you see, we are doing calculations. We are calculating multipliers. So, what happens in multipliers is that we are doing division by diagonal elements. Now, like we did this, here the diagonal element is one. When we did this, we got our multiplier.

(Refer slide time: 2:06)



So, in this case, if we take this system, and by chance, I change the system a little bit and the values here—suppose in this system the value is 0.01 x1 plus this same—then what would happen?

If the coefficient of x1 becomes very small as compared to other elements—this element is in the same column—or the element becomes even smaller than this, then what would happen in this case? That if you see, the values of the multipliers will be affected. Those values will become very large. So, the values that will come will be very large because we have to use diagonal elements to do the division.

So, if we use diagonal elements, then the values will become very large. Now, if I take any example, I define such an example. So, I take any matrix—just write 0.01 like this and 3 and 4, then wrote 2, 0.1, 3, 4, 1, 4 like this. Now, I have to convert this matrix in the form of U, in the form of upper triangular.

So now see, what will be the multiplier of this? What will be the m21? I will first divide it, and it will be -2 divide by 0.01. So, it will come close to -200. So, the multiplier which is m31 will come close to -400. So, what happened in this? The values of the multipliers will be very large. And if we are doing division by small numbers, then what happens is that if you ever enter a small number or a very small number in the computer, then the computer gives the message of NaN—NaN not a number. Or, division by small numbers also causes loss in the significant digits.

So, we want to avoid this. So, we say that we want to avoid this. We want to avoid this. So, what will we do to avoid this? So, what should we be told for this?

So, we can use another process so that we don't have to use the diagonal element here. Similarly, we will have to use this as well. So, what would you do? If I want to get rid of this, what should I do? What should I do?

I should swap the rows of the first row, second row, or third row with each other. So, if I make A like this—say 4 here—then I bring this 4, 1, 4 here at the top, in which the largest element was, and I bring 2, 0.1, 3, 0.01, 3, 4 here. So, what did I do? I swapped the R13 with each other, and we have this system. The right-hand side b will also change.

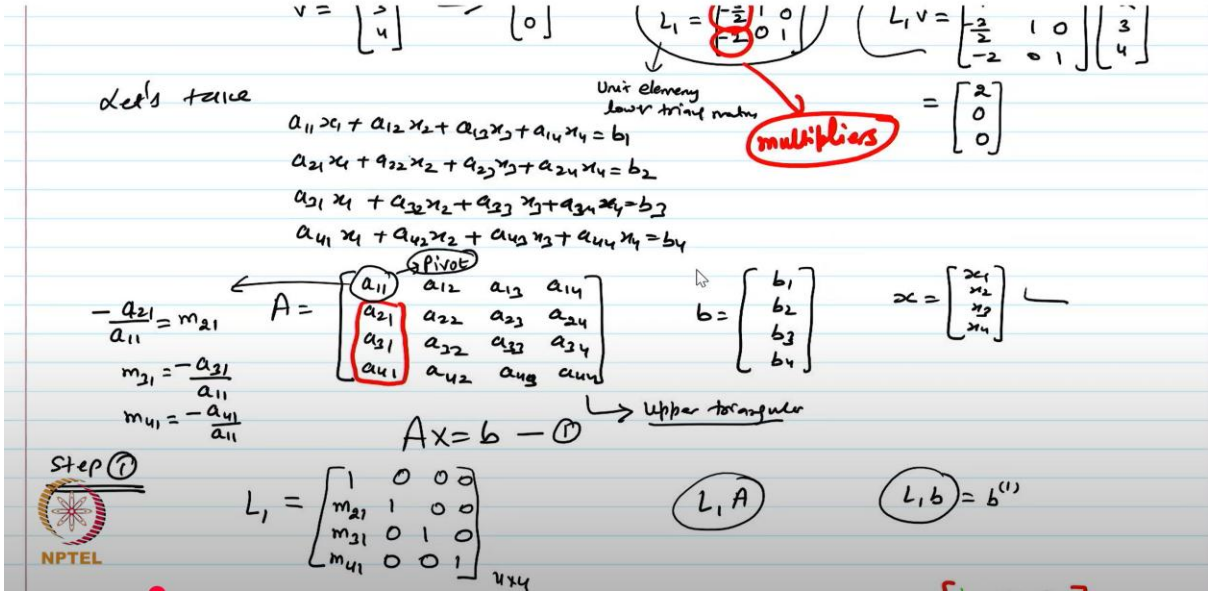
Now, see what has happened. These elements have come here. Now, I need to multiply it with m21.

So, what will happen? -2/4. So, -1/2. -0.5 came here. How much was 200 coming here? So, what happened with this? The quantity of the multiplier. So, generally, what do we do? We try to keep the absolute value of any multiplier less than one, so that the calculation remains stable, and our purpose is to keep its absolute value less than one.

So now, if we look at this process, what we did was to interchange the rows. So, we give a name to this work, and that name is partial pivoting. We know that these are diagonal elements; we call them pivots because with their help only we will make all the elements below zero. So, that is why these are the most important numbers in this Gauss elimination process. So, they are named pivots—very important numbers.

So, what do we do in partial putting? In partial putting, we will interchange the rows. And how will we interchange the rows? That in any matrix—for example, if I take this matrix, which we have called general matrix—then what will we do in this?

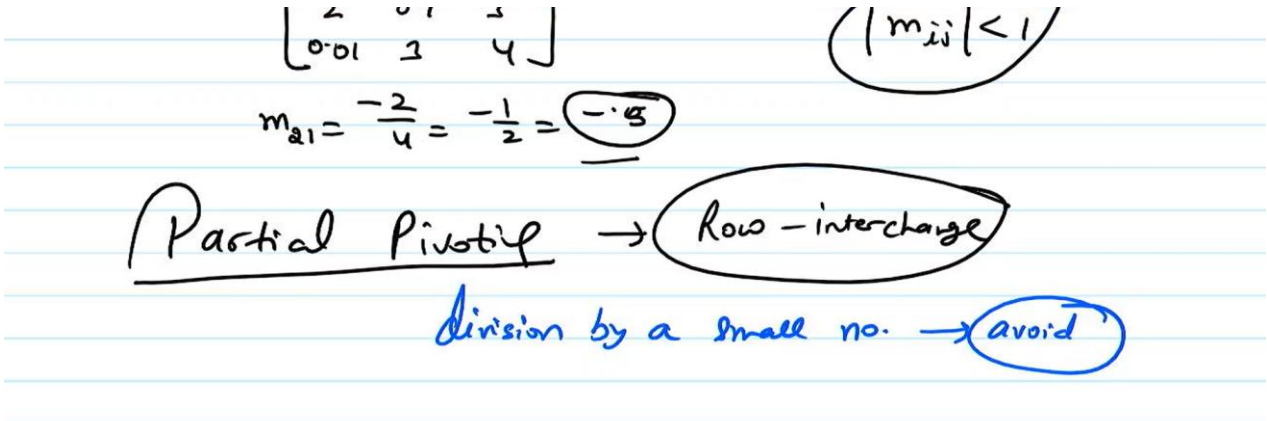
(Refer slide time: 8:07)



First, I will look at the column. I saw the first column, okay. So, I saw the first column, and after looking at the first column, I saw in which row the biggest number is in this fourth column. I swapped the first row with that row. Then, I applied Gauss elimination. Then after that, I did the same thing in the next one. Then, I did the same thing in the next one, okay.

So now, we keep doing this work that after every step, we are interchanging. So, all this work that is happening comes in the name of partial pivoting. So, we call it partial pivoting. So, what happens with partial pivoting is that—here, I can write—that the division by a small number, we avoid it. So, we also use partial pivoting for this thing, okay. So, we will use partial pivoting, and after that, we will find out its solution.

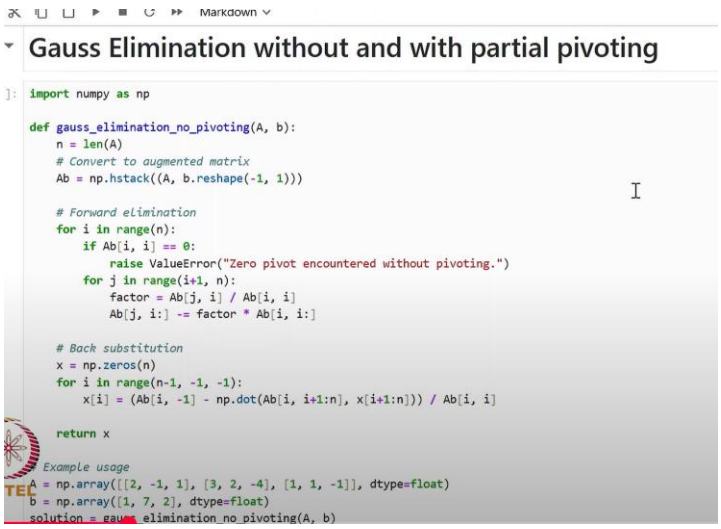
(Refer slide time: 9:38)



So, if I look at this thing, now let us see how to solve this in Python—this thing. This is Gauss elimination. So, what did we do? Look at the Gauss elimination.

So, look at this. We wrote the code for the Gauss elimination method, without and with partial pivoting.

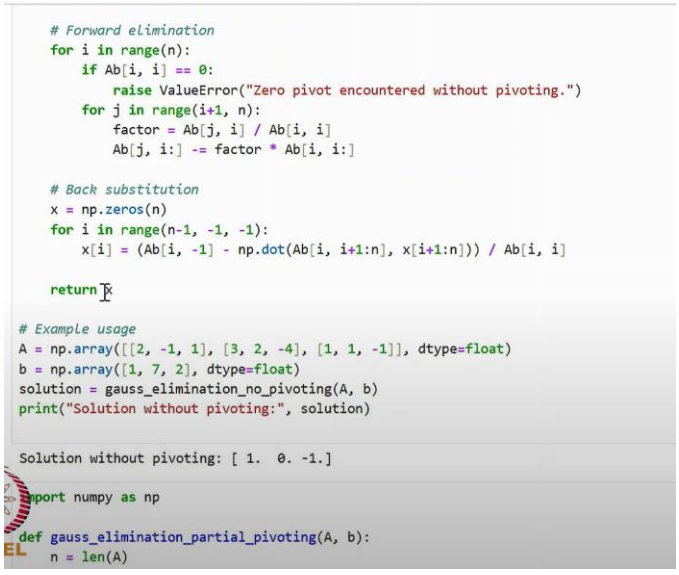
(Refer slide time: 10:18)



Look at both. This is Gauss elimination with no pivoting. No pivoting is happening. We have seen its length. What is its dimension? Okay. After that, we created an augmented matrix so that the same work that is happening can be done automatically on b as well. We don't have to do it separately. So, what we do is we create an augmented matrix and do it on that. After that, look at what we did because the problem was coming on the diagonal elements. So, what are the diagonal elements? Where there are same i and j. So, I wrote ii, if that value becomes zero, then we will say that zero pivot is encountered without pivoting. So, zero will be encountered, and as soon as we apply the multiplier, it will become infinity. The method will stop.

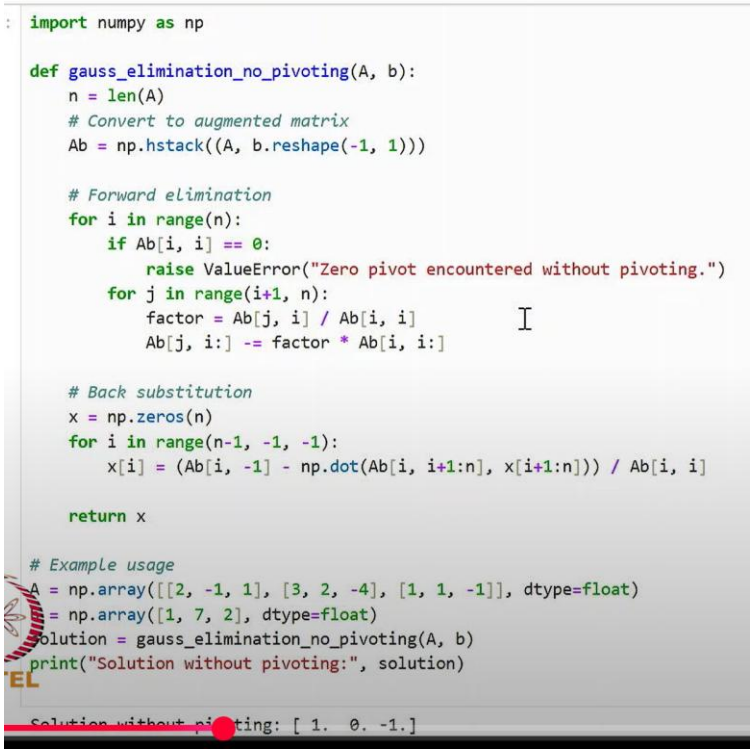
Otherwise, if this is not the case, then it can be multiplied by the multiplier as we had defined. Ah, divide them by absolute value and put a minus sign in front of it, and we will keep processing this thing. This process will keep happening in all our columns. After that, our matrix will be formed. That matrix, which will be ours, we solved it by back substitution and our x came from there, right?

(Refer slide time: 11:39)



So, we have done this work, and we can find it out with the help of this. So, now if I took this example, let's run it. Then its solution came 1, 0, -1 . We solved it, and the solution of it came is 1,0,-1, we had just taken an example, and I am seeing where it is.

(Refer slide time: 12:05)

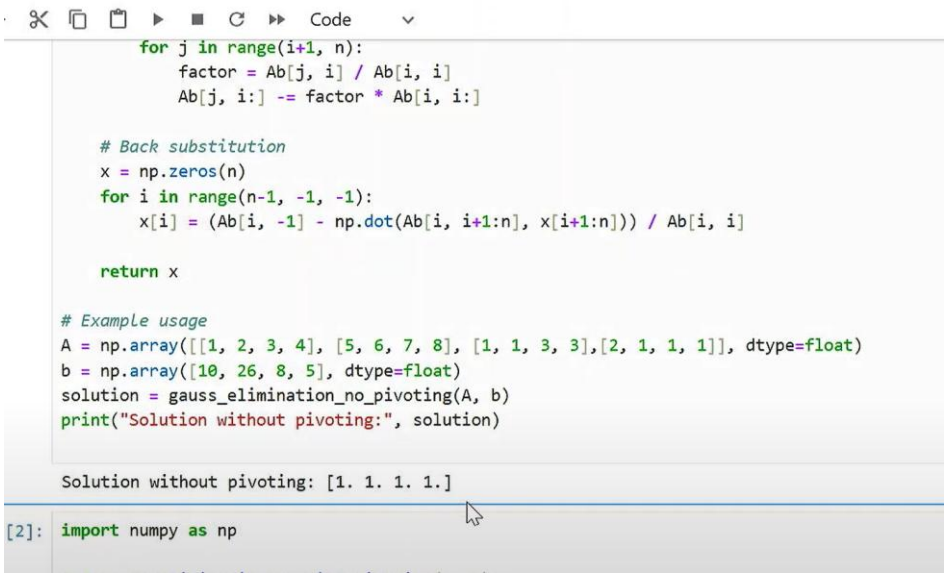


Just what we did was, I think, 1, 2, 3, 4, right? Like this. Yes, this is it. So, we have done it. Then 1, 2, 3, 4, then next 5, 6, 7, 8. This came. After that, the quantity was 1, 1, 3, 3 right?

Now, what happened? Three came. Now we have to apply it in the fourth. So, we will take the fourth in it, and it is 2, 1, 1, 1. So, this has come to us, okay. So, let's write it down properly.

Now this 4 cross 4 matrix, the vector on the right-hand side, what has it become? It was our 10, 26, 8, 5. This has come to us. So, we did this, and I solved it. So, look, 1, 1, 1, 1, we got the answer.

(Refer slide time: 13:48)

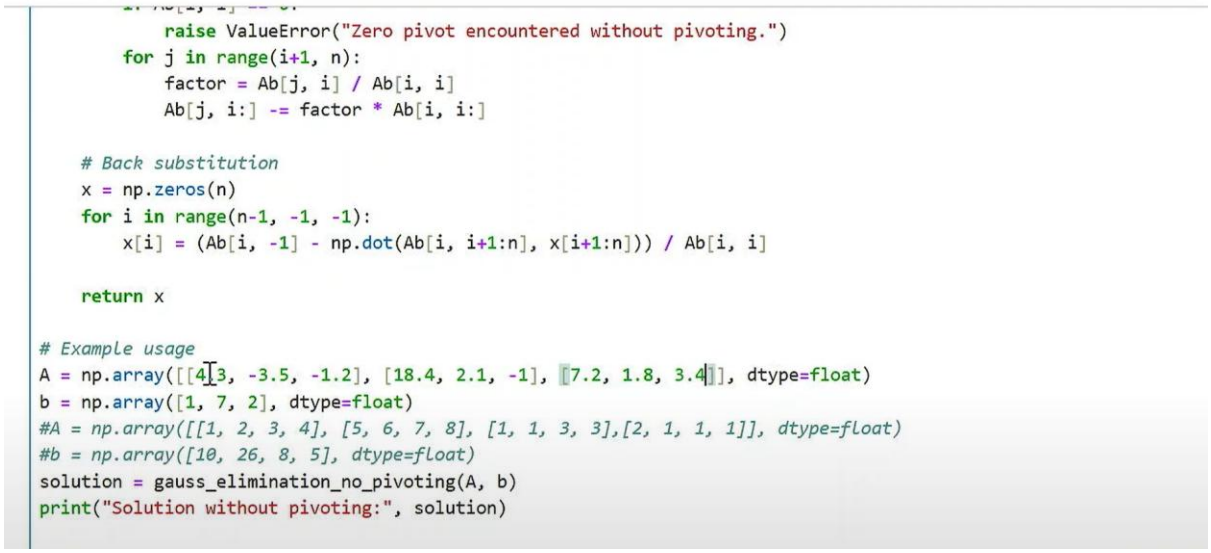


```
        for j in range(i+1, n):
            factor = Ab[j, i] / Ab[i, i]
            Ab[j, i:] -= factor * Ab[i, i:]

    # Back substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (Ab[i, -1] - np.dot(Ab[i, i+1:n], x[i+1:n])) / Ab[i, i]

    return x

# Example usage
A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [1, 1, 3, 3],[2, 1, 1, 1]], dtype=float)
b = np.array([10, 26, 8, 5], dtype=float)
solution = gauss_elimination_no_pivoting(A, b)
print("Solution without pivoting:", solution)

Solution without pivoting: [1. 1. 1. 1.]

[2]: import numpy as np
```

We just tried solving it by Gauss elimination, and the solution came from there. So, just like this, we worked on the values. Now, we did this work very easily, and we did not even realize it.

Now, if we take a system that you will not be able to do with pen and paper, it will be very difficult, right? It means that if you calculate it, it will be very difficult. Now, I will write it down like this, and I will take the suppose value. Let me change it like this. We will change it again. So, what are we doing? I will take the quantity. So, I will change it in its value. I will take its value: 4.3, minus 3.5, and minus 1.2, this is the first one. The second one is 18.4 and 2.1, and -1, the third quantity is 7.2 and 1.8 and 3.4.

(Refer slide time: 15:32)

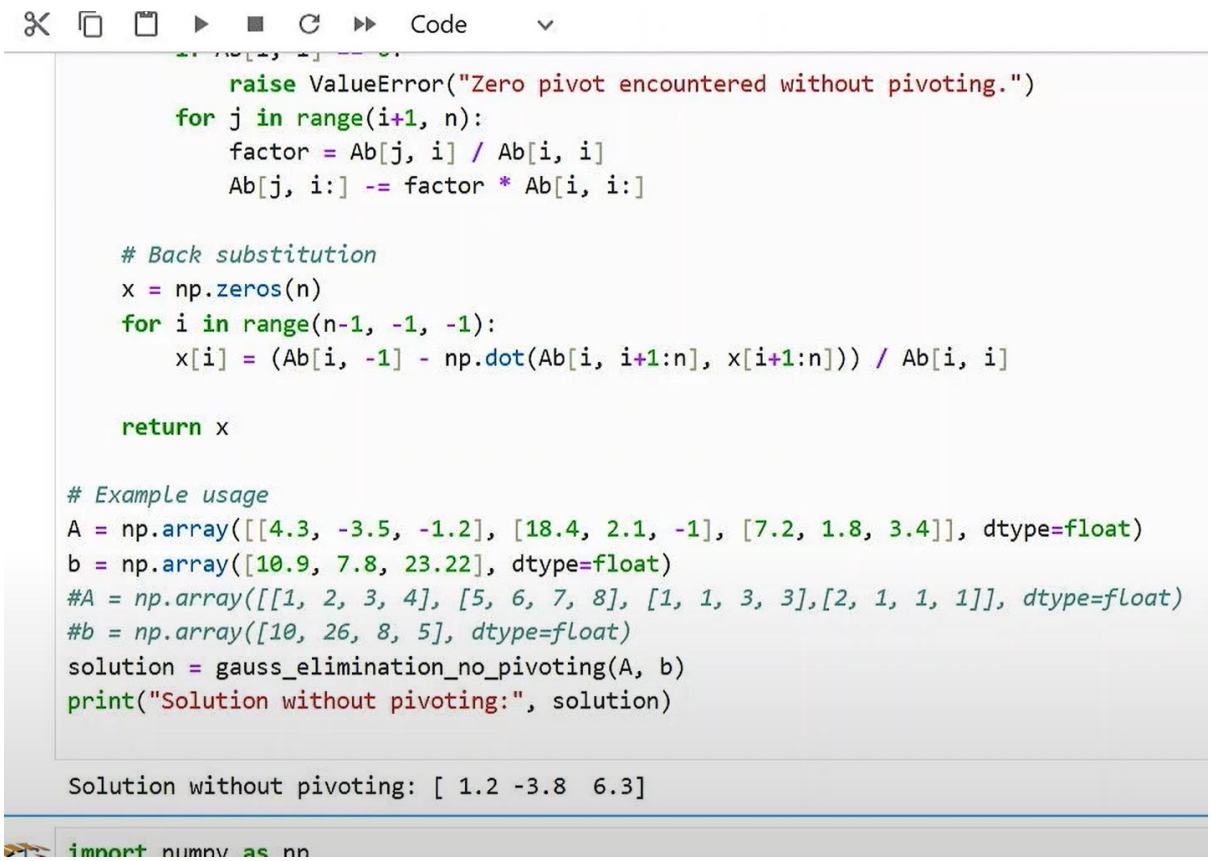


```
            raise ValueError("Zero pivot encountered without pivoting.")
        for j in range(i+1, n):
            factor = Ab[j, i] / Ab[i, i]
            Ab[j, i:] -= factor * Ab[i, i:]

    # Back substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (Ab[i, -1] - np.dot(Ab[i, i+1:n], x[i+1:n])) / Ab[i, i]

    return x

# Example usage
A = np.array([[43, -3.5, -1.2], [18.4, 2.1, -1], [7.2, 1.8, 3.4]], dtype=float)
b = np.array([1, 7, 2], dtype=float)
#A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [1, 1, 3, 3],[2, 1, 1, 1]], dtype=float)
#b = np.array([10, 26, 8, 5], dtype=float)
solution = gauss_elimination_no_pivoting(A, b)
print("Solution without pivoting:", solution)
```

Now see, the values that have come are in fractions. So now, it will be very difficult to calculate it with pen and paper. Let me take it on the right side: 10.9, and I took 7.8 and 23.22, this value. Now you see, this is a floating point. It is a fraction. So, if we take it with pen and paper, then the calculation that we have to do is very difficult.

So, we made a code for this. Once we substituted it in that code and saw what the answer is. So, look, the way we did it, its answer came: 1.2, minus 3.8, and 6.3. This value is okay, right? Now, these are the quantities. We did that. That means we did not apply any partial pivoting and we directly solved it with the help of it, okay.

(Refer slide time: 16:38)

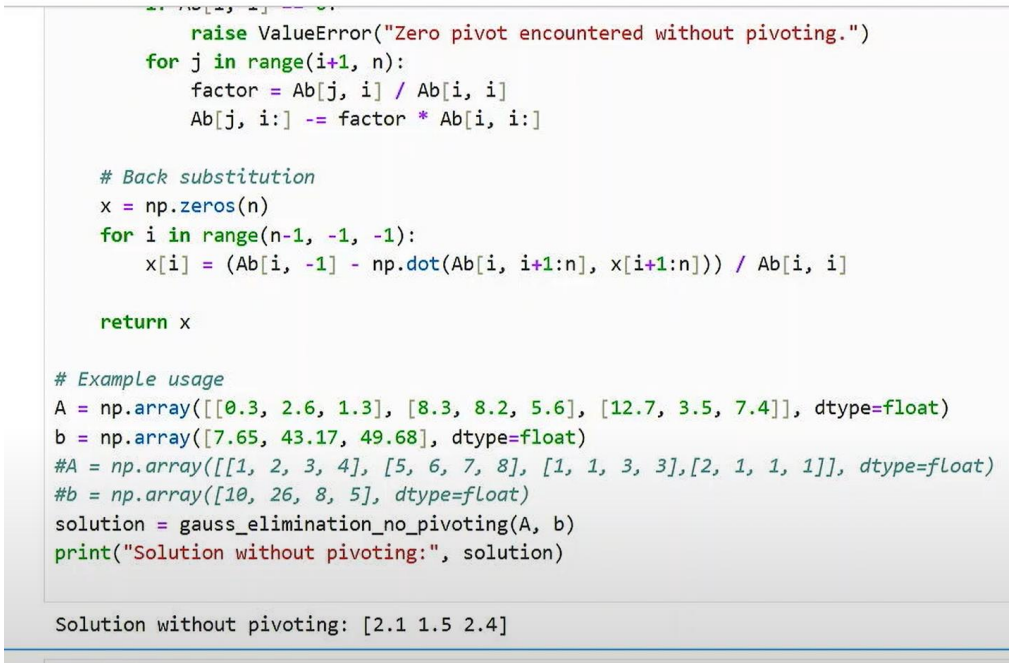


So, in this way, we can do the second method. I can see an example as to how we apply it in this. So, let me take an example in which we will apply both the methods: with and without partial pivoting. Then we will see what change is coming.

Let me take this. I took this. I took 0.3, plus 2.6, 2.6, and 1.3. That's right. 8.3, 8.2, and 5.6. And 12.7, 3.5, and 7.4. And on the right side, I took 7.65 and 4, 43.17, and 49.68. That's right.

So, in this, now we will see that the first element is 0.3, the second element is 8.3, the third element is 12.7. So, are we not applying partial pivoting in this? So, the solution for this is 2.1, 1.5, and 2.4

(Refer slide time: 18:46)



```
            raise ValueError("Zero pivot encountered without pivoting.")
        for j in range(i+1, n):
            factor = Ab[j, i] / Ab[i, i]
            Ab[j, i:] -= factor * Ab[i, i:]

    # Back substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (Ab[i, -1] - np.dot(Ab[i, i+1:n], x[i+1:n])) / Ab[i, i]

    return x

# Example usage
A = np.array([[0.3, 2.6, 1.3], [8.3, 8.2, 5.6], [12.7, 3.5, 7.4]], dtype=float)
b = np.array([7.65, 43.17, 49.68], dtype=float)
#A = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [1, 1, 3, 3],[2, 1, 1, 1]], dtype=float)
#b = np.array([10, 26, 8, 5], dtype=float)
solution = gauss_elimination_no_pivoting(A, b)
print("Solution without pivoting:", solution)

Solution without pivoting: [2.1 1.5 2.4]
```

So, what did we do in this? Because the value in this is not too small. So, we solved it and our solution came. Now, I applied partial pivoting in this case.

I will give it to you. Okay, so I have applied partial pivoting. Let's see. This system is a small system, and it is a good system. So what did we do? We will do the same thing with partial pivoting. This method is called Gauss elimination partial pivoting. So what do we have to do in partial pivoting? Whatever rows are there with the maximum element, swap them with each other.

So see what it is doing: find the maximum element in the current column. Okay, so calculate it, find it out, and swap that row. And if all the values are coming zero, then it is possible that the matrix becomes a singular matrix. Okay, so this thing also has to be taken care of. So now, as soon as we get to know the maximum values of its value rows, we will swap them. We swapped them with each other. Okay?

(Refer slide time: 20:14)

```python
import numpy as np

def gauss_elimination_partial_pivoting(A, b):
    n = len(A)
    # Convert to augmented matrix
    Ab = np.hstack((A, b.reshape(-1, 1)))

    # Forward elimination with partial pivoting
    for i in range(n):
        # Partial pivoting: find the max element in the current column
        max_row = np.argmax(np.abs(Ab[i:n, i])) + i
        if Ab[max_row, i] == 0:
            raise ValueError("Singular matrix detected.")

        # Swap rows
        if max_row != i:
            Ab[[i, max_row]] = Ab[[max_row, i]]

        # Elimination
        for j in range(i+1, n):
            factor = Ab[j, i] / Ab[i, i]
            Ab[j, i:] -= factor * Ab[i, i:]

    # Back substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (Ab[i, -1] - np.dot(Ab[i, i+1:n], x[i+1:n])) / Ab[i, i]

    return x
```

And after that, we applied the method. Then we will do this work. Then we will keep on calculating all these and will keep on doing substitution. We will apply Gauss elimination. From there, we have all the methods, all the steps. We will be completed. Then what did we do? We took back substitution and calculated from that, and see what is its solution. So its solution—2.1, 1.5, 2.4—will be the same because this is not a very big system. Right?

(Refer slide time: 20:48)

```python
            Ab[j, i:] -= factor * Ab[i, i:]

    # Back substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (Ab[i, -1] - np.dot(Ab[i, i+1:n], x[i+1:n])) / Ab[i, i]

    return x

# Example usage
A = np.array([[0.3, 2.6, 1.3], [8.3, 8.2, 5.6], [12.7, 3.5, 7.4]], dtype=float)
b = np.array([7.65, 43.17, 49.68], dtype=float)
#A = np.array([[2, -1, 1], [3, 2, -4], [1, 1, -1]], dtype=float)
#b = np.array([1, 7, 2], dtype=float)
solution = gauss_elimination_partial_pivoting(A, b)
print("Solution with partial pivoting:", solution)
```

```
Solution with partial pivoting: [2.1 1.5 2.4]
```

So there are not many changes in it, but if the quantities become very small—like I don't know—if I solve this system and see 0.0, then what is the result? It is coming out to be 0.03, and here I see that it will be almost the same. So then we will have to make a system like this with this quantity.

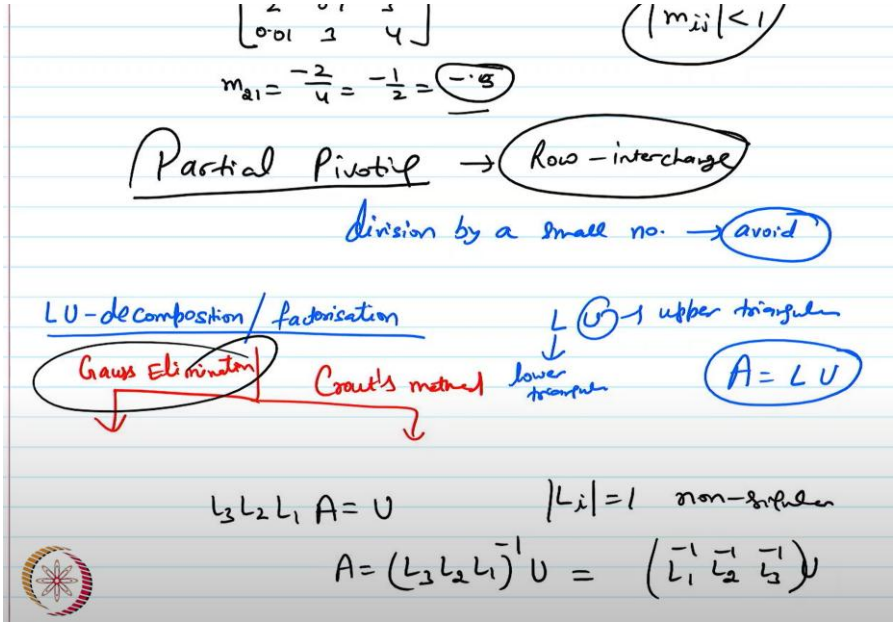
So partial pivoting is very useful because we know for sure that we will definitely get the solution. If we don't apply partial pivoting and somewhere the value of the solution comes zero, then in that case we will come to know that this cannot happen. So we will see this thing further, that what can we do next?

So we came to know from partial pivoting, we can use the method of elimination, which means Gauss elimination. Now we need partial pivoting, so if our diagonal element comes out to be zero by chance, then without partial pivoting, we cannot proceed further. So we will not be able to move ahead. So we will not be able to do anything with it. So partial pivoting is essential. So we have done the Gauss elimination method.

So our next step is LU decomposition. LU decomposition, or we can also call it factorization. What does it mean? This means LU, L means lower triangular, and this is upper triangular. So what did we do? We have to write our matrix like this. What does it mean? We have factorized that matrix into the product of two matrices, in which one is the lower matrix and one is the upper triangular matrix. So we have done this work. Now we have to see how we can do it. So LU decomposition is done in two ways. One is our Gauss elimination, and the second one is our Crout's method with these two. Now we can understand the Gauss elimination very easily. Now see what we did in this Gauss elimination.

What we did was we took a matrix A, suppose I took 4 by 4, applied L1 to it, then applied L2, then applied L3. Here we got U. Now I know that the determinant of Li is one and non-singular. So what do I do? I take the inverse of these. So from here, I write A like this—L3, L2, L1—and their inverse I write this into U. Now what does this become? It becomes the inverse of L1, the inverse of L2, the inverse of L3, and U into U.

(Refer slide time: 24:54)



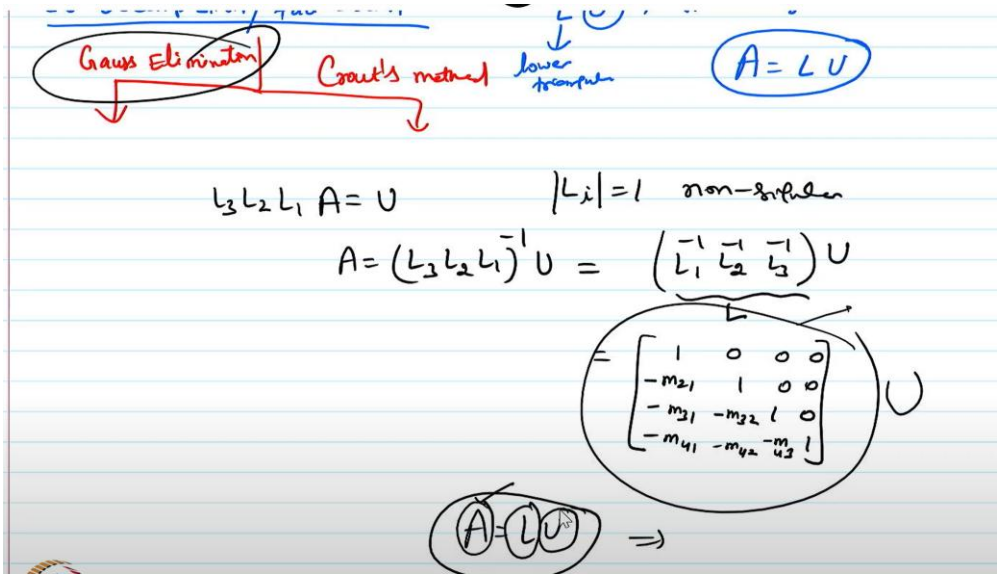
Now we know that the lower triangular matrix this is a unit lower triangular matrix. Its inverse is a unit lower triangular matrix. So this is a matrix, and the product of these three will also be a unit lower matrix.

So if we product all three and take their inverse, then we will get a matrix. We will calculate the new matrix by L. We will write this, which will be a unit lower triangular matrix in itself. And what will this matrix be? This will be a matrix of multipliers. So the L that will be formed, we will have this 1, 0, 0, 0, 1, 1, 1. Here we will have m21. And we are also taking the inverse, so let me do it a little. In the inverse, there will be minus m31, minus m41, minus m32, minus m42, minus m43.

This matrix that will be formed after multiplication, we will first take out its inverse and then multiply it. Then this matrix will come into—so you see, what is this matrix? This is a lower triangular matrix with unit diagonal elements, and this upper triangular we already know, which we got.

So from here, we said that A became LU; it became a LU decomposition. Right? So from here, we said that we got a matrix. We got a decomposition. We have a matrix A. We decomposed it. We factorized it into the product of two matrices, in which there is a lower triangular matrix. This is an upper triangular matrix.

(Refer slide time: 27:11)



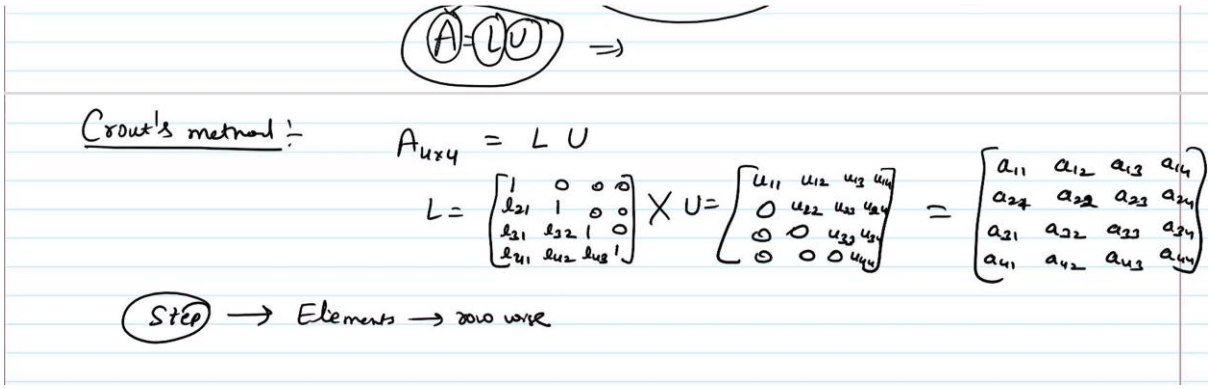
So the work that we have done is done automatically by Gauss elimination. Right?

So we can do this work very easily. Now the second method is a very important method, and that is Crout's method. So what Crout's said was that we should not go by Gauss elimination. So what he did was that we took the matrix A. Okay, let me take 4 by 4. So what we did with him was that we wrote it in this form. Here it is L. Now, we know that in the previous case it was a unit lower triangular matrix, so the same was done in this as well. But this element is there, and he said that we will take it out. So, l21, l31, l32, l41, l42, l43 — this and U is left. We have upper triangular, so in upper triangular, u11, u22, u33, and u44 came. This came: u12, u13, u14, u23, u24, u34 — and all these elements are zero.

So, we have converted this matrix into this form. So, this and A is ours — that is, a matrix. So, you see, its and its product will become equal to the matrix we have is a11, a12, a13, a14, a22, — sorry, a12, a22, a23, a24, a31, a32, a33, a34, a41, a42, a43, a44 . So, now we have to multiply these two matrices with each other.

So, now we have to see how to calculate this. So, the steps in this are okay. So, what will be the steps? That we will get our elements. So, let's get the elements. We will find out the elements row-wise. Row-wise means first we will get u11, u12, u13, u14. Then we will get l21, u22, u23, u24 — like this. So, we have to get them row-wise. So, we will have to go to this step. So, for this, we will see what the steps are.

(Refer slide time: 30:44)



And step one — now see — we multiply these with each other. So, what did we do? We multiplied the LU with each other. So, what happened? First, take the first row and nd the first column — we will get u11. Multiply this with this. So, what should always be a11? U11 should be a11. From here, we got to know.

Now, we have to use this. So, what will we do? We will calculate the second one by multiplying it with this. What will we get, u12, and what will happen a12, u13 will come a13, u14 will come a14. So, in the first time, all of them will come out automatically. Okay.

Now, in step two — in the first step, we got these four quantities. Now, from now on, what will we do in the second? We will multiply the second first column with the second row. So, this will come. We will have l21 u11, if we multiply it with zero, then tjis will remain. And this one that comes out — this comes out equal to a21.

Okay, two matrices are equal. If we say that two matrices A is eual to B, then it is possible only when the elements of both are exactly the same. So, this is what we are doing. We wrote that L is equal to A. So, after multiplying them, the element that comes should be the same, right?

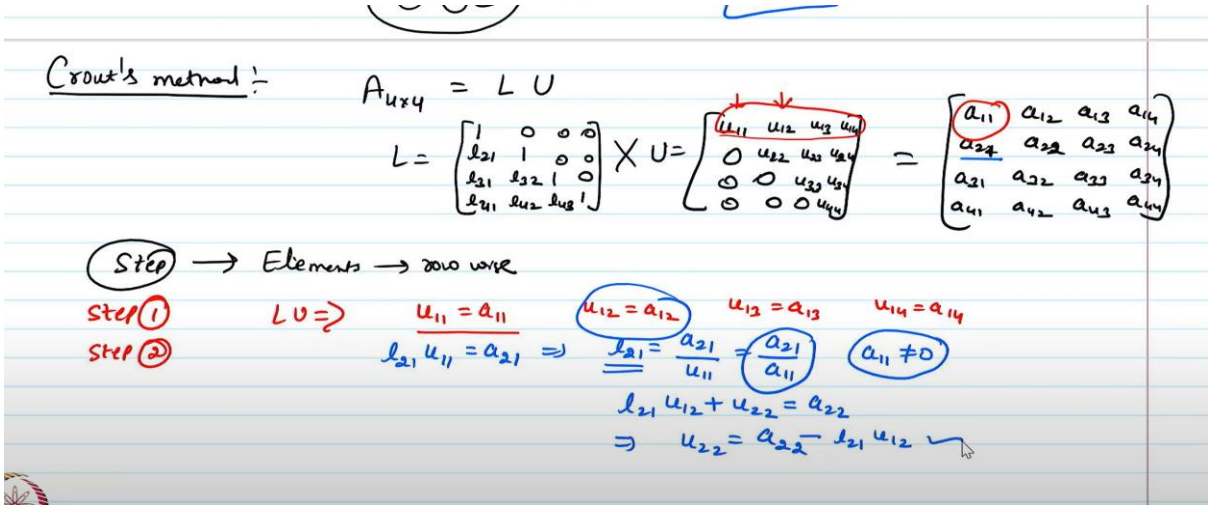
So, what happens to us from here — see, we have got the value of l21. And what is that, a21 divided by u11, or a21 divided by a11. So, you see, a21 has been divided by u11, right? And the multipliers were also coming like this. Okay.

So, there it was that a negative quantity was coming. So, now let's see what happens. So, the l21 that came is this — it is just a possibility in this. We always have to keep in mind that a11 cannot be zero. Okay? If it becomes zero, then we will have to do partial pivoting. So, we have to keep this thing in mind.

So, l21 has come to us. Now, let's see what happens as soon as 121 comes. Now, see the second one — which is done with the first — A has come. If we do the second column from the second row, then from here we will get l21 u12 +u22, okay, equal to a22.

l21 we have found out. u22 — so what will we get from here? We also know u12. So, what will happen to u22 from here? That will come — a22 minus, l21, now we have found out. So, whatever comes as l21 is the quantity, and u12 — we already know this. So, from here we have u22.

(Refer slide time: 34:23)



Now, we will take the third one and do it with this. So, from here, we will get l21 u13 + u 23= a23. Now see, l21 we know, u13 we already know from here. So, from here we will get u23, which will be a23 – l21 u13.

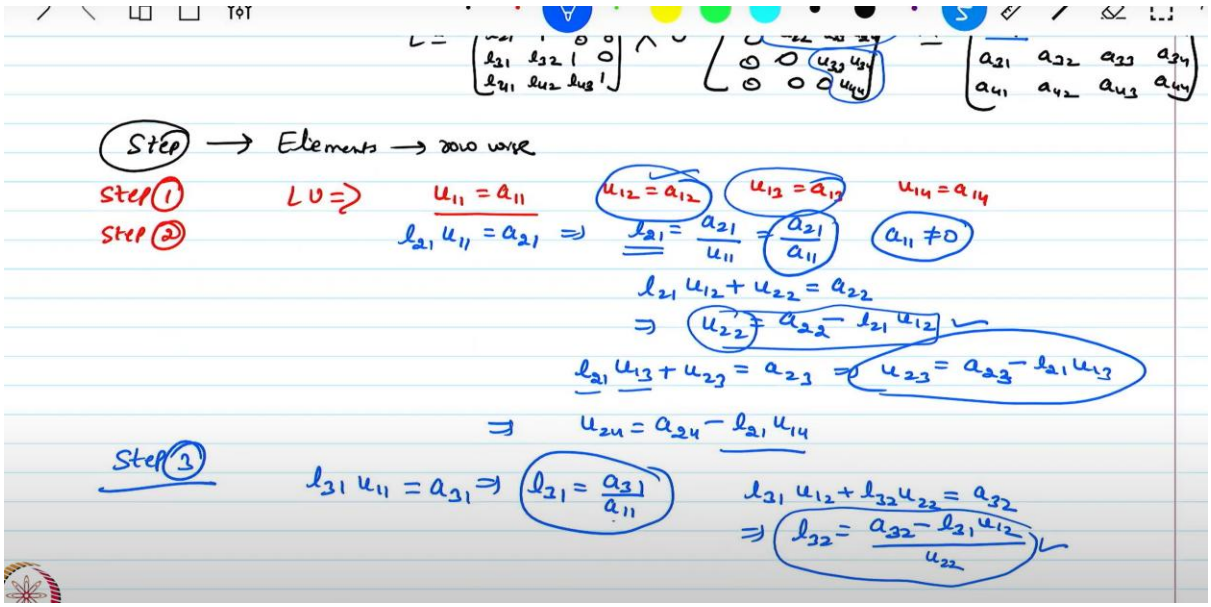
U24 — if we see like this, then we will have u24, okay? What will it become, a24- l21 u14 — we have. So, this quantity. So we have the second row complete.

So, we have got all these elements. Now, we have got this as well. Now, these three are left. So, now we will go in the same way. Okay? So, step two is done. Similarly, step three will come…

So, what will happen in step three? Now look, this is what we have in step three. Now look, in step three we have l31, we will multiply it by this, and from the third row to the first column, l31 u11 = a31. So, from here we get l31. We have a31 divided by u11 — that is, a11. Okay, so we have this quantity.

Now, if we go to the next, we will get l31 u12 + l32 = u22, and this will become equal to a32. Now look, here we get l31. From u12 we already know. Okay, l32 and what is u22 — we have already found out u22. So, from here, we get the value of l32. So, L32 comes. We have a32 minus l31 u12 divided by u22. This comes. See — and u22, we had this or this quantity. So, from here we got l32. So, we got this l32 as well. Okay.

(Refer slide time: 37:22)



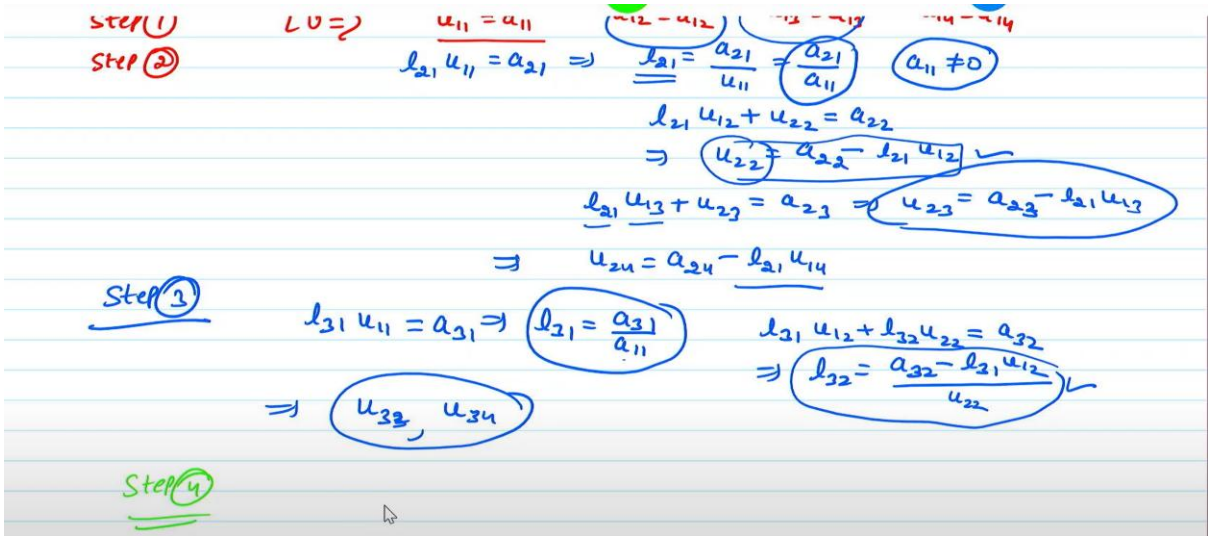
So, what happens by getting l32? We got these quantities. Now, if we use it, then we have u32 and — sorry — u33 and u34, which we have. We will calculate when we multiply these two with each other.

Now, what do we have to do? This is the third one, and we have to multiply it by this. Okay, so what will come in it? u33 will come as quantity. Then, if we have to do it in the next one, then u34 will also come in it, because in the last one it is zero. So, it will reach here, then the quantity will come from there. Okay.

So, we will keep doing it in this way, and then we will be left with the last one. If we do it with this, then we will get l41, l42, l43, and by using it, we will get u44. Okay.

So, if we keep taking these steps, then step three has come. In this way, step four will come. So, in these steps, all the values that we have will come.

(Refer slide time: 38:29)



So, if you see, then all the quantities that we have are here. Now look at the variables that were unknown — sorry — in this case, what were they? They were 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16. So, if we see here, we had 16 unknowns. And in the 4 by 4 matrix too, there are 16 unknowns — there are 16 elements — sorry.

So, that is why, with the help of this, we will easily and uniquely take out the 16 unknowns. It is not that they will be different, and we can take them out in a unique way. The only problem that we have to keep is — what is its limitation? The limitation of LU decomposition, and that is that the elements, the diagonal elements — and what they are — a11 should not be zero. Okay? And they should not be zero.

The diagonal element should always keep increasing so that we don't have a22, a33, a44 as zero. So, we have to keep in mind that this a11 should not be zero. What does it mean? That we can do partial pivoting in this also. First, we did partial pivoting before starting — we cannot do it in between — we did it first, and after that, we started the LU decomposition process, and after that, we got the solution. So, we can always do this work.

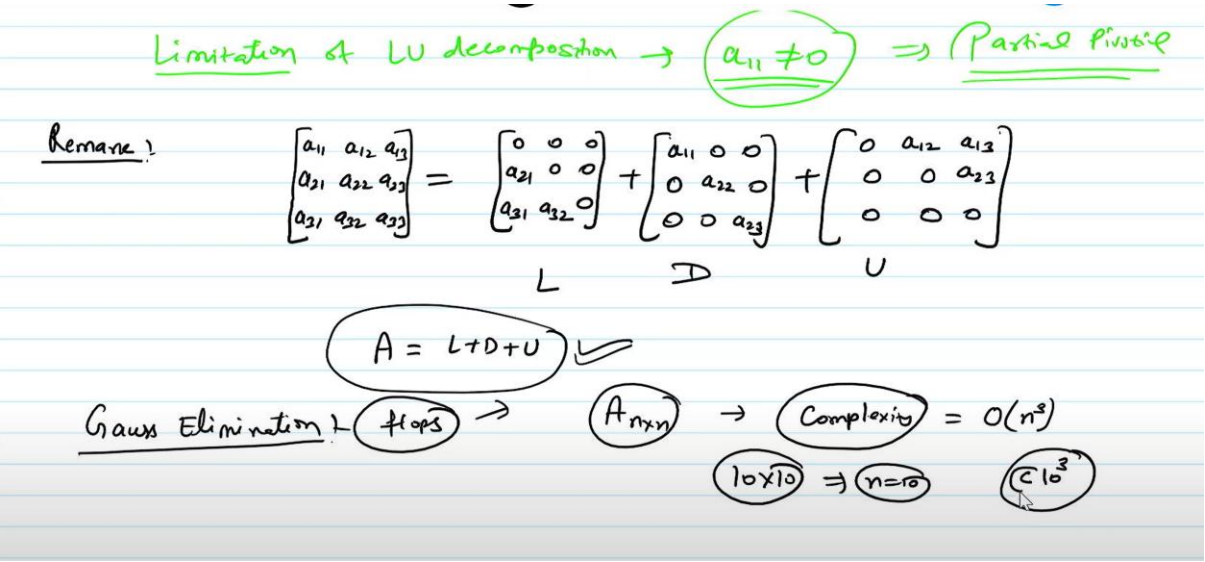
So, this is our LU decomposition. Just like this, we have just one — that is the outcome. And the remark is that — look — we can always write any matrix like this. Any matrix like a11, a12, a13, a21, a22, a23, a31, a32, a33, or write it like this: 0, 0, 0, 0, 0, 0, a21, a31, a32 plus a11, a22, a2 3 — zero will come here as well — and this is what we have, 0, a12, a13, a23 — this 0, this 0, this is also zero, this is also zero — this comes. If we see, then what is this? This is a lower triangular matrix. But this matrix is singular because its diagonal element is zero. But so — it is a lower triangular — so I can write it as — what is this? This is a diagonal matrix D. And what is this? This is an upper triangular matrix U.

So, we can write in it that if A is there, then I can convert it in this form: L + D + U. I can always write it in this form. Okay. So, here there is a matrix — we can always decompose it in this form: L plus D plus U. This always happens. So, if we do LU decomposition or not, we know about it or not — but doesn't matter — we can always process it. Okay.

So, now we have this — that the number of steps for this. How many steps do we have to take? If someone asks us, "Bhaiya, how many steps do we have in Gauss elimination?" — if we look at it — how many flops does the computer have to use, how many calculations does the computer have to do — if we have a matrix of A, n cross n — then how much work does the computer have to do?

So, it is said that the number of steps, or we call it a word, complexity of the algorithm or process — so this is the order of n cube. What does it mean? That if we have a matrix of 10 cross 10 — okay — if n becomes 10, then the computer will have to do approximately some constant multiplied by 10 to the power of 3. That means 1000, this is the minimum — it can be more than that also, depending on what this constant is.

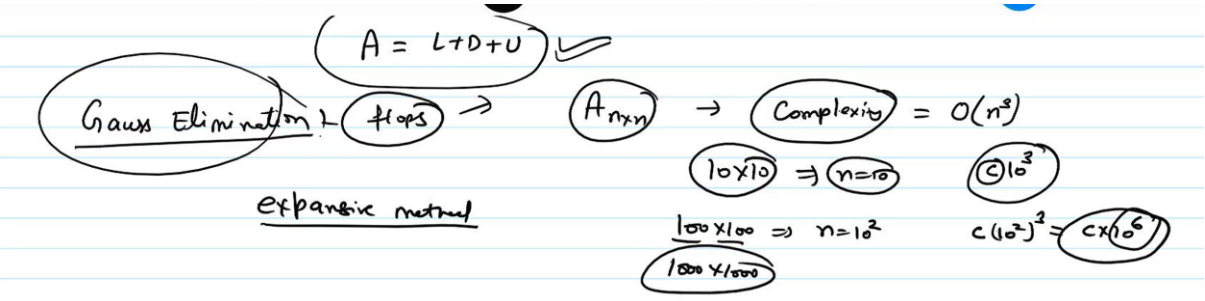
(Refer slide time: 44:32)



So, similarly, if we have 100 cross 100 — okay — then 100 cross 100 means what happens? n becomes 10 to the power 2. So, the number of calculations in this — the number of steps that the computer will have to take — will be some constant into 10 to the power 3. So, c into 10 to the power is 6. So, this much calculation will have to be done. And 100 cross 100 — so the matrix is very small. If we go to the 1000 cross 1000 matrix, then you can imagine how much calculation will have to be done.

So, that is why the Gauss elimination method is said to be a very expensive method. We say that this method is a very expensive method. So, but the advantage of this method is that we know that the solution that comes from this method will definitely be the solution. And in this case, if the diagonal elements of the matrix are much greater than other elements — and not that column — then it will definitely give us the solution. So, this is a direct method.

(Refer slide time: 46:07)



Now, after this comes another method, which is just a by-product of this — means from here we can do it very easily. We name that method as Cholesky method.

So, Cholesky method — what I said is that if matrix A is a symmetric matrix, then what will happen? If matrix A — we take the LU decomposition — and symmetric matrix is A transpose equal to A — what did I do? It will have to take the A transpose then have to take this side also also. So, it becomes U transpose L transpose.

So, what does it mean? If it becomes this — so, from here we can say that either U transpose is L, or L transpose is U , right? Right. Means, from here we can see the condition in this way — that from here we have the meaning that we have drawn the conclusion, and it happened that we can take the matrix A like this. If so, I take L, then its transpose will be U or of I take A, taking U then it will be U transpose U.

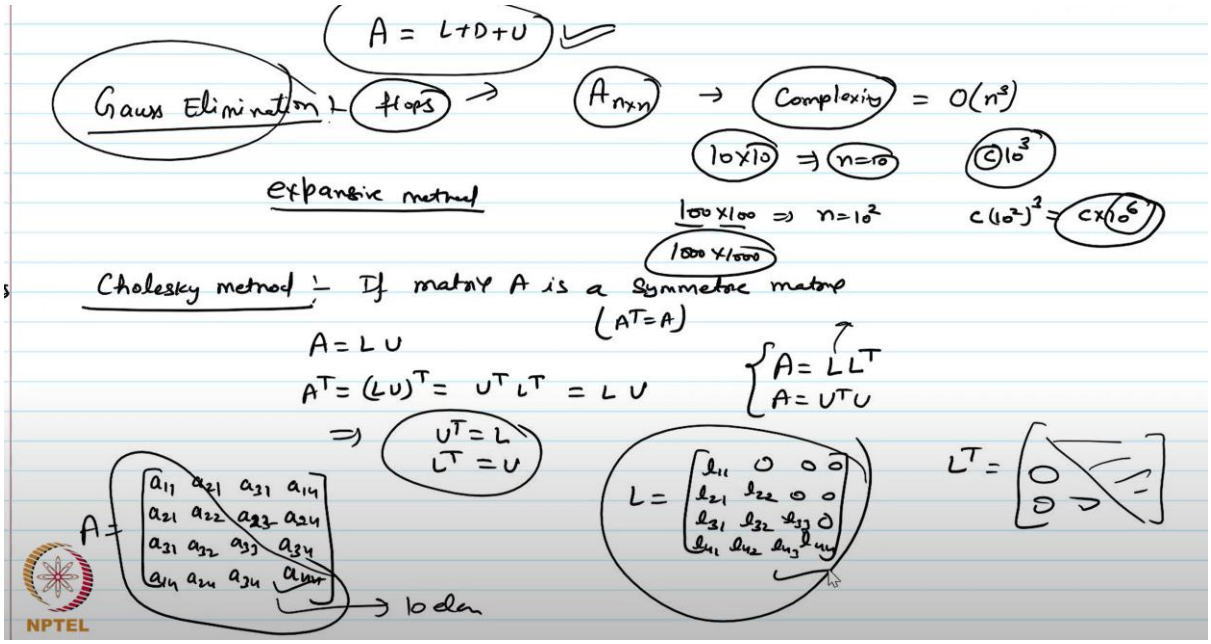
So, why? Because we already have the LU decomposition, right? Right. So, in this case, we have to keep in mind that L is not necessarily a unit lower matrix. So, we can do this — that here I will take L like this: l11, l22, l44 suppose — so, l21, l31, l32 — right? Isn't it? l33 — this is zero, l41, l42, l43.

So, if we take this L, then if we transpose it, we will get an upper triangle matrix. Okay, and we will find it out using LU decomposition. So now, look in this: 1, 2, 3, 4, 5, 6, 7, 8, 10 elements are required — it is unknown. We have L transformation, which we will take, that is, this matrix. This quantity will go here, and all that is below it will become zero. So, we have to find out 10 elements.

So, if the matrix is A, which is a symmetric matrix, then if we see, how many elements will be there in it? So, a11, a22, a33, a44 — these are already there. Now, if I write here, if I write l21, then l21 will come. But okay, if I write a31, if I write a32, then what will come here? a32 will come, a31 will come. Okay, so it goes on like this. So like this, I want to write this in four. Okay, let me make it a little bigger — a44 — so a34 and a34 will come, a24, a24, and a14, a14 this quantity will come.

So, you will see that in this also, we have the same elements left which we have to use, or it is the same. So this also has 10 elements. So from here also, we will do this and this, and with the help of this, the matrix which we have, L and U, will come out.

(Refer slide time: 50:51)



So, what happened in this case — it will become a little easy because only 10 quantity has to be found out. So, this method which has been given, we call it Cholesky method, and there is no other change. The only change is that the matrix is symmetric matrix. So, if the matrix is

symmetric and if we are applying LU decomposition in it, then basically we are using Cholesky method only.

So now, this is done. Now, let us see how to do this with Python. So now, with the help of Python code, we will see how to do this. So now, let us see in our code. We will find out. So, we have seen the method of LU decomposition. LU decomposition, we have seen that it can be done either by Gauss elimination or by crout's. So, we did it by Gauss elimination.

(Refer slide time: 52:10)

```python
import numpy as np

def lu_gauss_elimination(A):
    n = len(A)
    L = np.eye(n)  # Initialize L as identity matrix
    U = A.astype(float)  # Copy A to U

    for i in range(n):
        for j in range(i+1, n):
            factor = U[j, i] / U[i, i]
            L[j, i] = factor
            U[j] = U[j] - factor * U[i]

    return L, U

def lu_crout(A):
    n = len(A)
    L = np.zeros((n, n))
    U = np.eye(n)

    for i in range(n):
        for j in range(i, n):
            L[j, i] = A[j, i] - sum(L[j, k] * U[k, i] for k in range(i))
        for j in range(i+1, n):
            U[i, j] = (A[i, j] - sum(L[i, k] * U[k, j] for k in range(i))) / L[i, i]

    return L, U
```

So, what we did was — in the code that we wrote, we defined a function LU Gauss elimination. In the other, we did LU crout. So, what we will do is — we will give a matrix as input, and in that, we will get two matrices in the output, which will be L and U. Right? So, how do we do it? That means, we will write such code, and the criteria that we had told — that LU decomposition is very simple by Gauss elimination, the factors of that were made. So from there, we will use the multipliers that we had, and we will get L, U.

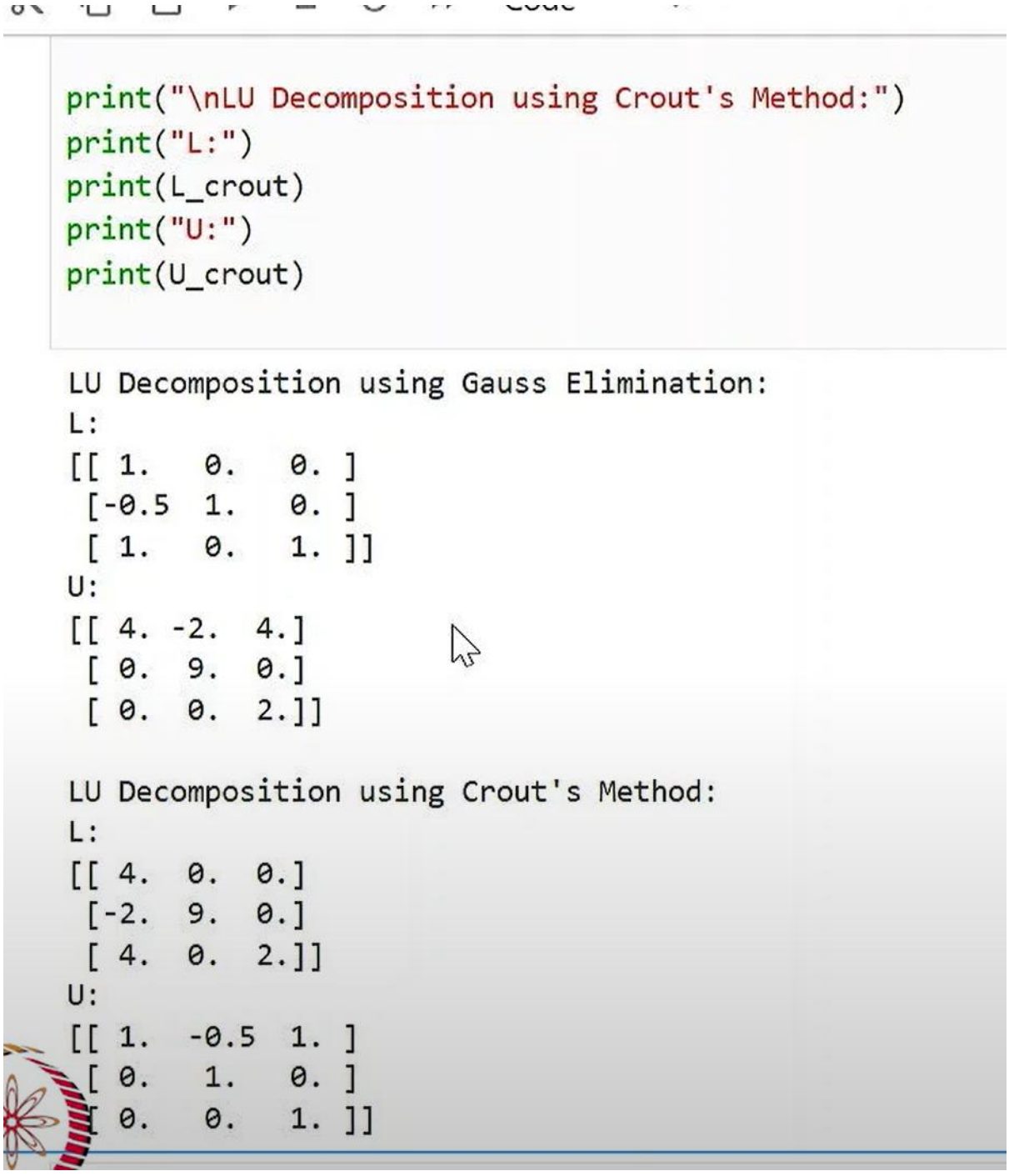
(Refer slide time: 52:57)

```python
import numpy as np

def lu_gauss_elimination(A):
    n = len(A)
    L = np.eye(n)  # Initialize L as identity matrix
    U = A.astype(float)  # Copy A to U

    for i in range(n):
        for j in range(i+1, n):
            factor = U[j, i] / U[i, i]
            L[j, i] = factor
            U[j] = U[j] - factor * U[i]

    return L, U

def lu_crout(A):
    n = len(A)
    L = np.zeros((n, n))
    U = np.eye(n)

    for i in range(n):
        for j in range(i, n):
            L[j, i] = A[j, i] - sum(L[j, k] * U[k, i] for k in range(i))
        for j in range(i+1, n):
            U[i, j] = (A[i, j] - sum(L[i, k] * U[k, j] for k in range(i))) / L[i, i]

    return L, U
```

And similarly, we will get U by gauss elimination. And in crout's also, we have seen that we will follow the steps that we had, and we will get L and U. So now, what will we do? We will do the same with the same matrix. We will calculate it with the help of both the methods. Okay? From Gauss and from LU decomposition — from the crout's.

So, we saw that this came. So we took the same matrix. The L that came in it came out to be this. Okay? And see, in this, L is 1, - 0.5,1, 1, so this is a lower triangular matrix with the unit elements will come on the diagonal, and U is this. Okay? So we came to know this from Gauss elimination.

(Refer slide time: 53:55)



In crout's, we saw that L is this — 4.2. So, what did we do? That the diagonal elements can be taken in any one — you can take one in L as well as in U. So, what was done in this case — if one is not taken in L, then it should be taken in the diagonal of U. There is no problem in that. The calculation will be the same.

So from here, the L we have is this, and U is this. Okay? So we have calculated this method. In this way, we can take any other matrix. Let's say I copy this, and here I define a matrix like this — let's say 3, 2, 1, -4 taking 4 by 4. Okay? And I took 1, -5, 2, and 1. Okay? And I took 5, 1, -3,and 2. I took this. Okay?

And I define another element — I take the last row here, 2, 3, 1, 5 — I defined. Okay? So what do I have to do now? Calculate it from both and see what is happening. If it comes, then we will do LU decomposition.

So we have this quantity. See, all these multipliers that were coming — the multipliers came here.

(Refer slide time: 56:21)

```
print("\nLU Decomposition using Crout's Method:")
print("L:")
print(L_crout)
print("U:")
print(U_crout)
```

```
LU Decomposition using Gauss Elimination:
L:
[[ 1.          0.          0.          0.        ]
 [ 0.33333333  1.          0.          0.        ]
 [ 1.66666667  0.41176471  1.          0.        ]
 [ 0.66666667 -0.29411765 -0.15384615  1.        ]]
U:
[[ 3.          2.          1.         -4.        ]
 [ 0.         -5.66666667  1.66666667  2.33333333]
 [ 0.          0.         -5.35294118  7.70588235]
 [ 0.          0.          0.          9.53846154]]

LU Decomposition using Crout's Method:
L:
[[ 3.          0.          0.          0.        ]
 [ 1.         -5.66666667  0.          0.        ]
 [ 5.         -2.33333333 -5.35294118  0.        ]
 [ 2.          1.66666667  0.82352941  9.53846154]]
U:
[[ 1.          0.66666667  0.33333333 -1.33333333]
 [ 0.          1.         -0.29411765 -0.41176471]
 [ 0.          0.          1.         -1.43956044]
 [ 0.          0.          0.          1.        ]]
```

And this upper triangular matrix came, whose first row — 3, 2, 1, -4— is the same. We already know that. Okay, a little chance comes from here. So this LU decomposition we did — we did it by Gauss elimination, we did it by crout's. So what is in crowd — the first column, same here, the quantity has changed. This upper triangular has come in the upper triangle — we have taken one here.

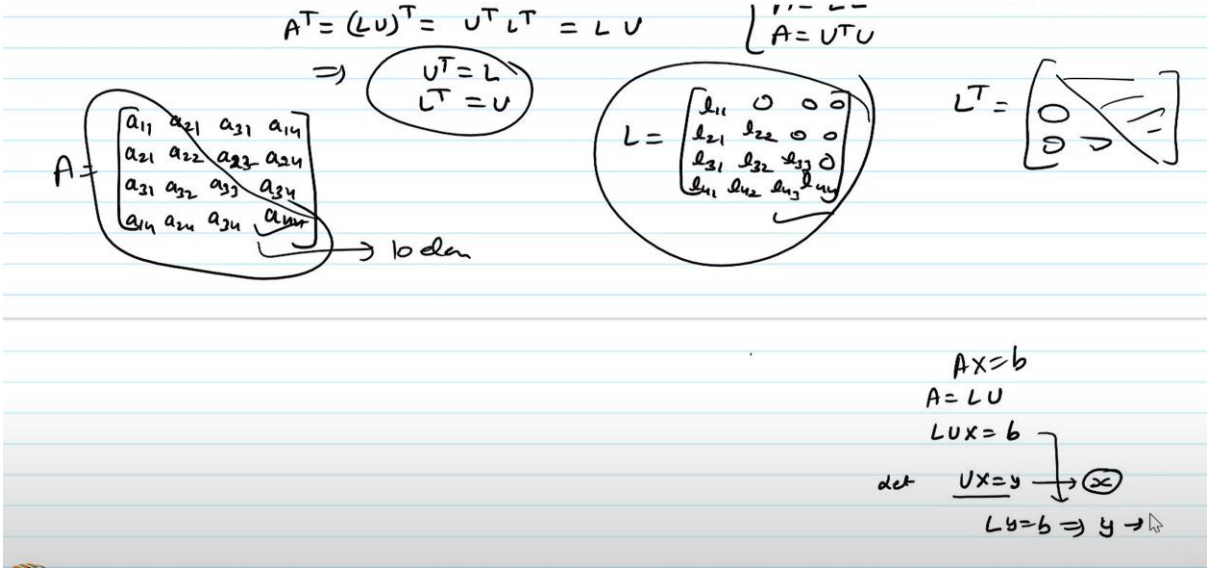
So, we can do the values of both in any way. It is not always lower because in this, it is well defined in Gauss elimination that our L will be one one always — we have checked that. But it is not like this in crout's. If you want to take one-one on the diagonal, then you can take it in L as well as in U. So, depending on the type you want to take, you can calculate it accordingly.

So, this matrix of ours has come. We have calculated this by LU decomposition. Now, we have calculated it, so we can solve it very easily. How will we solve that one?

Now look, we have calculated it by LU decomposition. So, we have A x = b system. So, what happened by having A x = B system — we have to solve it. So, I made A, LU. So, we have this LU x = b, what do I do? Let Ux be equal to y, which is unknown to us. Okay?

x is unknown, U is known, y is also unknown. So, I first let it be. So, if I let this quantity be this, which will become Ly=b. Now, b is known, L is known. So, from here we will have y find out, which we will put here. From here, x will become known.

(Refer slide time: 58:29)



So, first we will calculate y, which is unknown, and then we will calculate x. So, from here, y is calculated, then x is calculated. So, in this way, we can solve any system very easily. Okay?

So, by LU decomposition also, we can solve the system easily. From here, we took out y, and after that, we calculated the value of x.

So today, in the lecture, we did LU decomposition by using Gauss elimination. Okay? We used the Crout's method, discussed the Cholesky method, and those which — the codes are — that we discussed Python code. So, I hope you understood this lecture, and thank you for watching this lecture.