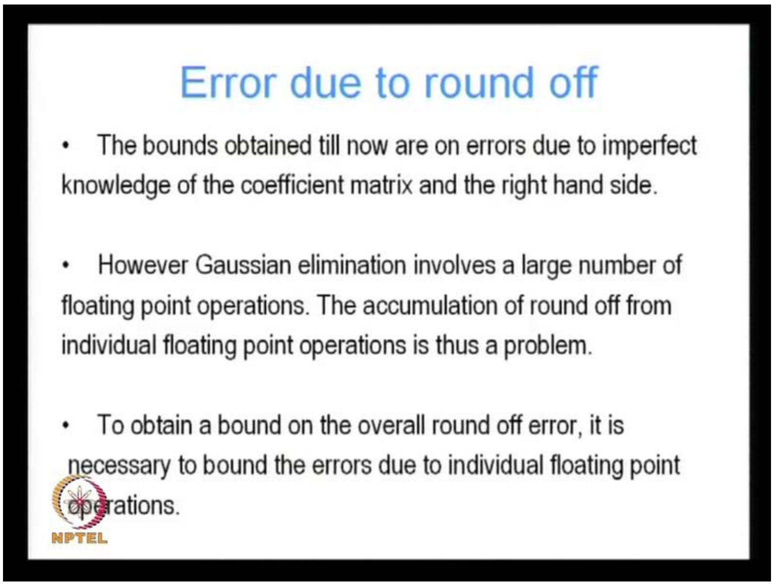
Numerical Methods in Civil Engineering
Prof. Arghya Deb
Department of Civil Engineering
Indian Institute of Technology, Kharagpur

Lecture - 7
Error Bounds and Iterative Methods for Solving Linear Systems

In the seventh lecture in our series in numerical methods in civil engineering, we are going to continue our discussion on error bounds for direct methods in addition. We are going to introduce iterative methods for solving linear systems.

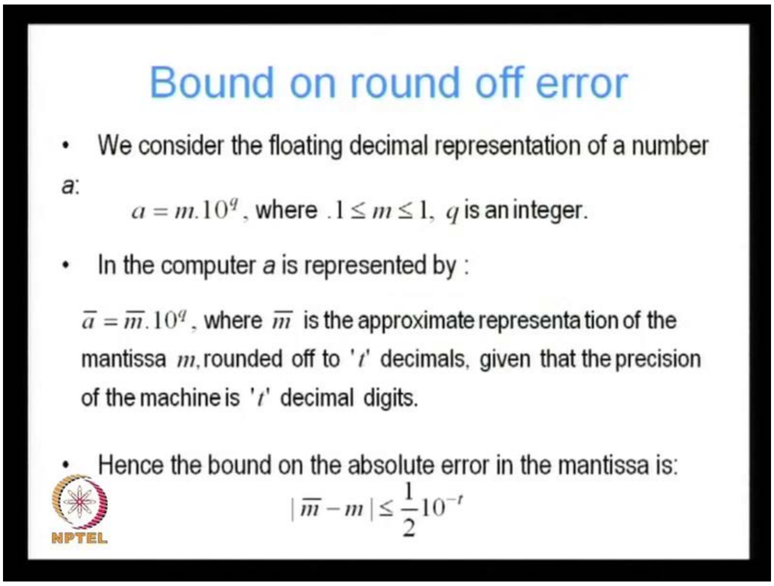
(Refer Slide Time: 00:31)



The bounds we obtained till now on the errors due to our on the errors on the solution of linear systems, were due to imperfect knowledge of the coefficient matrix, and the right hand side. Basically what we what we found was that if the coefficient matrix has an error delta a or the right hand side has an error delta b, we try to find bounds on the errors that will be introduced to the solution x due to the errors in a and b, due to the error delta a in a, and the error delta b in b that is going to result in solutions to errors to the solution x. And we try to find the bounds on those errors and what we found was that if the matrix is depending on the condition number, if the matrix is well conditioned then errors due to in the coefficient matrix, and the errors due to the right hand side will be bounded, and it is going the magnitude of the bound is going to depend on the condition number of the matrix of the coefficient matrix.

Today, we are going to talk about errors in Gaussian elimination, because of the large number of floating point operations, which take place during Gaussian elimination the accumulation of round off errors due to these floating point operations lead to overall errors in the solution, and is therefore a problem to obtain a bound on the overall round off error, it is necessary to bound the errors due to individual floating point operations. Since the Gaussian elimination process is a combination of a large number of floating point operations, we have to find bounds on the error of individual floating point operations in order to find a bound on the error of overall error, due to in the solution due to round off during Gaussian elimination.

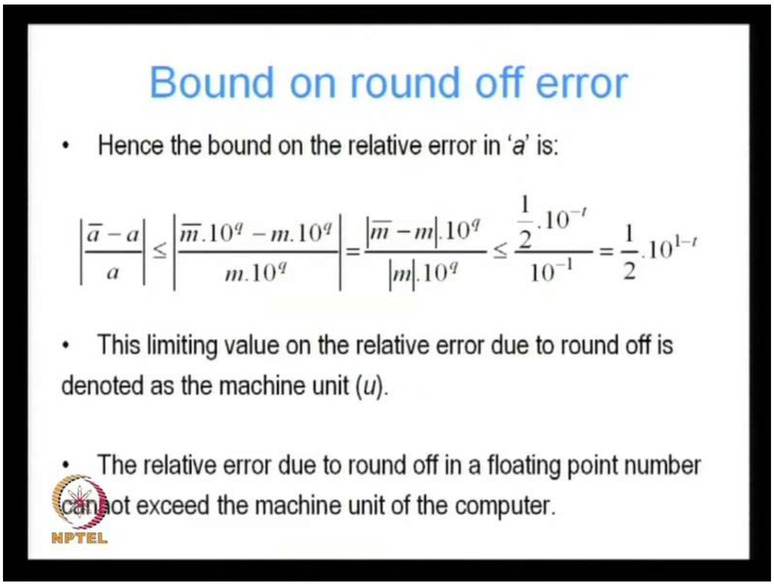
(Refer Slide Time: 02:45)



We consider the floating point decimal representation of a number a suppose we have a floating point number a how does the computer represent that number internally, well the computer almost always represents the number using this format. It represents it as some number m and times 10 depending on the base of the computer, if 10 is the base then that will be 10 there if instead of the instead of 10, it is some other number some other base then that is going to be the appropriate base. So, 10 to the power q where m lies between 0.1 and ones m m is a number between 0.1 and 1 and q is an integer. So, m is known as the mantissa and q is known as the exponent. So, this is how the computer represents any floating point number a, but in reality the computer cannot represent m up to infinite precision.

Because the computer has finite precision for instance if the computer rounds off floating point numbers to t decimals, then a will not actually we the computer will not actually be storing a, it will be storing an approximation to a which is denoted by a bar and the approximation arises; because m is only the computer only stores an approximate representation of the mantissa m m bar, which is rounded off to t decimals given that the precision of the machine is t decimal digits. So, the So, m bar is the approximation of m which is stored in the computer. Hence the bound on the absolute error in the mantissa m is given by m bar minus m, which is less than or equal to half 10 to the power minus t this we have obtained in a previous lecture. So, the error is bounded the round of error is bounded by half into 10 to the power minus t.

(Refer Slide Time: 04:59)

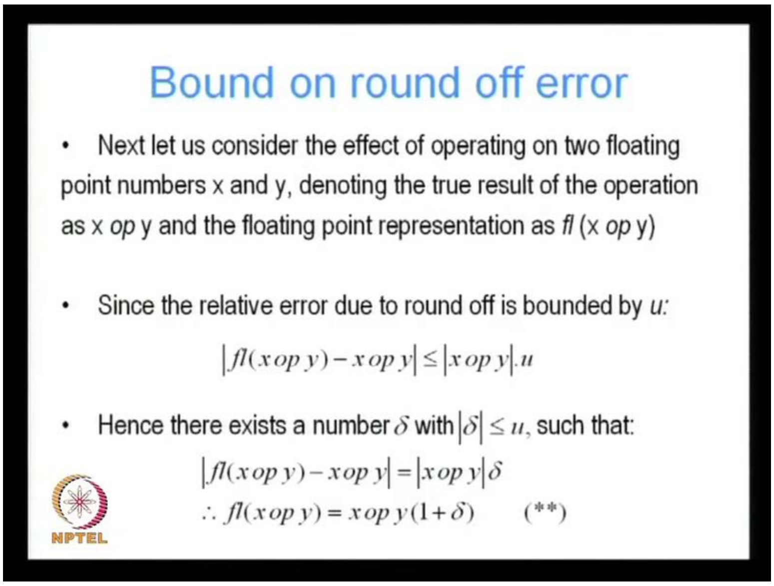


That was the bound on the absolute error in the mantissa, if we look at the relative error in a that is given by a bar minus a by a which is by definition the relative error. So, this has got to be less than or equal to m bar times 10 to the power minus q minus m dotted with 10 to the power q m dotted with 10 to the power q. That is simply because m bar minus m is less than or equal to 10 to half 10 to the power minus t sorry. So, we can write that as this is the m bar minus m dotted with 10 to the power q 10 to the power q divided by m bar 10 to the power q is lesser than or equal to half 10 to the power minus t 10 to the power 10 to the power q 10 to the power q cancels out, this is we know is lesser than or equal to half 10 to the power minus t, and since mod of m if we go back to the previous slide and see m is lesser than or equal to point one and greater than or equal to point one less than or equal to one.

So, if we if we replace mod of m by point one this has always got to be greater than, because we are taking the smallest possible value of m in the denominator. So, this thing has to be less greater than or equal to this thing this thing has to be lesser than or equal to this thing, and this gives me a bound half in to 10 to the power one minus t. A correction this is this should not be lesser than or equal to this is actually equal to right. So, mod of a bar minus a by a is equal to m bar dotted with 10 to the power q minus m 10 to the power q divided by this which is equal to this, but this is lesser than or equal to this because on in the denominator, we have replace mod of m by its smallest possible value which is 10 to the power minus 1. So, we get this bound on the relative error in a.

This limiting value on the relative error due to round off is denoted as the machine unit as you can see this is totally dependent on the machine precision t. So, the relative error due to round off in a floating point number cannot exceed the machine unit of the computer denoted as this whole thing denoted as u.

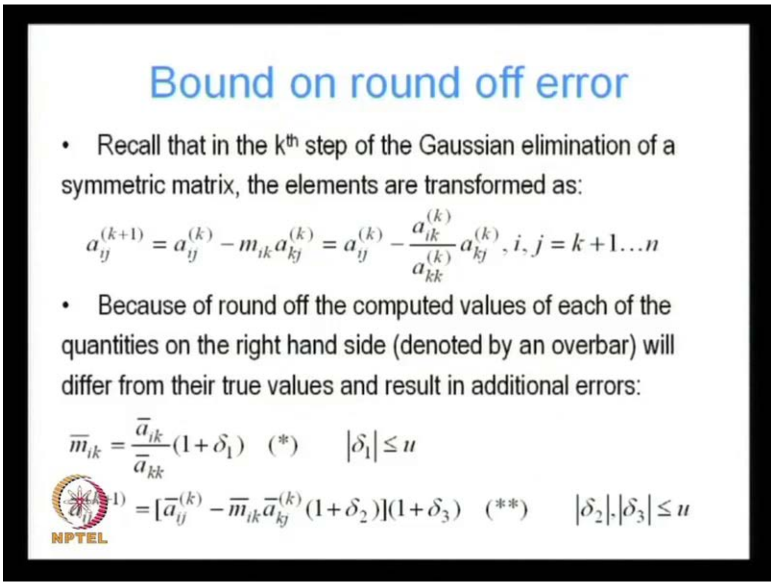
(Refer Slide Time: 07:51)



Next let us consider the effect of operating on two floating point numbers x and y, and let as denote the true result of the operation as x op y, and the floating point representation as float of x op y. So, x op y is the true solution and this is how the computer is going to represent this.

So, the relative error due to since the relative error due to round off is bounded by u, we can say that f l x op y minus x op y is lesser than or equal to mod of x op y times u; which is exactly what we get if we replace a bar here a bar here by f l x op y, and a which is the true solution by x op y right. So, this is what we get this is the bound on the relative error now; because this is less than x op y times u we can be sure that, there is a there is number delta with mod of delta lesser than or equal to u such that this becomes equal this is lesser than or equal to mod of x op y times u. So, there must exists a number for which this becomes equal to that right, and this gives this gives rise to this expression floating point of x op y is equal to x op y one plus delta. So, this is how the computer represents x op y, this is the true solution and the true solution times this error one plus delta is the floating point representation of the result of the operation of x op y.

(Refer Slide Time: 09:45)

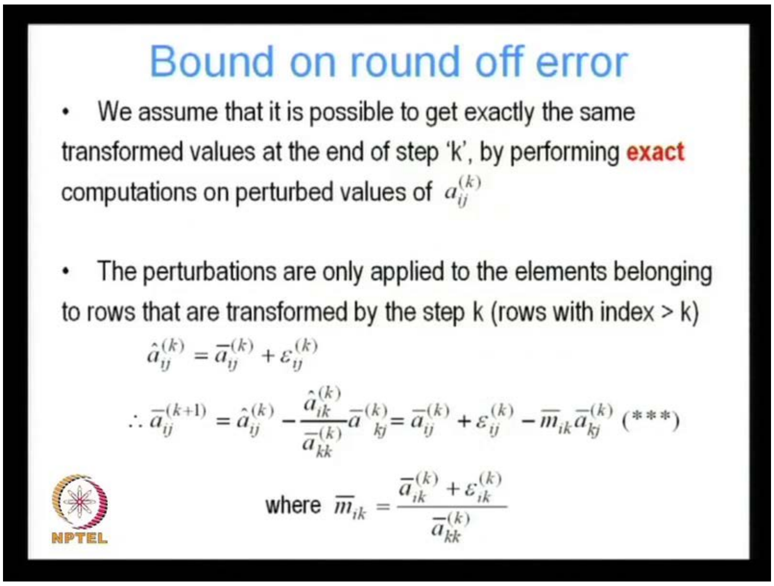


Next let us go back to the Gaussian elimination, and let us recall that in the k-th step of the Gaussian elimination of symmetric matrix the elements are transformed as the following a i k k plus one is equal to a i j k minus m i k, where m i k is the multiplier times a k j k at the k-th iterations k-th step; which is equal to a i j k minus a i k k by a k k times a k j k, and you can see that we have summing we have j j goes from k plus one to n, where we have taken advantage of the fact that the matrices symmetric, because of round off the computed values of each of the quantities on the right hand side will differ from this true values and result in additional errors. So, instead of a i j k what is actually stored in the computer is a bar i j k right, where a bar i j k includes the floating point errors right it includes the round off errors. So, it is the floating point approximation right.

So, a bar i j k plus one is equal to a bar i j k minus m bar i k a bar k j k. So, here when we when we compute m bar i k from a bar i k divided by a bar k k. Since this is the floating point operation, we introduce certain errors certain round off errors right, and this round off error is denoted by one plus delta one. So, this is the error in the computation of m bar i k then we subtract m bar i k a bar k j from a bar i j k. So, this operation the subtraction operation introduces additional floating point error, which is denoted by one plus delta two and then on top of let me let me take a step back.

So, this subtraction operation introduces additional floating point error which is given by 1 plus delta 3, and this operation m bar i k a bar k j k introduces floating point of error which is given by m bar i k a bar k j k one plus delta two. So, this operation introduces of floating point error which is given by one plus delta two this subtraction operation introduces a floating point error, which is given by 1 plus delta 3 and this division operation introduces a floating point error which is given by 1 plus delta 1; and we are guaranteed that each of these errors it must be less than or equal to the machine precision.

(Refer Slide Time: 12:53)



Now, we say that if we do these operations, then we are going to get these floating point errors right. So, instead of that we say that let us see if we can get exactly the same transform values of the end of step k; by performing the exact computations on perturbed values of a i j. So, the idea is like this.

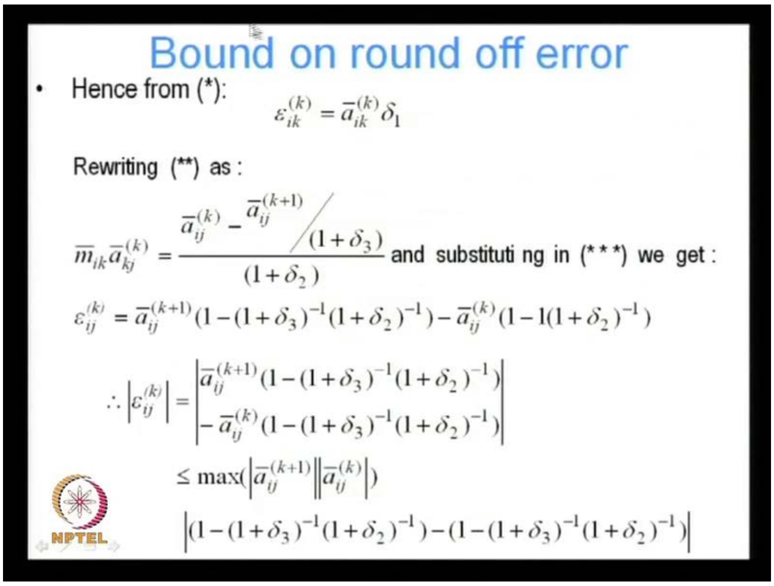
So, we operate on the values that a stored in the computer and we operate we go through this operation this operation, and we end up with additional errors due to the which are governed by this delta 1 delta 2 delta 3. So, now, we are saying we are saying that instead of working with the instead of assuming that there are instead of assuming that there are operating on two float no sorry. I am going back I have instead of getting instead of performing on the exact values a i j k, what I am saying is that I am going to operate on the some perturbed values on the perturbed values; and I am going to going to go through the exact same transformation equations, but I will assume that I am not introducing any additional floating point errors during the transformations right.

So, I am going to operate on some perturbed values instead of operating on the exact a i j k a i j superscript k. I am going to operate on certain perturbed values, but I will assume that after by I will get the same results by performing exact operations on the perturbed values. So, the perturbations; however, only applied to the elements belonging to the rows that are transformed by the step k that is rows with index greater than k. So, basically I am saying that instead of operating on a i j k. I will operate on a bar i j k plus epsilon i j k. So, now I am get I am going to get a bar i j k plus 1 is equal to a hat i j k minus a hat i k k divided by a bar k k k times a bar k a k j a bar k these I have already be obtained. So, they have superscript k right. So, they have already been obtained from the previous step in the Gaussian elimination right.

So, these values are known now what I am saying is that the values that I am going to transform right, I am going to I am not going to operate on those values themselves. I am going to operate on those values plus some perturbed values, and I am going to assume that my floating point operations. I am not going to introduce any errors. So, I am going to get the exact solution by operating on these perturbed values perturbed values and I am going to get the exact same solution. I hope that is cleared but basically the idea is that instead of looking at the effect of the floating point operations, we say that we are trying to get this same solution by considering perturbations in my original matrix components.

So, I get some value after my operations right those operations, typically include floating point operations operating on the original numbers original numbers that was there on the matrix. So I say that instead of operating on the original numbers on the matrix, I will operate on some original numbers plus perturbations on some perturb numbers, but during the operations I will not introduce any floating point operation any floating point errors, and I want to get the same values as a result of this operation. So, the idea is that instead of we are transferring the problem to the perturbations. So, we want to find what perturbations in my original system will give me the same errors same floating point errors, as I would get during round off right. So, so what changes should I make to the my original matrix elements, in order to get the same error which I would have got due to round off.

(Refer Slide Time: 18:15)

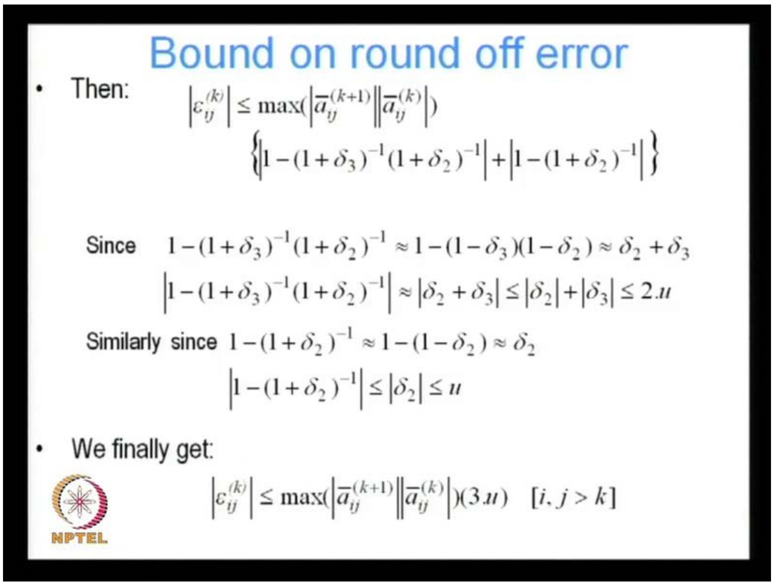


From this expression m bar i k is equal to a bar i k by a bar k k 1 plus delta 1. We therefore, get epsilon i k k is equal to a bar i k k times delta 1 how do we get this well we compare this expression this expression with sorry, this expression this expression with this expression right, when we compare this expression with this expression we get epsilon i k k is equal to a bar i k k times delta 1. Then we can rewrite this expression this expression as this is just a question of substitution right we are going to substitute those values here and we get this and finally, substituting all this in this expression we are going to get finally.

This expression which you can see gives me an expression for the perturbation epsilon i j k, which gives me an expression for the perturbation epsilon i j k. So, that is the basic purpose of this exercise the basic purpose of this exercise is to try to find bounds on the perturbation epsilon i j k, because we know that the perturbations are equivalent the end result of the perturbations is going to be the same errors, which would have a proved if I had the floating point errors right. So, instead of finding try to find bounds on the floating point errors themselves, I am going to try to find bounds on the perturbations. So, this is just I am transferring the problem transferring the problem of finding the bounds on the perturbations, because I am saying the end result of the perturbations is equivalent to the floating point errors in the operations.

So, we get this and then if we take bounds on both sides we get an expression like this and this must be lesser than or equal to. So, this is a bar i j k plus one times this term minus a bar i j k times this term. So, this has to be lesser than or equal to maximum of this and this times this term right, because this is maximum this and this must be larger than this minus this times this minus this times this must be lesser than the maximum of this and this times this minus this right. So, this is lesser than that.

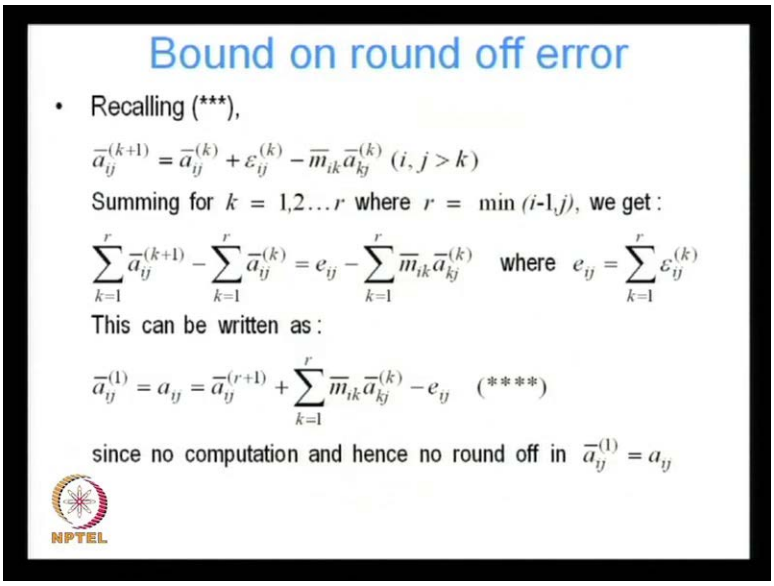
(Refer Slide Time: 21:08)



So, we get the same expression I have written here. So, mod of epsilon i j is lesser than or equal to max of this times this and then we know that since delta 1 delta 2 delta 3 are less than the machine precision u. So, these must be small numbers. So, I can do a binomial expansion of this and if I do that I can write 1 plus delta 3 minus 1 as 1 minus delta 3 one plus delta 2 minus 1 as 1 minus delta 2 and this is approximately equal to delta 2 plus delta 3. Where I have ignored terms which involved 2 delta is delta 2 times delta 3. Similarly. So, if I take bounds on that I get this is approximately equal to mod of delta 2 plus delta 3 which is lesser than or equal to mod of delta 2 plus mod of delta 3. And since both delta 2 and delta 3 are lesser than u this must be less than two times the machine precision.

Similarly, the second term 1 plus 1 plus delta 2 inverse I can write it as 1 minus 1 minus delta 2, again using binomial expansion taking into account the fact that delta two has is very small much smaller than 1, which is going to give me approximately delta 2. So, again I take bounds on that this going to be lesser than or equal to delta 2 and again delta 2 is lesser than the machine precision. So, that is going to be less than u.So, we finally, get mod of epsilon i j to the power not to the power mod of epsilon i j at the k-th step is lesser than or equal 2 max of this times 3 times u where u is the machine precision.

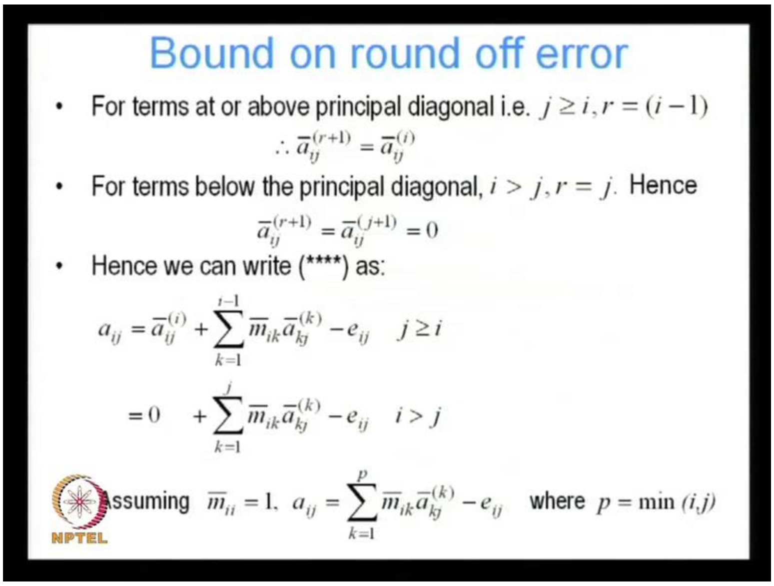
(Refer Slide Time: 23:12)



Let us recall this equation this equation, which was my update formula for Gaussian elimination and which says that a bar i j k plus 1 is equal to a bar i j k plus epsilon i j k minus this if we sum this expression for k is equal to 1 to r, where r is the minimum of i minus 1 j i being the row index j being the column index. We are going to get sigma k equal to one to r a bar i j k plus 1 minus sigma k equal to 1 1 to r a bar i j k is equal to e i j, where e i j is basically I have summing this term from k equal to one to r e i j minus sigma k equal to 1 to r m bar i k a bar k j k.

This everything is going to cancel except the r for except the term which is going to be for k equal to r, which is going to give me a bar i j r plus 1 and the term which involves one a bar i j 1. So, I have a bar i j r plus 1 minus a bar i j 1 the rest of the terms are going to cancel, the rest of the terms from this first term is going to cancel the rest of the terms from this second term. So, we are left with a bar i j r plus one minus a bar i j one, but a bar i j 1 is going to be a y j; because that is the first that this that is the first the first step and the first step there are no round off errors. So, a bar i j one is going to be equal to a i j. So, we can get an expression like a i j is equal to a bar i j r plus one bringing changing the sides right bringing this to the left hand side. So, we get a bar i j r plus one plus sigma k equal to one to r m bar i k a bar k j k minus e i j.

(Refer Slide Time: 25:35)

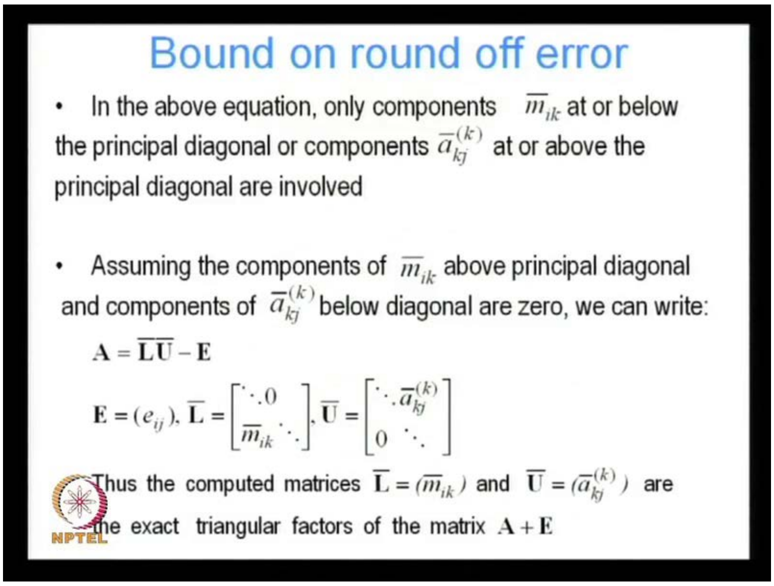


For terms at or above the principal diagonal j is greater than or equal to I and since r is equal to minimum of i minus 1 and j if j is greater than i. So, r must be equal to i minus 1, because r is minimum of i minus one in j so, r is equal to i minus 1. So, in that case a bar i j r plus 1 is going to be a bar i j I, because r is equal to i minus 1. So, this gives me a bar i j r plus 1 is equal to a bar i j i for terms below the principal diagonal the row index is going to be greater than the column index i is going to be greater than j therefore, minimum of i minus 1 j is going to be j. So, r is going to be j hence in that case a bar i j r plus 1 is equal to a bar i j j plus 1, which we know from our Gaussian elimination is going to be 0. So, beyond the j-th step the a bar i j is going to be 0 the terms which are below the principal diagonal are going to be 0.

Hence we can write this previous expression this previous expression we can write it as a i j which is equal to a i j one is equal to a bar i j i plus this term, which does not change this is true for j greater than or equal to i and this is equal to zero plus this term when i is greater than j. So, basically I have split it up if split this equation in to two parts one for j greater than or equal to i and 1 for i greater than j and if we assume that m bar i i is equal to 1, I can put this term inside the summation and change this index from k equal to one to i minus 1 to k equal to 1 to i.

So, in that case we can write a i j is equal to sigma k equal to one to i m bar i k a bar k j k minus e i j provided m bar i i is equal to 1 that is true, then we can write combine these two equations together to write a i j is equal to sigma k equal to 1 to p m bar i k a bar k j k minus a i j, where p is equal to minimum of i and j if j is greater than i then p is going to be i when i is greater than j then p is going to be j. So, when p when j is greater than i, I am going to recover this first equation, when i is greater than j. I am going to recover the second equation. So, i finally get a i j is equal to sigma k equal to one to p m bar i k a bar k j k minus e i j.

(Refer Slide Time: 28:45)

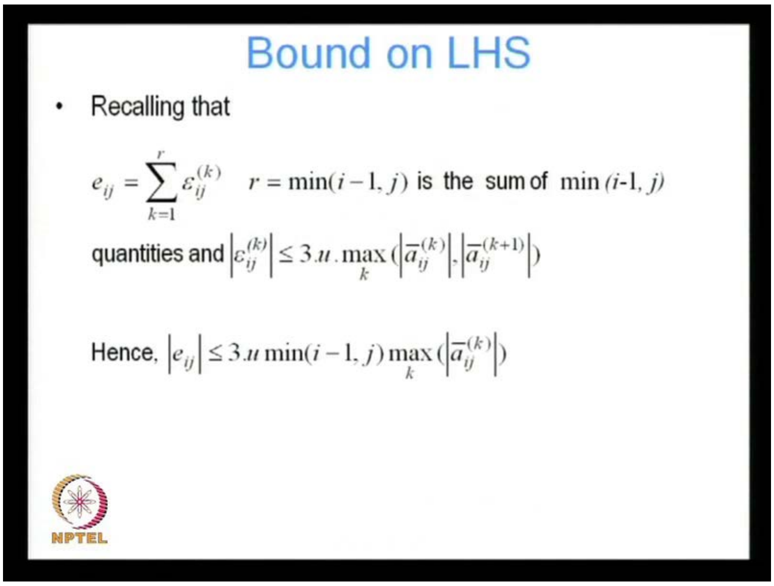


So, in the above equation we can see that only components m bar i k are needed when they are m bar i k terms, we only involve the m bar i k terms which are below the principal diagonal and a bar k j terms, which are above the principal diagonal we can see because k is equal to 1 to p and p is equal to minimum of i j, here p k is the second index and k is and k k can be as high as only as p and p is bounded by this. So, this term only involves a term, which are at or below the principal diagonal while here this term. Since the first index is k and k can be only as high as p and p is bounded by minimum of i j. So, this term only involves terms which are above the principal diagonal. So, in that because of that this equation only involves components of m bar i k at or below the principal diagonal and components of a bar k j k at or above the principal diagonal.

Since we do not use components of m bar which are which are above the principal diagonal and components of a bar, which are below the principal diagonal we can assume them to be 0 in which case this expression this expression is a L U decomposition this is the lower triangular matrix that is an upper triangular matrix. So, in that case we are going to get, a is equal to L bar U bar minus E where E is the matrix is components. I given by E i j let us go back to the previous slide this components i given by E i j and i bar, what we just discussed from is made of m is components are m bar i k and u bar has components a bar k j k.

Thus the computed matrices L bar and U bar are the exact triangular factors of the matrix a plus e. So, LU we thought was the triangular decomposition of a right, but because of round off errors, we are going to get L bar U bar and L bar U bar is the exact decomposition of a right a plus E it is the exact decomposition of a plus E. So, now we are transfer the error into a perturbation in a right. So, a the original matrix a plus some perturbation matrix E is going to give me L bar U bar L bar U bar includes the effects of all the floating point errors.

(Refer Slide Time: 32:07)

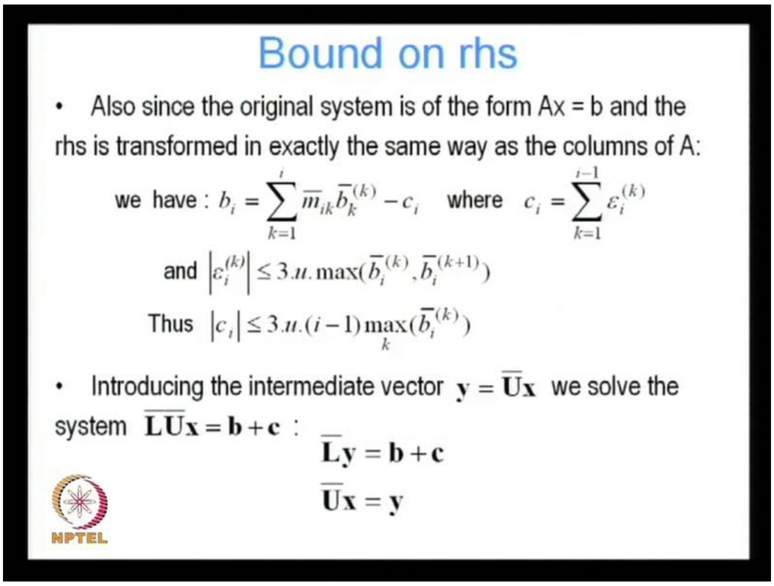


So, recall that we have defined e i j is the sum of all the epsilon i j k is that is when what we defined exactly. Here when we defined E i j equal to sigma epsilon i j k and r is equal to minimum of i minus one j. So, it we can write this as the sum of minimum i minus 1 j quantities and we also remember that epsilon i j k is has this bound which we obtained earlier. So, we can say that mod of e i j is lesser than or equal to three u times minimum i minus 1 j times maximum of maximum of mod of a i j k over k basically, we are saying that since this is bounded by that right and this is form by the sum of these epsilon i j k s; then we get a bound on e i j which is three u times are times are minimum of i minus 1 time j, because this sum is over r and r is minimum of i minus 1 times minimum of I minus 1 and j. So, we are taking this value which is the maximum value of r right r r is bounded r is given by this.

So, this value times the maximum of all these sums right this is the maximum of a i j k a i j a bar i j k plus one and that is maximum of a bar i j k over all the k s. So, we are taking the maximum possible. So, we have a sum of sum of epsilon i j k which involve terms like this right maximum of this and this. So, we are taking we are looking at all the sums and we are saying that we are going to take of all the terms which comprise the sums we are going to take the maximum value that this maximum of all those values. So, that maximum value times i minus one is going to be an upper bound on mod of e i j.

So, that. So, that is just the absolute upper bound because we this term comprises a number of sums. So, we are taking the largest term in that sum we are taking the largest term in that sum and multiplying it multiplying that by the number of terms in the sum. So, that is going to be greater than or equal to mod of e i j. So, again let me repeat we this term comprise is a sum of a number of terms right we are taking the largest term in that sum and multiplying it where the number of terms in that sum, and we are saying that has to be a bound on the left hand side which is the sum of all those terms.

(Refer Slide Time: 35:25)

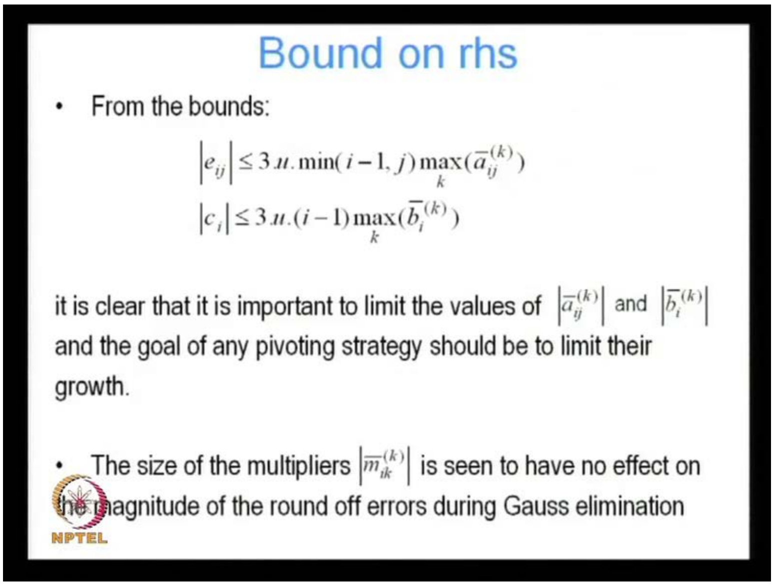


So, that was the bound on the left hand side let us consider, a bound on the right hand side since the original system is of the form a x is equal to b, and the right hand side is transformed in a it remember we recall from Gaussian elimination the right hand side is transformed in exactly the same way as the left hand side right. So, the right hand side because the right hand side is transformed in exactly the same way as the columns of a, we can develop a bound on the error in the right hand side, because we are performing operations on the right hand side every time at every step, we are operating on the left hand side as well as in the right hand side.

So, when we when we operate on the right hand side we introduce floating point errors on the right hand side, and these are the bounds on the similar to the errors we get on the bounds on the errors, we get on the bounds we get on the errors in the left hand side we can similarly get bounds on error on the right hand side in a very similar fashion, and they have a similar form. So, mod of c i is the it is sum of the errors on the right hand side and that is bounded by something like this which is very similar to the bound on the error in the right hand side.

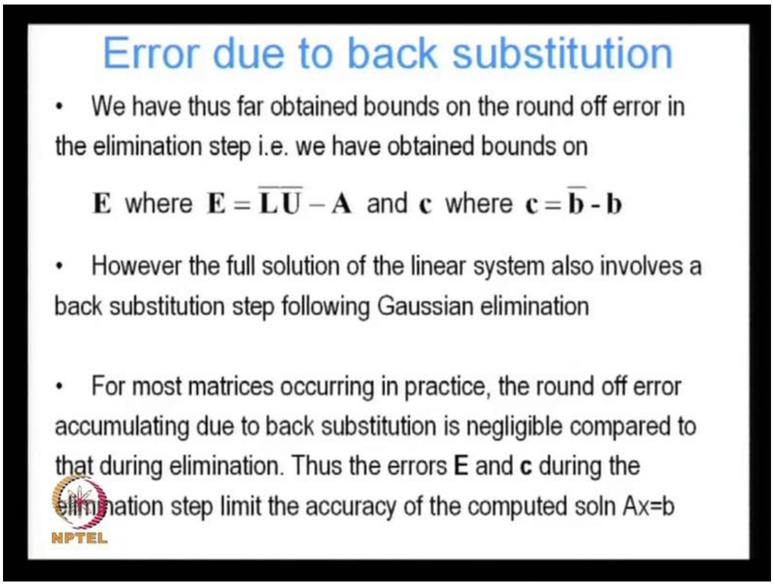
So, introducing the intermediate vector y is equal to U bar x we solve the system L bar U bar x is equal to b plus c right now l bar u bar include the floating point errors right plus b plus c, why do we have this c because c is because of the round off errors in b right. So, this is the actual system that we are solving and introducing an intermediate vector y is equal to u bar x we get L bar y is equal to b plus, and then we solve for y here we solve for y from this equation put y on the right hand side and finally, we solve for x U bar x is equal to y.

(Refer Slide Time: 37:40)



So, summarizing we have obtained bounds on e i j and on the right hand side c i and these bounds have the following form. Now, it is clear that if these errors are going to be small then these bounds this right these bounding values must be small two right. So, in order to bound these errors these terms a bar i j k and b bar i k must be small. So, it is very important that the pivoting strategy whatever pivoting strategy we adopt should limit the values of mod of a i j k and b bar a bar i j k and b bar i k, but what is interesting to note is that these bounds do not involve the multipliers m bar i k, thus the magnitude of the multipliers has no effect on the magnitude of the round off errors during Gaussian elimination; because these terms these terms do not involve the multipliers they only involved coefficients, and the right hand side the approximations to coefficients on the right hand side it do not involve the multipliers.

(Refer Slide Time: 39:04)



We have thus far obtained bounds on the round off error in the elimination step that is we have obtained bounds on e where e is equal to L bar U bar minus a. So, L bar U bar is the appropriation to L U L U is the exact decomposition of A. So, L bar U bar minus a is equal to e write that e matrix and we have obtained Boundson the terms of the e matrix each term of the e matrix is E i j. We obtained Boundson the error matrix in the in the in the coefficient and we have also obtained errors on c bounds on c where c is the error in the right hand side.

So, that. So, we have obtained bounds both on the coefficient matrix as well as on the right hand side errors due to round off, we know now know that those error due to round off cannot exceed these bounds right. So, those errors due to round off cannot exceed these bounds we have found bounds on those errors. However the full solution of the linear system also involves the back substation following Gaussian elimination. So, this is the bound on the errors due to Gaussian elimination, there will be additional errors due to back substitution which follows Gaussian elimination.

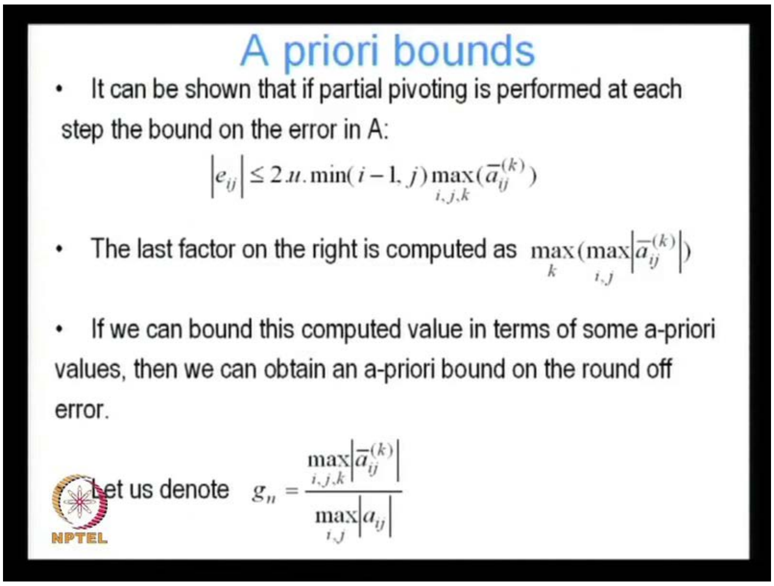
However, for most matrices that occur practice the round off error due to Gaussian due to back substitution is negligible compared to the round off error, due to Gaussian elimination thus the errors E and c during the elimination step limit the accuracy of the computed solution A x equal to b. So, the most of the errors round off errors occur during the elimination step and if we obtained bounds on those errors whatever additional errors have proved during back substitution.

They are going to be negligibly small compared to they are still going to be bounded by bound which we have obtained during for Gaussian elimination. So, let us look again at those bounds mod of a i j lesser than or equal to three times machine precision U times minimum 5 minus 1 j then maximum of a bar i j k, overall the iterations k similarly mod of c i less than or equal to three times machine precision u times I minus 1 times maximum of b bar i k over all the iterations k these are basically. What are known as posteriori bounds why because we cannot predict those bound before, we have actually computed these values right because these values are the maximum over all the all the steps in the Gaussian elimination.

So, unless we have actually compute unless we have actually gone through the steps of the Gaussian elimination we do not know what these max values are we do not know what max k a bar I j k is or max k b bar i k k is. So, these are posteriori bounds posteriori bounds and these have to be these can only be obtained after we have perform the Gaussian elimination. However it is desirable to obtained a priori bounds that is we want to know what my error will be I want to know what my bounds in my what the bounds in my error will be before I have actually started doing the computations right. So, a priori bounds are always more useful than posteriori bounds.

So, let us see how we can n obtained a priori bounds on the error in Gaussian elimination it has been shown that a priori bounds can only be obtained if mod of m bar I won not go into the derivation for that, but let us take it for granted it can be there are references and the references I have referred to in this course you can find discussions on this. So, it can be shown that if mod of m bar i k k is always lesser than or equal to 1 then we can obtained a priori bounds. But when is mod of m bar i k k will going to less than or equal to one when are those multipliers going to be less than or equal to one they are going to be less than or equal to one. Only when we have performed pivoting when at least we have performed partial pivoting right at every step, we choose the largest element in a column as the pivot only then can be assured that the multipliers, that we are going to use are always going to be lesser than or equal to one and in that case we can obtained a priori bounds.

(Refer Slide Time: 44:17)

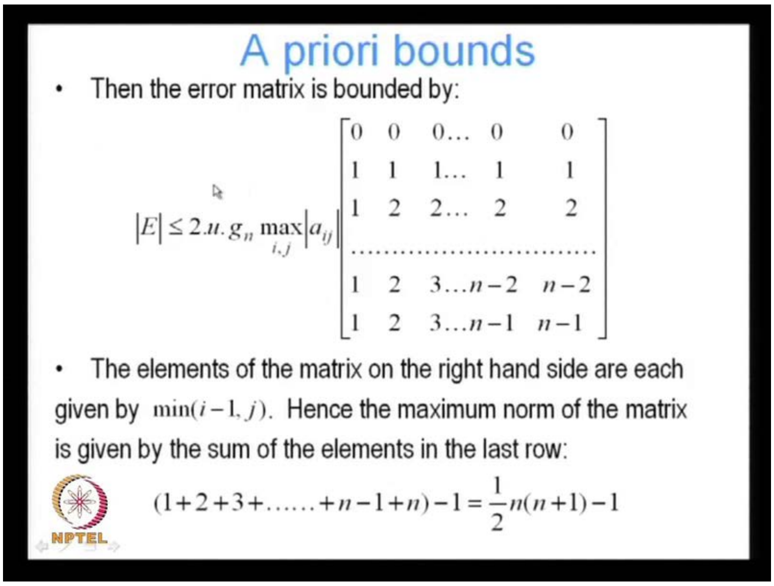


So, in the in case we perform partial pivoting then we can show that mod of e i j now be it changes. So, it the bound changes the bound becomes narrow instead of this lesser than or equal three u times this, now we are going to get this lesser than or equal two u times minimum of i minus one j, but most interesting is this part. Let's look at what this was this was maximum of a bar i j k over all the steps k ah right. So, now, instead of that I have maximum of a bar i j k over i j k. So, not only is the maximum over the steps k it is over all the elements of the matrix right it is over all the elements of the matrix i j k.

So, now I am saying that mod of e i j is lesser than or equal to the largest value in the coefficient matrix not only over the steps, but over each element not it is not only concerned with each element it is it is the maximum of over all the elements in that matrix it is a maximum of i j as well as k. So, it is not only a maximum of over the steps, it is also a maximum over the row and column indices. So, basically it is the maximum absolute maximum element in that matrix over all the steps. It is the maximum element in that matrix over all the steps. So, the last factor on the right is computed as maximum of i j mod of a bar i j k. So, at each step k we find the largest element in the matrix and then we find the largest over all the steps, so maximum of k over i and j. So, if we can bound this computed value in terms of some a-priori values then we can obtain an a-priori bound on the round off error.

So, if we can we can bound this in terms of some a-priori values in terms of some values which are known which are known before I do my Gaussian elimination typically some values, which are part of my original matrix a right using some components of my original matrix a if I can obtain a bound on this I am going to get in a-priori bound on my error due to Gaussian elimination to do, that is to do that let us denote g n is equal to maximum of i j k mod of a bar i j k divided of divided by maximum of i j mod of a i j.

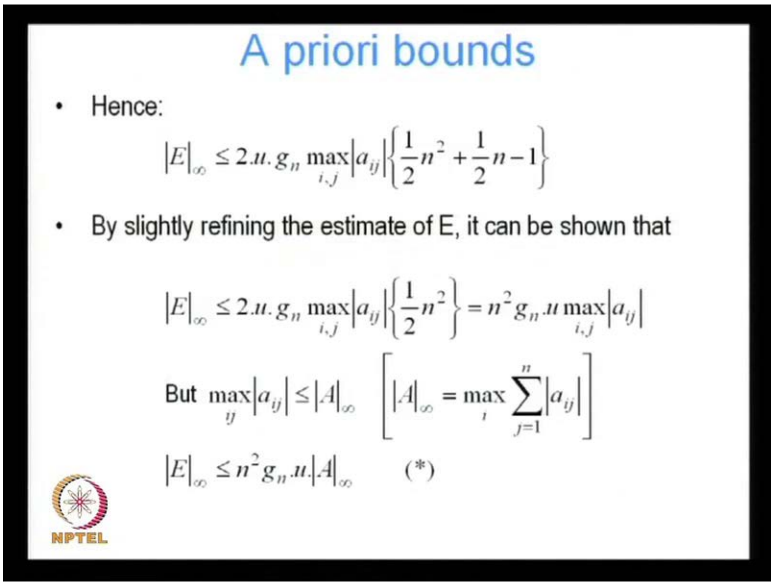
(Refer Slide Time: 47:22)



If we do that we can write this expression we can re write this expression as mod of e mod of e lesser than or equal two u two u remains the same maximum of a bar i j k i j k. I am going to replace this by g n times maximum of i j mod of a i j. So, that is going to be g n times maximum of i j i j and what is this matrix this matrix is nothing, but minimum of i minus 1 j. So, at for each element I have computed I have obtained this element by taking minimum of the row index minus one and the column index. So, the row index here is one the column index is one. So, I have one minus one is 0 0 and 1 minimum of 0 and 1. So, that is going to be 0. Similarly, I have computed all the rest of the terms in this matrix right which is nothing, but minimum of i minus 1 and j elements of the matrix on the right hand side or each given by minimum of i minus 1 j.

Hence the matrix norm of the maximum norm of the matrix is given by the sum of the elements in the last row recall what is the maximum norm it is the sum of the sum of the elements in each row maximum of that right. So, I have taken. So, that. So, that the my last row is going to contribute that and that is going to be 1 plus 2 plus 3 plus n minus 1 plus n minus 1. I have group these terms together you can see these are the sum of the first n natural numbers which is given by half n n plus one. So, this is the half n n plus one minus one. So, we get mod of E infinity why is why have we taken e infinity because we have computed the infinite norm of this matrix right.

(Refer Slide Time: 49:33)

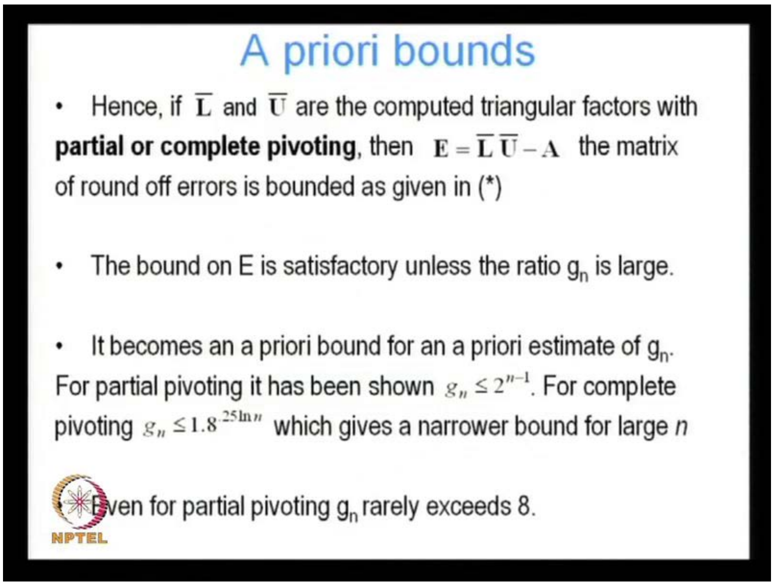


So, mod of e infinity is lesser than or equal to 2 u g n times max i j over mod of i j this term remains the same and then the infinite norm of this matrix infinite norm of this matrix, which we just calculated to be half of n square plus half of n minus 1 by slightly refining the estimate of E, which I am not going to go into again it is slight change actually we can show that mod of e infinity is lesser than or equal to 2 u g n max i j mod of i j times half of n square.

So, it is actually here we have obtained half of n square plus half of n minus 1 actually it is less than half of n square this is a sharper bound right it can be shown that we can write it like this and then 2 2 cancels out we get n square g n u maximum of i j mod of i a i j, but maximum of a mod of a i j i j is lesser than or equal the infinite norm of my original coefficient matrix why because the infinite norm is defined as sum of over all the columns sum of sum of each row sigma j equal to one to n mod of a i j and then the maximum of that right. So, maximum of i j mod of a i j is going to be bounded by the infinite norm of a right.

So, I can replace this replace this on the right hand side by the infinite norm of a because this is greater than that. So, I finally, get this mod of e infinity is lesser than or equal to n square g n u times a infinite norm of a, but we still have this factor g n right what is this factor g n.

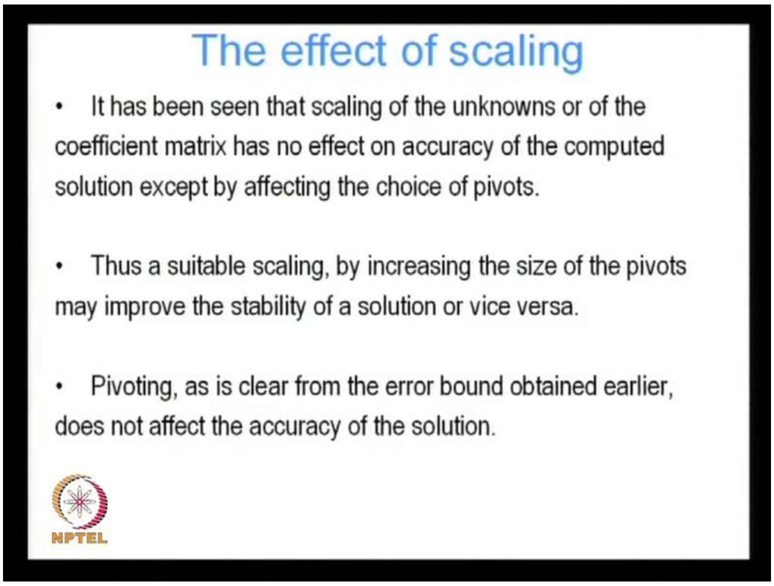
(Refer Slide Time: 51:38)



Let's recall g n we define to be this we define this factor g n to be this and it can be it has been it can by just by slowing a large number of problems finding out the typical values for g n, it has been shown, that for partial pivoting g n is lesser than or equal to 2 to the power n minus 1, where n is the dimension of the matrix and for complete pivoting g n is lesser than or equal to 1 point 8 to the power point 25 l n, which gives a narrower bound, but even if we do just partial pivoting this bound is sufficiently narrow because g n rarely exceeds eight right.

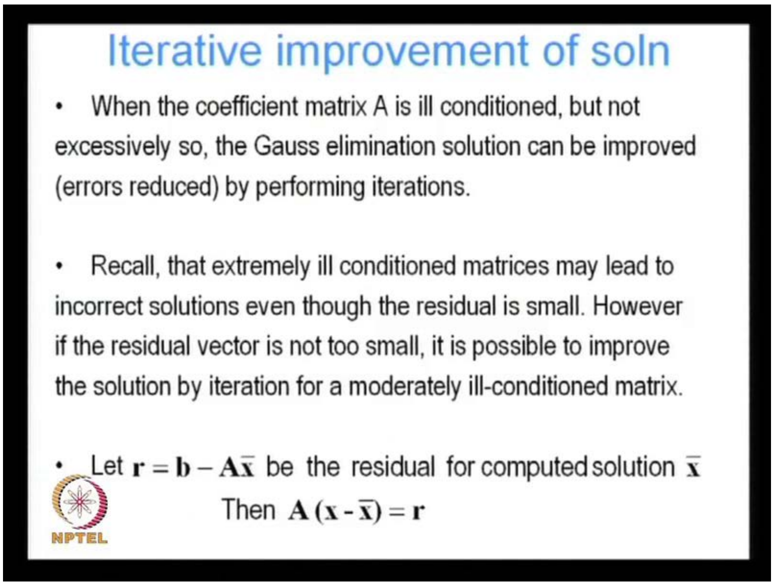
So, the bound on e is satisfactory, because g n is usually quite small lg n rarely exceeds 8 g n rarely exceeds 8 so in that case, we have we have obtained bounds on e bar on E L L bar and U bar are the computed triangular factors with partial or complete pivoting why do we say partial or completing pivoting, because we have try to find a a-priori bound a-priori bound right, and we have mentioned that we can only obtain an a-priori bound and all those multipliers are less than 1. So, partial pivoting is necessary so this bound is satisfactory unless the ratio g n is large.

(Refer Slide Time: 53:38)



So, that was. So, we have found a bound on the Gaussian error in Gaussian elimination due to round off some people have often try to improve the solution improve the results of Gaussian elimination they often try scaling, basically they either scale the coefficient matrix or they scale the right hand side. And they assumed they sometimes think that before because their doing scaling they somehow going to get better solutions they are going to get smaller errors, but it has been seen that scaling the unknowns or the coefficient matrix has no effect on the accuracy of the computed solution except by affecting the choice of the pivots. So, it can improve the stability of the solution because pivot is going to the size of the pivot is going to determine the stability of the solution. So, if the pivots are large we are going to get more stable solution the pivots are small the error this solution is going to be less table, but it is not going to effect the accuracy of the solution the accuracy of the solution is not going to be effected by scaling.

(Refer Slide Time: 54:55)



So, that was that was discussion on the error estimate due to round off during Gaussian elimination. Next from next lecture onwards we are going to talk about iterative methods for solving linear systems, but before we talk about iterative methods for instance Gauss Seidel iterations or Jacobi iterations. We will briefly talk about a simple technique to improve the solution of direct methods improve the solution of obtain using direct methods such as Gaussian elimination by some iterations. So, we solve the problem using Gaussian elimination, but then we can improve the solution by doing some simple iterations. So, we will talk about that first before going into directly into iterative methods for solving Gaussian for solving linear systems.

And we will find that these iterative methods are particularly suited for problems which have coefficient matrix which are very spars right which are very spars because Gaussian elimination, we know one of the problem with Gaussian elimination is that it destroys sparseness of this system right. So, so we want if the matrix is really spars we want to use iterative methods because those methods preserve this sparseness of this system. So, we can take advantage of this sparseness in reducing the number of mathematic mathematical operations as well as the storage required.

Thank you.