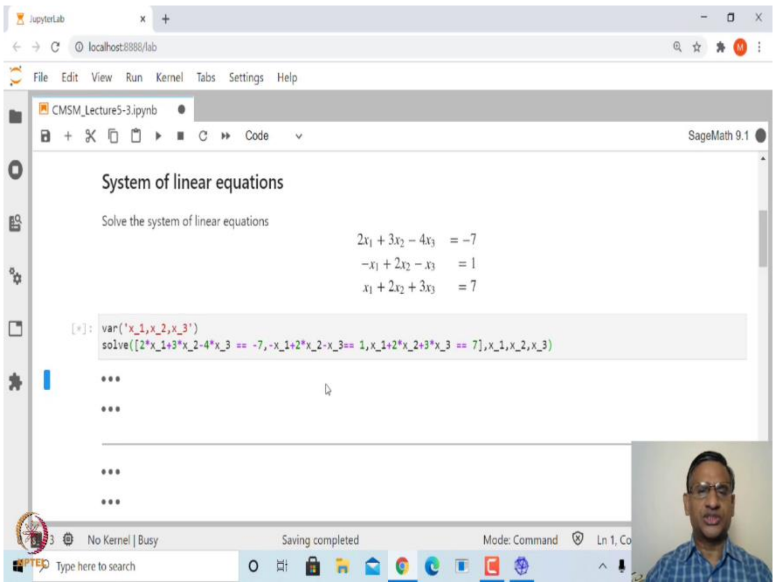
Computational Mathematics with SageMath

Prof. Ajit Kumar
Department of Mathematics
Institute of Chemical Technology, Mumbai

Linear Algebra with Sagemath
System of linear equations
Lecture – 30
Solving system of linear Equations in SageMath

Welcome to the 30th lecture on Computational Mathematics with SageMath. In this lecture, we will explore solving system of linear equations. So, let us start with an example.
(Refer Slide Time: 00:29)



System of linear equations

Solve the system of linear equations

$$\begin{aligned} 2x_1 + 3x_2 - 4x_3 &= -7 \\ -x_1 + 2x_2 - x_3 &= 1 \\ x_1 + 2x_2 + 3x_3 &= 7 \end{aligned}$$

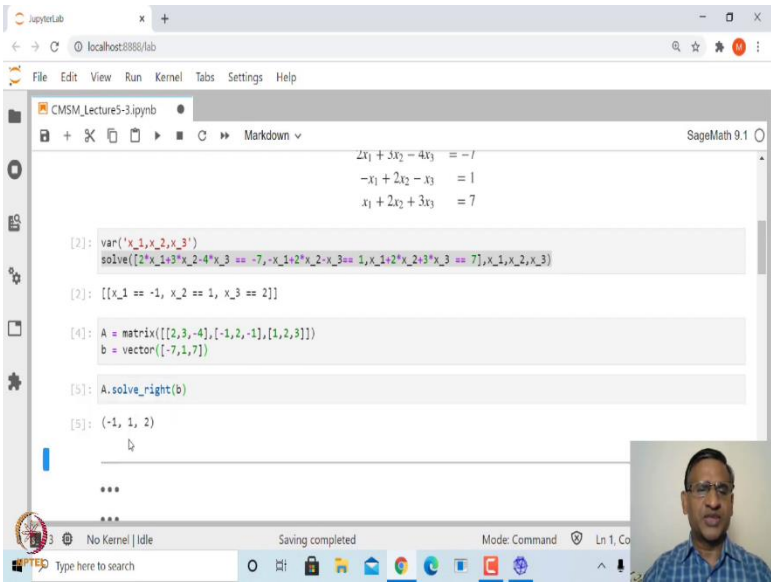
```
[*]: var('x_1,x_2,x_3')
     solve([2*x_1+3*x_2-4*x_3 == -7,-x_1+2*x_2-x_3== 1,x_1+2*x_2+3*x_3 == 7],x_1,x_2,x_3)
```

So, suppose you have a system of linear equations. These are 3 equations in 3 variables x1, x2, x3.

So, we have seen how to solve such a system of linear equations using solve command. So, let us just recall that, and solve the system using inbuilt function solve. So, how do we do that first we need to declare x1 x2, x3 as variables. So, var x1 underscore, x2, x underscored 2, and so on.

And then we use solve function, solve, write the set of equations inside square brackets separated by comma, and then we want to solve with respect to variable x1, x2, x3, so let us run this. So, when we run this, it will give me solution of this system. It takes little bit time to begin with, and later on it will become faster, right?

(Refer Slide Time: 01:23)



$$\begin{aligned} 2x_1 + 3x_2 - 4x_3 &= -7 \\ -x_1 + 2x_2 - x_3 &= 1 \\ x_1 + 2x_2 + 3x_3 &= 7 \end{aligned}$$

```
[2]: var('x_1,x_2,x_3')
     solve([2*x_1+3*x_2-4*x_3 == -7,-x_1+2*x_2-x_3== 1,x_1+2*x_2+3*x_3 == 7],x_1,x_2,x_3)

[2]: [[x_1 == -1, x_2 == 1, x_3 == 2]]

[4]: A = matrix([[2,3,-4],[-1,2,-1],[1,2,3]])
     b = vector([-7,1,7])

[5]: A.solve_right(b)

[5]: (-1, 1, 2)
```

So, the solution is : x1 is equal to 1, x2 is equal to 1, sorry x1 is equal to minus 1, x2 is equal to 1, and x3 is equal to 2. If you want to get the solution in terms of dictionary, you can say solution underscore dictionary is equal to true. This is solving system using inbuilt function solve.

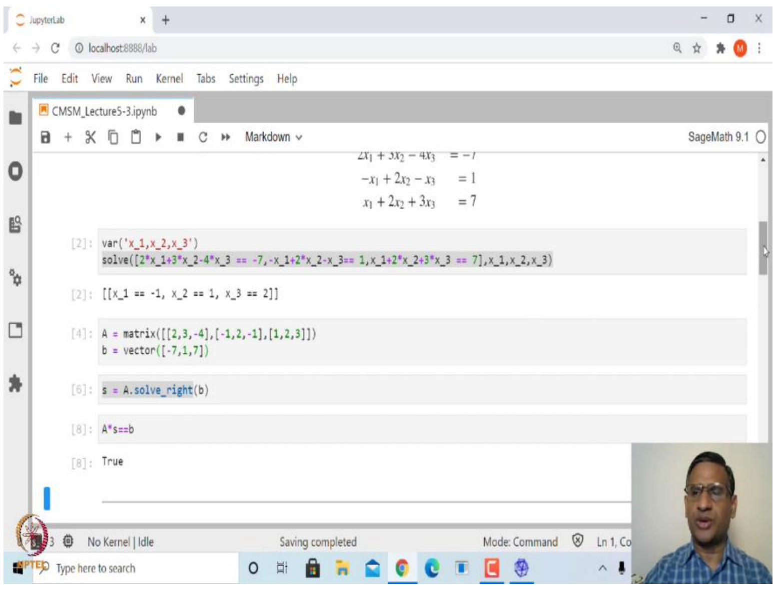
However, you can also write this equation as a matrix equation. So, you can write the coefficient matrix A, which will be a 3 by 3 matrix whose first row is 2, 3, minus 4, second row is minus 1, 2, minus 1, third row will be 1, 2, 3, and then the b is the column matrix minus 7, 1, 7.

So, let us see how we are going to declare. So, we declared this matrix A, which is the coefficient matrix, and the matrix b, this we will declare as a vector because by default, when you declare a vector, it is thought of as a column matrix. So, therefore, we, we have a coefficient matrix, and b is the right-hand side.

Now we can solve again using this solve function, but slightly different. So, there is a function called solve underscore right, and in the bracket, you write b. So, A dot solve underscore right, see there is also notion of solve underscore left, because sometimes you write x instead of x equal to b. If you write x transpose A is equal to b, then in that case you can use solve underscore left, but most commonly we use solve underscore, right?

That means we write the set of equations as Ax equal to b, ok? So, let us run this. When we run this, we should just, this I did not execute. So, let me execute this again. When we execute A dot solve underscore right, it will give you solution as a vector.

(Refer Slide Time: 03:27)



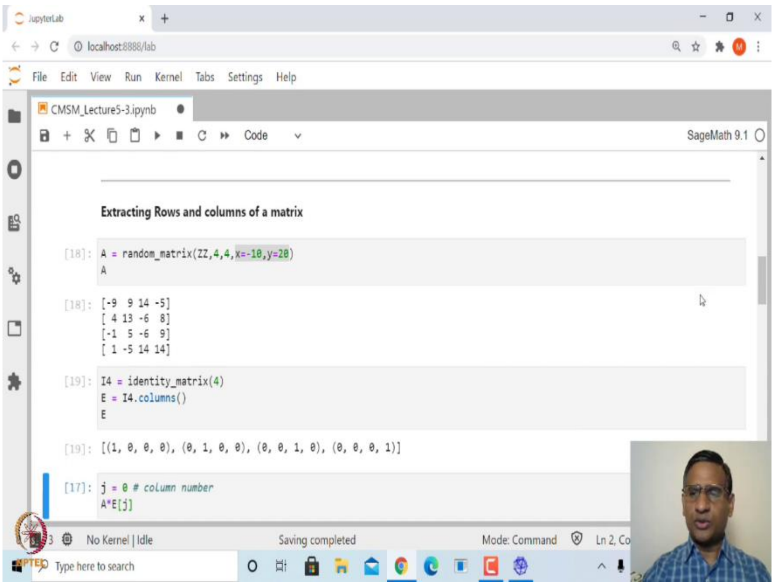
$$\begin{aligned} 2x_1 + 3x_2 - 4x_3 &= -7 \\ -x_1 + 2x_2 - x_3 &= 1 \\ x_1 + 2x_2 + 3x_3 &= 7 \end{aligned}$$

```
[2]: var('x_1,x_2,x_3')
     solve([2*x_1+3*x_2-4*x_3 == -7,-x_1+2*x_2-x_3== 1,x_1+2*x_2+3*x_3 == 7],x_1,x_2,x_3)

[2]: [[x_1 == -1, x_2 == 1, x_3 == 2]]

[4]: A = matrix([[2,3,-4],[-1,2,-1],[1,2,3]])
     b = vector([-7,1,7])

[6]: s = A.solve_right(b)

[8]: A*s==b

[8]: True
```

Now, suppose if I want to store this in s is equal to solution, and then we can check whether this is a solution. So, if it is a solution, I should get A star b. This should be equal to A star s, this should be equal to b, and that is what you can see here.

So, you can check whether this is double equal to b, right? The answer is true. So, there are two ways of solving system of linear equations in, in SageMath. One, you can use solve command, and other one, you can use solve underscore, right? In that case, you have to think of the system as a system declared as A x equal to b, right?

(Refer Slide Time: 04:33)


Extracting Rows and columns of a matrix

```
[18]: A = random_matrix(ZZ,4,4,x=-10,y=20)
      A

[18]: [-9  9 14 -5]
      [ 4 13 -6  8]
      [-1  5 -6  9]
      [ 1 -5 14 14]

[19]: I4 = identity_matrix(4)
      E = I4.columns()
      E

[19]: [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)]

[17]: j = 0 # column number
      A*E[j]
```

Now, let us play with this matrix, because the solution at present we are using this inbuilt function, and we do not know how this solution is obtained, using this inbuilt function. So, let us look at some concepts that leads to solving the system of linear equations using what is called Gaussian elimination method.

Next, let us look at how to extract rows and columns of a matrix by matrix multiplication. So, we have already seen that Sage provides a function called A dot columns, which will extract columns as vectors. Similarly, A dot rows will extract rows of A, but when you want to perform matrix multiplication and various other operations, it is very, I mean it is convenient to think of the rows and column-wise.

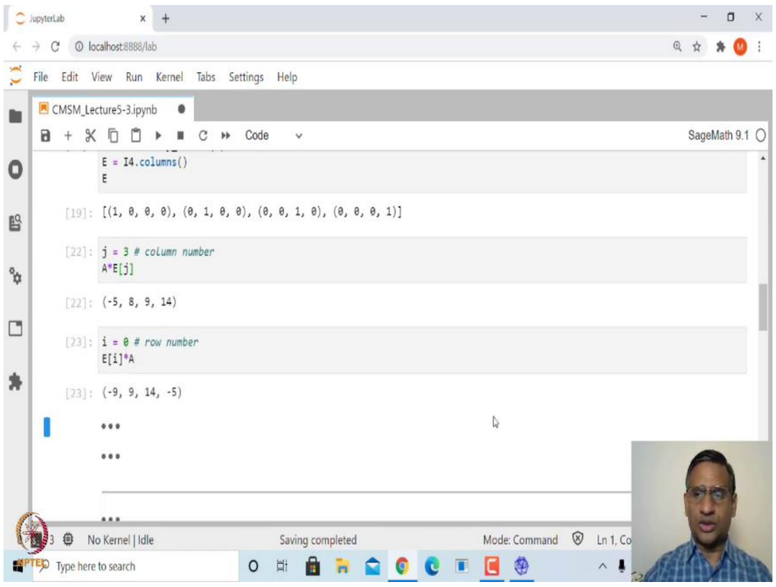
So, let us see what, what we can do. So, suppose you have a matrix, random matrix, a 4 cross 4 random matrix, which is matrix of integers whose entries varies between minus 10 and 20. So, in that case, let us see how do we extract first column, second column, third column, without using Sage inbuilt function, just as a matrix multiplication.

So, what we can do is, let us write 4 cross 4 identity matrix. Let us declare that as I4. This can be obtained using identity underscore matrix, and in the bracket you write the order of that matrix, and then E is equal to I4 dot columns. So, we are, we are extracting E as column vectors.

So, first column vector E is generally, we write, we write as E1 which is 1, 0, 0, 0, second E2, which is 0,1,0,0, and so on. So, in general, Ei is ith unit vector along the coordinate xi, right? Now, let us see what happens, suppose we multiply A by E, by first column of the identity matrix, namely 1 0 0 right, so what will you get?

So, if we say, for example, j is equal to 0, and look at Ej, that means, E0, that is the first column of A.

(Refer Slide Time: 06:35)



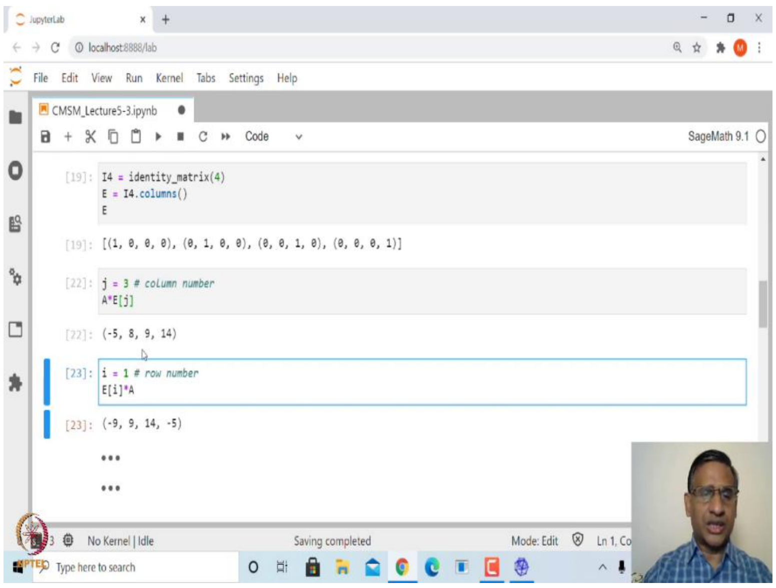
```
E = I4.columns()
E

[19]: [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)]

[22]: j = 3 # column number
      A*E[j]

[22]: (-5, 8, 9, 14)

[23]: i = 0 # row number
      E[i]*A

[23]: (-9, 9, 14, -5)
```

And when you multiply by A, what will you get? It is the first column of A, that is what you can see here, minus 9, 4, minus 1, 1. Similarly, if I take j is equal to 1, this will give me the second column. If I take j is equal to 3, it will give me fourth column, and that is what you can show, you can see.

So, you can obtain ith column of a matrix A, by A times Ei, right? So, that is a vector obtained by multiplying A with Ei on the right-hand side, that is what it means. So, ith column of a matrix can be thought, thought of as A times Ei. Similarly, instead of multiplying on the right-hand side, suppose we multiply on the left-hand side.

Suppose we multiply Ei by A, right? So, what will you get? So, if I multiply by Ei on the left of A, and what you get, you can see here, this is the first row of this matrix. So, E0 times the first row.

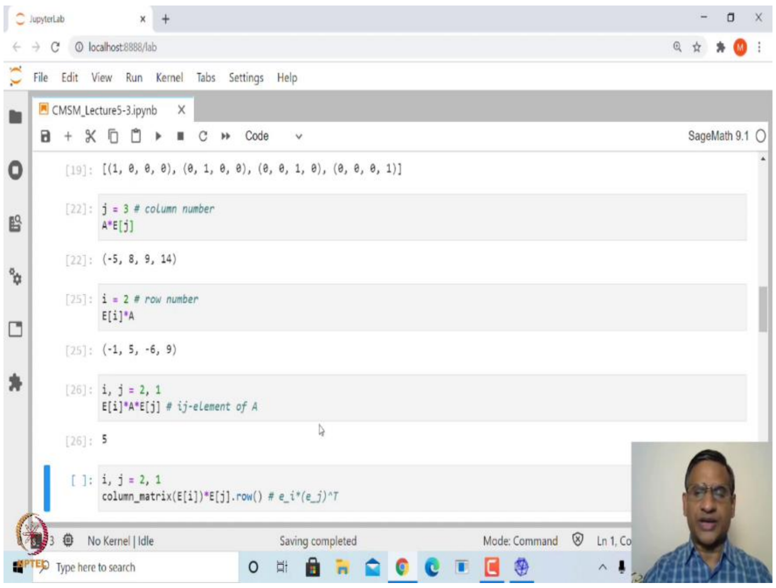
(Refer Slide Time: 07:45)



```
[19]: I4 = identity_matrix(4)
      E = I4.columns()
      E

[19]: [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)]

[22]: j = 3 # column number
      A*E[j]

[22]: (-5, 8, 9, 14)

[23]: i = 1 # row number
      E[i]*A

[23]: (-9, 9, 14, -5)
```

If I take E equal to i equal to 1, and then it will give me second row.

(Refer Slide Time: 07:51)



```
[19]: [(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)]

[22]: j = 3 # column number
      A*E[j]

[22]: (-5, 8, 9, 14)

[25]: i = 2 # row number
      E[i]*A

[25]: (-1, 5, -6, 9)

[26]: i, j = 2, 1
      E[i]*A*E[j] # ij-element of A

[26]: 5

[ ]: i, j = 2, 1
     column_matrix(E[i])*E[j].row() # e_i*(e_j)^T
```

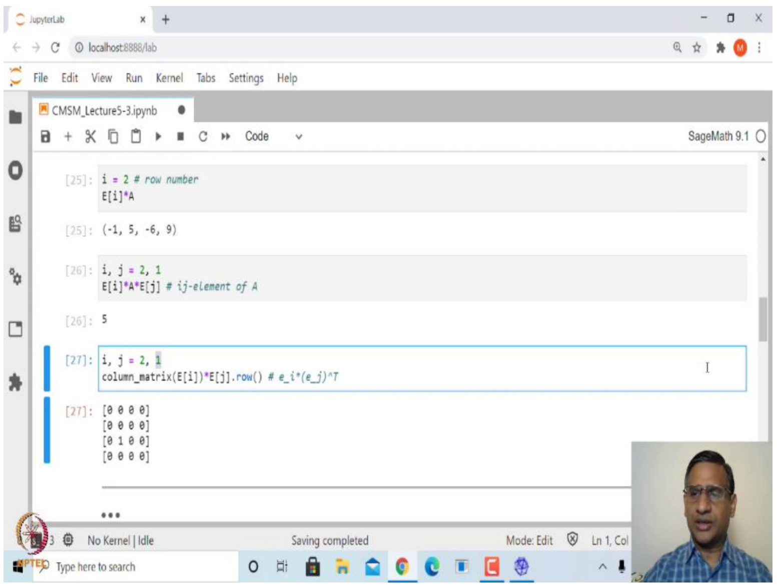
If I take i equal to 2, it will give me third row, and so on. So, we can write any row and any column by multiplying Ei with A on the left, or on the right, ok? When you multiply on the left, it will give you row, when you multiply on the right, it will give you columns.

So, that is one way of thinking of rows and columns of a matrix, this actually. So, basically, we are visualizing as vectors right, ok? In case you multiply, let us say, Ei with with A, and with Ej, what will you get? A times Ej is a vector, column vector, and then you are multiplying by Ei. So, what you are getting, this will give you, actually a dot product. So, it will be ijth entry.

So, for example, in this case, when i is equal to 2, j is equal to 1, what will you get? What you are going to get is third row, second column element, which is 5, and you can verify that third row, and second column, third row, and second column entry is 5. This is what you got.

So, ijth entry you can also obtain using this. Now suppose you multiply, look at Ei, and multiply by Ej, ok? So, Ei you think of, as a column matrix, and they are multiplied as Ej as a row matrix. So, Ei times Ej transpose this will give you again a 4 cross 4 matrix whose, in this case, i plus 1th row, and j plus 1th column will be will be 1, remaining all will be 0.

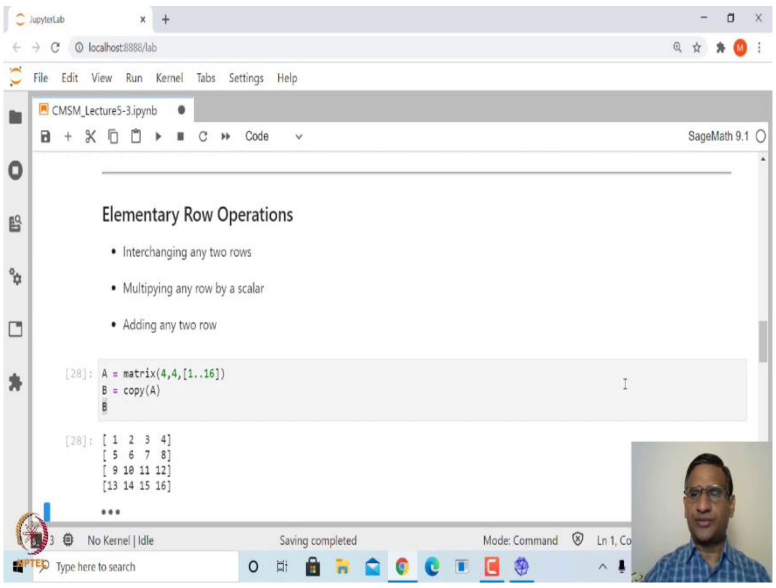
(Refer Slide Time: 09:29)



```
[25]: i = 2 # row number
      E[i]*A

[25]: (-1, 5, -6, 9)

[26]: i, j = 2, 1
      E[i]*A*E[j] # ij-element of A

[26]: 5

[27]: i, j = 2, 1
      column_matrix(E[i])*E[j].row() # e_i*(e_j)^T

[27]: [0 0 0 0]
      [0 0 0 0]
      [0 1 0 0]
      [0 0 0 0]
```

So, in this case, this is i equal to 2, that is why you can see the third row, and j is equal to 1. So, that is second column third row, and second column entry is 1, remaining all are 0. So, that is how you can obtain Ei times Ej, right? So, this actually helps us to understand matrix multiplication and many other things.

(Refer Slide Time: 10:11)



Elementary Row Operations

- Interchanging any two rows
- Multiplying any row by a scalar
- Adding any two row

```
[28]: A = matrix(4,4,[1..16])
      B = copy(A)
      B
```

```
[28]: [ 1  2  3  4]
      [ 5  6  7  8]
      [ 9 10 11 12]
      [13 14 15 16]
```

Now, when you want to solve system of linear equations, generally you must have seen that in order to solve this, we generally do some kind of operations, row operations. So, there are 3 row operations which leaves the solution unchanged, what are they?

You can interchange any two rows. It is same as saying interchange any two equations. When you interchange any two linear equations, the solution does not change. You can multiply any equation by a scalar, that again the solution does not change, or you can add any two rows. You can add one row to another row, and, or you can take any combination of all these things.

So, these are called elementary row operations, because when you apply these operations on, on a set of linear equations, the solution does not change, and, and this can be translated to row operations, to the system of linear equations Ax equal to b.

So, and I am sure you must have solved many problems by using elementary row operation, and the idea behind using elementary row operation is that you want to convert the system into a simpler system. Of course, I mean you can ask what is meaning of simple system.

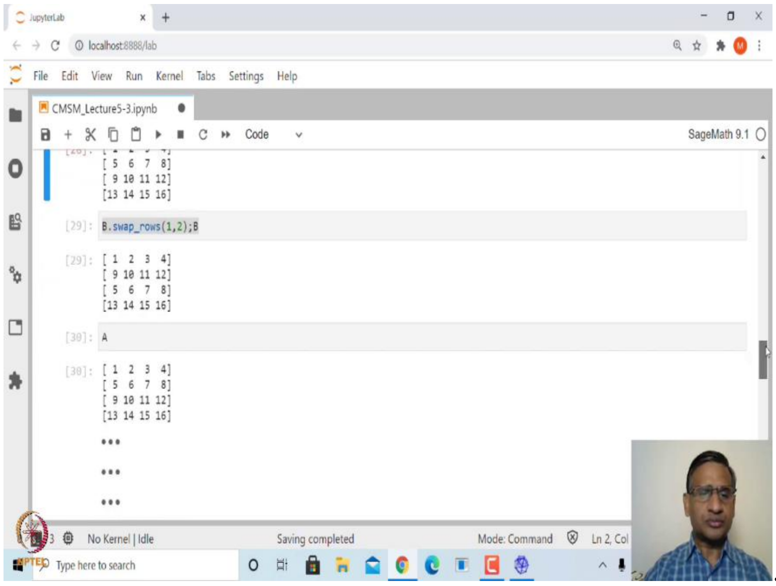
So, for example, if it, if the coefficient matrix is identity matrix, all you are looking at, if the coefficient matrix A is identity matrix, and your system of linear equation Ax equal to b, that is same as saying x1 is equal to b1, x2 is equal to b2, and so on, xn is equal to bn. So, that is the, the easiest system you can think of, right?

So, if some way, if you can convert the coefficient matrix A into identity matrix by system of elementary row operation, that is the best thing you can ask for. So, that is the basic idea. Even if you have upper triangular matrix or lower triangular matrix as a coefficient matrix, then also solving system of linear equations becomes much simpler.

For example, if you have upper triangular matrix, then you can solve the system by what is called back substitution, if you have lower triangular matrix you can solve this by forward substitutions. So, that is the basic idea of converting this matrix into matrix which is simpler to deal with, right?

So, let us look at what are the elementary row operations that we have. So, suppose you have a matrix A, which is 1, 2, 3, 4, up to 16. So, 4 cross 4 matrix, and this is another way of declaring a matrix. So, we have declared the elements of this matrix as a list, and then we are declaring that whether the order is 4 cross 4, right?

And then let us store this A into B, so copy of A into B, right? So, let us do that. So, B is this 1, 2, 3, 4, 5, 6, 7 up to 16, 4 cross 4 matrix. (Refer Slide Time: 13:15)



```
      [ 5  6  7  8]
      [ 9 10 11 12]
      [13 14 15 16]

[29]: B.swap_rows(1,2);B

[29]: [ 1  2  3  4]
      [ 9 10 11 12]
      [ 5  6  7  8]
      [13 14 15 16]

[30]: A

[30]: [ 1  2  3  4]
      [ 5  6  7  8]
      [ 9 10 11 12]
      [13 14 15 16]
```

.

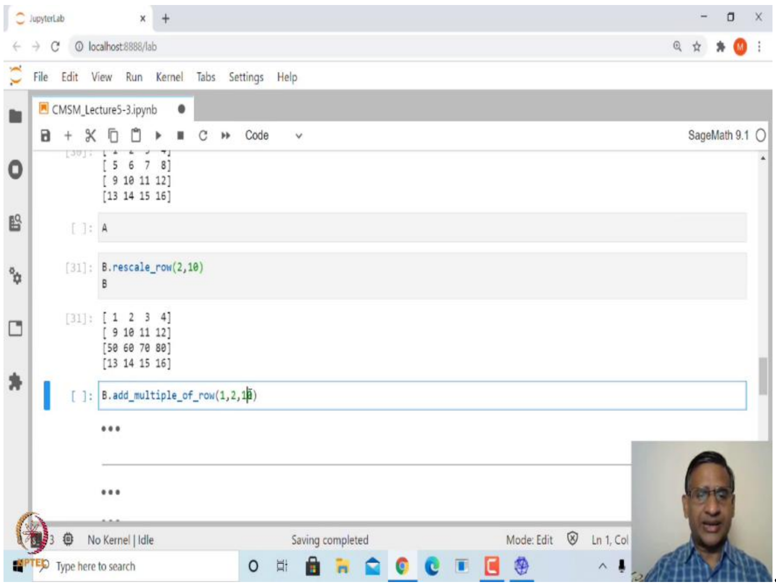
Now, suppose we want to do some elementary row operation. So, if I, for example, there is one elementary row operation called swap rows, swap rows i,j will swap ith row and j th row, ok?

So, if I, if I run this, A dot swap rows 1 comma 2, what are we expecting? We are expecting third row and second row to be interchanged, ok? So, let us run this, and when you call, what happens to B? So, you can see here, the, the second row was, was starting with 5.

Now that has become third row, and the third row was starting with 9, that has become second row. Now you can ask for what has happened to the matrix A. So, if I, if I, let me insert a cell here, and if I asked what is, what had happened to the matrix A, the matrix A has not been changed. It remains as it is; only B has changed.

Whereas, if we had declared B equal to A, then whatever operations that you are going to perform on B, will all also be reflected on A. So, A will also change. That is the reason why we are taking copy of A rather than declaring A equal to B, right, ok?

(Refer Slide Time: 14:29)



```
       [ 5  6  7  8]
       [ 9 10 11 12]
       [13 14 15 16]

[ ]:   A

[31]:  B.rescale_row(2,10)
       B

[31]:  [ 1  2  3  4]
       [ 9 10 11 12]
       [50 60 70 80]
       [13 14 15 16]

[ ]:   B.add_multiple_of_row(1,2,10)
```

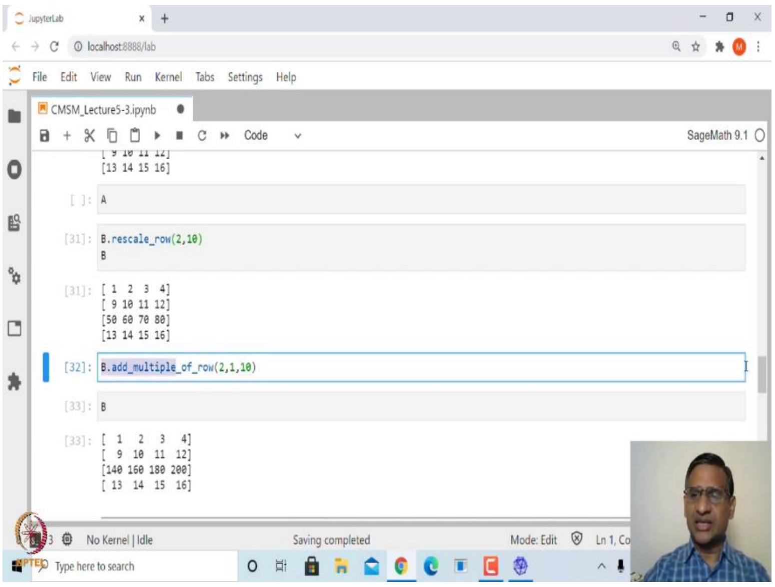
Let us look at other elementary row operations.

So, for example, if I, if I look at B dot rescale, and then underscore row 1 comma 10, means the second row will be multiplied by 10. Let us look at, we want to multiply the third row by 10, and then see what happens. So, third row was starting with 5. It was 5, 6, 7, 8, and it has been multiplied by 10. Again you can, you can check that A would not have changed. A will remain as the original matrix.

So, this is the, the scaling any row. Now suppose you want to, for example, multiply a row by some scalar, and add to another row. For that, you can use B dot add multiple of underscore row 1, comma 2, comma 10 means what?

The second row in this case will get multiplied by 10, and will get added to sorry the here 2 means third row will be multiplied by 10, and it will get added to the second row.

(Refer Slide Time: 15:41)



```
 [ 9 10 11 12]
 [13 14 15 16]

[ ]: A

[31]: B.rescale_row(2,10)
      B

[31]: [ 1  2  3  4]
      [ 9 10 11 12]
      [50 60 70 80]
      [13 14 15 16]

[32]: B.add_multiple_of_row(2,1,10)

[33]: B

[33]: [  1   2   3   4]
      [  9  10  11  12]
      [140 160 180 200]
      [ 13  14  15  16]
```

Now, let let me interchange here let me say here 2, and this is 1, and then let us ask for what happens to to B.

So, what has happened to B you can see here now this this first the the second row that is multiplied by 10. So, this is 90 plus 50, 140, and then 10 into 10, this is 100, plus 60, 160, and so on. So, this, this B dot add underscore multiple, underscore of, underscore row, i comma j, means what? This comma, let us say s, this, then what will happen?

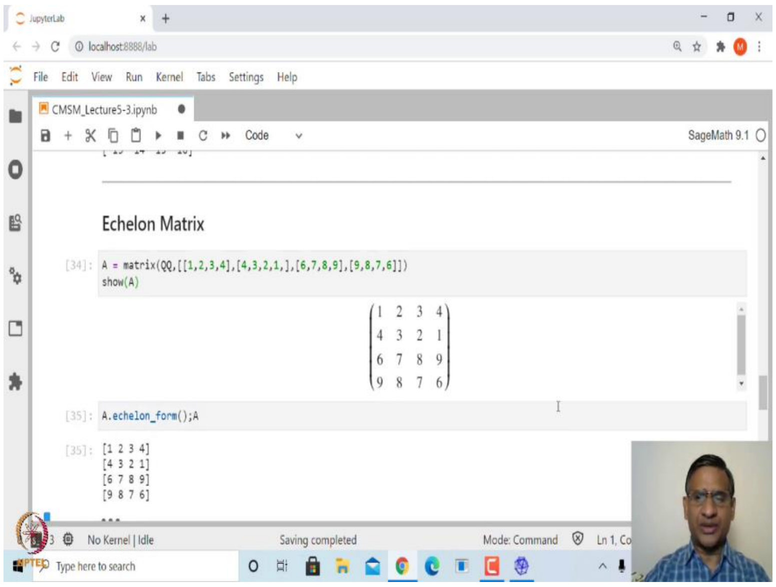
It will multiply this sth row, s plus 1th row in this case, by, sorry, j plus 1th row by s, and add to r plus 1th row, that is what it, it is doing. So, this is, these are the elementary row operations that already are inbuilt, and you can perform them, right, ok? So, and you can keep on applying these operations.

Now when you apply these operations, as I said, when, the idea of applying these operations in terms of finding solution, is to convert this system into simpler form, so that solving becomes easy. Of course, in case of my 3 cross 3 matrix, 4 cross 4 matrix it is not very difficult.

But if you, you are dealing with, let us say 1000 linear equations in 1000 many variables, then solving will be an uphill task. In that case, if you can convert the matrix A into simpler form, it would become much simpler. So, in that sense, and when you obtain, when you apply any row operation. Let us say we applied row operation to A, and obtain a matrix B, then A and B are called row equivalent matrices.

Row equivalent matrices means what? The solution will not change the A and B. If Ax equal to b, and if I look at Ax equal to, let us say, b, and another matrix M, which is row equivalent to A, Mx plus, Mx equal to B, then this Ax these two systems are actually equivalent. They represent, actually the system of linear equations, right?

(Refer Slide Time: 18:01)



Echelon Matrix

```
[34]: A = matrix(QQ,[[1,2,3,4],[4,3,2,1,],[6,7,8,9],[9,8,7,6]])
      show(A)
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 6 & 7 & 8 & 9 \\ 9 & 8 & 7 & 6 \end{pmatrix}$$

```
[35]: A.echelon_form();A
```

```
[35]: [1 2 3 4]
      [4 3 2 1]
      [6 7 8 9]
      [9 8 7 6]
```

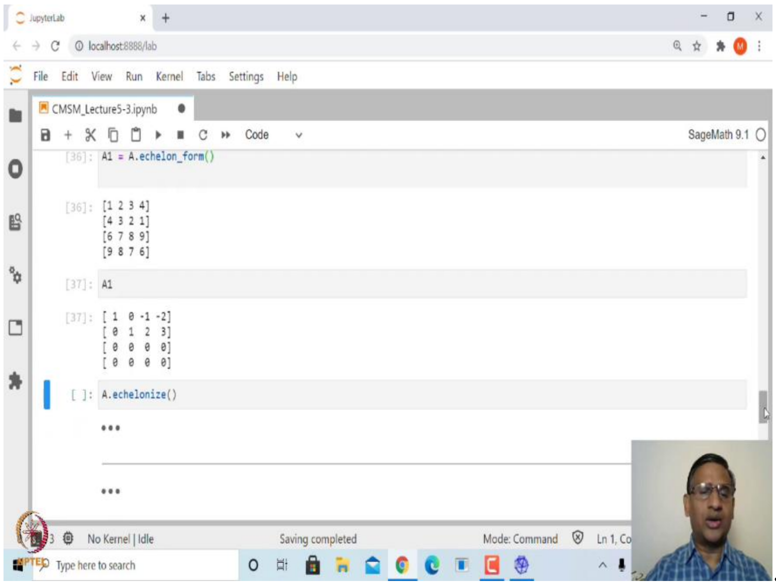
So, there is a matrix called echelon matrix to a given matrix. The echelon matrix is actually a matrix, it is in, which in case there are some 0 rows that will appear at the bottom, and nonzero rows will be at the beginning, and also if you look at in any row, if I take any two rows, let us say ith row, and jth row, if there is a nonzero element in ith row, and nonzero element in jth row, then what should happen if in case j is bigger than i?

Then the, the nonzero element on the ith row should be to the left of nonzero element to the jth row ok, and any element after if you look at, in any row if there is a nonzero element. Then in that particular column, all the elements following that should be 0, and that element generally we call it as a leading element.

So, leading element, if it is non, non zero leading element, if you look at that particular column, all the entries after that will be 0 that, those are called echelon matrices. And if you are given a matrix A which is, let us say this is the matrix A. Now you can convert this matrix into an echelon matrix using inbuilt function, which is called echelon form. A dot echelon form of the matrix A.

So, it will, what it will do? It will apply elementary row operations, and it will convert into equivalent matrix which is an echelon matrix, so let us apply this. So, when we applied this, and look at what has happened to A. Of course, A remains as it is when you apply. This is echelon matrix.

(Refer Slide Time: 19:49)

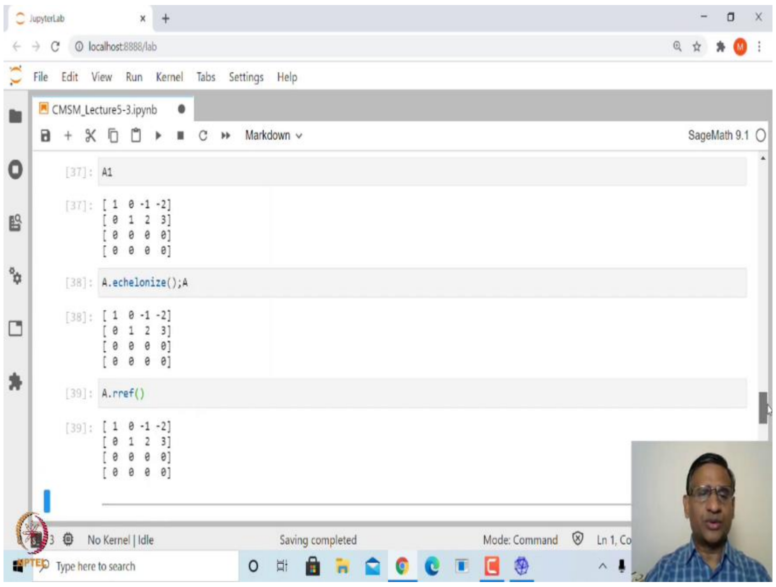


```
[36]: A1 = A.echelon_form()

[36]: [1 2 3 4]
      [4 3 2 1]
      [6 7 8 9]
      [9 8 7 6]

[37]: A1

[37]: [ 1  0 -1 -2]
      [ 0  1  2  3]
      [ 0  0  0  0]
      [ 0  0  0  0]

[ ]: A.echelonize()
```

Let me call this as A1, and let us see what is A1 now. So, A remains as it is, and if I say what happened to A1, then A1 is actually echelon matrix, which is in the, which is a row equivalent to the matrix A. A and A1 are row equivalent. You can also write another thing. You can call A dot echelonize.

So, what it will do? When we, when we saw that, when we say A dot echelon form, A does not change, right? Whereas, if I say A dot echelonize, A itself will change.

(Refer Slide Time: 20:31)



```
[37]: A1

[37]: [ 1  0 -1 -2]
      [ 0  1  2  3]
      [ 0  0  0  0]
      [ 0  0  0  0]

[38]: A.echelonize();A

[38]: [ 1  0 -1 -2]
      [ 0  1  2  3]
      [ 0  0  0  0]
      [ 0  0  0  0]

[39]: A.rref()

[39]: [ 1  0 -1 -2]
      [ 0  1  2  3]
      [ 0  0  0  0]
      [ 0  0  0  0]
```

So, let me, let me see, what is A in this case? So A itself has changed.

So, you can see here what, this is echelon matrix. It is that, there are two 0 rows which are appearing at the bottom here. First two rows are nonzero, and then leading entry in, in the first row is 1, and here also 1, and this entry is to the left of this nonzero entry, that is why it is called echelon matrix, ok?

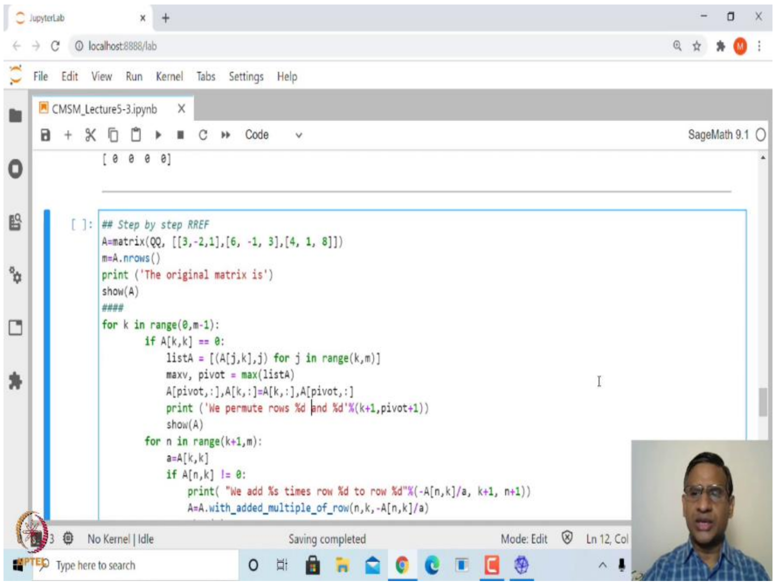
You can, if you have bigger example, you can take some arbitrary matrix, try to convert into echelon form. You will be, you will understand much better. Of course, this echelon form of a matrix need not be unique, but what is unique is actually called RREF reduced row echelon form. So, you keep on reducing this till you cannot proceed any further, right?

So, for example, it is, it is possible that I can make these elements also 0. If it is possible, I do not, I do not know. So, in that case, we, we try to keep on applying this elementary row operation till the ,the point from which you cannot reduce any further, ok?

So, this is in case, for example, if A is invertible matrix, RREF of that will be the identity matrix. So, let us see what is RREF in this case. RREF is same as the echelon form of A.

So, that in this case both are the same so it hardly matters, but however, in many cases it may be different, and RREF, that means, Reduced Row Echelon Form of a matrix is unique. That is one advantage of applying RREF, right?

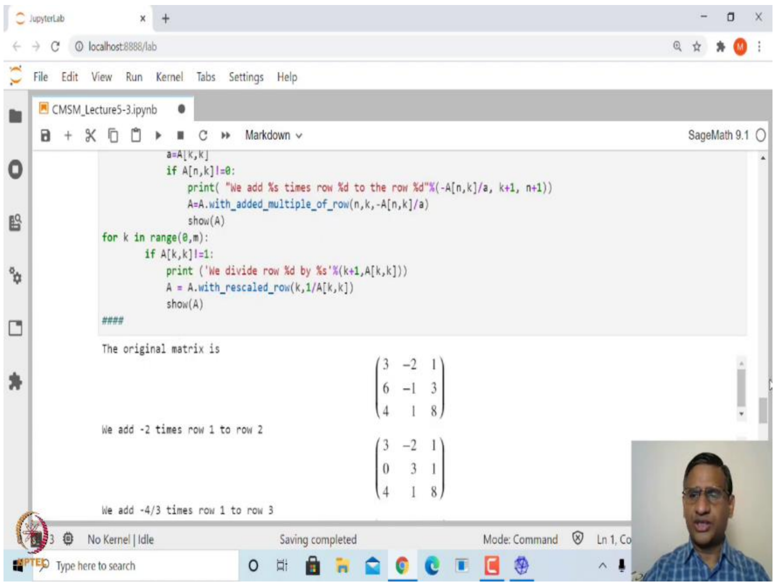
(Refer Slide Time: 22:21)



```
[ 0  0  0  0]
```

```
[ ]: ## Step by step RREF
     A=matrix(QQ, [[3,-2,1],[6, -1, 3],[4, 1, 8]])
     m=A.nrows()
     print ('The original matrix is')
     show(A)
     ####
     for k in range(0,m-1):
             if A[k,k] == 0:
                 listA = [(A[j,k],j) for j in range(k,m)]
                 maxv, pivot = max(listA)
                 A[pivot,:],A[k,:]=A[k,:],A[pivot,:]
                 print ('We permute rows %d and %d'%(k+1,pivot+1))
                 show(A)
             for n in range(k+1,m):
                 a=A[k,k]
                 if A[n,k] != 0:
                     print( "We add %s times row %d to row %d"%(-A[n,k]/a, k+1, n+1))
                     A=A.with_added_multiple_of_row(n,k,-A[n,k]/a)
```

Now, suppose if you want to write your own function to convert any matrix into RREF, then it is quite simple. So, this particular code gives you step by step conversion of a given matrix to RREF. So, all these steps like converting the matrix in row, interchanging the rows, and scaling a row by adding any the scalar multiple of row to another row, etcetera, you can, you can perform, and then you can run this. (Refer Slide Time: 23:05)



```
                 a=A[k,k]
                 if A[n,k]!=0:
                     print( "We add %s times row %d to the row %d"%(-A[n,k]/a, k+1, n+1))
                     A=A.with_added_multiple_of_row(n,k,-A[n,k]/a)
                     show(A)
     for k in range(0,m):
             if A[k,k]!=1:
                 print ('We divide row %d by %s'%(k+1,A[k,k]))
                 A = A.with_rescaled_row(k,1/A[k,k])
                 show(A)
     ####
```

```
The original matrix is
```

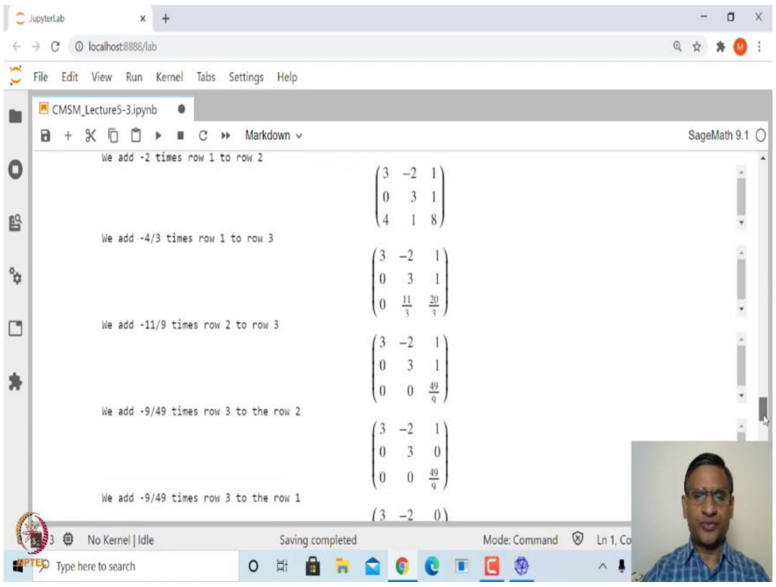
$$\begin{pmatrix} 3 & -2 & 1 \\ 6 & -1 & 3 \\ 4 & 1 & 8 \end{pmatrix}$$

```
We add -2 times row 1 to row 2
```

$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 1 \\ 4 & 1 & 8 \end{pmatrix}$$

```
We add -4/3 times row 1 to row 3
```

So, let me just show you this code. Let me just run this code, and then see here. (Refer Slide Time: 23:09)



```
We add -2 times row 1 to row 2
```

$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 1 \\ 4 & 1 & 8 \end{pmatrix}$$

```
We add -4/3 times row 1 to row 3
```

$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 1 \\ 0 & \frac{11}{3} & \frac{20}{3} \end{pmatrix}$$

```
We add -11/9 times row 2 to row 3
```

$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 1 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We add -9/49 times row 3 to the row 2
```

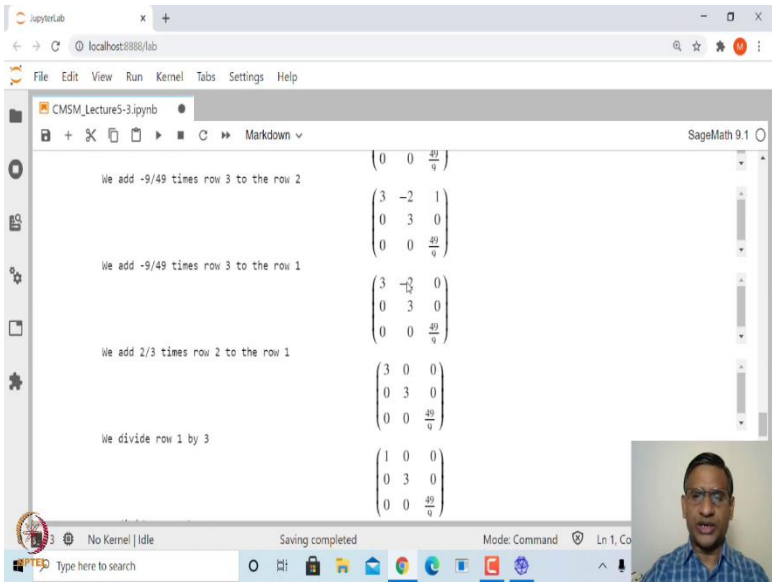
$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We add -9/49 times row 3 to the row 1
```

$$\begin{pmatrix} 3 & -2 & 0 \end{pmatrix}$$

You can, you can see here, it is performing step by step. So, let us look at what it is doing. So, this was the original matrix, and what it is doing? It is converting this into 0 first, that means, the second entry, then this into 0, and then it will go to the next step, and next step once this is making this 0.

(Refer Slide Time: 23:33)



$$\begin{pmatrix} 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We add -9/49 times row 3 to the row 2
```

$$\begin{pmatrix} 3 & -2 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We add -9/49 times row 3 to the row 1
```

$$\begin{pmatrix} 3 & -2 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

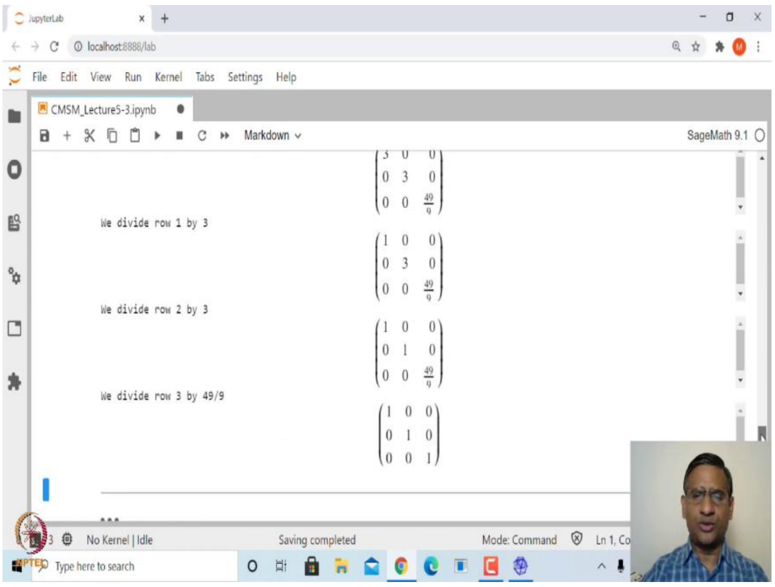
```
We add 2/3 times row 2 to the row 1
```

$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We divide row 1 by 3
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

And then next step will make, make this particular entry 0, and things like that. (Refer Slide Time: 23:37)


$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We divide row 1 by 3
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

```
We divide row 2 by 3
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{49}{9} \end{pmatrix}$$

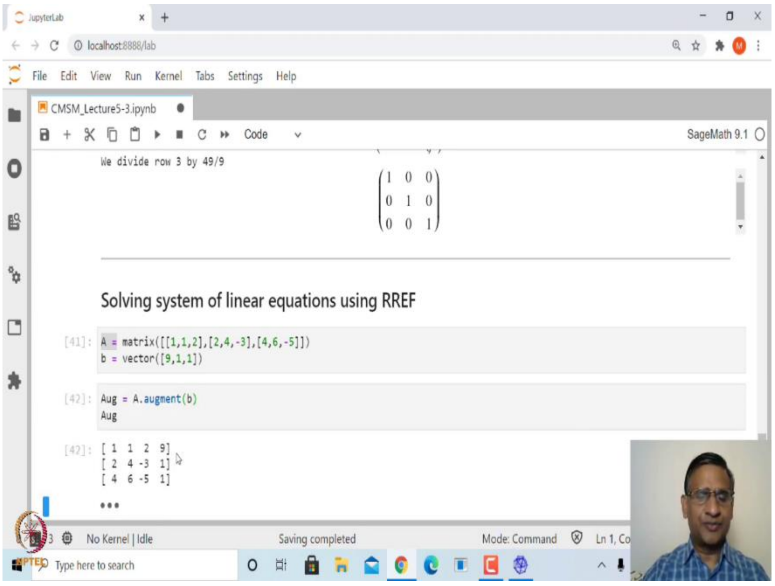
```
We divide row 3 by 49/9
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So, you can keep on continuing this till you get identity matrix. In this case, this since the matrix is invertible, row echelon form of this matrix, reduced row echelon form of this matrix will be identity matrix.

So, this is the row echelon form, or reduced row echelon form, is very useful. This, you can use it for solving system of linear equations and many other computation in linear algebra actually boils down to elementary row operation. So, this we will keep coming, we will keep using this RREF in several concepts, especially when we want to do certain computations.

(Refer Slide Time: 24:19)


```
We divide row 3 by 49/9
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

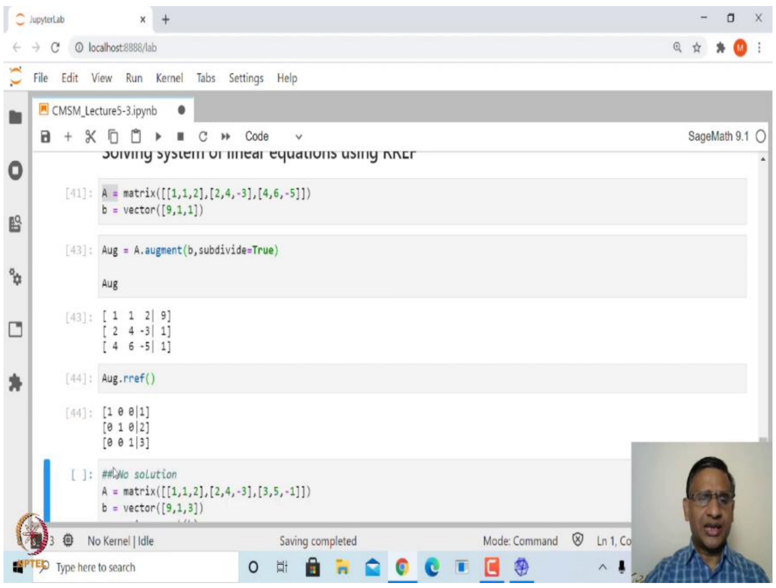
Solving system of linear equations using RREF

```
[41]: A = matrix([[1,1,2],[2,4,-3],[4,6,-5]])
      b = vector([9,1,1])

[42]: Aug = A.augment(b)
      Aug

[42]: [ 1  1  2  9]
      [ 2  4 -3  1]
      [ 4  6 -5  1]
```

So, let us look at how we can solve this system of linear equations using RREF. So, this I am sure you must have already done it, but let me demonstrate this using SageMath, so how do we do that? So, let us take a system of linear equations Ax equal to b. In this case, this, these are 3 equations in 3 variables, and what do we, do we convert? We make an augmented matrix.

So, we augment this vector b to this matrix A. So, that is the inbuilt function to find augment, to find augmented matrix. So, you simply say A dot augment, and in the bracket, you write b, in the bracket you write b. So, for example, let me just run with only this. When you do that, you can see here this column b is augmented to the matrix A, but you can also mention.

So, many times when you do this augmented matrix, generally you put also a vertical bar just to differentiate between the coefficient matrix, and the, the last column which is inserted. So, that option also is there in a, in a, in Sage, you can, you can say what is called. (Refer Slide Time: 25:29)



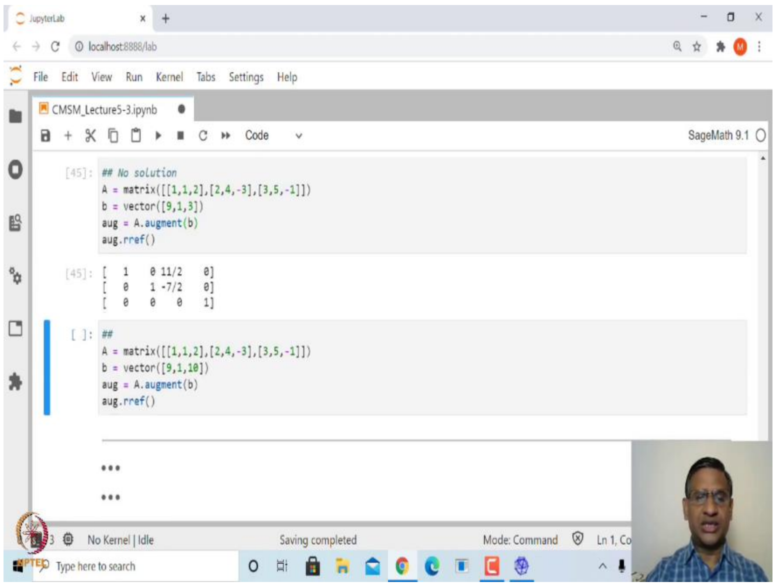
```
[41]: A = matrix([[1,1,2],[2,4,-3],[4,6,-5]])
      b = vector([9,1,1])

[43]: Aug = A.augment(b,subdivide=True)

      Aug

[43]: [ 1  1  2| 9]
      [ 2  4 -3| 1]
      [ 4  6 -5| 1]

[44]: Aug.rref()

[44]: [1 0 0|1]
      [0 1 0|2]
      [0 0 1|3]

[ ]: ##No solution
     A = matrix([[1,1,2],[2,4,-3],[3,5,-1]])
     b = vector([9,1,3])
```

Let me just say here, this, yeah, let me say this is called sub divide, sub divide is equal to true.

So, when you say sub divide equal to true, now you can see there is a vertical bar that is put in order to distinguish the matrix A, and this vector b, right? Now once you have created this augmented matrix, then what you, you should, you can do? You can apply

RREF to this augmented matrix, row reduced echelon form to this augmented matrix, and whatever you get on the last column will actually reflect the solution.

So, let us say, in this case, when we do RREF to this augmented matrix, last column is 1, 2, 3. So, what this equation represents? The Ax equal to b has been converted into this: x1 is equal to 1, x2 is equal to 2, and x3 is equal to 3. So, that, that means, 1, 2, 3, is a solution of this system of linear equation, right? So, that is, and now we can extract the last column using the sub matrix, or using just the indices, right? (Refer Slide Time: 26:43)

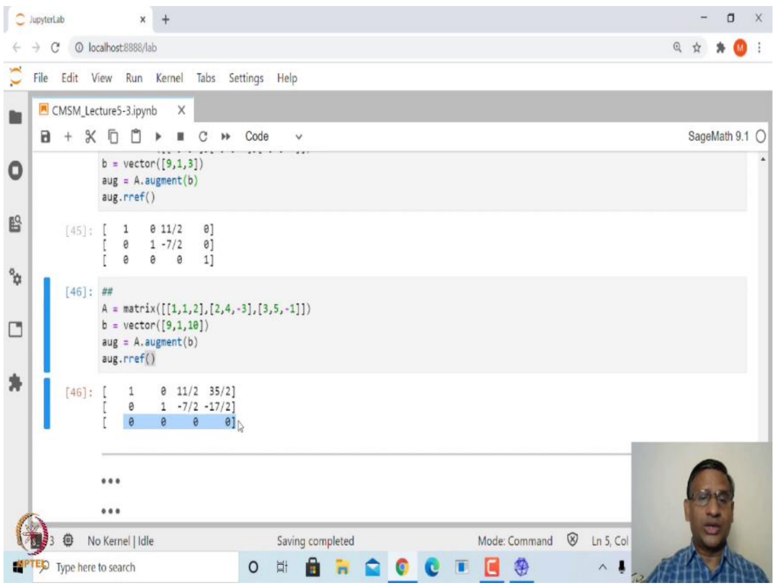


```
[45]: ## No solution
      A = matrix([[1,1,2],[2,4,-3],[3,5,-1]])
      b = vector([9,1,3])
      aug = A.augment(b)
      aug.rref()

[45]: [   1    0 11/2    0]
      [   0    1 -7/2    0]
      [   0    0    0    1]

[ ]:  ##
      A = matrix([[1,1,2],[2,4,-3],[3,5,-1]])
      b = vector([9,1,10])
      aug = A.augment(b)
      aug.rref()
```

Now, suppose you have another set of system of linear equations Ax equal to b, and in this case, suppose we applied RREF to augmented matrix in this case, what are we getting? We, we, we are getting here, you can see here what we are getting. In the last row, A has become all 0 whereas, this side is 1.

So, this last row will represent 0 times x1, plus 0 times x2, plus 0 times x3, is equal to 1. So, that cannot happen, therefore, this means that this system of linear equations does not have a solution. So, in, when you apply RREF to, to augmented matrix, and if you see that on the right hand side, if there is something nonzero on the left hand side, that is the, the row corresponding to the matrix A.

If it is 0, that reflects that this system does not have a solution. However, in case, let us say, last row has become all 0, let us take an example. So, suppose we take this system of linear equations. So, A is 1, 1, 2, B is 2, 4, minus 3, C is sorry, the first row is 1, 1, 2, second row is 2, 4, minus 3, third row is 3, 5, minus 1, and so on. And now if I apply RREF to this system of linear equations then what we get? (Refer Slide Time: 28:05)



```
b = vector([9,1,3])
aug = A.augment(b)
aug.rref()
```

```
[45]: [   1    0 11/2    0]
      [   0    1 -7/2    0]
      [   0    0    0    1]
```

```
[46]: ##
A = matrix([[1,1,2],[2,4,-3],[3,5,-1]])
b = vector([9,1,10])
aug = A.augment(b)
aug.rref()
```

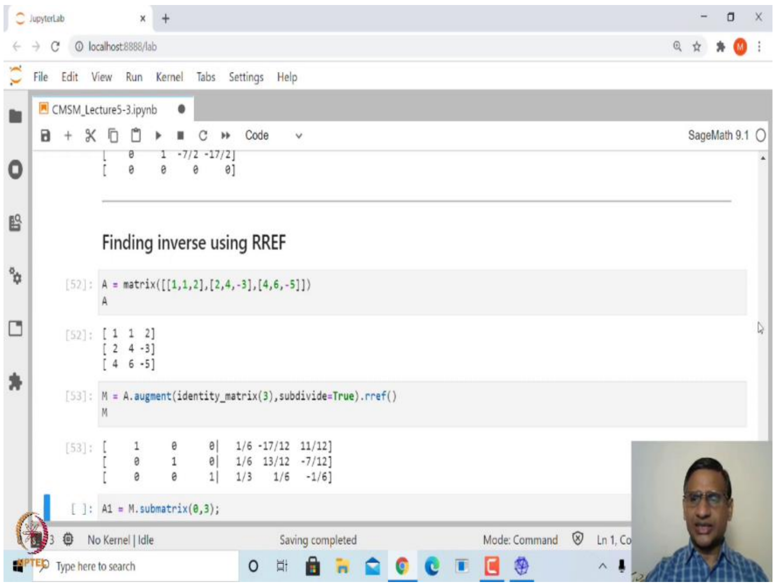
```
[46]: [    1     0  11/2  35/2]
      [    0     1  -7/2 -17/2]
      [    0     0     0     0]
```

We get last row is completely 0; that means, what does it imply? It, it implies that this last row is x1 times 0, plus x2 times 0, plus x3 times 0, is equal to 0, which is obviously true, which is always true, but this actually has only 2 equations in 3 variables.

So, for last equation, what does it represent? x2 times x2 plus, minus 7 by 2 times x3, is equal to minus 17 by 2. So, and then the first equation is x1 plus 11 by 2 times x3 is equal to 35 by 2. So, this system actually there are two equations in 3 variables. It will have infinitely many solutions.

So, in such situation, you have infinitely many solutions, and there are many other things that can be explained using this notion of row reduced echelon form. So, for example, you can find inverse of a matrix, you can find rank of a matrix, rank of a matrix will be nothing but nonzero rows in row echelon form, right?

So, so for example, if you look at this, this matrix A, and when we applied RREF to this matrix then what we, we got was this matrix is, first row, two rows are nonzero, remaining are all 0s. Therefore, the rank of this matrix will be 2. So, let us look at another use of RREF in finding inverse of a matrix. (Refer Slide Time: 29:35)



```
[     0     1 -7/2 -17/2]
[     0     0    0     0]
```

Finding inverse using RREF

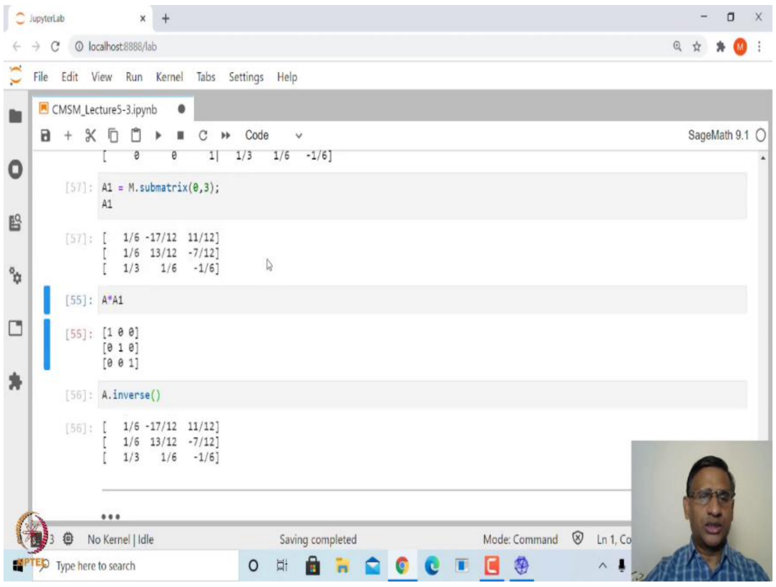
```
[52]: A = matrix([[1,1,2],[2,4,-3],[4,6,-5]])
      A

[52]: [ 1  1  2]
      [ 2  4 -3]
      [ 4  6 -5]

[53]: M = A.augment(identity_matrix(3),subdivide=True).rref()
      M

[53]: [     1      0      0|   1/6 -17/12  11/12]
      [     0      1      0|   1/6  13/12  -7/12]
      [     0      0      1|   1/3    1/6   -1/6]

[ ]: A1 = M.submatrix(0,3);
```

So, suppose you want to find inverse of a matrix. Let us say A is this matrix, how do we find its inverse? What we can do, is we can augment the identity matrix to A. Ee can, we augmented one column, rather than one column, you can argument even a matrix of appropriate size, and then apply RREF to this.

So, A dot augment identity matrix 3 by 3, and then subdivide equal to true, and then apply RREF. So, that is what you get, and what you have got here, actually these last 3 rows and 3 columns, is the inverse. So, let us extract that matrix, let us extract that matrix in A1. So, what what are we going to extract? We will extract these last 3 rows and 3 columns.

So, how do I do that? We can use sub matrix, and this is starting from 0th column, that is the first column, sorry 0th row, first row, and take all the, the columns, right? And this, take from 0th row, take, and third column, that is the fourth column, take all of them.

(Refer Slide Time: 30:53)



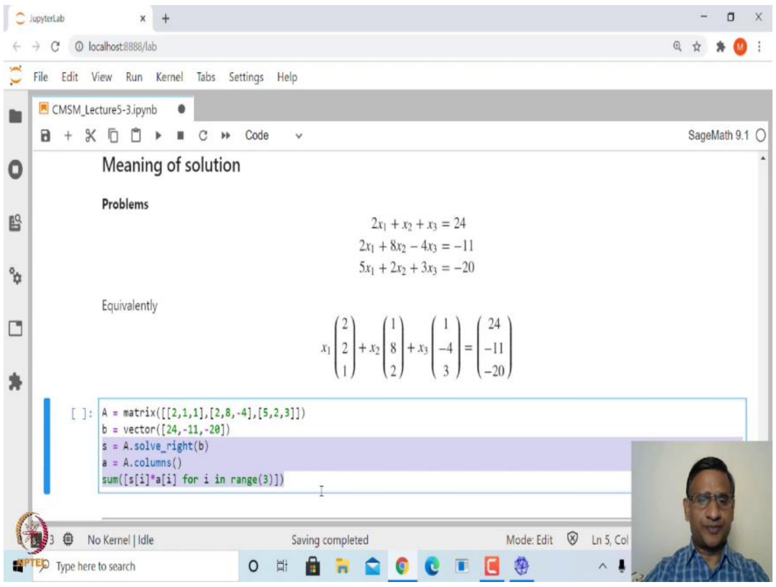
```
[     0      0      1|   1/3    1/6   -1/6]

[57]: A1 = M.submatrix(0,3);
      A1

[57]: [   1/6 -17/12  11/12]
      [   1/6  13/12  -7/12]
      [   1/3    1/6   -1/6]

[55]: A*A1

[55]: [1 0 0]
      [0 1 0]
      [0 0 1]

[56]: A.inverse()

[56]: [   1/6 -17/12  11/12]
      [   1/6  13/12  -7/12]
      [   1/3    1/6   -1/6]
```

So, this is your A1, and now you can check whether A times A1, can check whether A times A1, is it identity matrix? The answer is true.

Therefore, this is the inverse of the matrix A. So, actually this is nothing but, when we find the inverse, it is nothing but solving system of linear equations. So, actually what we are doing? We are solving A times some first column of B which is equal to identity. So, that is why you augmented here identity, first column of identity matrix here, augmented second column of identity matrix, and, and so on, right?

So, that is actually, it is solving these 3 systems of linear equations in 3 variables, and in which coefficient matrix A is common, right? So, this is one way of finding inverse. Of course, Sage has inbuilt function to find inverse of a matrix. You can, for example, look at A dot inverse, and then see whether you are getting the same thing. This should be equal to the same thing A1. Let us print this A1, A1, and this A dot inverse they are the same thing. So, we have, what we got using RREF is same as inverse of a matrix using inbuilt function, right?

(Refer Slide Time: 32:07)


Meaning of solution

Problems

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 24 \\ 2x_1 + 8x_2 - 4x_3 &= -11 \\ 5x_1 + 2x_2 + 3x_3 &= -20 \end{aligned}$$

Equivalently

$$x_1 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ 8 \\ 2 \end{pmatrix} + x_3 \begin{pmatrix} 1 \\ -4 \\ 3 \end{pmatrix} = \begin{pmatrix} 24 \\ -11 \\ -20 \end{pmatrix}$$

```
[ ]: A = matrix([[2,1,1],[2,8,-4],[5,2,3]])
     b = vector([24,-11,-20])
     s = A.solve_right(b)
     a = A.columns()
     sum([s[i]*a[i] for i in range(3)])
```

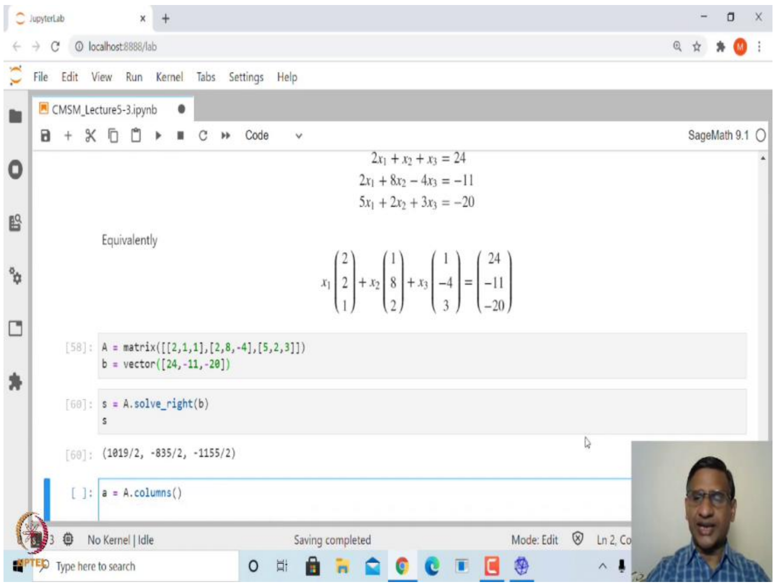
Next, let us look at just another way of visualizing, or looking at this solution of system of linear equations. So, what we have? Suppose we have this system of linear equations, 3 equations in 3 variable, this set of linear equations. You can also write as A times x1, x2, x3 is equal to B.

So, if I look at this, I can write the same equations as x1 times the first column of A, plus x2 times second column of A, plus x3 times third column of A. So, this is, this is first column of A, this is second column of A, this is third column of A. So, x1 times first column of A plus, x2 times second column of A, plus x3 times third column of A, is nothing but b.

So, what it means is that b is linear combination of columns of A, that is what exactly it means. So, in case let us say alpha 1, alpha 2, alpha 3 is a solution of the system, then alpha 1 times first column, plus alpha times second column, plus alpha 3 times third column, should be equal to b, ok?

So, that is another way of visualizing this system of linear equation. So, let us just verify this quickly. So, how do we verify this? It is quite simple. So, let us first declare A and b for the system Ax equal to b, and then what we will do? We will find solution of this.

(Refer Slide Time: 33:39)



$$2x_1 + x_2 + x_3 = 24$$
$$2x_1 + 8x_2 - 4x_3 = -11$$
$$5x_1 + 2x_2 + 3x_3 = -20$$

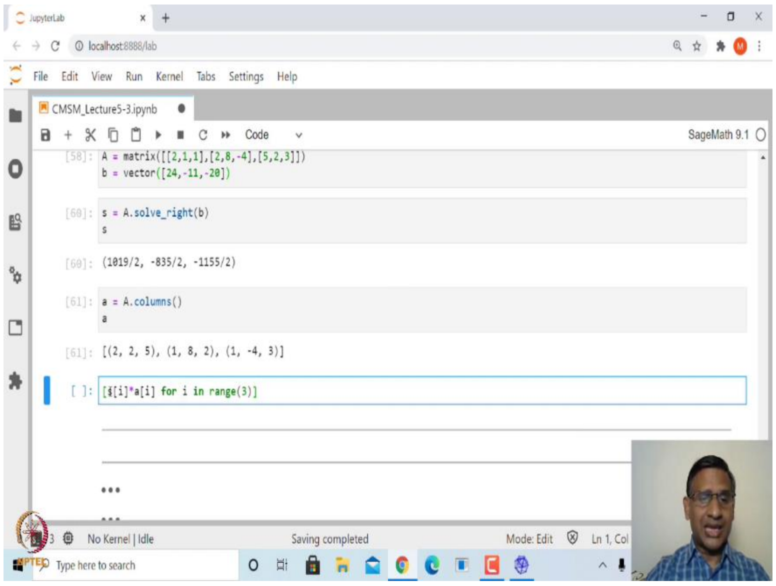
Equivalently

$$x_1\begin{pmatrix}2\\2\\1\end{pmatrix} + x_2\begin{pmatrix}1\\8\\2\end{pmatrix} + x_3\begin{pmatrix}1\\-4\\3\end{pmatrix} = \begin{pmatrix}24\\-11\\-20\end{pmatrix}$$

```
[58]: A = matrix([[2,1,1],[2,8,-4],[5,2,3]])
      b = vector([24,-11,-20])

[60]: s = A.solve_right(b)
      s

[60]: (1019/2, -835/2, -1155/2)

[ ]: a = A.columns()
```

So, let us find a solution using inbuilt function A dot solve underscore right. So, that is the solution, and once you have the solution, let us see what is the solution. In this case, s is the solution, which is a vector, and now what we will do? We will look at the first entry of s multiplied by first column of A, plus second entry of s multiplied by second column of A, and third entry of s multiplied by third column of A. That should give me b, which is 24, minus 11, minus 20. So, let us verify that. How do we do that? So, we will write a simple, first we will extract columns of A, how do I extract columns of A? So, columns of A we can extract as A dot columns, A dot columns.

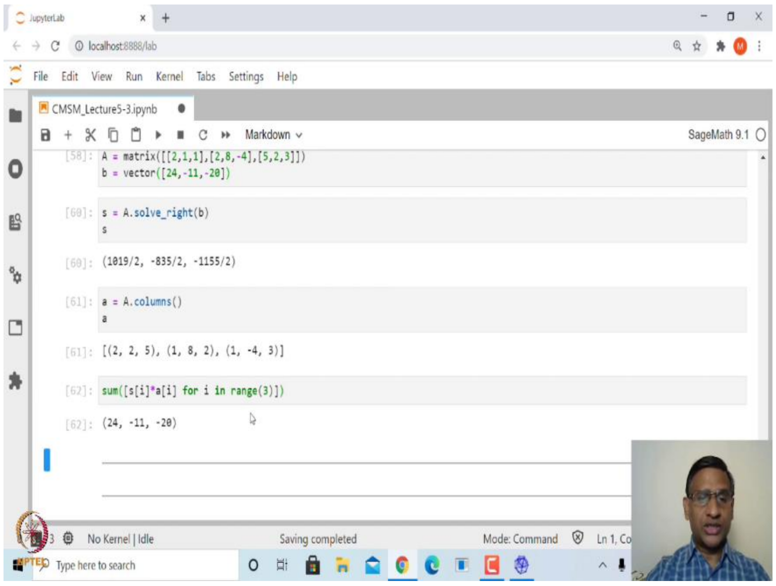
(Refer Slide Time: 34:31)



```
[58]: A = matrix([[2,1,1],[2,8,-4],[5,2,3]])
      b = vector([24,-11,-20])

[60]: s = A.solve_right(b)
      s

[60]: (1019/2, -835/2, -1155/2)

[61]: a = A.columns()
      a

[61]: [(2, 2, 5), (1, 8, 2), (1, -4, 3)]

[ ]: [s[i]*a[i] for i in range(3)]
```

And that I am putting it in small a, so these are the 3 columns of A. Now, let us look at we, we can write a simple code. So, ith entry of s multiplied by ith column of A right, and this do it for all of all these s. So, for I, for i in, in range, in this case 3, range 3.

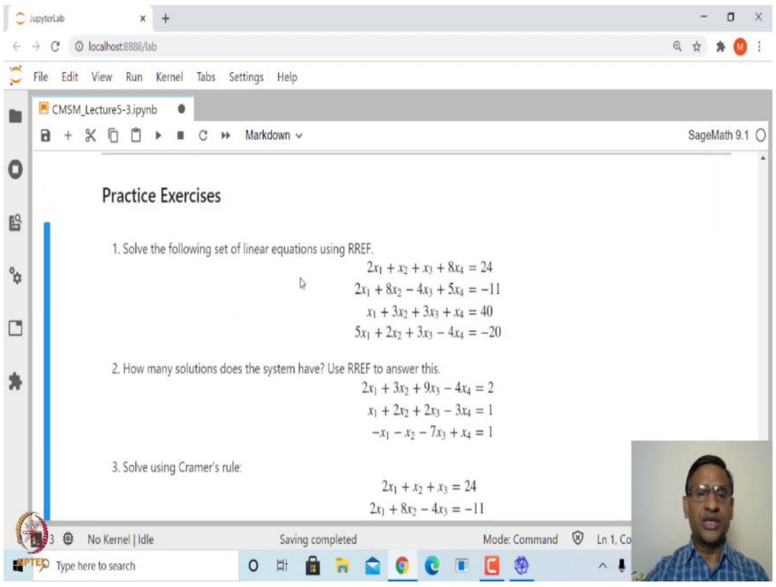
So, this will give me list of s1 times a first column, plus s 2 times second comma, s 2 times second column, plus s 3 times third column, and then sum all of these. (Refer Slide Time: 35:11)



```
[58]: A = matrix([[2,1,1],[2,8,-4],[5,2,3]])
      b = vector([24,-11,-20])

[60]: s = A.solve_right(b)
      s

[60]: (1019/2, -835/2, -1155/2)

[61]: a = A.columns()
      a

[61]: [(2, 2, 5), (1, 8, 2), (1, -4, 3)]

[62]: sum([s[i]*a[i] for i in range(3)])

[62]: (24, -11, -20)
```

So, sum, this take the sum, and this would give me b, that is correct. This is, this is the b, 24, minus 11, minus 21.

So, that is a very convenient way of actually visualizing a solution, and the many things, especially in the concepts in, in vector, in a, in a vector space will be very easy to kind of, or very convenient to look at, in this form, ok? (Refer Slide Time: 35:45)



Practice Exercises

1. Solve the following set of linear equations using RREF.

$$\begin{aligned} 2x_1 + x_2 + x_3 + 8x_4 &= 24 \\ 2x_1 + 8x_2 - 4x_3 + 5x_4 &= -11 \\ x_1 + 3x_2 + 3x_3 + x_4 &= 40 \\ 5x_1 + 2x_2 + 3x_3 - 4x_4 &= -20 \end{aligned}$$

2. How many solutions does the system have? Use RREF to answer this.

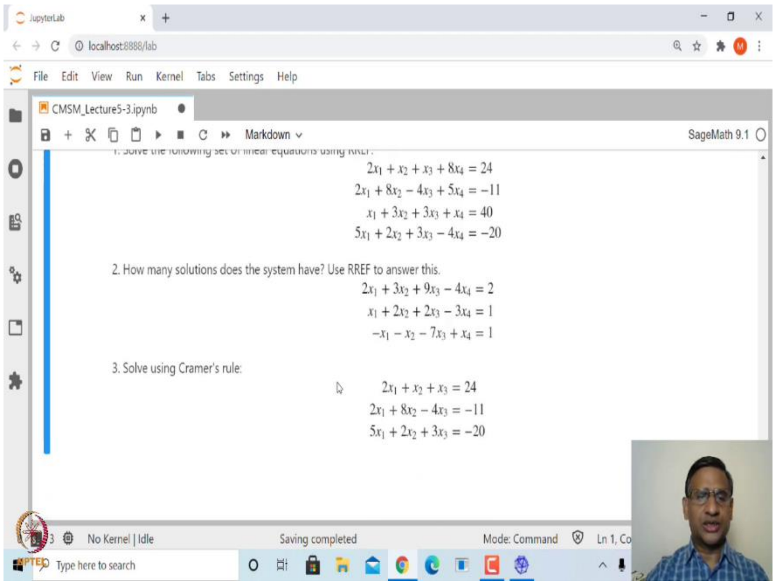
$$\begin{aligned} 2x_1 + 3x_2 + 9x_3 - 4x_4 &= 2 \\ x_1 + 2x_2 + 2x_3 - 3x_4 &= 1 \\ -x_1 - x_2 - 7x_3 + x_4 &= 1 \end{aligned}$$

3. Solve using Cramer's rule:

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 24 \\ 2x_1 + 8x_2 - 4x_3 &= -11 \end{aligned}$$

So, let me leave you with few exercises these are again simple exercises to explore this RREF.

So, look at this system of linear equations. Solve this using RREF, and look at this system of linear equations, check how many solutions does it have. (Refer Slide Time: 36:03)



1. Solve the following set of linear equations using RREF.

$$\begin{aligned} 2x_1 + x_2 + x_3 + 8x_4 &= 24 \\ 2x_1 + 8x_2 - 4x_3 + 5x_4 &= -11 \\ x_1 + 3x_2 + 3x_3 + x_4 &= 40 \\ 5x_1 + 2x_2 + 3x_3 - 4x_4 &= -20 \end{aligned}$$

2. How many solutions does the system have? Use RREF to answer this.

$$\begin{aligned} 2x_1 + 3x_2 + 9x_3 - 4x_4 &= 2 \\ x_1 + 2x_2 + 2x_3 - 3x_4 &= 1 \\ -x_1 - x_2 - 7x_3 + x_4 &= 1 \end{aligned}$$

3. Solve using Cramer's rule:

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 24 \\ 2x_1 + 8x_2 - 4x_3 &= -11 \\ 5x_1 + 2x_2 + 3x_3 &= -20 \end{aligned}$$

And the last one is, you must have also solved system of linear equations using Cramers rule. So, try to see how we can solve this system of linear equations using Cramers rule.

So, in case of this, you need to learn how to replace a particular column of A by this vector b. That is the only thing which is needed in this case, but that I am sure you should be able to figure it out, ok?

So, thank you very much we will see you in the next class.