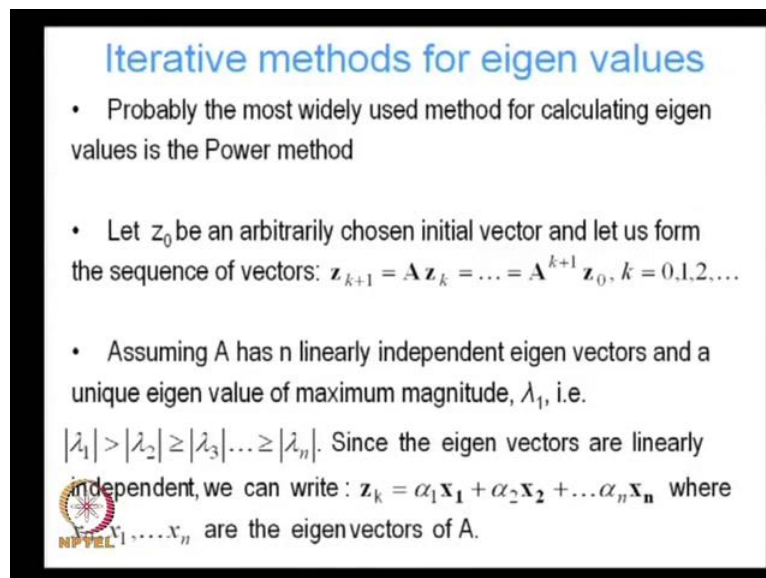


**Numerical Methods in Civil Engineering**  
**Prof. Arghya Deb**  
**Department of Civil Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 11**  
**Iterative Methods for Eigen Value Extraction**

Let us see 11 of our series on Numerical Methods in Civil Engineering, we are going to talk about Iterative Methods for Eigen Value Extraction. And at the end, if you have some time left over, we are going to introduce some net methods in a numerical solution of non-linear equations.

(Refer Slide Time: 00:39)



**Iterative methods for eigen values**

- Probably the most widely used method for calculating eigen values is the Power method
- Let  $z_0$  be an arbitrarily chosen initial vector and let us form the sequence of vectors:  $z_{k+1} = A z_k = \dots = A^{k+1} z_0, k = 0, 1, 2, \dots$
- Assuming  $A$  has  $n$  linearly independent eigen vectors and a unique eigen value of maximum magnitude,  $\lambda_1$ , i.e.  
 $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \dots \geq |\lambda_n|$ . Since the eigen vectors are linearly independent, we can write:  $z_k = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n$  where  $x_1, \dots, x_n$  are the eigen vectors of  $A$ .

Last time, we looked at the Eigen value problem and ways to solve the Eigen value problem, basically ways to find the Eigen values and the Eigen vectors. And we found that one way to do it was write out the characteristic equation of the system and then to find out roots of that characteristic equation. But for an  $n$  by  $n$  matrix we know, that the characteristic equation is a polynomial of order  $n$  and since it is a polynomial of order  $n$ , it is going to have  $n$  roots. And we talked about abele's theorem, which tells us that for a polynomial of order greater than 4, it is not possible to get closed form solutions.

So, we have to rely on some numerical method for solving the characteristic polynomial, probably the most widely used method for calculating Eigen values is what is known as the power method. So, this is an iterative method where, we repeatedly use, we

repeatedly, we start with any arbitrary assumption for the Eigen vector and we repeatedly iterate using our the coefficient matrix to improve the guess to the Eigen vector.

And can be shown that after repeated iteration, the initially chosen vector, it actually converges to the Eigen vector. For instance, let us  $g_0$  be any arbitrarily chosen initial vector and let us, form a sequence of vectors, which is given by  $z_{k+1}$  is equal to  $A z_k$  and so on and so forth. So, from  $g_1$  from  $g_0$ , we get  $g_1$  by taking the product of  $g_0$  with  $A$  from  $g_1$ , we get  $g_2$  by taking the product of  $g_1$  with  $A$  and so on and so forth.

Until we get  $z_{k+1}$  is equal to  $A z_k$ , where  $A$  is the coefficient matrix, the matrix for which, I want to find the Eigen values and the Eigen vectors. Now, for the sake of a, let us assume that  $A$  has  $n$  linearly independent Eigen vectors and unique Eigen value of maximum amplitude  $\lambda_1$ . So,  $A$  is a non singular matrix, it has linearly independent Eigen vectors and it has Eigen values, such that the largest Eigen value is unique, that it that is there are no other Eigen values, which are equal to  $\lambda_1$ .

So,  $\lambda_1$  is strictly greater than  $\lambda_2$ , but  $\lambda_2$  may be greater than or equal to  $\lambda_3$  and so on and so forth, but  $\lambda_1$  is the largest Eigen value is unique in the sense, that there are no other Eigen values of the matrix equal to  $\lambda_1$ . So, that is what we assumed and since, we have assumed that  $A$  has linearly independent Eigen vectors that means, if we consider the Eigen vectors of  $A$  to be  $x_1$  through  $x_n$ , we know that  $x_1, x_2, \dots, x_n$  form a basis for the vector space of dimension  $n$  right. So, it forms the basis for the vector space of dimension  $n$ , so any  $n$  dimensional vector can be represented as the linear combination of my basis vectors, which are  $x_1, x_2$  through  $x_n$ .

So, for instance the  $z_k$  vector the  $z$  vector after at the  $k$ th iteration, I can write it as a linear combination of  $x_1$  through  $x_n$ , I can write it as  $\alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n$  where,  $\alpha_1$  through  $\alpha_n$  are some constant. Similarly, I can write any vector in the iteration  $z_{k-1}, z_{k-2}$  any vector in the iteration, I can write it as a linear combination of my Eigen vectors, which are linearly independent.

(Refer Slide Time: 04:57)

### Power method


Then,  $\mathbf{z}_k = \alpha_1(\lambda_1)^k \mathbf{x}_1 + \alpha_2(\lambda_2)^k \mathbf{x}_2 + \dots + \alpha_n(\lambda_n)^k \mathbf{x}_n$

$$= (\lambda_1)^k \left( \alpha_1 \mathbf{x}_1 + \sum_{j=2}^n \left( \frac{\lambda_j}{\lambda_1} \right)^k \alpha_j \mathbf{x}_j \right), \quad k = 0, 1, \dots, n \quad (*)$$

Since  $\left| \frac{\lambda_j}{\lambda_1} \right| < 1 \quad \forall j \geq 2$  as  $k$  increases the direction of  $\mathbf{z}_k$  tends to that of  $\mathbf{x}_1$ , provided  $\alpha_1 \neq 0$

The ratio  $\frac{\mathbf{z}_k^T \mathbf{A} \mathbf{z}_k}{\mathbf{z}_k^T \mathbf{z}_k}$  thus tends to  $\frac{(\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{A} \mathbf{x}_1 (\lambda_1)^k}{(\lambda_1)^k (\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{x}_1} = \lambda_1$

and is known as Rayleigh's quotient.



So, we can write then in that case, we can write  $\mathbf{z}_k$  is equal to this why because why can, we write that well.  $\mathbf{z}_k$  is nothing but  $\mathbf{A}^k \mathbf{g}_0$  right,  $\mathbf{A}$  to the power  $k$  operating on  $\mathbf{g}_0$ ,  $\mathbf{A}$  operating on  $\mathbf{g}_0$ . So,  $\mathbf{A}$  operating on  $\mathbf{z}_k$  therefore, can be written as  $\alpha_1 \mathbf{A}$  operating on  $\mathbf{x}_1$  plus  $\alpha_2 \mathbf{A}$  operating on  $\mathbf{x}_2$  plus  $\alpha_3 \mathbf{A}$  operating on  $\mathbf{x}_3$  through  $\alpha_n \mathbf{A}$  operating on  $\mathbf{x}_n$ . And since,  $\mathbf{x}_1, \mathbf{x}_2$  and  $\mathbf{x}_n$  are Eigen vectors of  $\mathbf{A}$  operating on  $\mathbf{x}_1$  is nothing but  $\lambda_1 \mathbf{x}_1$  where,  $\lambda_1$  is some Eigen value of  $\mathbf{A}$ . Similarly,  $\mathbf{A}$  operating on  $\mathbf{x}_2$  can also be written some other line Eigen vector times  $\mathbf{x}_2$  and so on and so forth. So,  $\mathbf{A}^k$  operating on  $\mathbf{z}_k$  can be written as  $\alpha_1 \lambda_1^k \mathbf{x}_1$  plus  $\alpha_2 \lambda_2^k \mathbf{x}_2$  plus  $\alpha_3 \lambda_3^k \mathbf{x}_3$  plus  $\alpha_n \lambda_n^k \mathbf{x}_n$ . Where  $\lambda_1$  is the Eigen value corresponding to the Eigen vector  $\mathbf{x}_1$ ,  $\lambda_2$  is the Eigen value corresponding to the vector  $\mathbf{x}_2$  and so on and so forth.

(Refer Slide Time: 06:22)

### Power method


Then,  $\mathbf{z}_k = \alpha_1(\lambda_1)^k \mathbf{x}_1 + \alpha_2(\lambda_2)^k \mathbf{x}_2 + \dots + \alpha_n(\lambda_n)^k \mathbf{x}_n$

$$= (\lambda_1)^k \left( \alpha_1 \mathbf{x}_1 + \sum_{j=2}^n \left( \frac{\lambda_j}{\lambda_1} \right)^k \alpha_j \mathbf{x}_j \right), \quad k = 0, 1, \dots, n \quad (*)$$

Since  $\left| \frac{\lambda_j}{\lambda_1} \right| < 1 \quad \forall j \geq 2$  as  $k$  increases the direction of  $\mathbf{z}_k$  tends to that of  $\mathbf{x}_1$ , provided  $\alpha_1 \neq 0$

The ratio  $\frac{\mathbf{z}_k^T \mathbf{A} \mathbf{z}_k}{\mathbf{z}_k^T \mathbf{z}_k}$  thus tends to  $\frac{(\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{A} \mathbf{x}_1 (\lambda_1)^k}{(\lambda_1)^k (\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{x}_1} = \lambda_1$

and is known as Rayleigh's quotient.



So, in that case, we can write  $\mathbf{z}_k$  like this, taking out  $\lambda_1^k$  outside, so we get  $\lambda_1^k$  times  $\alpha_1 \mathbf{x}_1$  plus a sum of these sort of terms where, we have quotients of the subsequent Eigen vectors with my initial Eigen vector  $\lambda_1$  rise to the power  $k$ . So, basically I am dividing each of these terms by  $\lambda_1^k$  and what do we know about these terms, well we know that  $\lambda_1$  is my largest Eigen vector right.

So, each of these terms  $\lambda_j^k$  by  $\lambda_1^k$  must have modulus less than 1, therefore I as I increase the number of iterations, as I increase  $k$  as I increase the number of iterations this terms, this is less than 1 becomes smaller and smaller. And as it becomes smaller and smaller, eventually, this part goes away this part becomes negligibly small as  $k$  increases and we can say, we can see that as  $k$  increases the direction of  $\mathbf{z}_k$  tends to that of my first Eigen vector, which is  $\mathbf{x}_1$  provided of course, this coefficient  $\alpha_1$  is not equal to 0 right.

Thus the ratio, let us look at the ratio  $\mathbf{z}_k^T \mathbf{A} \mathbf{z}_k$ , what does that become divided by  $\mathbf{z}_k^T \mathbf{z}_k$ , therefore  $\mathbf{z}_k$  tends to  $\lambda_1^k$  times  $\alpha_1 \mathbf{x}_1$ , because this term goes to 0, therefore  $\mathbf{z}_k^T \mathbf{A} \mathbf{z}_k$  is nothing but  $\lambda_1^k \alpha_1^2 \mathbf{x}_1^T \mathbf{A} \mathbf{x}_1$  right.  $\mathbf{z}_k^T \mathbf{z}_k$  is nothing but again another  $\lambda_1^k \alpha_1^2 \mathbf{x}_1^T \mathbf{x}_1$ , but now I have a  $\mathbf{z}_k^T$  operating on  $\mathbf{x}_1$  right. So, this is the numerator and the denominator, I get  $\lambda_1^k \alpha_1^2 \mathbf{x}_1^T \mathbf{A} \mathbf{x}_1$  divided by  $\lambda_1^k \alpha_1^2 \mathbf{x}_1^T \mathbf{x}_1$ . So, these terms cancel out,  $\lambda_1$  to

$k$   $\lambda_1$  to  $k$  cancels out,  $\alpha_1$  square cancels out, so I eventually get this term, which is equal to  $\lambda_1$  and is known as Rayleigh's quotient is that clear. So,  $\mathbf{A} \mathbf{x}_1$  is nothing but  $\lambda_1 \mathbf{x}_1$ .

So,  $\lambda_1$  is  $\mathbf{x}_1^T \mathbf{A} \mathbf{x}_1 / \mathbf{x}_1^T \mathbf{x}_1$  let goes out  $\mathbf{x}_1^T \mathbf{x}_1$  cancels out, so I am left with  $\lambda_1$ , so this ratio, so the that if I start with an initial vector right. And I keep on iterating on, that initial vector I start with an initial arbitrary vector, I operate each time on that initial vector with my coefficient matrix  $\mathbf{A}$ , I get a series of  $\mathbf{z}_k$ 's right. Eventually, when I take enough number of  $\mathbf{z}_k$ 's, I am assured that, if I compute this product and this product and divide them together, I compute the Rayleigh's quotient, I am going to get my first my largest Eigen value, which is  $\lambda_1$ .

(Refer Slide Time: 09:39)


### Power method

- If  $|\lambda_1| < 1$  or  $|\lambda_1| > 1$ , (\*) may become zero or unbounded for large  $k$  and the  $\mathbf{z}_k$  may no longer tend to  $\mathbf{x}_1$
- In order to ensure that the scheme does not break down, it may be necessary to scale the sequence  $\mathbf{z}_k$  to ensure that its norm remains bounded

This is done by using a sequence  $\{\mathbf{y}_k\}$  with  $\|\mathbf{y}_k\| = 1$ :

$\mathbf{y}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|}$ ,  $\mathbf{z}_{k+1} = \mathbf{A} \mathbf{y}_k$ ,  $k = 0, 1, 2, \dots, n$ . Hence the Rayleigh's

quotient becomes  $\frac{\|\mathbf{z}_k\| (\mathbf{y}_k)^T \mathbf{A} \|\mathbf{z}_k\| \mathbf{y}_k}{\|\mathbf{z}_k\| (\mathbf{y}_k)^T \|\mathbf{z}_k\| \mathbf{y}_k} = \frac{(\mathbf{y}_k)^T \mathbf{A} \mathbf{y}_k}{(\mathbf{y}_k)^T \mathbf{y}_k}$



If mod of  $\lambda_1$  is less than 1 or mod of  $\lambda_1$  is greater than 1 star may become 0 or unbounded for large  $k$  and the  $\mathbf{z}_k$  may no longer tend to  $\mathbf{x}_1$ , why would this happen well.

(Refer Slide Time: 09:52)

### Power method


Then,  $\mathbf{z}_k = \alpha_1(\lambda_1)^k \mathbf{x}_1 + \alpha_2(\lambda_2)^k \mathbf{x}_2 + \dots + \alpha_n(\lambda_n)^k \mathbf{x}_n$

$$= (\lambda_1)^k \left( \alpha_1 \mathbf{x}_1 + \sum_{j=2}^n \left( \frac{\lambda_j}{\lambda_1} \right)^k \alpha_j \mathbf{x}_j \right), \quad k = 0, 1, \dots, n \quad (*)$$

Since  $\left| \frac{\lambda_j}{\lambda_1} \right| < 1 \quad \forall j \geq 2$  as  $k$  increases the direction of  $\mathbf{z}_k$  tends to that of  $\mathbf{x}_1$ , provided  $\alpha_1 \neq 0$

The ratio  $\frac{\mathbf{z}_k^T \mathbf{A} \mathbf{z}_k}{\mathbf{z}_k^T \mathbf{z}_k}$  thus tends to  $\frac{(\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{A} \mathbf{x}_1 (\lambda_1)^k}{(\lambda_1)^k (\lambda_1)^k (\alpha_1)^2 \mathbf{x}_1^T \mathbf{x}_1} = \lambda_1$

and is known as Rayleigh's quotient.



Let us look at this, if  $\lambda_1$  becomes smaller and smaller  $\lambda_1$  is less than 1, as  $k$  increases this term becomes smaller and smaller, so it may, so happen that this term may become almost 0 right. So, in that case  $\mathbf{z}_k$  may not, we are saying that  $\mathbf{z}_k$  tends to look from this, that  $\mathbf{z}_k$  tends to  $\mathbf{x}_1$ , but incase this term goes to 0  $\mathbf{z}_k$  would actually, tend to 0. Similarly, if  $\lambda_1$  is greater than 1, in that case  $\lambda_1$  to the power  $k$ , if  $k$  is sufficiently large would become very, very large, it will be it become infinite large for a sufficiently large number of iterations. In that case also  $\mathbf{z}_k$  would not be, it would not be tending to  $\mathbf{x}_1$ .

(Refer Slide Time: 10:36)


### Power method

- If  $|\lambda_1| < 1$  or  $|\lambda_1| > 1$ , (\*) may become zero or unbounded for large  $k$  and the  $\mathbf{z}_k$  may no longer tend to  $\mathbf{x}_1$
- In order to ensure that the scheme does not break down, it may be necessary to scale the sequence  $\mathbf{z}_k$  to ensure that its norm remains bounded

This is done by using a sequence  $\{\mathbf{y}_k\}$  with  $|\mathbf{y}_k| = 1$ :

$$\mathbf{y}_k = \frac{\mathbf{z}_k}{|\mathbf{z}_k|}, \quad \mathbf{z}_{k+1} = \mathbf{A} \mathbf{y}_k, \quad k = 0, 1, 2, \dots, n.$$

Hence the Rayleigh's quotient becomes

$$\frac{|\mathbf{z}_k| (\mathbf{y}_k)^T \mathbf{A} |\mathbf{z}_k| \mathbf{y}_k}{|\mathbf{z}_k| (\mathbf{y}_k)^T |\mathbf{z}_k| \mathbf{y}_k} = \frac{(\mathbf{y}_k)^T \mathbf{A} \mathbf{y}_k}{(\mathbf{y}_k)^T \mathbf{y}_k}$$



So, what do we do to solve that problem well in order to do, that we normalize in order to ensure, that the scheme does not break down. It may be necessary to scale the sequence  $z_k$  to ensure, that its norm remains bounded that is every time, we computed a new  $z_k$  we normalize it, we compute its norm divide it by divide each term in the vector by the norm and we get you need a vector with a unit norm right. So, this we get the sequence instead of working with the sequence  $z_k$ , we work with the sequence  $y_k$  where, the norm of  $y_k$  is equal to 1.

So, now my question is change, my iteration algorithm is slightly change, so  $z_{k+1}$  is no longer equal to  $A$  operating on  $z_k$ , it is  $A$  operating on  $y_k$  right. And therefore, the Rayleigh's  $k$  quotient also automatically changes, it becomes  $y_k^T A y_k / y_k^T y_k$ , it is clear from this expression that, this is going to be bounded why is that, because the norm of  $y_k$  is 1, norm of  $y_k$  is 1 a has finite norm, so this can never be 0 or never be infinity.

(Refer Slide Time: 11:49)

### Inverse power method

- If the power method is used on the matrix  $A^{-1}$  then in a similar way, we obtain the smallest eigen value of  $A$  instead of the largest eigen value, provided that  $|\lambda_n| < |\lambda_{n-1}|$
- This is because if  $A$  has eigen values  $\lambda_1 \geq \lambda_2 \dots \geq \lambda_{n-1} > \lambda_n$   
 $A^{-1}$  has eigen values  $1/\lambda_n > 1/\lambda_{n-1} \dots \geq 1/\lambda_1$
- More generally, if we form a sequence:
 
$$y_k = \frac{z_k}{\|z_k\|}, (A - \lambda^* I)z_{k+1} = y_k, k = 0, 1, \dots, n$$
 we get a power method with matrix  $(A - \lambda^* I)^{-1}$



Similarly, if instead of using the power method and my original matrix  $A$ , I use the power method on my matrix  $A$  inverse, then instead of getting the largest Eigen value and the largest Eigen vector, I will get the smallest Eigen value and the smallest Eigen vector. Why is that, because  $A$ , if  $A$  has Eigen values  $\lambda_1, \dots, \lambda_n$   $A$  inverse has Eigen values  $1/\lambda_1, \dots, 1/\lambda_n$ , that should be clear right. If you do a spectral decomposing of  $A$ , it becomes, we have get a diagonal


matrix, it becomes  $\lambda_1$ , through  $\lambda_n$  on the principle diagonals, if we invert that matrix to invert  $A$  diagonal matrix, then what we have is another diagonal matrix, but now the diagonal terms are  $1/\lambda_1$  through  $1/\lambda_n$ , so we are trying to find, so instead of using  $A$ , if you use  $A^{-1}$  then we are going to get the, we are going to get the Eigen value, which has the largest  $1/\lambda$  term right.

So, that means, that will be the smallest  $\lambda$  term smallest Eigen value of  $A$  right. So, more generally, if you form a sequence instead of using the matrix  $A$  or  $A^{-1}$ , if we use a matrix  $A - \lambda^* I$ , as my coefficient matrix then we get a power method with a matrix  $(A - \lambda^* I)^{-1}$  right. So, this is going to be my new coefficient matrix, in that case what am I going to get.

(Refer Slide Time: 13:36)

### Inverse power method

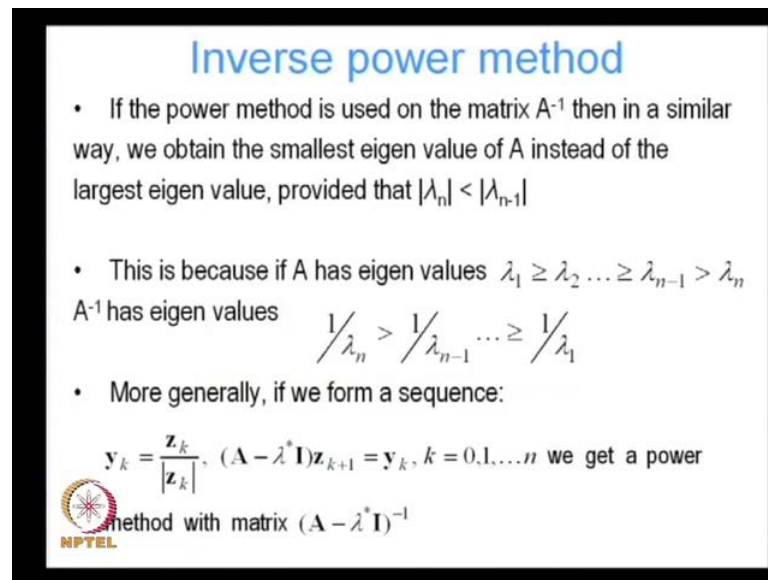
- This matrix has eigen values  $1/(\lambda_i - \lambda^*)$  and hence this inverse iteration will yield eigen values close to  $\lambda^*$  and the corresponding eigen vector.
- The fact that  $(A - \lambda^* I)$  is very near a singular matrix when  $\lambda$  is approximately equal to  $\lambda^*$  will not in general affect the accuracy of  $x_i$ , the corresponding eigen vector.



In this case, this matrix has Eigen values  $1/(\lambda_i - \lambda^*)$ , so it is going to be largest for that Eigen value, which is closest to  $\lambda^*$ . So, if I want, to if I am given a coefficient matrix  $A$  and I am giving some arbitrary number  $\lambda^*$  and I want to find the Eigen value of  $A$ , which is closest to  $\lambda^*$  and the Eigen vector of  $A$ , which is closest to  $\lambda^*$  then instead of using  $A$  or  $A^{-1}$ .




(Refer Slide Time: 14:09)



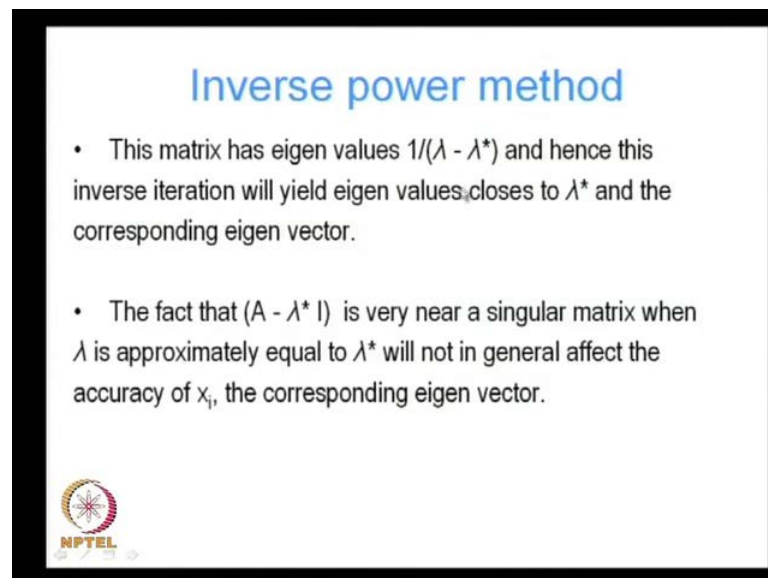
**Inverse power method**

- If the power method is used on the matrix  $A^{-1}$  then in a similar way, we obtain the smallest eigen value of  $A$  instead of the largest eigen value, provided that  $|\lambda_n| < |\lambda_{n-1}|$
- This is because if  $A$  has eigen values  $\lambda_1 \geq \lambda_2 \dots \geq \lambda_{n-1} > \lambda_n$ ,  $A^{-1}$  has eigen values  $\frac{1}{\lambda_n} > \frac{1}{\lambda_{n-1}} \dots \geq \frac{1}{\lambda_1}$
- More generally, if we form a sequence:  
$$\mathbf{y}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|}, \quad (A - \lambda^* I)\mathbf{z}_{k+1} = \mathbf{y}_k, \quad k = 0, 1, \dots, n$$
 we get a power method with matrix  $(A - \lambda^* I)^{-1}$

 NPTEL


I will iterate with  $A - \lambda^* I$  inverse, in which case, I am going to get my Eigen value then that case I am going to maximize this term right.

(Refer Slide Time: 14:19)



**Inverse power method**

- This matrix has eigen values  $1/(\lambda - \lambda^*)$  and hence this inverse iteration will yield eigen values, closes to  $\lambda^*$  and the corresponding eigen vector.
- The fact that  $(A - \lambda^* I)$  is very near a singular matrix when  $\lambda$  is approximately equal to  $\lambda^*$  will not in general affect the accuracy of  $x_i$ , the corresponding eigen vector.

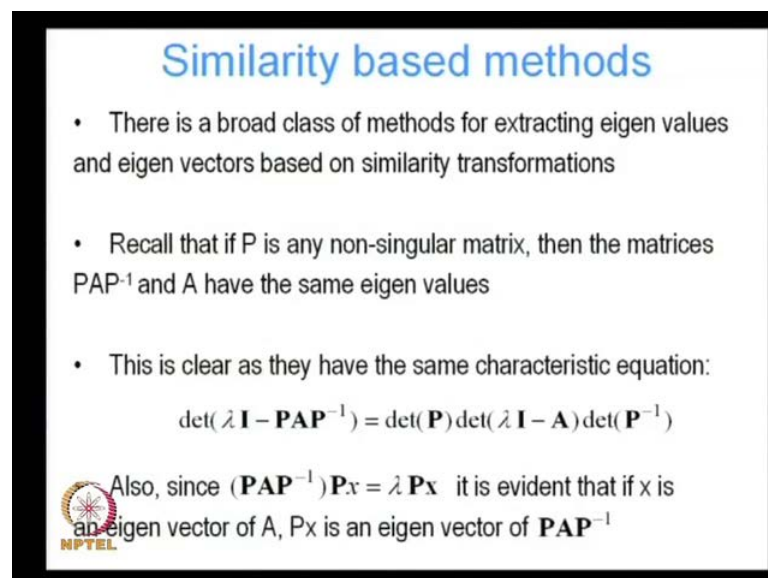
 NPTEL

And when do I maximize this term, when  $\lambda$  becomes as close as possible, when I get the Eigen value, which is as close as possible to my constant  $\lambda^*$ . So, I am going to get the Eigen value, which is closest to  $\lambda^*$  and the corresponding Eigen vector. So, the power method can be generalized, it can be generalized not only to compute the basically, we can use the power method not only to compute, the largest

Eigen value, the smallest Eigen value, we can use it to compute any Eigen value or Eigen vector right.

If, but the fact I might ask that well, when I when if we choose lambda star such that lambda star is very close to an Eigen value of A right. In that case, this term is going to become very large right. But, when the problem when the problem become singular well, that is true but it will not in general affect the accuracy of my corresponding Eigen vector, I am still going to get my corresponding Eigen vector corresponding to that, Eigen value with sufficient accuracy.


(Refer Slide Time: 15:34)



**Similarity based methods**

- There is a broad class of methods for extracting eigen values and eigen vectors based on similarity transformations
- Recall that if P is any non-singular matrix, then the matrices  $PAP^{-1}$  and A have the same eigen values
- This is clear as they have the same characteristic equation:  
$$\det(\lambda \mathbf{I} - \mathbf{PAP}^{-1}) = \det(\mathbf{P}) \det(\lambda \mathbf{I} - \mathbf{A}) \det(\mathbf{P}^{-1})$$

Also, since  $(\mathbf{PAP}^{-1})\mathbf{P}\mathbf{x} = \lambda \mathbf{P}\mathbf{x}$  it is evident that if x is an eigen vector of A,  $\mathbf{P}\mathbf{x}$  is an eigen vector of  $\mathbf{PAP}^{-1}$



So, that was the power method, there are another class of methods for computing Eigen values and Eigen vectors, which are based on similarity transformations. So, what are similarity transformations well, similarity transformations are typically transformation of this form where, we operate on a matrix A with another matrix P and its inverse P inverse. So, there is a broad class of methods for extracting Eigen values and Eigen vectors, which are based on similarity transformations, in our last lecture, we solved we saw that, if P is a non singular matrix, then the matrices P A P inverse and A have the same Eigen values, we proved that last time. So, this can be shown again, in it is easy enough to show.

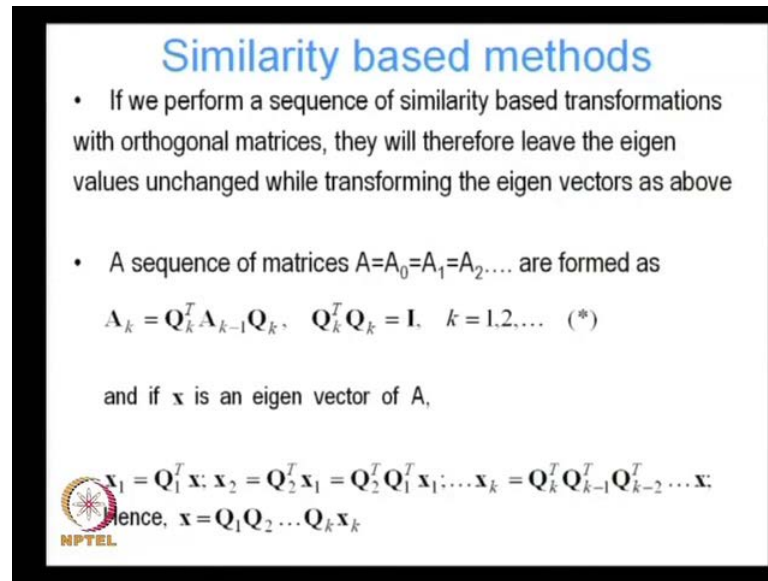
So, we can this is clear, if you look at the characteristic equation of P A P inverse and A, what is the characteristic equation of P A P inverse, it is determinant of lambda, I minus

$P^{-1}AP$  inverse, what is the characteristic equation of  $A$ , it is determinant of  $\lambda I - A$  minus  $A$ , why is  $\lambda$  the same, because they have the same Eigen values right. So, I can write determinant of  $\lambda I - P^{-1}AP$  as basically, determinant of  $\lambda I - P^{-1}AP$  inverse minus  $P^{-1}AP$  inverse  $P^{-1}P$  inverse being  $I$ .

So, in that case, I can write this as determinant of  $\lambda I - P^{-1}AP$ . So, if I take out  $P^{-1}P$  inverse minus  $P^{-1}AP$  inverse, I take out  $P$  outside, so I get  $P^{-1}(\lambda I - A)P$  inverse minus  $A^{-1}P$  inverse, I take out  $P^{-1}$  to the left, so I get determinant of  $P$  times determinant of  $\lambda I - A$  times determinant of  $P^{-1}$  right. So, just because taking into account the idea that, the product determinant of the product of 2 matrices is the product of its determinants, determinant of  $a \cdot b$  is equal to determinant of  $a$ , times determinant of  $b$  right.

So, and we also know that determinant of  $P$  determinant of  $P^{-1}$  is equal to 1 by determinant of  $P^{-1}P$  right. So, determinant of  $A$  is equal to 1 by determinant of  $A^{-1}$ , so these cancel out, this determinant of  $P$  and determinant of  $P^{-1}$  cancel out and I get determinant of  $\lambda I - A$ , which means that  $P^{-1}AP$  and  $A$  have the same characteristic equation. And we know that the Eigen values are nothing but the roots of the characteristic equations, so the Eigen values have to be the same. Also since,  $P^{-1}AP$  inverse operating on  $P^{-1}x$  is equal to  $\lambda P^{-1}x$  right, again this should be obvious  $P^{-1}P$  is  $I$ . So, we have  $P^{-1}AP$  operating on  $\lambda P^{-1}x$ , but again  $A^{-1}x$  is nothing but  $\lambda x$ . So, this means that,  $P^{-1}x$  is an Eigen vector of  $P^{-1}AP$  inverse right,  $P^{-1}x$  is an Eigen vector of  $P^{-1}AP$  inverse.

(Refer Slide Time: 19:01)




**Similarity based methods**

- If we perform a sequence of similarity based transformations with orthogonal matrices, they will therefore leave the eigen values unchanged while transforming the eigen vectors as above
- A sequence of matrices  $A=A_0=A_1=A_2\dots$  are formed as
$$A_k = Q_k^T A_{k-1} Q_k, \quad Q_k^T Q_k = I, \quad k = 1, 2, \dots \quad (*)$$

and if  $x$  is an eigen vector of  $A$ ,

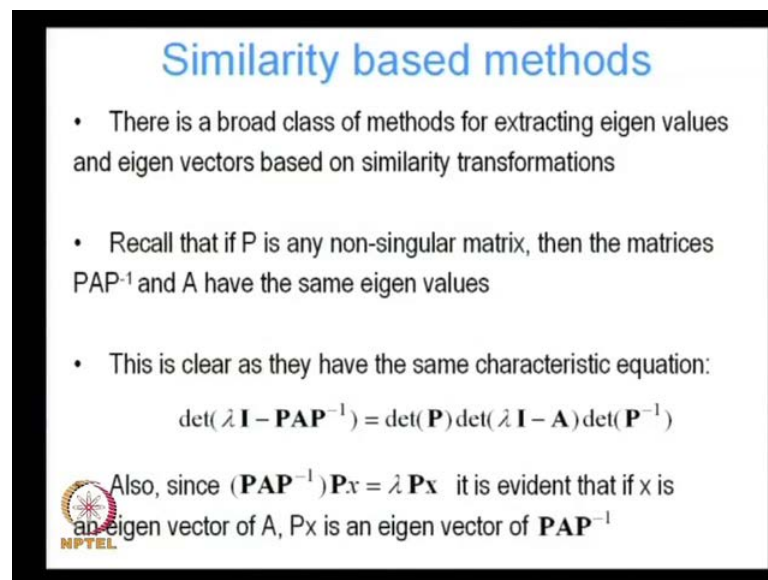
$$x_1 = Q_1^T x; \quad x_2 = Q_2^T x_1 = Q_2^T Q_1^T x; \dots x_k = Q_k^T Q_{k-1}^T Q_{k-2}^T \dots x;$$

Hence,  $x = Q_1 Q_2 \dots Q_k x_k$



So, we in similarity, base methods for extracting Eigen values, we perform a series of similarity based transformations with orthogonal matrices, that will therefore, leave the Eigen values unchanged by transforming the Eigen vectors as above.


(Refer Slide Time: 19:20)



**Similarity based methods**

- There is a broad class of methods for extracting eigen values and eigen vectors based on similarity transformations
- Recall that if  $P$  is any non-singular matrix, then the matrices  $PAP^{-1}$  and  $A$  have the same eigen values
- This is clear as they have the same characteristic equation:
$$\det(\lambda I - PAP^{-1}) = \det(P) \det(\lambda I - A) \det(P^{-1})$$

Also, since  $(PAP^{-1})Px = \lambda Px$  it is evident that if  $x$  is an eigen vector of  $A$ ,  $Px$  is an eigen vector of  $PAP^{-1}$



We know that, if  $P$  is any non singular matrix, then it leaves the Eigen values unchanged and it transforms the Eigen vectors like this right. Now, we are saying that, instead of using any non singular matrix, we are going to use only orthogonal matrices right. We are only going to use orthogonal matrices.

(Refer Slide Time: 19:43)

### Similarity based methods


- If we perform a sequence of similarity based transformations with orthogonal matrices, they will therefore leave the eigen values unchanged while transforming the eigen vectors as above
- A sequence of matrices  $A=A_0=A_1=A_2\dots$  are formed as

$$A_k = Q_k^T A_{k-1} Q_k, \quad Q_k^T Q_k = I, \quad k = 1, 2, \dots \quad (*)$$

and if  $x$  is an eigen vector of  $A$ ,

$$x_1 = Q_1^T x; \quad x_2 = Q_2^T x_1 = Q_2^T Q_1^T x; \dots x_k = Q_k^T Q_{k-1}^T Q_{k-2}^T \dots x;$$

Hence,  $x = Q_1 Q_2 \dots Q_k x_k$



And orthogonal matrices will also therefore, leave the Eigen values unchanged while transforming the Eigen vectors as above right, assumption is that, I am using an orthogonal matrix with it is non singular right. Orthogonal matrix always is non singular its determinant can only be, if its ortho-normal, its determinant is plus or minus 1 right. So, sequence of matrices, what we are doing is that, we are repeatedly operating on a using  $A$ , similarity transformation, but  $A$  transformation matrix is an orthogonal matrix right.

So, we start, so we obtain a sequence of matrices starting with  $A_0$  equal to  $A$ , we operate on  $A_0$  with  $A$  similarity transformation using  $Q$ , we get  $A_1$ , we again operating on  $A_1$  with similarity transformation  $Q$ , we get  $A_2$  and so on and so forth. So,  $A_k$  is nothing but  $Q_k^T A_{k-1} Q_k$  by  $Q_k^T Q_k = I$  right, it is an orthogonal matrix,  $k$  is equal to 1 to 2 and if  $x$  is an Eigen vector of  $A$ .


So, the only criteria is that, this transformation matrix be a orthogonal matrix right, at each step we can use, if we want a different orthogonal matrix, but we are using a orthogonal matrix for each similarity transformation. So,  $x_1$  is nothing but  $Q_1^T x$ , why is that, because I knew that, my Eigen vectors are  $P$  times  $x$  right. So,  $Q_1^T x$  is equal to  $Q_2^T Q_1^T x$ , similarity transformation means, that my Eigen vectors are the original Eigen vectors times, the transformation matrix right. What is the matrix here here, it is  $Q Q^T$ , why is that.

(Refer Slide Time: 21:51)

### Similarity based methods

- There is a broad class of methods for extracting eigen values and eigen vectors based on similarity transformations
- Recall that if P is any non-singular matrix, then the matrices  $PAP^{-1}$  and A have the same eigen values
- This is clear as they have the same characteristic equation:
 
$$\det(\lambda \mathbf{I} - \mathbf{PAP}^{-1}) = \det(\mathbf{P}) \det(\lambda \mathbf{I} - \mathbf{A}) \det(\mathbf{P}^{-1})$$

Also, since  $(\mathbf{PAP}^{-1})\mathbf{P}\mathbf{x} = \lambda \mathbf{P}\mathbf{x}$  it is evident that if  $\mathbf{x}$  is an eigen vector of A,  $\mathbf{P}\mathbf{x}$  is an eigen vector of  $\mathbf{PAP}^{-1}$



Because, my similarity transformation is of the form P A P inverse right, in case of an orthogonal matrix, it is P A P transpose right. So, P A P transpose instead of P I am using Q transpose right.

(Refer Slide Time: 22:06)


### Similarity based methods

- If we perform a sequence of similarity based transformations with orthogonal matrices, they will therefore leave the eigen values unchanged while transforming the eigen vectors as above
- A sequence of matrices  $A=A_0=A_1=A_2\dots$  are formed as
 
$$A_k = Q_k^T A_{k-1} Q_k, \quad Q_k^T Q_k = \mathbf{I}, \quad k = 1, 2, \dots \quad (*)$$

and if  $\mathbf{x}$  is an eigen vector of A,

$$\mathbf{x}_1 = Q_1^T \mathbf{x}; \mathbf{x}_2 = Q_2^T \mathbf{x}_1 = Q_2^T Q_1^T \mathbf{x}; \dots \mathbf{x}_k = Q_k^T Q_{k-1}^T \dots Q_1^T \mathbf{x};$$

hence,  $\mathbf{x} = Q_1 Q_2 \dots Q_k \mathbf{x}_k$



So, instead of my Eigen vectors are going to be transpose transformed as Q transpose operating on x right. So,  $\mathbf{x}_1$  is equal to  $Q_1^T \mathbf{x}$ ,  $\mathbf{x}_2$  is equal to  $Q_2^T \mathbf{x}_1$ , which I, if I expand it out, I am going finally, going to get  $Q_k^T Q_{k-1}^T \dots Q_1^T \mathbf{x}$ . So, after k similarity

transformation my original Eigen vector  $x$  is going to be transformed to this. So, in that case, I can just by doing a transpose of both sides, I get  $x$  is equal to  $Q_1 Q_2 \dots Q_k$  operating on  $x_k$ .

(Refer Slide Time: 22:58)

### Similarity based methods

- The sequence of transformations (\*) also preserve symmetry since  $A_k^T = (Q_k^T A_{k-1} Q_k)^T = Q_k^T A_{k-1}^T Q_k, \quad k = 1, \dots, n$
- Thus if the initial matrix  $A_0 = A$  is symmetric, the sequence of similarity transformations will yield a symmetric matrix
- The orthogonal transformations used in practice are mainly of two types – plane rotations and reflections
- The aim of both types of transformations is reduce the initial matrix to as close to a diagonal form as possible – with the Eigen values occupying the diagonal locations

The sequence similarity, transformations also by definition performs symmetry why because we can show this  $A_k$  transpose is equal to  $Q_k$  transpose  $A_{k-1}$   $Q_k$  transpose, which is equal to  $Q_k$  transpose  $A_{k-1}$   $Q_k$ . Thus if the initial matrix  $A_0$  is symmetric, the sequence of similarity transformations will yield a symmetric matrix, so as we do our transformation, if we start with a symmetric matrix, we are going to get symmetric matrices. The orthogonal transformations used in practice are mainly of 2 types plane rotations and reflections.

So, usually what people do, in as is a very well known method, which we are going to talk about later on 1 is Jacobi's method, which basically just uses plane rotations, rather transformation techniques, which use reflections. So, we operate, so our transformation orthogonal transformation matrix is usually a rotation or reflection. So, we repeatedly operate on my original matrix  $A$  with repeated rotations or repeated reflections until I get a diagonal system, which has the Eigen values on its diagonal principle diagonals. So, the orthogonal transformations used in practice are mainly of 2 types plane rotations and plane reflections, the aim of both types of transformations is to reduce the initial matrix

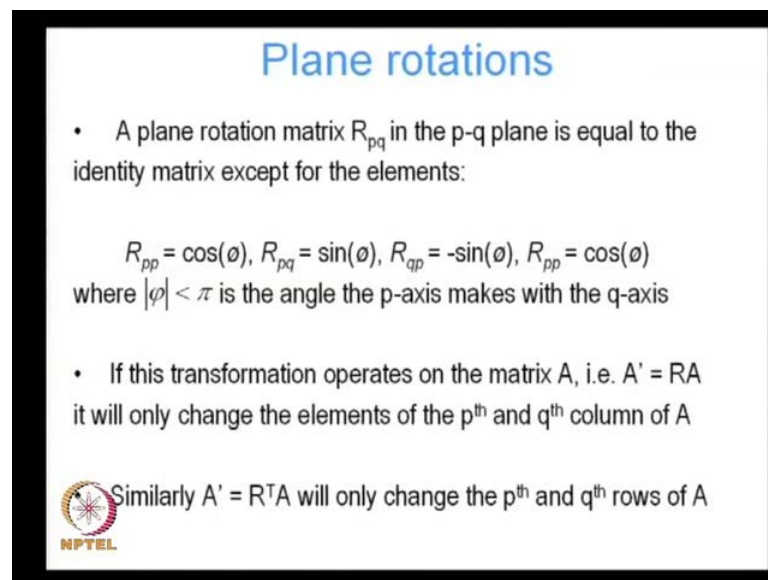


to as close to a diagonal form as possible with the Eigen values occupying the diagonal locations.

You can think of it like this, you know that when you have a symmetric matrix and you have linearly independent Eigen vectors, you can transform the symmetric matrix to a diagonal matrix by operating on the symmetric matrix with a matrix whose columns are my Eigen vectors right. So, if I have a matrix A and I which we looked at last time right and we have another matrix x an each column of x corresponds to an Eigen vector, if I do  $x^T A x$  eventually end up with a diagonal matrix, whose principle diagonal elements are my Eigen values right.

Here we do not know up priory, what are my Eigen vectors right, so I am by using this repeated, similarity transformations I am trying to get as close as I can to my Eigen vector matrix right. And then eventually, I can going to get a diagonal matrix, which is going to contain all the principle values, I mean Eigen values and its principle diagonals right.


(Refer Slide Time: 26:01)



**Plane rotations**

- A plane rotation matrix  $R_{pq}$  in the p-q plane is equal to the identity matrix except for the elements:  
$$R_{pp} = \cos(\theta), R_{pq} = \sin(\theta), R_{qp} = -\sin(\theta), R_{qq} = \cos(\theta)$$
where  $|\theta| < \pi$  is the angle the p-axis makes with the q-axis
- If this transformation operates on the matrix A, i.e.  $A' = RA$  it will only change the elements of the p<sup>th</sup> and q<sup>th</sup> column of A

Similarly  $A' = R^T A$  will only change the p<sup>th</sup> and q<sup>th</sup> rows of A



So, what is it, so I said that, we repeatedly operate on this matrix using orthogonal transformations, using plane rotations and plane and reflections. So, what is a plane rotation matrix, A plane rotation matrix is a matrix in the p q plane, which is equal to the identity matrix except for the elements on the pth row and qth pth row pth column and qth row and qth column. So, only 2 only, it is an identity matrix except for the elements



$R_{p,p}$ . So, the intersection of the  $p$  row and the  $q$  column and the  $p$  row and the  $p$  column right, only those terms are going to change.

The rest of my terms are going to remain 1 and 0 like my usual identity matrix. So, only terms, which are going to be different are  $R_{p,p}$ ,  $R_{p,q}$ ,  $R_{q,p}$  and  $R_{q,q}$  and these will be like a plane rotation right, which you know A plane rotation matrix is like a matrix  $\cos \phi$ ,  $\sin \phi$ ,  $-\sin \phi$ ,  $\cos \phi$  right. So, in this case, we have that on a  $p$  intersection of my  $p$ th row and  $p$  and  $q$  columns, similarly  $q$ th row and  $p$  and  $q$  columns, so those are and then we have control over the rotation angle  $\phi$  right, which is less than  $\pi/2$ .

So, the idea is to give A plane rotation to choose these  $\phi$ 's to choose this choose these  $\phi$ 's meaning choose this rotation matrix, such that my it rise to annihilate the off-diagonal terms of my original matrix, what do I, what I have an original matrix right, which has got off-diagonal terms, I want to get to a diagonal form when my diagonal elements occupy the my when my Eigen values occupy the diagonal elements.

So, what I what do I want to do, I want to annihilate my off-diagonal terms, I want to make them 0. So, I can choose my  $\phi$ 's, I am going to choose my  $\phi$ 's in these each iteration, I am going to choose my  $\phi$  with a target to annihilate a particular off-diagonal term and how am I going to choose, which off-diagonal term to annihilate well, the natural choice is I will try to choose the off-diagonal term, which is largest right.

The largest off- diagonal term, I will try to annihilate in it, at any iteration  $k$  I will choose my  $R$  matrix  $R_k$ , such that it will try to make the largest off-diagonal term 0 and if I do this sufficient number of times, it turns out, that I can in most cases, I can annihilate all my off- diagonal terms and I left, I am left with a diagonal matrix. And since I know that my the similarity transformation leaves the Eigen values unchanged, those Eigen values of a diagonal matrix, the diagonal matrix has the Eigen values and its diagonal and those are those Eigen values must be the Eigen values of my original matrix right.

So, if this transformation operates on the matrix  $A$ , so we it only changes the elements of the  $p$ th and  $q$ th, so if I form  $A$  matrix like this, with only the  $p$ ,  $R_{p,p}$ ,  $R_{p,q}$ ,  $R_{q,p}$  and  $R_{q,q}$  different from 1 and 0, then it is going to only change the elements on the  $p$ th and  $q$ th column of  $A$ . Similarly if you operate on  $A$  with  $R$  transpose, if I form  $R$  like this, but then instead of operating on  $A$  with  $R$ , I operate an  $A$  with  $R$  transpose, I am only going

to change the pth and qth rows of A right. So, I can change the pth and qth rows or the pth and qth columns.

(Refer Slide Time: 30:01)

### Reflections


- Reflections use transformation matrices of the form:

$$P(\mathbf{w}) = \mathbf{I} - 2\mathbf{w} \otimes \mathbf{w}, \quad \mathbf{w} \cdot \mathbf{w} = 1$$

This matrix is orthogonal since

$$\begin{aligned} (\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w})^T (\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w}) &= (\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w})(\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w}) \\ &= \mathbf{I} - 4\mathbf{w} \otimes \mathbf{w} + 4(\mathbf{w} \otimes \mathbf{w})(\mathbf{w} \otimes \mathbf{w}) = \mathbf{I} - 4\mathbf{w} \otimes \mathbf{w} + 4\mathbf{w} \otimes \mathbf{w}(\mathbf{w} \cdot \mathbf{w}) \\ &= \mathbf{I} \end{aligned}$$

The transformation  $P(\mathbf{w})\mathbf{A} = \mathbf{A} - 2(\mathbf{w} \otimes \mathbf{w})\mathbf{A}$  and reflects the matrix  $\mathbf{A}$  in the hyperplane through the origin orthogonal to  $\mathbf{w}$



So, that was about plane rotations, what about reflections well for reflections the transformation matrix has this form, it is of the form  $P$   $P$  depends on a particular vector  $w$  and the  $P$  of  $w$  is  $I$  minus  $2$ ,  $w$  vector product  $w$  right, returns a product right. So, it is basically the outer product, it is when you take a dot product of  $2$  vectors, you get a scalar right. But, this is not a dot product, this is a vector product sorry, it is not a vector product it is a tensor product right, it is a tensor product.

So, basically, if you think of it like this, if you have a vector, if you write a vector in row form then you have write a vector in column form, if you take the product of the row vector with the column vector, you end up with a scalar right. That is your dot product, but instead of that, if you take the vector in the column form and then you take the product with the another vector in the row from then you end up with a matrix right.

So, it is changing the order right. So, it is going from  $1$  order to  $2$  orders, so it is called a tensor product right, it increases the order from vector we go to matrix right. So, omega  $w$ , so we are saying that, if we construct  $P$  to be  $I$  minus  $2$   $w$  tensor product  $w$ , at if I get tensor, it is a outer product  $2$   $I$  minus  $I$  minus  $2$   $w$  outer product  $w$  with the condition, that  $w$  is a vector of unit norm, that is  $w$  dotted with  $w$  is equal to  $1$ .

This matrix is an orthogonal matrix, why is this matrix an orthogonal matrix, we can show that, if I take the transpose of this matrix and take the product of that matrix with its with its transpose, I am going to get an I am going to get an identity matrix, well you can show that, I am not going too many details of that, it is obvious right. So, you can just see, if you take this, so first term is going to be I, the second term is going to be, so this operating on this is going to be  $4 w$  tensor outer product  $w$ , operating on  $w$ , outer product  $w$  and this is going to be minus  $2 w$  tensor outer product  $w$  operating on I.

This is going to be another minus  $2 w$  tensor outer product, so this is going to be my this part and this part, using the operation of using the rules for our tensor products, we can write  $w$  tensor product  $w$ , operating on  $w$  tensor product  $w$  as  $w$  tensor product  $w$  times  $w$  that  $w$  scalar product  $w$ , since  $w$  scalar product  $w$  is equal to 1. So, we have again this 2 terms are going to cancel out and we are going to get I.

So, the transformation matrix is orthogonal, if we construct it like this, if we take any vector  $w$  of unit norm and we construct a matrix by  $I$  minus  $2 w$  tensor product  $w$ , we are going to end up with a orthogonal matrix, it also we can, we will go we have later, that is also leaves a symmetric matrix symmetric right. So, it preserves symmetry, but what it does is that when this operates on any matrix  $A$ , it basically reflects the matrix  $A$  in the hyper plane through the origin, which is orthogonal to  $w$ .

So, we consider in that space in that  $n$  dimensional space, since we are considering an  $n$  dimensional matrix as space is  $n$  dimensional, in that  $n$  dimensional space, I have an  $n$  dimensional vector  $w$ , which is suppose passing through the origin. And I have a plane, which is orthogonal to that vector  $w$ , in that  $n$  dimensional space and what this transformation does any is that, it reflects the matrix  $A$  in the hyper plane through the origin orthogonal to  $w$ . So, that is why it is called a reflection matrix right. You can think of it like this minus  $w$  tensor product  $w$  operating on  $2$ , so it takes out the projection, so you can see from here, it takes out the part of  $A$ , in the  $w$  direction twice, that is why it gets reflected right. It gets reflected about the hyper plane, through the hyper plane passing, through  $A$  with orthogonal  $w$ .

(Refer Slide Time: 34:36)

## Reflections


- $P(\mathbf{w})$  is also symmetric since:

$$(\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w})^T = (\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w})$$

The transformation  $\mathbf{A}' = P(\mathbf{w})\mathbf{A}$  thus transforms each column of  $\mathbf{A}$  independently according to

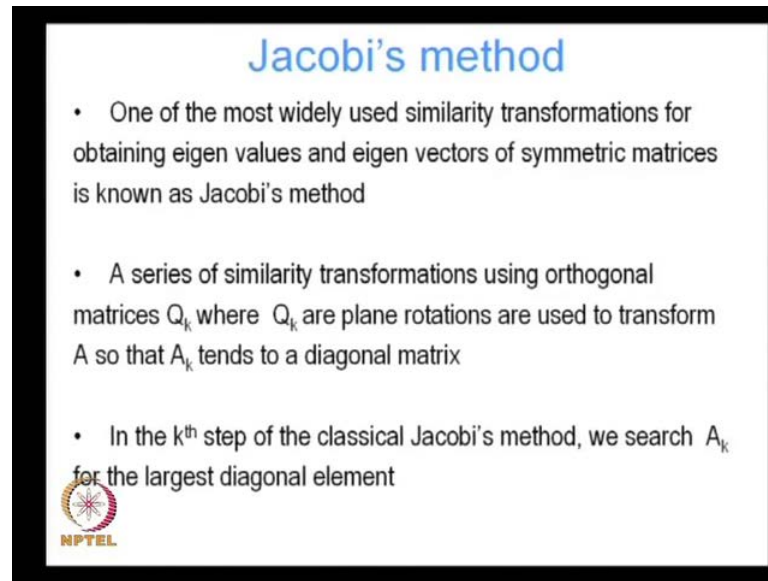
$$\mathbf{a}'_k = (\mathbf{I} - 2\mathbf{w} \otimes \mathbf{w})\mathbf{a}_k = \mathbf{a}_k - 2(\mathbf{w} \cdot \mathbf{a}_k)\mathbf{a}_k$$

Similarly  $\mathbf{A}' = \mathbf{A}P(\mathbf{w})$  transforms each row of  $\mathbf{A}$




$P(\mathbf{w})$  is also symmetric well, that can be shown trivially again using the properties of outer product, the transformation  $\mathbf{A}' = P(\mathbf{w})\mathbf{A}$ , thus transforms each column of  $\mathbf{A}$  independently according to this. So,  $\mathbf{a}'_k$  is nothing but  $P$  operating on  $\mathbf{a}_k$ . And again using the transformation rules for outer products separating on vectors, we can show that, this is equal to this. Similarly, each similarly, if I instead of using  $P$  operating on  $\mathbf{A}$ , if I use  $\mathbf{A}' = \mathbf{A}P(\mathbf{w})$  instead of operating on  $\mathbf{A}$  with  $P$  on from the left hand side, you operate on  $\mathbf{A}$  on I operate on  $\mathbf{A}$  with  $P$  from the right hand side, I transform each row of  $\mathbf{A}$  right.

(Refer Slide Time: 35:38)



**Jacobi's method**

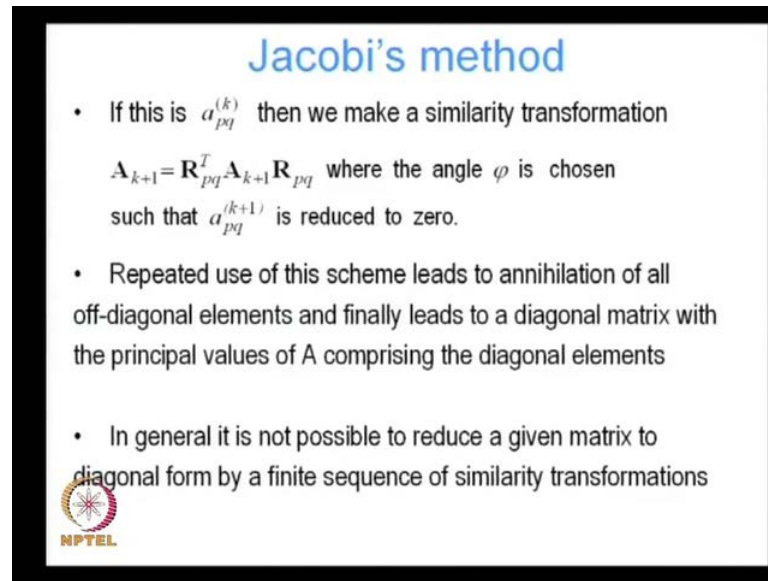
- One of the most widely used similarity transformations for obtaining eigen values and eigen vectors of symmetric matrices is known as Jacobi's method
- A series of similarity transformations using orthogonal matrices  $Q_k$  where  $Q_k$  are plane rotations are used to transform  $A$  so that  $A_k$  tends to a diagonal matrix
- In the  $k^{\text{th}}$  step of the classical Jacobi's method, we search  $A_k$  for the largest diagonal element

 NPTEL

So, one of the most widely used, similarity transformations for obtaining Eigen values and Eigen vectors of symmetric matrices is known as Jacobi's method right, as I mentioned a little while earlier. So, Jacobi's method, we do a series of similarity transformation using orthogonal matrices  $Q_k$  where,  $Q_k$  are plane rotations and they are used to transform  $A$ , so that it finally, becomes a diagonal matrix.


In the  $k^{\text{th}}$  step of the classical Jacobi's method, we search  $A_k$  for the largest, I am sorry, this is  $A$  type, this should be the largest of diagonal element right. You search for the largest off-diagonal element and then I construct my  $P$  matrix taking care to choose my angle  $\phi$ , in such a manner that, my largest diagonal element becomes 0 right. I annihilate my largest off- diagonal element becomes 0.

(Refer Slide Time: 36:39)



**Jacobi's method**

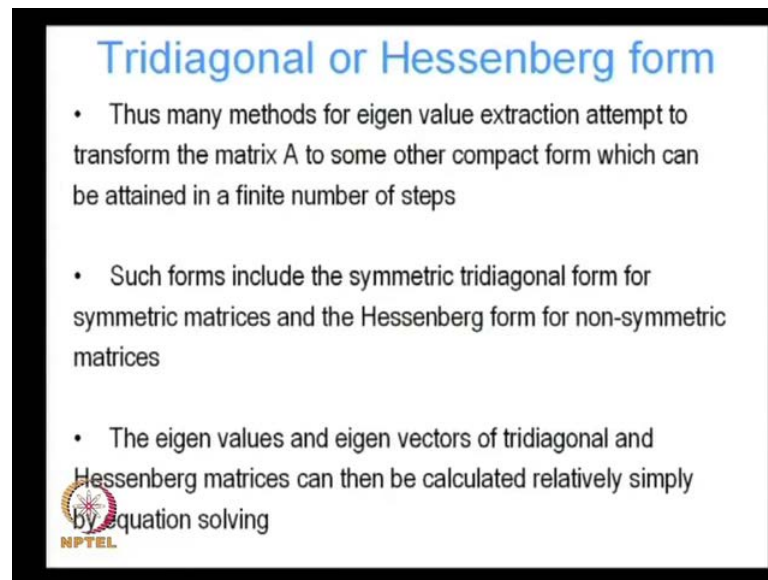
- If this is  $a_{pq}^{(k)}$  then we make a similarity transformation  $\mathbf{A}_{k+1} = \mathbf{R}_{pq}^T \mathbf{A}_k \mathbf{R}_{pq}$  where the angle  $\varphi$  is chosen such that  $a_{pq}^{(k+1)}$  is reduced to zero.
- Repeated use of this scheme leads to annihilation of all off-diagonal elements and finally leads to a diagonal matrix with the principal values of  $\mathbf{A}$  comprising the diagonal elements
- In general it is not possible to reduce a given matrix to diagonal form by a finite sequence of similarity transformations



If this largest off-diagonal element is a  $p$   $q$   $k$  then we make a similarity transformation  $\mathbf{A}_{k+1}$  is equal to  $\mathbf{R}_{pq}^T \mathbf{A}_k \mathbf{R}_{pq}$  where,  $\mathbf{R}$  is a planer rotation right, which transforms just the  $p$  and  $q$  columns of  $\mathbf{A}$  right. Where angle  $\phi$  is chosen, such that  $\mathbf{A}_{pq}^{k+1}$  is reduced to 0. So, repeated use of this scheme leads to annihilation of all off-diagonal elements and finally, we get a diagonal matrix with the principle values of  $\mathbf{A}$  comprising the Eigen values of  $\mathbf{A}$  comprising diagonal elements.


And since diagonal elements have to have the Eigen values, those are the Eigen values of  $\mathbf{A}$  right, because I knew that my similarity transformations leave my Eigen values unchanged right. Sometimes it is not possible it is not generally true that by doing a finite sequence of similarity transformations, we are going to get a diagonal matrix, it does not work out that way, it does not work on all the time.

(Refer Slide Time: 37:57)



**Tridiagonal or Hessenberg form**

- Thus many methods for eigen value extraction attempt to transform the matrix  $A$  to some other compact form which can be attained in a finite number of steps
- Such forms include the symmetric tridiagonal form for symmetric matrices and the Hessenberg form for non-symmetric matrices
- The eigen values and eigen vectors of tridiagonal and Hessenberg matrices can then be calculated relatively simply by equation solving

 NPTEL

So, in that case, what is the second best option, well in that case instead of trying to get a diagonal form, we try to reduce it, as simpler form as possible right. And when using, that simpler matrix, we construct out characteristic polynomial right, characteristic polynomial and then try to solve the characteristic polynomial numerically right. So, we know that, if we direct solve the characteristic polynomial of a large matrix that lots of components and we are susceptibility will conditioning right.

We saw that last time, because the characteristic polynomial is susceptible is very sensitive right, minor perturbations in the elements of a can lead to large changes in the Eigen values right. If you try to solve obtain the Eigen values after solving the characteristic polynomial, so for you for a full matrix, it is not a good idea, we saw to use the characteristic polynomial approach and try to find the roots of that polynomial.

However, if we reduce, that full matrix to sufficiently compact form to as close the diagonal form is possible, then in that case, the characteristic polynomial is not as well as susceptible to ilposeness right, it is not as imposed right. So, in that case, we can go ahead and solve the characteristic polynomial to find the Eigen values which in case, we cannot reduce a to diagonal form just by performing similarity transformations.

So, what are some of these forms, we try to reduce transform the matrix  $A$  to some other compact form, which can be obtained in a finite number of steps, such forms include the symmetric tri-diagonal form for symmetric matrices and the Hessenberg form for non-

symmetric matrices. Then once we have obtained the symmetric tri-diagonal form or the Hessenberg form you go ahead and write the characteristic polynomial for those simple matrices and solve them to obtain Eigen.

(Refer Slide Time: 40:08)


**Tridiagonal or Hessenberg form**

Symmetric tridiagonal matrix :

$$\begin{bmatrix} a_1 & b_2 & & & \\ b_2 & a_2 & b_3 & & \\ & b_3 & \dots & & \\ & & \dots & & \\ & & & \dots & b_n \\ & & & b_n & a_n \end{bmatrix}$$

Hessenberg form :

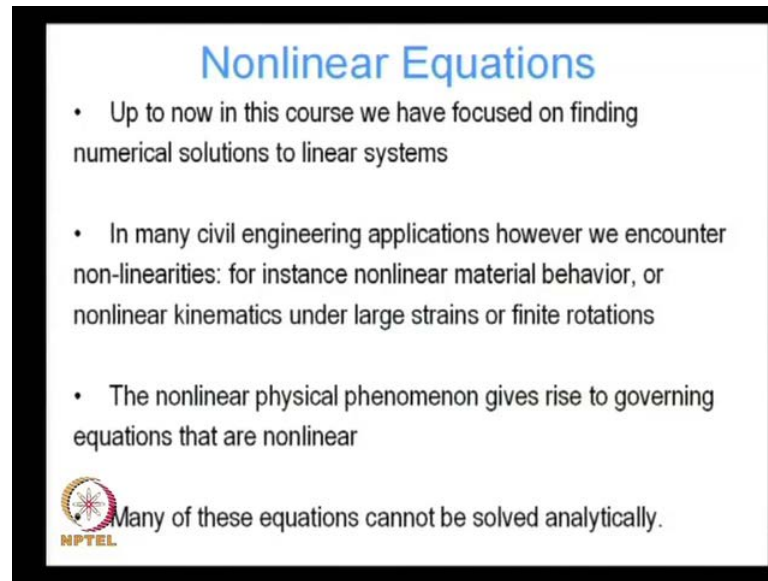
$$\begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ & h_{32} & \ddots & \vdots \\ & & \dots & \\ & & & h_{n-1} & h_n \end{bmatrix}$$



So, what are symmetric tri-diagonal matrix is of this form, so you can see it is pretty close to a diagonal matrix, however, it is not exactly a diagonal matrix. So, doing transformations, we can reduce any symmetric matrix to symmetric tri-diagonal matrix symmetric matrix, non-symmetric matrix to Hessenberg form and then we can use the try to write down the characteristic for, equation for this characteristic polynomial for this and try to find the roots of that equation.




(Refer Slide Time: 40:43)



**Nonlinear Equations**

- Up to now in this course we have focused on finding numerical solutions to linear systems
- In many civil engineering applications however we encounter non-linearities: for instance nonlinear material behavior, or nonlinear kinematics under large strains or finite rotations
- The nonlinear physical phenomenon gives rise to governing equations that are nonlinear

 Many of these equations cannot be solved analytically.

So, now, we are going to change track and try something else, so up to now, we have focused on solution of linear systems right. We have looked at a variety of ways to solve linear systems, we have looked at convergence, we have looked at airway propagation, we have try to find out, which are the best possible ways, most efficient ways of solving a linear system.

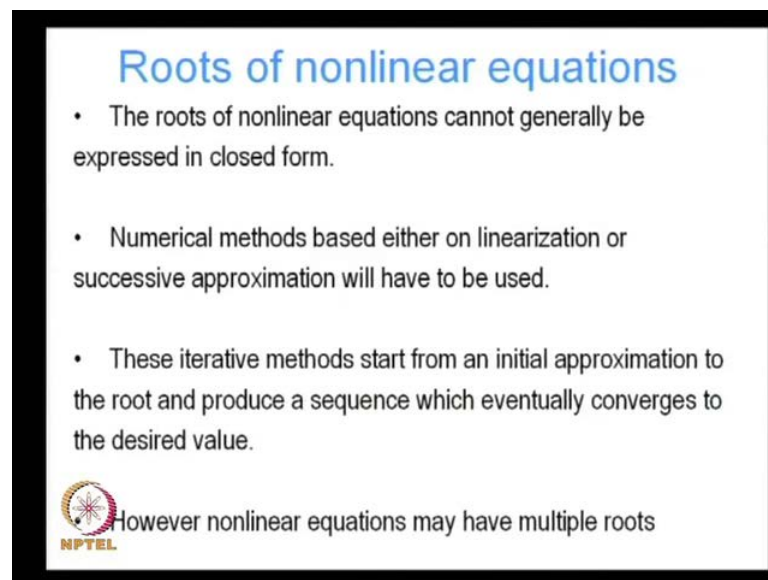
So, now we are going to change switch over to non-linear systems, in many civil engineering applications when you come to nonlinearities for instance. We have non-linear material behavior, whether in solid or fluid mechanics, we encounter non-linear material behavior or for finite strains or finite rotations, we encounter non-linear kinematics right. So, in that case the non-linear physical phenomenon gives rise to governing equations, which are non-linear right.

if we have a matrix  $a$  to in a simplest gives a simplest possible example, if we have a solving  $a x$  equal to  $b$  where, the coefficient matrix  $A$  is a function of the solution right. So,  $a$  is a function of  $x$ ,  $a$  depends on  $x$ , right and we are trying to solve  $a x$  equal to  $b$ . So, every time the solution change a solution changes my coefficient matrix is also going to change right.

Every time  $x$  changes my coefficient matrix is going to change, so it is no longer a linear system, it is a non-linear system. So, how do we solve those systems, well many of these equations cannot be solved analytically, so we have to solve most interesting problems, I


mean it cannot be solved analytically, unless there is simplest possible problems to the simplest possible nonlinearities and the simplest possible geometries and things like that. Simplest possible material properties and things like that, we cannot solve them analytically. So, we have to solve them numerically.

(Refer Slide Time: 42:47)



**Roots of nonlinear equations**

- The roots of nonlinear equations cannot generally be expressed in closed form.
- Numerical methods based either on linearization or successive approximation will have to be used.
- These iterative methods start from an initial approximation to the root and produce a sequence which eventually converges to the desired value.

 However nonlinear equations may have multiple roots


And if you have to solve the numerically, we have to find the roots of non-linear equations right and the roots of non-linear equations cannot generally, be expressed in closed form, we have seen that before. So, numerical methods based either on linearization basically, we take our original non-linear system and we linearize it, in the first class, we introduced some basic ideas about linearization.

We are going to talk about in about it in greater detail later on that, we can use either linearization the successive approximation to solve these non-linear equations. So, these iterative methods start from an initial approximation to the root and produce a sequence, which eventually converges to the desired value. However non-linear equations may have multiple roots, what are multiple roots well.

(Refer Slide Time: 43:48)

### Roots of nonlinear equations

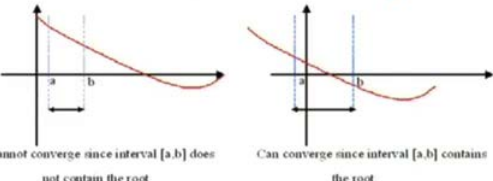
- Unless the starting value of the iteration is within a certain interval, say  $[a, b]$  that contains the root, the iteration may not converge to the desired root.
- Other algorithms may require an initial approximation that is close to the desired root in order to converge. These methods typically have the advantage that they converge more quickly.
- Thus one can begin with a method that is relatively simple to use, which may not converge very fast, but is sure to converge if starting iterate is within a certain interval that contains the root.



We will talk about that, we will give a precise definitions of multiple roots later on, but I just wanted to mention that, we have you can have simple roots and multiple roots and we will define, what exactly a simple and multiple roots later on. But, unless the starting value of the iteration is within a certain interval say that contains the root iteration may not converge to the desired root.

(Refer Slide Time: 44:16)


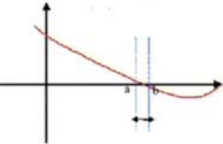
### Roots of nonlinear equations



Cannot converge since interval  $[a, b]$  does not contain the root

Can converge since interval  $[a, b]$  contains the root

But for some algorithms not only must the interval  $[a, b]$  contain the root, it must be sufficiently small for convergence to occur



Say for instance suppose, I have a non-linear function like this and I am going to find the root of this non-linear function, root meaning the value at of I want to find the value of

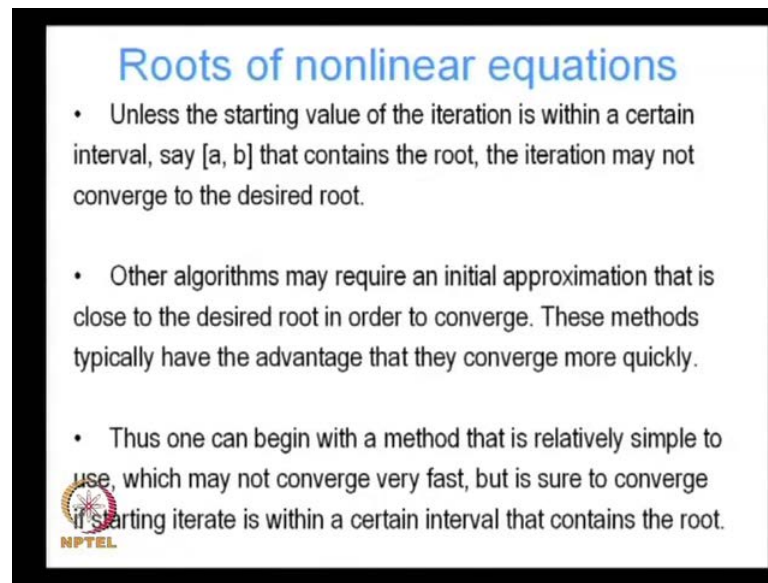
my abscissa for, which my ordinate becomes 0. So, I want to find  $x$  such that,  $f$  of  $x$  becomes equal to 0 right. So, I want to find this  $x$  right, such that  $f$  of  $x$  become 0, but if my if I restrict my iteration interval to between  $a$  and  $b$  where,  $a$  and  $b$  do not bound the root, I have no chance of finding the root right.

If my root lies outside my iteration interval  $a$  to  $b$ , I have no chance of finding the root right. So, I better make sure, that my iteration interval bounds my root right, so if my root lies within my iteration interval. So, many algorithms, if the root lies within an iteration interval, however, large be the iteration interval, my iteration interval can may be as large as I want right.

That if my root lies within that iteration interval, it is not this situation, but it is this, but with my  $a$   $b$  far apart as far as possible as I want, but still those methods are going to after suppose  $n$  large number of iterations, we are going to converge to the root right. Come what are they are going to converge to the root somehow right, but there are some other methods, which are not going to converge to the root unless I start sufficiently close to the root, unless my interval is bounds the root that is relative narrow bound on the root right.


And the problem is that most good iteration methods, the good iteration methods means methods, which converge fast to the root, which have fast rates of convergence, they need a small initial interval. So, that means, that they converge faster near the root right, but far away from the root, they do not perform well right, they converge slowly right. So, for they are most efficient most efficient methods, they were best near the root.

(Refer Slide Time: 46:43)



**Roots of nonlinear equations**

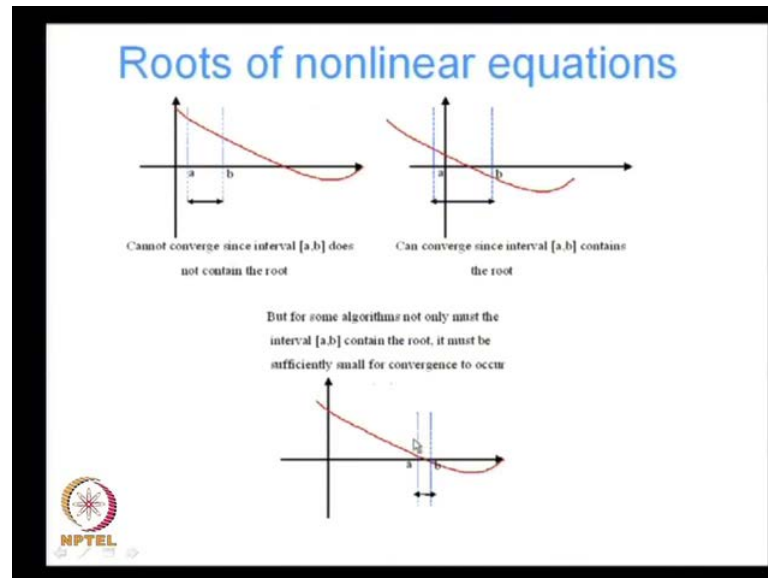
- Unless the starting value of the iteration is within a certain interval, say  $[a, b]$  that contains the root, the iteration may not converge to the desired root.
- Other algorithms may require an initial approximation that is close to the desired root in order to converge. These methods typically have the advantage that they converge more quickly.
- Thus one can begin with a method that is relatively simple to use, which may not converge very fast, but is sure to converge if the starting iterate is within a certain interval that contains the root.

 NPTEL

So, unless the starting value of the iteration is within a certain interval say  $a$   $b$ , that contains the root, the iteration may not converge to the desired root, other algorithms may require initial approximation that is close to the root in order to converge, these methods typically, have the advantage that, they converge more quickly. So, what is a good alternative may be to start with a method, that may be does not, we because we do not know the interval, which bounds the root right.

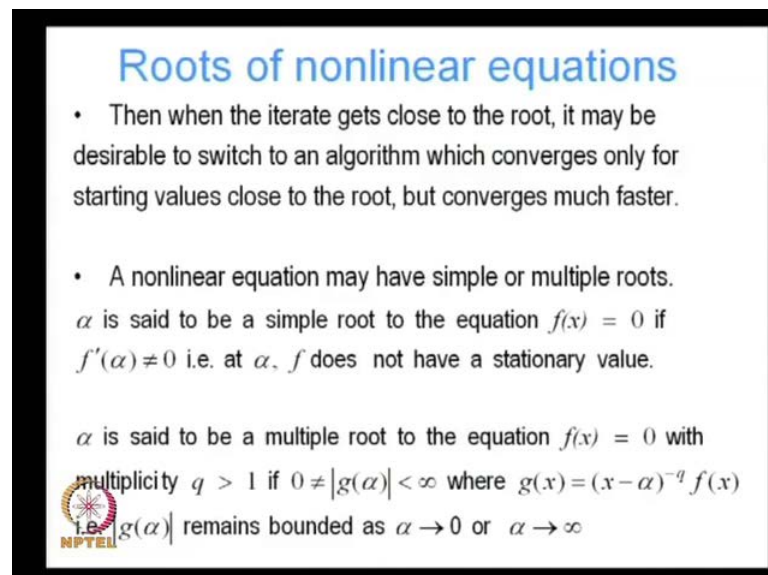
So, but we only know that the root lies somewhere between  $a$  and  $b$   $a$  and  $b$  is big right, we start with  $a$  and  $b$  big right and then we use a relatively unsophisticated method to narrow the bound right. And then we use a sophisticated method near the when the bound is sufficiently small right. A sophisticated method means more expensive right, so I do not want to use my sophisticated method when my bound is large, so I use my relatively unsophisticated method to narrow down, that bound right and when I have a sufficiently narrow bound, then I have reached.

(Refer Slide Time: 47:59)



Somewhere here, then I can use my sophisticated method, because that is because I am, I know even though, I am using the sophisticated method and it is more expensive, I know that, it has very good convergence property. So, I am going to, I am going to converge faster to the root.

(Refer Slide Time: 48:21)



So, then when the iterate gets close to the root, it may be desirable to switch to an algorithm, which converges only for starting values close to the root, but converges much faster. So, I said, I am going to define, what are simple and multiple roots, so let

me do that, alpha is said to be a simple root of this equation  $f(x) = 0$ , if  $f'(x)$  at alpha is not equal to 0, that is at alpha the derivative of  $f$  is not equal to 0.

So, that is that does not have a stationary value at alpha, alpha is said to be a multiple root to the equation  $f(x) = 0$ , if this multiplicity  $q$  greater than 1, if we can write, we can write  $f(x)$  as  $(x - \alpha)^q g(x)$  where  $g(\alpha) \neq 0$ . That is we can write  $f(x)$ , in terms of another function  $g(x)$  times  $(x - \alpha)^q$  and in that case, we say that  $f$  has roots alpha with multiplicity  $q$  that means, there are  $q$  roots of  $x$  equal to alpha right. And this function  $g(x)$  remains bounded as  $x$  goes to alpha, so as  $x$  goes to alpha  $g(x)$  does not become 0 or infinity, so  $g(x)$  has some finite value right. If this condition is satisfied, we say that  $f(x)$  has multiple roots alpha of order  $q$  right,  $q$  multiple roots  $q$  roots alpha.

(Refer Slide Time: 50:15)


### Systems of nonlinear equations

- We will focus mostly on iterative schemes for roots of one-dimensional nonlinear equations.
- Multi-dimensional nonlinear equations give rise to a system of nonlinear equations. For instance if we have nonlinear equations in variables  $x_1, x_2, \dots, x_n$  it is clear that we need a system of  $n$  denoted as:  $\mathbf{F}(\mathbf{x}) = 0$  i.e.
 
$$f_1(x_1, x_2, \dots, x_n) = 0$$

$$f_2(x_1, x_2, \dots, x_n) = 0$$

.....

$$f_n(x_1, x_2, \dots, x_n) = 0$$

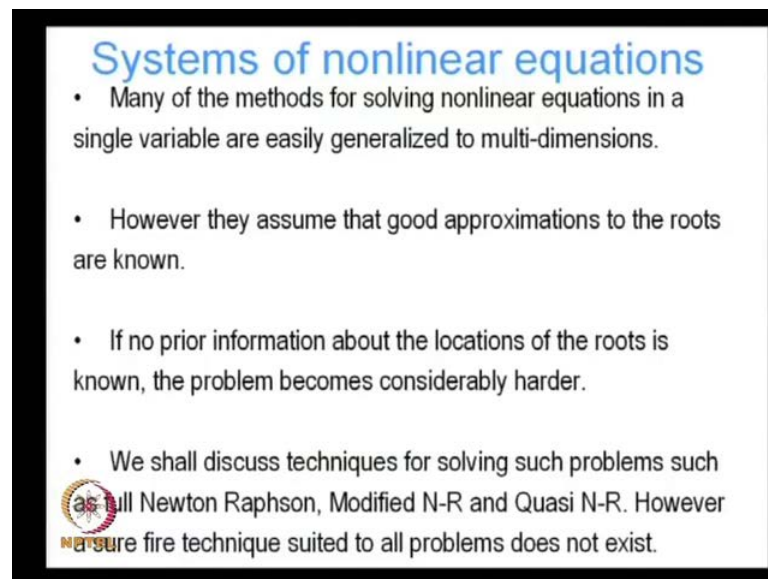
 Solving a multidimensional system of nonlinear equations is considerably more difficult than solving a 1D nonlinear equation

So, we will focus mostly on iterative schemes, so in while derive while looking at algorithms, we are going to first look at algorithms for 1 dimensional non-linear equations, because they are the easiest to understand right. And the most of those ideas, which we obtained for 1 dimensional non-linear equations get generalized to multiple dimensions, but we are going to look at multiple dimensions later on, but the basic ideas, we are going to see with 1 dimensional non-linear equations.

Multidimensional non-linear equations, what do I mean may I say multidimensional, which means that, there are more than one independent variables right. Instead of just  $x$  there are this there is a vector of independent variables  $x_1 x_2$  through  $x_n$  right. So, multidimensional non-linear equations give rise to a system of non-linear equations, for instance, if we have non-linear equation in variables  $x_1 x_2$  through  $x_n$ , it is clear that, we need a system of equations for 1 variable, I need 1 equation right, for  $n$  variables I need  $n$  equations.

So, I need a system of equations and that is that I can represent as a vector,  $f$  now becomes a vector  $f$  of  $x$  equal to 0, so I want to find the roots, I want to find  $\alpha$  as a vector of  $\alpha$ 's, such that  $f(\alpha)$  becomes equal to 0, if I now  $\alpha$  is a vector. And I can write this as  $f_1 f_2 f_n$  equal to 0, so this is the system of equations. So, solving a multidimensional system of non-linear equations is considerably more difficult, than solving a 1 dimensional non-linear equation, which should be sort of obvious.

(Refer Slide Time: 52:05)



**Systems of nonlinear equations**

- Many of the methods for solving nonlinear equations in a single variable are easily generalized to multi-dimensions.
- However they assume that good approximations to the roots are known.
- If no prior information about the locations of the roots is known, the problem becomes considerably harder.
- We shall discuss techniques for solving such problems such as full Newton Raphson, Modified N-R and Quasi N-R. However sure fire technique suited to all problems does not exist.

as full Newton Raphson, Modified N-R and Quasi N-R. However sure fire technique suited to all problems does not exist.

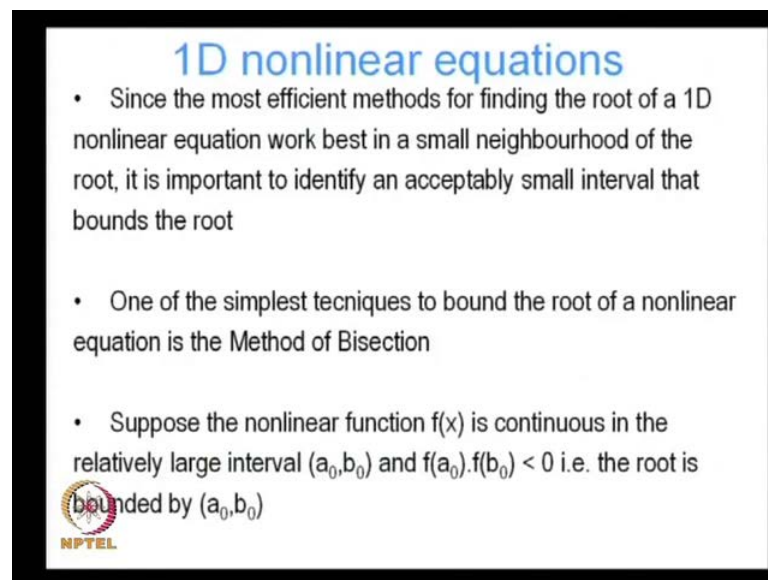
However many of the methods for solving non-linear equations in a single variable are easily generalized to multi-dimensions, however they all assume that good the methods that, we are going to talk about, they all assume that good approximations to the roots are known, If no prior information about the location of these roots are known the problem becomes considerably harder. We shall discuss techniques for solving non-linear



problems, such as full Newton Raphson modified Newton Raphson and quasi Newton Raphson.


However, I should mention that, there is no technique, no non-linear technique right, which can claim that given any non-linear problem, it is going to be able to solve it right. So, that I am not aware of any such techniques till now, this was better we have better techniques and not, so good techniques, but there is no technique, which can solve all non-linear problems without fail, because nonlinearities are really hard to handle.

(Refer Slide Time: 53:14)



**1D nonlinear equations**

- Since the most efficient methods for finding the root of a 1D nonlinear equation work best in a small neighbourhood of the root, it is important to identify an acceptably small interval that bounds the root
- One of the simplest techniques to bound the root of a nonlinear equation is the Method of Bisection
- Suppose the nonlinear function  $f(x)$  is continuous in the relatively large interval  $(a_0, b_0)$  and  $f(a_0) \cdot f(b_0) < 0$  i.e. the root is bounded by  $(a_0, b_0)$



So, in our next class, we are going to talk about the most efficient methods for solving 1 dimensional non-linear equations, we are going to talk about the Newton Raphson method. We are going to look at the Newton Raphson method very closely, we are going to look at its convergence properties right. Then we are going to look at the secant method its convergence properties and then we are going to move on to more general methods.

Thank you.