MITOCW | MITRES2_002S10linear_lec12_300k-mp4

PROFESSOR: Ladies and gentlemen, welcome to lecture number 12. In this lecture, I would like to discuss with you solution methods for eigenproblems. This is a very large field, and all I can do really in this lecture is to give you or discuss this you a few of the solution methods that we are using in finite element analysis. So it will be a very brief introduction, a brief survey of the methods of solution that we're using for finite element eigenvalue problems. The generalized eigenvalue problem, K phi equals lambda M phi, is a problem that we encountered in the multiple position analysis. Please recognize that I'm calling lambda here, the eigenvalue, which was our omega squared in the last lecture.

This is really the eigenproblems that we are primarily addressing ourselves to finite element analysis. The standard eigenproblems problem here, EVP stands for eigenproblem, has M equal to the identity matrix. So when we're talking about solution methods of this problem, we're really also talk about solution methods of this problem. If we have, in a [? multiple ?] position analysis, non-proportional damping, then we would have to solve this eigenproblem, which is a quadratic eigenproblem. I will not address myself to the solution of this problem, so in what follow, we are talking about a solution of this problem, and of course, the solution of this problem also, when M is equal to the identity matrix.

In particular, we are interested in the solution of large eigenvalue problems when N, is say, greater than 500. N, of course, being the order of K and M, and with the half bandwidth, I should say, being greater than 60.

The number of eigenvalues that we will be looking for, that we want to calculate, are say, 1 to 1/3 of N. 1/3 of N is already large number of eigenvalues. Generally, we are talking about 1 to and at most, 100 for large eigenproblems. When we have a small eigenvalue problem, in other words, N being small, say 50 or 100, there are a large number of different techniques that can be used to solve this eigenvalue problem, and it really does not make that much difference which technique you are using. The cost can still be quite significant, but certainly, when we're talking about very large eigenproblems, N being even larger than a 1,000 or 2,000, then it is of up most concern to select a technique that is very effective.

And I want to address myself really now towards those techniques that we feel are very effectively used in finite element analysis. In dynamic analysis, of course, with proportional damping, this was the eigenvalue problem that we wanted to solve. Now, if they are 0 frequencies present, in other words, when we have rigid body modes in the system, then we can use the following procedure to basically getting rid of these 0 frequencies. We simply add, on the left hand side, a mu M phi, and on the right hand side, a mu M phi. And taking these two together, we are getting this coefficient matrix, and a new lambda value. That lambda value now is omega squared plus mu, where omega squared is equal to lambda minus mu.

What we have been doing here, is basically, we have been applying a shift. In other words, if we were to plot the characteristic polynomial, giving us the eigenvalues, the characteristic polynomial of course, being p of lambda being determined K bar minus lambda M, where K bar is equal to K plus mu M.

If we plug this characteristic polynomial, we will see this curve here, schematically drawn, of course, where the omega squared values would start here, in other words, the lowest omega squared value is, in fact, 0. Whereas our lambda value now starts right here. This is the shift that we [? imposed. ?] So, basically, we now recognize that we have all eigenvalues being greater than 0, because this is our new starting value. All eigenvalues measured from here are greater than 0, the smallest one being mu, namely when omega squared is equal to 0. And the larger ones being here.

It is effective to perform such a shift when we have 0 eigenvalues, in other words, to operate on this eigenvalue, rather than on that one, because our techniques then are very stable and effective, and this shift here is, in general, therefore performed. So all we have to address ourselves to then from now on, is really, the solution of eigenvalues and eigenvectors, when all of the eigenvalues are greater than 0. If we start off with a problem where we have a 0 frequency, we apply the shift so that the eigenvalues now of this problem here are all greater than 0.

In buckling analysis, we have this eigenproblem. We really did not talk about-- we did not really at all about buckling analysis in the set of lectures, but you might just be interested in finding out how would we solve for these eigenvalues. Well, here is the characteristic polynomial. The smallest lambda value, of course, gives now a critical load, that is the one e would be interested in. It is effective here, to rewrite this problem in this form. All we do, is we are taking the lambda value onto the other side. In other words, we are dividing this equation here by a lambda. So that one of the lambda stands on this side, and KG phi on that side. The resulting problem is this one, where kappa is 1 over lambda and we have now here, is a K matrix.

K is positive definite, and now it is more effective to operate on this equation. How do we obtain now the largest kappa? Because the largest kappa is the one that we are interested in, which corresponds to the smallest lambda, the smallest buckling load. Well, we would now perform a shift to the right, subtracting mu K from this side, from this side, and from that side, to obtain this eigenvalue problem. That means we are really shifting from here up to there, and now we are interested in solving kappa N by operating on this eigenproblems problem. It is this eigenvalue problem that we can operate on directly, using the techniques that I would be discussing later, to determine search method and the subspace iteration method.

But before we get into those techniques, I would like to discuss with you some preliminary considerations that are important when we look at an eigenvalue solution. The important first point that I want to make is that there is a traditional approach for the solution of this eigenvalue problem here. Let's look at this eigenvalue problem. Where of course, when we're talking about a buckling analysis, we would have different matrices here, but basically, we still solve this eigenvalue problem.

The traditional approach lies in factorizing the M matrix into Cholesky factors. We talked about the Cholesky factor when we talked about solution of equations. So here it comes up again. We would factorize the M matrix into Cholesky factors, as shown here. We define now a new eigenvector, phi curl being equal as shown here, L curl transpose times 5. We substitute from here into there, and we directly obtain this eigenvalue problem here. Now we have the standard eigenproblem.

Now once we have a standard eigenproblem, there are a large number of techniques that can be used. For example, the QR method, the standard Jacobi method, et cetera. So this is, what we might call. a traditional approach to solve the generalized eigenproblem, this is the one here that we are really interested in. But it involves this transformation, and this transformation here can involve a very significant cost, because we have to find these Cholesky factors and then, of course, we have to calculate this K curl. Another way of performing a transformation of the generalized item problem to a standard form would be to solve the eigenproblem of M first. W here and W transpose [? store ?] the eigenvectors of the M matrix, and D squared is a diagonal matrix of the eigenvalues of M. And then having factored M in this form, an equation similar to this one here can be written. Where now, the transformation, of course, involves the eigenvectors, W.

Well, this approach is sometimes still quite effective, but in general, it cannot be recommended because there's too much cost involved in the factorizing of the M matrix, and then particular, in the transformation here, and the solution of this problem and the, of course, to get from this problem, the phi vector, we have to go back to this equation. A direct solution of the generalized eigenvalue problem that we want to solve is really more effective. And now, we are ordering the lambdas as shown here. Remember, our smallest lambda now is greater than 0, the eigonvectors are stored as shown here, and for most of a position analysis, we are really interested in solving only for a certain number of eigen pairs, i equals 1 to p, as discussed in the last lecture, or i equals r to s. If we have a vibration excitation problem, we really only want to solve for specific frequencies and corresponding mode shapes in a certain interval, as shown here.

The solution procedures on this generalized eigenvalue problem can be categorized into the following forms. The first solution procedure, or a whole class of solution procedures, that I should really say, which we might call the vector iteration techniques. The idea is the following-- if we want to solve this problem, then why not solve it in the following way? Start off with a vector on this side, that we assume, calculate a new vector on this side, then take that new vector, plug it back on the right hand side, and cycle through this way.

This is indeed inverse iteration. What do we do here? We are putting a vector on the right hand side, x1, when K is equal to 1, that vector we have to assume. Then we are multiplying out the right hand side. Now we have a set of simultaneous equations that we would solve just in the same way as we solve the equations in static analysis. We can think of this as a load vector. We are calculating this displacement vector, and then having got this displacement vector, and not taking care of this lambda, or having, basically, taken this lambda out of this equation. Which means that this vector will grow in lengths, or collapse in lengths. Therefore, want to scale this vector down to a good length-- a length that does not hit overflow, so to say, or underflow in the machine. And that scaling is achieved as shown here.

So, the final result then, this is just a scalar here, as you can see, is a vector here, which we can put on to the right hand side and keep on iterating around. In fact, we find as that iteration procedure converges to the lowest eigenvector, provided the starting vector is not orthogonal, is not M orthogonal to phi 1. This is one first iteration technique. Maybe I should put a side note down. A side note that many of you might have in your mind. Many of you might have the following question in your mind-- why he iterate, why does he not solve directly for the eigenvalues? Just like we saw a set of simultaneous equations. Well, there's one very important point, and this is the point-- if we are solving an eigenvalue problem, and shown here, then really that means the following-- we are saying that determinant, K minus lambda M shall be equal to 0.

Why? Because if we write this equation in the following form-- K minus lambda M times phi equals 0, then we have here a set of simultaneous equations with a 0 vector on the right hand side. And there is an important theorem in linear algebra that says, only a solution to this set of equations when this coefficient matrix is singular. Which means that the determinant of this coefficient matrix must be 0. So, what do we want? We really want to solve p of lambda determinant, K minus lambda M. We want to solve for the 0's of this polynomial. That's really what we want.

Now, if we want to solve for the 0's of this polynomial, then we also have to understand that when k is of order 4, there will be four 0's. The polynomial itself will be of order 4. If K is of order, N, then there will be N 0's. So the polynomial is order N. However, if we want to find the solution of a polynomial, or order N, then we very quickly find that, in fact, we can only look up the solution to such polynomials if they are of low enough order, lower than say, 4.

Above for larger order polynomials, there's no way you can look up the solution. You have to iterate to obtain the solution of the 0's of this polynomial. However, since this problem here, is completely equivalent to this problem. To solve this problem, you have to iterate. And this is the important conclusion. It is of up most importance that you understand that to solve an eigenvalue problem of order larger than say, 4, you will have to iterate. Because you're solving for the 0's of a polynomial of order larger than 4. Only for very small order polynomials, we can look up the solutions in handbooks.

So having grasped the fact that we have to iterate to solve for the 0's of this polynomial, to solve for the eigenvalues of this problem, we are stuck with iteration, and since we're stuck with iteration, we now want to come up with techniques to iterate. And the first technique to iterate is this vector iteration technique.

And the basic idea, once more, is if you want to solve this equation, well, then why not iterate in the following way? Put some starting value in here, this number is just a scalar, let's neglect that effect for the moment. And having put a starting value in here, calculate this one and keep on cycling around this way. And that's what we call inverse iteration. Inverse iteration here, a vector inverse iteration, because we have to solve a set of equations here, with K as the coefficient matrix.

There are other iteration techniques. There's also forward iteration. And basically, what we're saying there is-- well if we are starting off here and going around in this circle, why not start off here and go around in that circle, the red line. And that, in fact, gives us forward iteration.

One can prove that that forward iteration converges to the largest eigenvalue. We are usually interested in the smallest frequencies, the smallest eigenvalues, and therefore, we are using inverse iteration quite commonly.

Then there is Rayleigh Quotient iteration. The Rayleigh Quotient iteration is again, an inverse iteration, however, with an acceleration scheme in it, a very effective acceleration scheme. This Rayleigh Quotient iteration can be employed to calculate one eigenvalue and vector, and then, of course, we have to deflate afterwards the matrix, the coefficient matrix for the eigenvector that we have calculated already, and we then could go on towards the calculation of additional eigenvalues and vectors. The same deflation also would be used in inverse iteration alone, and in forward iteration alone. In a Rayleigh Quotient iteration, we converge to an eigen pair, but which one is not guaranteed.

So this is a class of techniques, iteration methods that we are using.

Another class of techniques, again iteration methods, we are stuck with iteration, as I mentioned earlier, is the polynomial iteration methods. Here, we are saying that this problem is equivalent to that, we want to solve for these 0's. Therefore, for the 0's of this polynomial, sketched out here, and one way is to use Newton iteration. Here we have p, here we have p prime. Well, of course, if we have p here, we have to also recognize we should have the coefficients here. Well, these coefficients are given here, A0 to a n. And we could rewrite this polynomial in this form, once we know the zeroes. Now, if we don't know the zeroes, of course, we would be stuck on iterating on this matrix, where we would have to calculate these coefficients.

However, there are now really, two different techniques. The first technique is explicit polynomial iteration. In the explicit polynomial iteration, we expand the polynomial and iterate for the 0's, by that I mean, we actually expand this polynomial. However, this technique is not suitable for larger problems because first of all, there's much work to obtain the ai's, and then it's an unstable process. Unstable because small errors in these coefficients here, give large errors in the zeroes.

So this is not a suitable technique, and we are better off using implicit polynomial iteration. In the implicit polynomial iteration, we evaluate the determinant all this matrix here. In others, p at mu i, via the L D L transpose factorization, which we discussed in the Gauss elimination solution, when we talked about Gauss elimination solution. To obtain the determinant of this matrix here then, we need simply to multiply all the diagonal elements. This technique is accurate, provided we do not encounter large multipliers L matrix. And if we do, we have to shift away, we have to change our mu i. Also we directly solve for lambda 1 and so on. And for the more we can use effectively a secant iteration, instead of calculating, in other words, the p prime at mu. This is here approximately equal to p prime, at mu i.

Instead of calculating this derivative, we are putting this value here, which gives us an approximation to that derivative, this is what we call the secant iteration. Once we have calculated one eigenvalue, of course, we will have to deflate to calculate the next eigenvalue, and this is the process that he would pursue. Pictorially, we have here, our p lambda, this is the polynomial. Let me show it to you here.

Here, we have the polynomial, here we have lambda 1, here we have lambda 2. In the secant iteration, we start off with the value of mu i minus 1 here, mu i, these two values are given. Usually, in a practical analysis, mu 0 would be this value here, and mu 1 would be a lower bound on lambda 1. And plugging these two values in, we get mu i plus 1, and you can see that this process would then converge to this eigenvalue. Once we have lambda 1, we can repeat the iteration. However, working now on p lambda divided by lambda minus lambda 1. And that means we are operating on this polynomial here, and polynomial then gives us the iteration on that polynomial then gives us the second eigenvalue. So we are deflating the actual p lambda by the lambda 1 value in order to obtain a convergence to the second value.

Well, this is the one class of methods. Another class of methods, the third one, are the Sturm sequence methods. And here, the basic idea is the following-- if we look at this eigenproblem, and I use a 4x4 matrix here, then if we take a shift on this eigenproblem, mu S, and this is the shift here. Then if we factorize K minus mu S M, equal to L D L transpose, again, we're using our Gauss elimination procedures basically. Then the number of negative elements in D is equal to the number eigenvalues smaller than mu S, a very important point.

Let us look at this example here of the 4x4 matrix. Here we have plotted characteristic polynomials. The characteristic polynomial of this 4x4 matrix, with that M matrix, of course, is drawn along here, four eigenvalues. And I have put mu S here as the red line. What this theorem says is if I factorize K minus mu S, M, this being the K matrix, the total K matrix, and the 4x4 M matrix goes in here. Then there would be one negative element in D. One only because mu S is large than lambda 1. If mu S were on this side here, there would be two negative elements in D. If mu S were on this side, there would be three negative elements in D-- very important point because qw can use this procedure to directly calculate eigenvalues the way I will be discussing it just now.

However, let us look a little bit more at why this hole is here. Well, it derives from the associated constraint problems with the original eigenproblem. The original eigenproblem corresponds to this beam problem, which we considered earlier already in the discussion of Gauss elimination. If I constrain this degree of freedom here, I get a new polynomial corresponding to this problem, and the important point is that the lowest root, the lowest value of this problem here lies in between these two-- lambda 1 and lambda 2.

The second one here, lies between lambda 2 and lambda 3, and the third one here lies between lambda 3 and lambda 4. The second associated constraint problem has two eigenvalues, and this one lies between these two, this one lies between these two. And the final eigenvalue problem here, the third associate constraint problem obtained by knocking out these degrees of freedom here, has one eigenvalue, and that one lies in between these two values here. It's this important fact that is used to derive this final result. Now, we have not time to go through the actual details, but this is the important fact here. In other words, that the [? i's ?] eigenvalue of this associated constraint problem, for example, the second eigenvalue of this associated constraint problem here, bisects or lies in between, not bisect-- lies in between lambda 2 and lambda 3 of this eigenproblem here, and so on.

That result is used to derive this final result, which we use effectively then in the eigen solution. And the class of techniques then that we are referring to using this fact are the Sturm sequence methods. So basically, what we are saying is the following-- take this coefficient matrix, now I make the mu x available. We factorize that coefficient matrix into L D L transposed, and the number of negative elements in D is equal to the number of eigenvalues smaller than mu S, I. How do we use this fact?

Well, if this is here, the polynomial, we could start off by putting mu S 1 into this point here, and we would find out the number of eigenvalues smaller than mu S 1. Then we are taking, mu S 2, and we find out the number of eigenvalues smaller than mu S 2. Of course, we would find that they are one more eigenvalue smaller than mu S 2. Then mu S 1, namely this one. So we know there's one eigenvalue in between mu S 1 and mu S 2. Now we could bisect this interval, and we would find that, in fact, there is still one more eigenvalue smaller than mu S 3, then there was smaller than mu S 1. Because this eigenvalue still lies on the left side of mu S 3.

So we bisect again, we get mu S 4. Notice this distance here, is equal to that distance here. And now, we would find that the eigenvalue still lies in between these two lines, between mu S 3 and mu S 4. And like that. we could continue bisecting the interval until we finally find a very small interval in which this eigenvalue must lie.

We, of course, have to take care in the L D L transpose factorization, because we are not working on a positive definite matrix. The multipliers must be small, the L, I, J elements must remain small in the factorization. But provided that is the case, the round of errors are small, and the procedure is very stable.

However, convergence can be very slow, can be extremely slow because you have to cut down the intervals further and further until you finally get a very small interval, in which you now know the eigenvalue does lie. Of course, at some point, you might want to switch to another iteration scheme. Once you know there is an eigenvalue in between say mu S 3 and mu S 4, you might want to switch to a iteration scheme that converges to this particular value much faster.

Finally, a whole class of methods that I would like to briefly mention to you are the methods of transformations, where we are operating on this eigenvalue problem, recognizing that phi transpose K phi is equal to lambda. And phi transpose M phi is equal to I. Where phi is defined as shown here, lambda is defined as shown here. This fact, of course, we use already earlier in the [? multiple ?] position analysis. Now how can we get phi and lambda? That is our objective. Well, if we know that this matrix diagonalizes the K matrix, and it also diagonalizes the M matrix, then why not try to construct it by iteration? And this is the basic idea in the transformation methods.

What we do is the following-- we take the original K matrix, we are operating with P1 transpose K P1. On K, and on M, such as to make this resultant matrix here closer to a diagonal form. Ultimately, we want to get a diagonal matrix. Well, one step in the iteration is to make this P1 transpose K P1 matrix, and P1 transpose M P1 matrix, both of these matrices a little closer to diagonal form. Then K and M were by themselves.

So this is the first step. Now, we have here, let's call this a K2 matrix, and this here an M2 matrix. Now, we are operating with P2 transpose on K2, and also there's a P2 here. So let me take a different color to show that now we are operating with this part here, and that part there. And this, of course, we are doing in such a way as to have that P2 transposed, K2, P2 is a little bit closer to diagonal form than was K2. I hope you can see these lines-- let me put them in once more. What I'm saying is we have K2 after having applied P1 transpose on P1, on K, and similar for M.

And now we want to make K2 still a little bit closer to diagonal form. Well, we are now applying P2 transposed on K2 and the P2 on K2, this one here has been replaced by K2, this one here has been replaced by M2. And what we're saying is that this final result, in this final result should be closer to diagonal form. In fact, this should be closer to I matrix, and this should be closer to the lambda matrix, ideally.

Well, this is the basic process that is used in the generalized Jacobi method. We can show that if we choose specific matrices, P I matrices. That zero always one [? of ?] diagonal element. In other words, the P I matrix, zeroes a particular off diagonal element in this current coefficient matrix, similarly for the M2, for the current M coefficient matrix. Then we know in the generalized Jacobi method that the process finally converges. It converges and we get a diagonal matrix here.

With proper scaling, it will be the identity matrix. And here, we are getting also a diagonal matrix, and if this is an identity matrix, this will actually be the matrix storing the eigenvalues. Well, this is the generalized Jacobi method here. They are other techniques that can also be used, of course, this is a very common approach taken when we have M being equal to the identity matrix.

However, the disadvantage of this approach is that we are calculating all eigen pairs simultaneously. In finite element analysis, we are only interested in specific eigenvalues, so lowest P, or a certain number of eigenvalues in an interval. Here, we have to calculate all of the eigenvalues stored in the lambda matrix, and all of the eigenvectors. And the eigenvectors here, would be these vectors here. These are the the matrix phi, which is P1 times P2, up to P K, after K steps of iteration. This gives us the matrix phi, storing all of the eigenvectors.

So the disadvantage is that we are calculating all of that eigenvectors, which in finite element analysis is hardly ever required. All we want is the total number of eigenvalues and eigenvectors. Remember, also if K is of [? order ?] 1,000, this would be a tremendous amount of work involved and, in fact, it would be beyond the current state of computing capabilities to use a generalized Jacobi method on a K matrix 1,000x1,000 and an M matrix 1,000x1,000. You will certainly stretch the current computing capabilities tremendously by using the generalized Jacobi method on large eigenproblems.

For large eigenproblems, it is really best to combine the basic approaches that I just mentioned. Let's recall the basic approaches. There were iteration methods, vector iteration methods, there were Sturm sequence methods, there was the transformation method. And then, of course, there were polynomial iteration methods. So really, four classes of methods that we are talking about.

Each of them, we find, have some advantages. Well, as engineer, surely we want to now combine the advantages of each of these methods to come up with optimum techniques for our specific problems. Well, there are a number of thoughts that we would go through. First of all, a determinant search, a polynomial iteration method would be effective to get near a root. Vector iteration then could be used to calculate an eigenvalue and an eigenvector. The transformation method would be good to orthogonalize the current iteration vectors if we iterate with more than just one vector. And the Sturm sequence method is extremely powerful to assure that we have actually calculated the required eigenvalues and eigenvectors.

Well, the first method then that I would like to discuss with you is the determinant search method. In this method, we have combined some of these techniques that I pointed out to you earlier in an optimum way. Here, we have the polynomial P lambda, we want to, of course, solve for the roots of that polynomial-- that is our objective. Here is lambda 1, there is lambda 2. I also show a point, A, because I will refer to it just now.

The basic scheme in the determinant search method is to look for the zeroes of the determinant, of the determinant K minus mu I M. Of course, these zeroes give us R equal to the eigenvalues. That's what we're interested in calculating. Well, we are calculating the determinant by the factorization of this matrix into L D L transpose, and we notice that this determinant is simply the product of the [? D I I's, ?] as I mentioned earlier.

Now, having a certain starting value, mu 0, say, and mu 1. We can directly use the secant iteration, that I mentioned to you earlier to get close to an eigenvalue. Now, I also slipped in here an acceleration factor. This acceleration factor would be equal to 1 in the normal secant iteration that I showed to you earlier already.

However, since we want to use this polynomial iteration technique only to get close to an eigenvalue, and we might actually obtain with eta equal to 1, sometimes slow convergence. What we do is we simply set it larger to accelerate the process. And this is what we're doing here. If we do jump beyond an eigenvalue because we have accelerated the secant iteration. In other words, if we do a jump across here, then since we are factorizing the matrix here, we can also look at the number of negative elements in the D matrix. These would tell us that we, in fact, have jumped across an unknown eigenvalue.

So, in this particular case, of course, there would be one negative element in the D matrix. And we would know now we have jumped across. The jumping across an unknown eigenvalue is not possible when eta is equal to 1. However, it is possible when eta is greater than 1. And however, that jumping does not hurt us. In fact, all we will find is that the Sturm sequence count will tell us that we have jumped across an eigenvalue, and then, of course, we have to go to another strategy of solving for the eigenvalue more accurately.

But the important point is that all we want to do via this procedure is to get into the vicinity of the eigenvalue. We do not want to calculate it accurately, all we want to do is get into the vicinity of an unknown eigenvalue. First, of course, lambda 1, then lambda 2, and so on.

There's also a concern, of course, that we do not want to jump beyond A, because if we, in our process of looking for lambda 1, we are jumping too close to lambda 2, then via the other solution techniques that we are using later on then to calculate the eigenvalue more accurately, we would actually converge to lambda 2. And, of course, that we don't want, we want to calculate lambda 1.

So, the process is then to start off with eta equal to 1, iterate, see how we are progressing towards an unknown eigenvalue. If that progressing is very slow, we are choosing eta larger. There are specific strategies that we're using to choose the right amount all eta. And until we find that we are close to an eigenvalue, either from the left or we have jumped across it already, but you should never jumped to far. And we are making sure that we never jump beyond that point A. and having jumped, we find that the Sturm sequence count tells us about it.

And now, we are going on to another strategy, and that strategy is the vector iteration, vector inverse iteration method that I discussed with you earlier. What we are doing now, is we have a new coefficient matrix, that's the one. We are sitting here, say, and now I have assumed that we have, in fact, jumped. It's not necessary. We might have also approached from this side and come very close to lambda J. And now, we are using vector iteration to get this eigenvalue accurately.

This coefficient here, or this value here corresponds to that length. And if you are adding this value here to this shift, we are getting an approximation to lambda J. And that, of course, is our objective. Once we have lambda J, we have calculated this eigenvalue-- the eigenvector comes out, in x k plus 1 as k increases, so we have calculated this eigen pair, and now, of course, we would use deflation method, just the way I showed it to your earlier, for the polynomial iteration method. To deflate the polynomial of this value, implicitly-- implicitly, that's important-- and we would continue our iteration process towards that eigenvalue next.

In this process, we also want to make sure that the iteration vector is deflated of the previously eigenvectors, for example, Gram-Schmidt orthogonalization. If the convergence is slow in this particular iteration here, we're using also the really Rayleigh-quotient iteration to speed up iteration itself. The advantage of the method-- well, it calculates only the eigen pairs that are actually required., there's no prior transformation of the eigenprobelm necessary. The disadvantage of the method is that we have to perform many triangular factorizations. Therefore, the method is really only effective for small banded systems.

If we want to calculate eigenvalues of large banded systems, we really need an algorithm with less factorizations, and more vector iterations. And that is the subspace iteration method. In fact, we have now extended the subspace iteration method to be also most effective for small banded systems. And I will just refer to it very briefly later.

The basic steps of the subspace iteration method are the following. Here, we have the first equation used. We have on the right hand side, the current iteration vectors. Now, notice that in this particular case, we're iterating with q vectors, where q ip larger than p simultaneously. This gives us here, q right hand side vector. We are solving for q vectors here. This is an inverse iteration step on q vector simultaneously. Then we are calculating the projections of these k and m matrices on these vectors.

Notice that this pat here, of course, is equal to this right hand side. So we would not calculate this here. We simply use the right hand side, which we have evaluated already and plug it right in there. But having calculated these two matrices, which are now of order q by q, we are solving eigenvalue problem all of these q by q matrices. And the solution of that eigenvalue problem is shown here. Notice this matrix here is a q by q matrix. We're using q vectors. This is a q by q matrix storing the eigenvectors of this eigen problem here, of the eigenproblem corresponding to these matrices.

This here is the diagonal matrix listing the eigenvalues of the eigenproblem corresponding to these matrices. Having calculated these matrices here, and of course, that one too, we are performing this operation to get a new set of iteration vectors, which then are plugged back into here. Notice that the solution of this eigenproblem, q by q, might be a 50x50 matrix here, 100x100 matrix, maybe a 200x200, but that already would be a large number of vectors if we're using-- a very large number of vectors. This solution of this eigenproblem can be effectively achieved via the generalized Jacobi method, which I referred to briefly earlier.

Of course here, we are using our solution algorithm for static analysis. There's L D L transpose factorization involved here. And here, this is a simple matrix multiplication that has to be carried out. This part here, is the eigenproblem solution, and there's another vector multiplication carried out here. Now, under specific conditions, this eigensolution algorithm converges and the vectors stored in x k plus 1, they are q vectors now here, with appropriate ordering, converge to the phi matrix. This is now an n by q matrix, storing the lowest eigenvectorr, or I should say, the q eigenvectors corresponding to the lowest eigenvalues. And here, we have it written out, and of course, this is a diagonal matrix. The condition that we have to satisfy here is that the starting subspace spanned by these vectors here must not be orthogonal to the subspace spanned by the p eigenvectors that we are interested in.

Of course, here, I'm talking about q eigenvectors. I really am only interested in p eigenvectors, in the lowest p eigenvectors. So in general, what happens is that I will only be interested in these eigenvectors here, and the lowest eigenvalues. One comes out here, I'm not too much concerned about. In fact, we are carrying these along only to accelerate the iteration scheme. And I will show you just know what the convergence rate is, and why we are carrying them along.

So since we only interested in the lowest p eigenvalues and corresponding eigenvectors, we want to get convergence towards those, and what that means is that our starting vectors in here must not be m orthogonal to the eigenvectors that we want to calculate, phi 1 to phi p. If that condition is satisfied, one can show, theoretically, and of course, we see it in practice also, that this iteration scheme will always converge to the lowest eigenvectors and corresponding eigenvalues.

One important point here, once we have calculated, say, a certain number of iteration vectors, and we have tested for convergence, we are saying that convergence is reached, say, when the current iterates on the eigenvalues don't change very much anymore. This tol might be 10 to the minus 6, as an example. Then once this condition is satisfied, we stop the iteration, and we really want to make then sure that we really converge to the eigenvalues of interest. And how is that accomplished? Well, if these are the p eigenvalues, we are calculating error bounds on these eigenvalues, indicated here by the blue lines here. And then just to the right of the error bound. Or we can really use that error bound, we apply a Sturm sequence shift, mu S. What this then says that k minus mu S factorize into L D L transpose. And we must now have in the D matrix, p negative elements. And only p, not more, not less, just p negative elements must be occurring in the D matrix.

This check is absolutely sure. If this check a satisfied, we can be absolutely sure that we have calculated the eigenvalues of interest, the lowest p eigenvalues, in this particular case. So I repeat, after we have satisfied this convergence rate, or this conversions limit, I should say-- typically 10 to the minus 6, or 10 to the minus 8, depending on what accuracy you want. We calculating eigenvalue bounds, as shown here. We are putting a mu S Sturm sequence shift just to the right of the last eigenvalue bound, and if this is the p's eigenvalue, we must have here, exactly p negative elements indeed.

The convergence rate of the iteration, we find is given by lambda i divided by lambda q plus 1. And the convergence rate to the eigenvalues of interest are just this value squared. Notice that we have here, q plus 1. So if we are interested in the p's eigenvalue,== the black pen doesn't work too well, let me take the blue one here-- if you're interested in the p's eigenvalue as the largest eigenvalue, notice that the convergence rate is that lambda p plus-- lambda p divided by lambda q plus 1. That is the worst convergence rate that we are reaching because the other eigenvalues below lambda p are smaller, giving us better convergence rate. So this is the worst that we can reach, if we are only interested in p eigenvalues and corresponding eigenvectors.

Notice however, that lambda q plus 1 must be greater than lambda p. If lambda q plus 1 is equal to lambda p, we would never converge. So this is the reason why we are choosing q to be larger than p, substantially large than p. For example, we might choose q to be typically, as an example, q being the maximum of 2p and p plus 8. This is a formula that we have used with quite a lot of success. In other words, when p is equal to 4, we would use-- no, sorry, I should say the minimum here. the In other words, when p is equal to 4, we would have 8 here. When p is equal to 8, it's the same. Because these two values give us the same. And when p is equal to 100, then we just have 8 more. Q is equal to 108. This 8 here is a safeguard, so that this cannot be equal to 1 in practice. It would only be equal to 1 if we would have, basically. 9 time the same root, and of course, that is a case which we would hardly encounter in practice.

Now, regarding the choice of the starting values. For the vectors, there are two choices. x1 is simply a vector with 1 only. xj would be a vector with zeroes everywhere, but with a 1 somewhere. And that one is in the j's entry-- sorry, the k's entry. The k's entry for ek, ek has in the case, entry of 1. And otherwise, everywhere zeroes. This would be the second to the q minus first vector. And the last vector, we take a random vector. That is one choice of starting vectors. The Lanczos method can also be used to generate starting vectors very effectively.

Once we have selected the starting vector, we go through the iteration schemes the way I have been displaying it. And there, it is important, once again, to perform after the convergence- the Sturm system sequence check, and also we might want to calculate error bounds on the eigenvalues and eigenvectors. These error bounds would be calculated by simply saying, well, if we want to satisfy-- or if we want to solve for lambdas and phis that satisfies this equation, then why not see how close we will satisfy. Make that a minus and put here an error, say, R, on the right hand side.

Substituting then the last values for lambda and phi that we just kind calculated, the last iterative values, we get this top line here. We're taking a norm the way I've been talking about earlier. And we are also taking a norm at the bottom to get, basically, a relative value, top to bottom. And then relative value gives us epsilon i. So epsilon i should be of order, 10 to the minus 3, if we want to have three-digit accuracy in the eigenvalues and eigenvectors. In fact, the eigenvalues will generally be more accurate than the eigenvectors. That is, of course, a very low accuracy. Ideally, we have much more accuracy after the iteration schemes. We would have to tighten the tolerance, the tol value that I referred to earlier sufficiently to make epsilon i small enough.

Now, I like to refer, very briefly also, to an accelerated subspace iteration method that we have been developing during the last years, which is a very substantial extension of this subspace iteration method. The idea here is, basically, that we wanted to combine the sum of the effective techniques that we are using in the polynomial iteration method, with the standard subspace iteration method that I have discussed with you here. Basically, in the accelerated subspace iteration method, we are not keeping a constant coefficient matrix, as we have done here. The k matrix never changed. Here, we are shifting the coefficient matrix, we are shifting through the spectrum. Much in the same way as we are pursuing it in the determined search method. And that is number one.

And number two-- we are also not iterating with vectors that are always larger than the number of eigenvectors that we want to calculated. In fact, if we want to calculate, say, 100 eigenvalues, we might just use four iteration vectors at a time, and shift through the spectrum, always with four iteration vectors, capturing the eigenvalues values and eigenvectors that we are really interested in, in bundles. So this is a way how we have very substantially accelerated the standard subspace iteration method that I have been talking to you about just now.

If you are interested in seeing some solution times, please we look at this reference. Ladies and gentlemen, this then brings me to the conclusion of this lecture. And in fact, it also brings us to the conclusion of the whole set of lectures that I have been presenting to you. These set up lectures have been brief and very compact. They've been a survey introduction to the finite element method-- a very brief and compact introduction and survey. Brief because there are only 12 lectures. Compact because I wanted to present to you as much as possible in these 12 lectures. And there is, of course, a lot of information that we could have talked about.

However, I hope that the information that I had given in the lectures has been of interest to you, and it will also be valuable to you in your work and general decision making process of whether you want to use a finite element method, how you want to use it, and so on. I like to take the opportunity at this point to thank the Center of Advanced Engineering Studies for the terrific effort they've made in preparing the tapes. I'm very happy that they have been cooperative with me in the way they did, in the preparation of the tapes. I also would like to thank you as listeners for your patience of listening to the tapes and, once again, I hope you found the tapes interesting and have obtained some value information on them. Best of luck in your further studies of the finite element method.