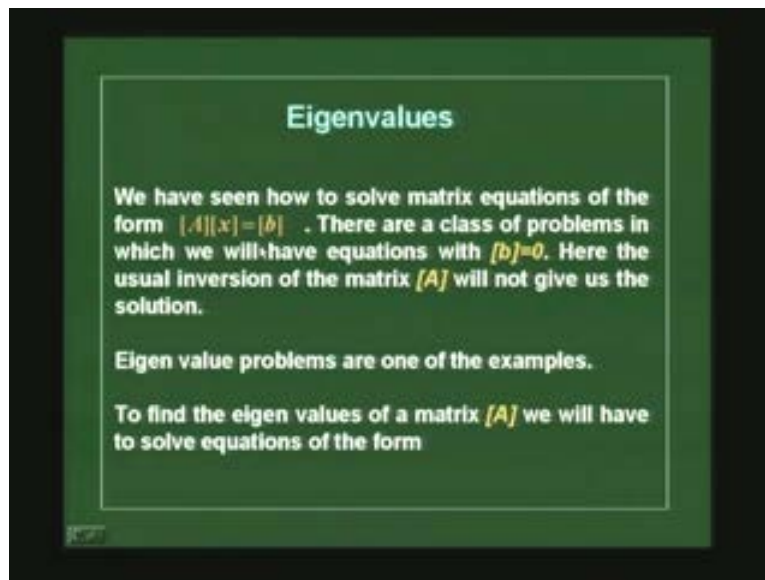


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 19
Eigen Values of a Matrix

We have seen how to solve the set of linear equation for using matrix inversion or by using elimination techniques. So now, we look at the special class of problem set of linear equation problems, is slightly different class of problem in which, we have the equation of this form $Ax = b$ but the right hand side of this equation is 0.

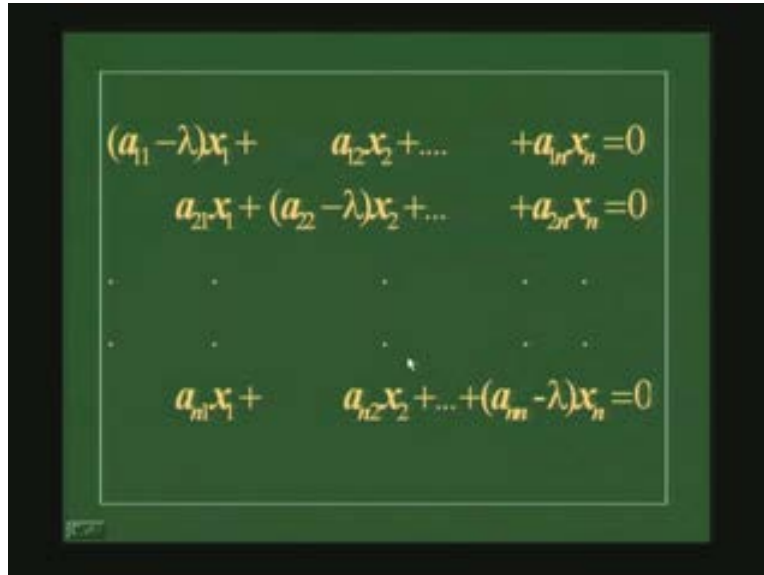
(Refer Slide Time: 02:58)



So in that cases such situation arises for example, when you want to find Eigen values is can Eigen vectors the matrix a right. In that situation right we have to solve equations of the form $(A - \lambda I)x = 0$. So set of linear equation again, once again is the matrix is $(A - \lambda I)$, where I being a unit matrix and A is the desire matrix to chosen eigenvalues we initiate find out. So the eigen values is eigen function of this matrix this is also we find out, so then we have eigen value equations of this form $(A - \lambda I)x = 0$ and, we know that such equation have non-trivial solution, that is trivial solution can $x = 0$ and non-trivial solution, if the determinant of $(A - \lambda I)$ vanishes.

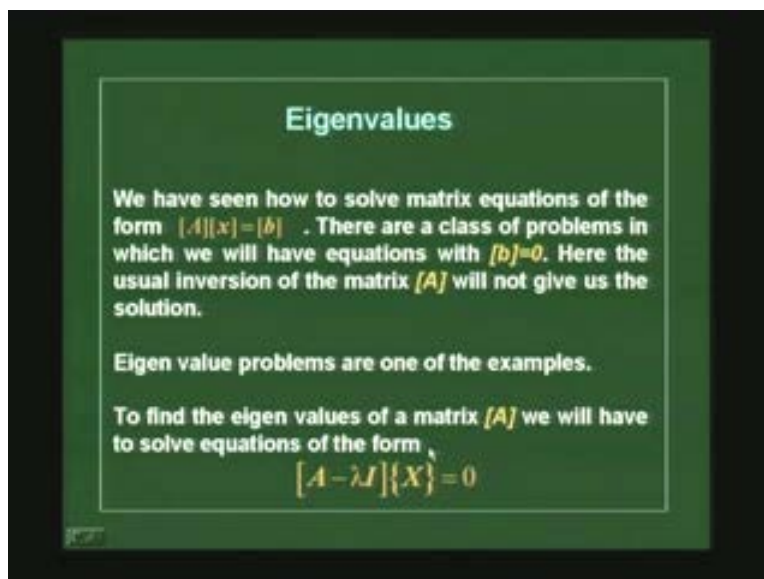
So that is, the condition we use to term and Eigen values λ . So that situation in which $Ax = b$ set kind of equation such has to be solve the matrix from matrix which can have non-trial solutions that is solution is different from $x = 0$ provided the determinant of $(A - \lambda I)$ is not equal to 0.

(Refer Slide Time: 03:19)


$$\begin{aligned}(a_{11} - \lambda)x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= 0 \\ a_{21}x_1 + (a_{22} - \lambda)x_2 + \dots + a_{2n}x_n &= 0 \\ \dots & \\ a_{m1}x_1 + a_{m2}x_2 + \dots + (a_{m} - \lambda)x_n &= 0\end{aligned}$$

So that is kind of things we want to look at so we could say that an equation here matrix equations of this form write $a_{11} - \lambda$, a_{12} , a_{1n} and a_{22} , $a_{22} - \lambda$, x_2 etcetera and that is equal to 0 that is equal to equation it solve set of equations okxy just we solve in the matrix form here.

(Refer Slide Time: 03:51)



Eigenvalues

We have seen how to solve matrix equations of the form $[A][x] = [b]$. There are a class of problems in which we will have equations with $[b]=0$. Here the usual inversion of the matrix $[A]$ will not give us the solution.

Eigen value problems are one of the examples.

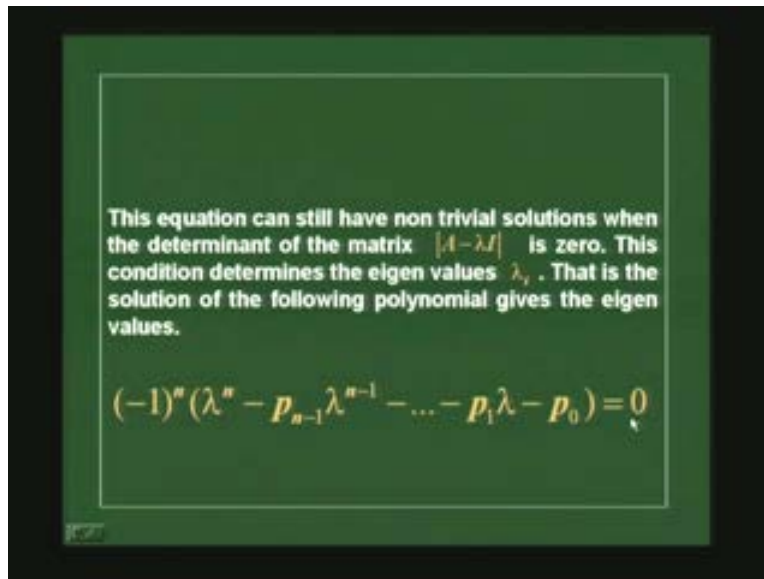
To find the eigen values of a matrix $[A]$ we will have to solve equations of the form

$$[A - \lambda I][X] = 0$$

So then, we now only what we do is equation of this form right, we just saw that to find inverse of the this matrix and then multiply inverse by b, multiply b by inverse then, we go x solutions that would to work here because right hand side 0 but, in the inverse itself it does in x is that is when the determinant of this is 0, we can still find a set of equations

which are a, set of solution for this x, which are non-trivial. So that is what we want to find out for constructing the determinant x the terminant of the lambda equal to 0 that is what you want to find, the constructing the determinant itself term of the non-trivial when a is very large. So you seen that the determinant a minus lambda i the determinant of a minus lambda i can be expanded in terms of lambda, a polynomial that is form and equate that 0, so has a set what do to you want to do is to wants to find out the values of the lambda such that, so we find out the values of lambda such that the determinant 0 right that is our aims right.

(Refer Slide Time: 04:19)



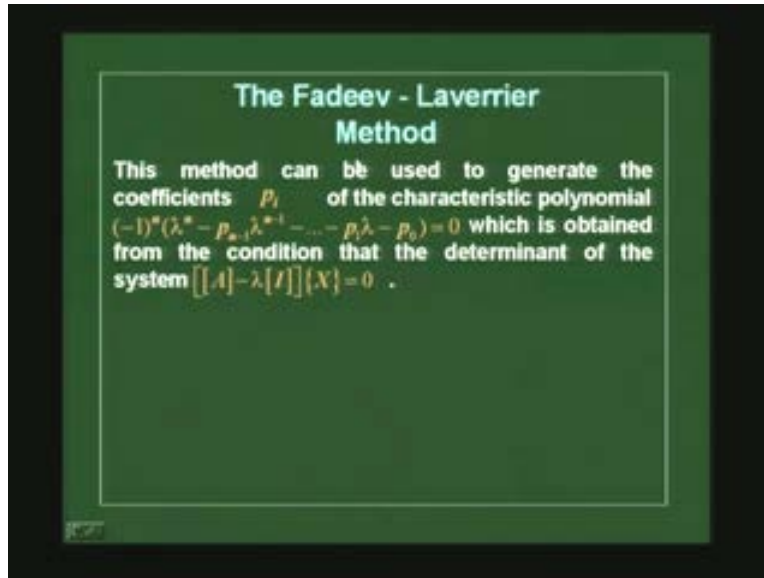
So such equations, okay that is polynomial equations this form that is when expanding this think for this polynomial equations this form and now we need to solve such polynomial equations okay such that is a discussion which have about this today. Okay, how to solve, how to arrive at such equations and then how to solve has a said that two problems the first thinks to actually arrive such equations right.

So here large matrix then the converting the determinant into the polynomial this form itself could be a large exercise. We first to be look at, how to arrive at, how to construct this polynomial equations from this conditional of the determinant a minus lambda i equal to 0 that is the first equations we would have to right okay so that is summaries here, so for large matrices constructing the polynomial is itself non-trivial excises ok we have this polynomial.

We can use one of the root finding algorithm to determinant in the solutions right so we have the discussion root finding in the general problem but we look at that in this specific case of eigen value from okay, so know loot at the one of this method to construct this polynomial which is called a fadeev laverrier method okay this is the boost, one of the easiest ways methods of the constructing the characteristic polynomial that is this

polynomial which determines the eigen values, so one of the easiest way to do the construction of this polynomial is use using, what is calls fadeev laverrier method.

(Refer Slide Time: 06:18)



We will not go into that details derivation or the logistic behind the fadeev laverrier method which just see, how such think can be implemented and what is method, what is algorithm and how this can be implemented and we have just see particular case why is been implemented. So the idea is the following, we want to determinant this equation that is a minus lambda i into x equal to 0 and we want to find the determinant of a minus lambda i and we want write it in this particular form.

Okay now, this polynomial in lambda, okay so we are find nth order matrix, then we will have the order of the polynomial also be n okay 4 by 4 matrix, the highest coefficients, the highest power of lambda is be 4 right. So we know that and then this are the coefficients of this power that is in, so you have p_n minus 1 p_1 , p_0 as a coefficients of this polynomial, okay so this polynomial coefficients will now be function of elements of this matrix, a idea is to actually the determinant this polynomial coefficients.

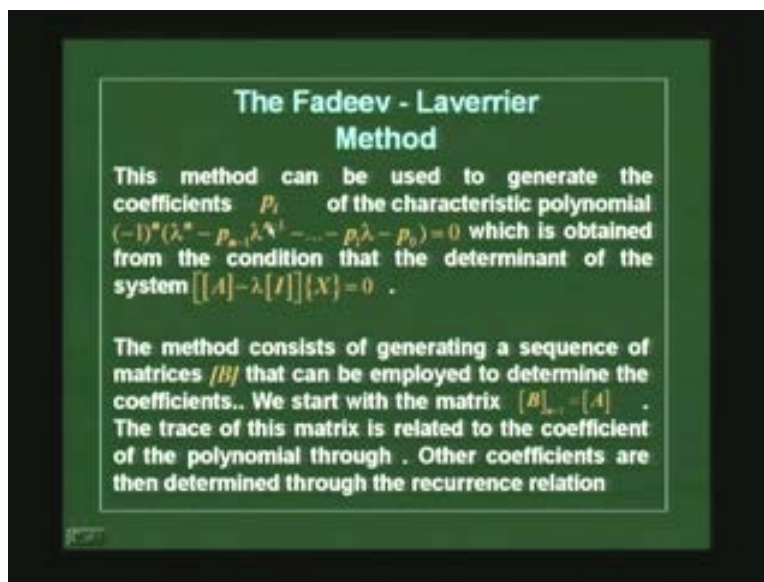
The fact that this equations that determinant a minus lambda i equal to 0 that the determinant of this, particular matrix, that is matrix a minus lambda terms identity matrix equal to 0 can be return into this form its a sure, how is all us, so now the question is good algorithm to determinant, what is the good algorithm to determinant this coefficients p_n minus 1 into p_n p_0 . The first coefficients always be 1, the high, the coefficients of high power, this lambda is p always p_1 is, such we can always identity this form let see okay.

So now, we want to construct the this think, okay so we will what we do we generate to sequence of matrices who is trace would be the coefficients of this matrix p_n minus 1 to p_0 okay that is a they will a fadeev laverrier method, so we will generate sequence of

matrices called b and then if have n okay b_n minus 1 to b_0 okay that is what do going to have mini matrices, so and that trace of the matrix ok would be a coefficients of this of this particular order.

So for example, this one the coefficients of this the first the next, the second term in this polynomial that is the coefficients of lambda to the power of n minus 1, there is p_n minus 1 is simply the trace of this matrix a . Now and then that is where a start with. Okay so this particular coefficients the trace of this matrix, you can prove this, so you can use this prove using the Calley Hamilton theorem that is, that all the all the characteristic equations, all matrices satisfied one characteristic equations that is what can use to proof this we had gone into the proof we just we want to looking here only at the algorithm.

(Refer Slide Time: 07:18)



So which say that, the first coefficients, that is coefficients of lambda to power n minus 1, so lambda power n terms coefficients is 1, ok and the coefficients of the lambda to the power n minus 1 to and that is the trace of the matrix a , ok first matrix b , that b_n minus 1 here and here okay we just trace the matrix itself, a itself ok and we take the trace of that. Now, other coefficients are determinant through a series of regression relations of this form there is, now the next one will be b_n minus 1, we just show, ok now b and minus 1 is a itself now b_n minus 2 is that construct b and minus i in general, is constructed from b and minus i plus 1 by this relation. So we started with b_n minus 1, we just a itself, now we take b_n minus 2 that is i equal to 2 and that is equal to the matrix a multiply by b_n minus 1, now i is 2.

So n minus i plus 1 is 1, n minus 1, so it is b_n minus 1 which was matrix a itself, we remember. Now minus, the coefficients which is just now got that is p_n minus 1 multiplied by the identity matrix. So that is we have construct this matrices b_n minus 1, b_x . Okay then we take the trace of the that matrix okay divided by I , so in this case p_n minus 2, we want to construct right, i is 2 there is 1 by 2 time the trace of b_n minus 2. So

then, we will continue this, so this is regression relation, so once we have the b_n minus 2, we can go back here right and we have p_n minus 2, b_n minus 2 the trace its remember is the sum of the diagonal elements, so the matrix so sum of the diagonal elements is trace the trace is the b is b_{ii} sum over i is, once we have p_n minus 1 and then we go back here.

(Refer Slide Time: 11:09)

$$[B]_{n-i} = [A][B]_{n-i+1} - p_{n-i+1}[I]$$

and

$$p_{n-i} = \frac{1}{i} \text{tr}[B]_{n-i}$$

Until we obtain

$$[B]_0 = [A][B]_1 - p_1[I]$$

and

$$p_0 = \frac{1}{n} \text{tr}[B]_0$$

Okay, so we have p_n minus 1 and p_n minus 2, then we can construct b_n minus 3, so b_n minus is 3 now matrix a time b_n minus 2 minus p_n minus 2 into I, the identity matrix okay now, we need to go till I reaches n there is until we get b_0 , okay now b_0 , finally we get and raise that b_0 will be then since n minus I_0 , n minus I_0 b_0 would be matrix a into b_1 minus p_1 of I, now just an interesting, so then p_0 and 1 is 1 by n times, so write n so I is n now so 1 by n times the trace of the b_0 .

So I repeat, we start with matrix b_n minus 1, we take the a write so the trace of that the sum of the diagonal elements of this matrix, that is sum of the diagonal elements of the matrix a itself is the coefficients of the lambda to the power of the n minus 1 terms and then we continue, right and then, we use the first we start with b_n minus 1 as a and then, we construct all b_n minus i is using this recursion relation that is a into b_n minus i plus 1 minus p_n minus i plus 1 into lambda into i.

Okay so you could substitute this back into the equation and see that the matrix satisfied the characteristic equations something, which as the will not go into but terms of the situations come from the come from the Cayley Hamilton theorem. Okay so now, we have if p_n minus 1 that is, the trace of this okay and once we have the p_n minus i, we can go and construct the next p_n minus i etcetera. Okay just we continue this till we reach the value d_0 and once we have b_0 and that is a into b_n minus p_1 and then we have all the coefficients p_0 and then, all the coefficients that is an interesting follow, how to this, that is the fact that a inverse, okay so now we just show that here the b_0 is a into b_1 minus p_1

times i right, that is what we just do this so and b_0 is that trace of for this and this equation. We can construct that a inverse right, so we will term of we will find out a inverse is then simply related to 1 by p_0 b_1 minus p_1 i because we just that, we can construct b_0 and trace of that okay is as p_0 so if is take the a inverse of that, so that the wide the right hand side by 1 by p_0 then, I should be getting and identity matrix right. So I will get a inverse then as 1 by p_0 b_1 minus p_1 i . We will just see this in an example is actually implemented this see this will, we will see the all the details of how we get much more better feeling of how will actually arrive this.

(Refer Slide Time: 15:28)

The method has the additional advantage that it gives the inverse of the matrix $[A]$ through

$$[A]^{-1} = \frac{1}{p_0} [[B], -p_1 [I]]$$

Complete explanation of the above algorithm is beyond the scope of this course. However it is easy to see that it follows from the fact that every matrix satisfy its own characteristic equation.

(Refer Slide Time: 16:28)

Let us look an example:

Use the Fadeev Laverrier method to find the characteristic polynomial of the matrix

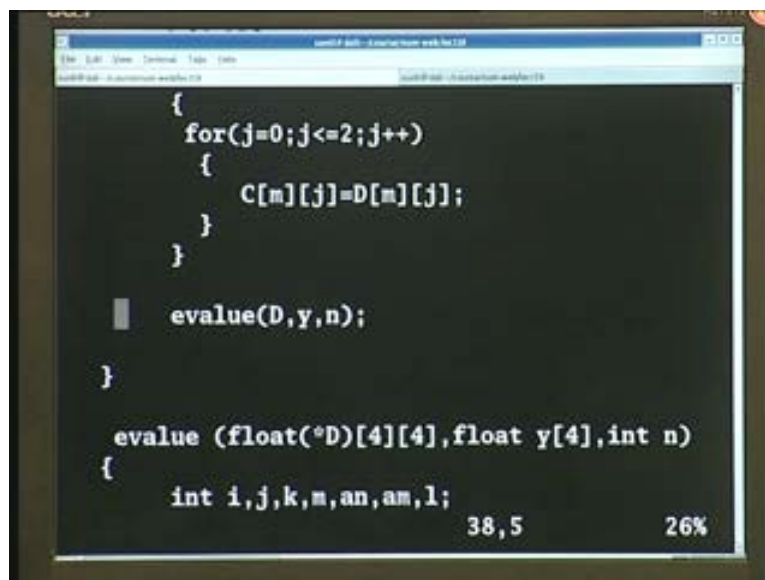
$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

So it say complete the explanation of this is b_1 , the scope of this occurs okay but, we could see that is from the whole, this whole idea come from the calley Hamilton theorem okay interested only looked at this calley Hamilton theorem, okay so now we just look at the example. Okay so we will example for matrix of this from 2 minus 1 0 0 and minus 1 2 minus 1 0 and 0 minus 1 2 minus 1 0 0 minus 1 2, so that the matrix. Now we want to construct the eigen values of this matrix, so we take the such 4 by 4 matrix because we can actually do this by hand but idea is that for this program can this time can easily this scale to large matrix n since, start with this, so as said that we will half b and minus 1 and set equal to a.

So I will also the same a program in which is the implemented and we will come back will go step by step. So here is the program which basically implements, so I have matrix d right, so I have the usual think which have the, when including this all the set of linear equation problem that we have the matrix the right and we have the elements of the matrix yes now as a said as and exactly this same as us see that arrive when you print saw then would it will be exactly the same us this same as this matrix we have 2 minus 1 0 0 minus 1 2 minus 1 0 0 minus 1 2 by minus 1 0 0 minus 1 2.

Okay so we have this matrix now, I just store the matrix before I do any things to matrix I just store this n_1 another variable c, so because I am going to find the, I going to fast this matrix d into function call eigen values, that is e value and that this function is takes into which as fast in d will return eigen values in an array y, okay that side here y is an array is one dimensional array 4 by 4 matrix and eigen values is come out in an one dimensional array y and since this program might alter, this program actually alter by matrix the I store that never to c before I pass it down .

(Refer Slide Time: 19:26)



```

{
    for(j=0;j<=2;j++)
    {
        C[m][j]=D[m][j];
    }
}

    evaluate(D,y,n);
}

evaluate (float(*D)[4][4],float y[4],int n)
{
    int i,j,k,m,an,am,l;

```

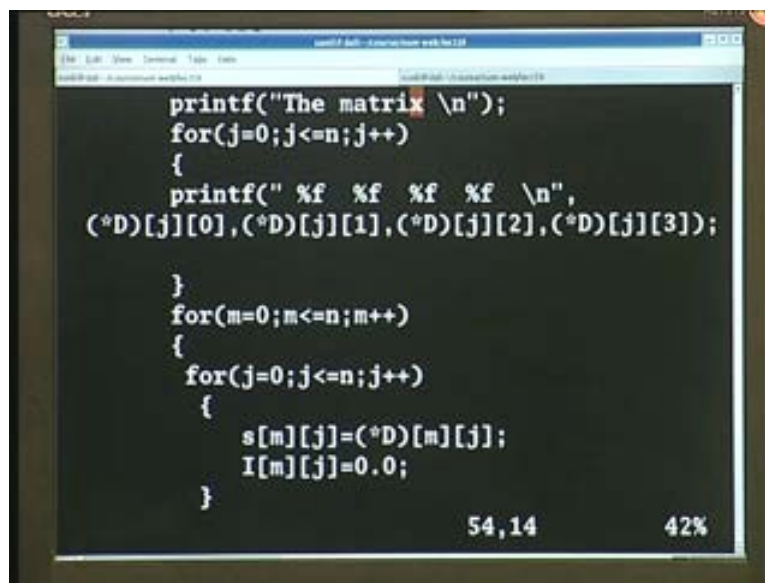
So this comparison then we required in an original program, in an actual program okay and then so what I do this is the main program here is very simple, the main program

basically just defines matrix d and store the whole elements of that in a matrix c and call is eigen values function and x is that is solve it as. So we can do for the things with this Eigen values is wise this will just y is can Eigen values and why just give see the coefficients of the characteristic polynomial.

So this particular function will just return y has the coefficients of the characteristic polynomial and then the main program n, so the main program n is here, ok so the n program n th here, so just calls the function to which pass is d and get back to coefficient of the characteristic polynomial in this matrix, in this one dimensional array y and n is the order of matrix, order of the matrix minus 1 in this case because start from 0. Okay it goes from 0 to 3, n is 3 particular thinks. So I have define n is 3 here, okay right, okay that is all it as.

Okay, so now the program does not anything more because, we would want append this program with the solution of the characteristic of polynomial later. So right now, we just determined in the characteristic polynomial x_n okay then, we go there this is now this is the function which does the calculation of the characteristic polynomial, so to which now this is function receives the matrix d has 2 dimensional array of pointers, so here is a another example in which you pass as 2 dimensional array has to a function and the function revises that has the two dimensional array of pointers, we have seen this in the early lecturer, if passing an array into function and the function c into has a two dimensional array of point. So here is a example of that the receives this functions, receives it has two dimensional array of pointers we can also receive has 2 dimensional array itself use this just show you how to, use how to implement this kind of passing two dimensional arrays.

(Refer Slide Time: 21:37)



```
printf("The matrix \n");
for(j=0;j<=n;j++)
{
printf(" %f %f %f %f \n",
(*D)[j][0],(*D)[j][1],(*D)[j][2],(*D)[j][3]);
}
for(m=0;m<=n;m++)
{
for(j=0;j<=n;j++)
{
s[m][j]=(*D)[m][j];
I[m][j]=0.0;
}
}
```

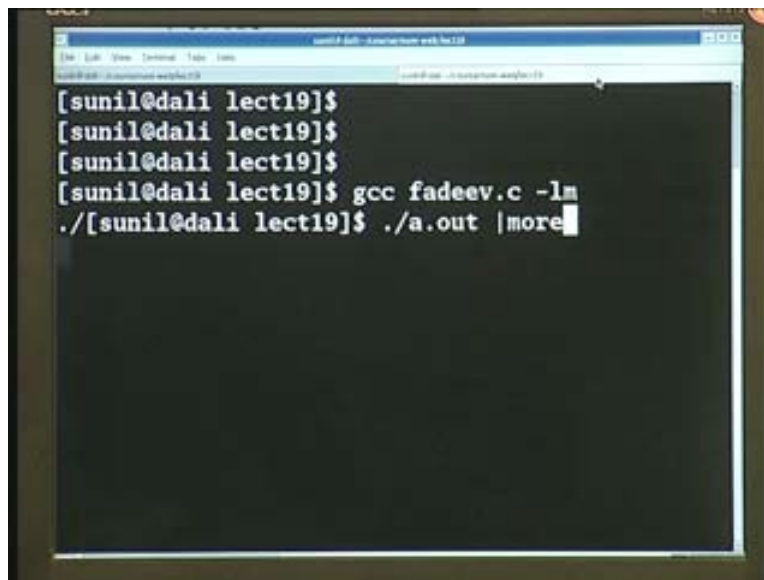
54,14 42%

Ok now this receives the one dimensional arrays just has in arrays of the simply has an array of dimensional 4 and then receives order of the matrix has an integer. So now, I

have another set of integer defined here and then have another matrix defined here and then identity matrix an then, there is the matrix which it all works working array which would required which is go here we will see that, and then this function calls another function and so that function point to here, so this function this is the another function and that is called multiply, okay just this function is basically is to multiply 2 matrices.

Okay, so this is my this function is idea is multiply give two dimensional matrices is multiply, we will see that ok then another function which see that calculates minus 12 something to power n ok so now it receives has said the matrix d right and the I just print that matrix d we will see that received at correct this function okay such has passing into that function just print that out okay the first thing I just to see in this is receives it correctly So now that is first part here, so that it is see for ignore all this so first part here there it is received the matrix okay everything as a floating, its floating point number. Okay so everything is floating point numbers so we can see to numbers minus 100 minus 102 minus 1000 minus 122 minus 100 minus 12 has the same things that just return down.

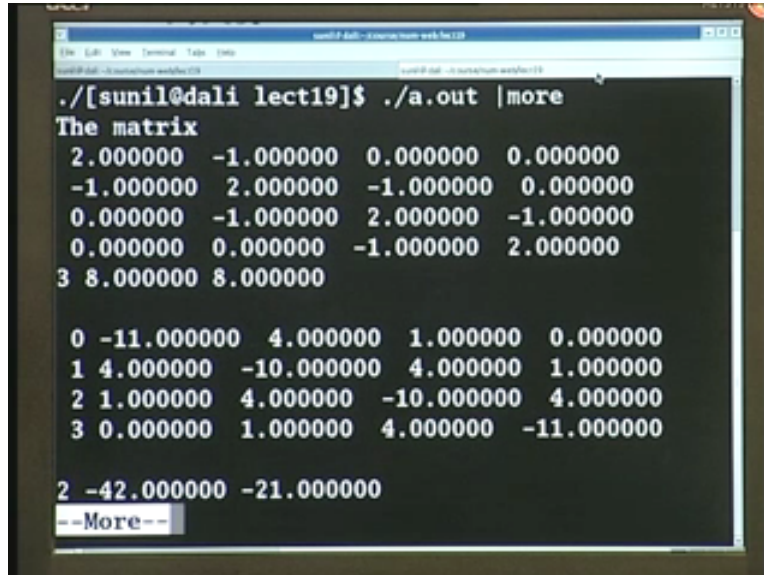
(Refer Slide Time: 21:57)



```
[sunil@dali lect19]$  
[sunil@dali lect19]$  
[sunil@dali lect19]$  
[sunil@dali lect19]$ gcc fadeev.c -lm  
./[sunil@dali lect19]$ ./a.out |more
```

Okay, so then we what we do then is to this matrix x is yes now how we have to construct our idea construct matrices b okay which is call I call s here. So the first order, so the first is the first matrix is simply the matrix itself right, so that is what I doing from here ok so I have in this loop that m goes to from 0 to n has a said now goes from 0 to that is order of the matrix n is 3. So the order of the matrix is 4 is 4 and jk was 0 to n and then we have for this matrix s is simply the same as matrix okay that is what b_n minus 1 was remember, is the matrix d itself right, so this is b_n minus 1, now s that the matrix d itself I was the uses the arrays to define my identity matrix.

(Refer Slide Time: 21:58)

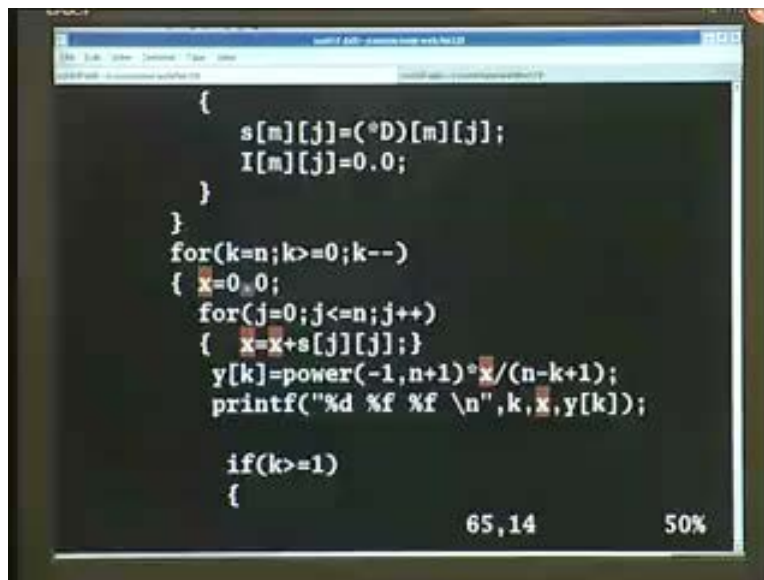


```
./[sunil@dali lect19]$ ./a.out |more
The matrix
2.000000 -1.000000 0.000000 0.000000
-1.000000 2.000000 -1.000000 0.000000
0.000000 -1.000000 2.000000 -1.000000
0.000000 0.000000 -1.000000 2.000000
3 8.000000 8.000000

0 -11.000000 4.000000 1.000000 0.000000
1 4.000000 -10.000000 4.000000 1.000000
2 1.000000 4.000000 -10.000000 4.000000
3 0.000000 1.000000 4.000000 -11.000000

2 -42.000000 -21.000000
--More--
```

(Refer Slide Time: 23:56)



```
{
    s[m][j]=(*D)[m][j];
    I[m][j]=0.0;
}
for(k=n;k>=0;k--)
{ x=0.0;
  for(j=0;j<=n;j++)
  { x=x+s[j][j];}
  y[k]=power(-1,n+1)*x/(n-k+1);
  printf("%d %f %f \n",k,x,y[k]);

  if(k>=1)
  {
65,14 50%
```

So right now identity matrix all the matrix define the matrix i with all the element 0, 0 that is what would be now this define matrix is the matrix here which is the all the elements 0 will see the use of the this right, so in a little while. So here this matrix b n minus 1 which is just d itself, right so and then what do you want you want compute that trace of the matrix, okay so there is a loop here.

(Refer Slide Time: 24:25)

$$[B]_{n-1} = [A][[B]_{n+1} - p_{n+1}I]$$

and

$$p_{n-1} = \frac{1}{i} \text{tr}[B]_{n-1}$$

Until we obtain

$$[B]_0 = [A][[B]_1 - p_1I]$$

and

$$p_0 = \frac{1}{n} \text{tr}[B]_0$$

Okay, so what I did first before enter the loop I just define my b has minus 1 has d itself ok and then I will now go for i right. So we have now remember this regression relation, so we have to used to regression relation right but b is for i equal to 1 that is b_n minus 1 that simply a is good define that out side and then I can understand I loop which wood which uses requisition relation to the term in the b_n minus 2, n minus 3 and n minus 4 our nor k will just up to n minus 3 because n is 3 okay. So I start with b_n minus 1, so that the first quantity which is the d here.

(Refer Slide Time: 26:01)

```

{
    s[m][j]=(^D)[m][j];
    I[m][j]=0.0;
}
}
for(k=n;k>=0;k--)
{
    x=0.0;
    for(j=0;j<=n;j++)
    {
        x=x+s[j][j];
    }
    y[k]=power(-1,n+1)^x/(n-k+1);
    printf("%d %f %f \n",k,x,y[k]);

    if(k>=1)
    {

```

67,9-16 50%

So which is the matrix d here, so that is the first quantity and then compute come here inside of the loop, okay other then actually and term the regression relation. So first I will do this compute p_n minus 1 that is trace of the matrix a, or the d here. Okay, so remember the matrix d is the a in the, in the in the notes and s is the b in a notes, ok so now uses s compute the compute the trace of that so I define x equal to 0 here and then sum all this quantities right.

So I sum all the diagonal elements j j, ok so all the diagonal elements of this matrix s is summed here, okay and that sum, so the loop ncr, okay we can see this is simply a loop which ncr right, okay that is 4j values. Okay, so sum the all the diagonal elements if x that gives me p_n minus 1 right, the p_n minus 1 is that determinant of the second the coefficient of the lambda to the power of n minus one term is now determinant here ok as first time and nothing to divide that because that i is 1 so p_n minus 1 by 1 is p_n minus 1 that the trace divide by 1 is the trace itself if right, so if p_n minus 1 simply the trace of the matrix d. so that what do done here.

(Refer Slide Time: 26:09)

$$[B]_{n-1} = [A][B]_{n-1} - p_{n-1}[I]$$

and

$$p_{n-1} = \frac{1}{i} \text{tr}[B]_{n-1}$$

Until we obtain

$$[B]_0 = [A][B]_0 - p_0[I]$$

and

$$p_0 = \frac{1}{n} \text{tr}[B]_0$$

So we just compute that trace of the matrix b_n minus 1, ok that is what the first, the first step is ok that is, what we should get and then, we will go on constructing others, first we look at the trace of that quantity. Okay that is, we will compute here, so it can see 2222 the trace of that, okay now, so n is 3 okay here. So i was 1, so n minus 1, so n minus i is 3 order is 4, so order minus 1. So this the coefficients of the lambda to the power the 3 term that is what this significance here okay. So fourth order matrix 4 by 4 matrix, so coefficient of the lambda to the power of 4 term is 1 and this the coefficient of the lambda to the power 3 term.

(Refer Slide Time: 28:07)

Solution: $n=4$ so the first step is $[B_1] = [A]$ and the coefficient $p_1 = \text{trace}[A] = 8$

Then from the recurrence relation we have

$$B_2 = [A](B_1 - p_1 I)$$

$$= \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

Okay that the trace of this matrix which is eight that is what I return, so that the trace by i that both 1, so printing of the matrix it is the order of the term, so looking at that the trace and trace divided by i okay, so then once we done that now we go back and then look at the next matrix right, so that is the trace we got now, so we just write it down here as the trace of the b 3, b 3 is a and the trace of that is 8, okay such that which we write it down, ok now next step is construct this matrix that is a into whatever the b_n minus 1, we had here minus the p_3 which we obtain here multiply by identity matrix.

(Refer Slide Time: 28:11)

$$= \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} -6 & -1 & 0 & 0 \\ -1 & -6 & -1 & 0 \\ 0 & -1 & -6 & -1 \\ 0 & 0 & -1 & -6 \end{bmatrix}$$

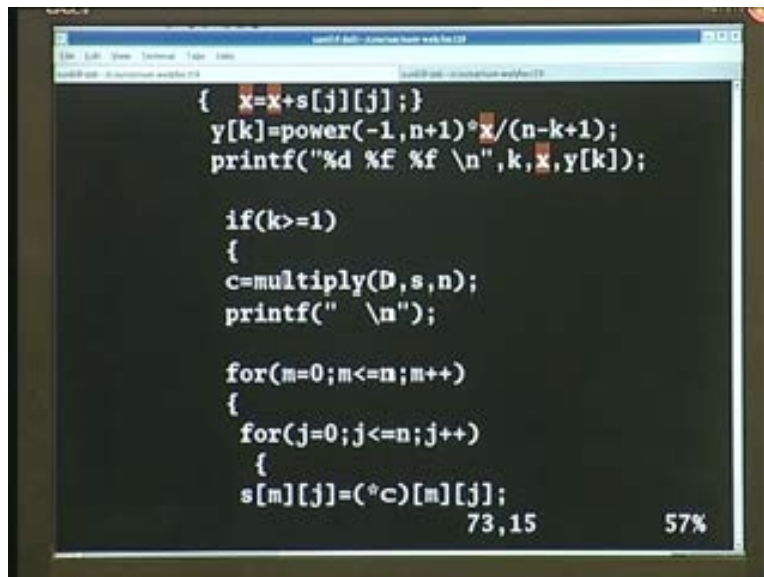
$$= \begin{bmatrix} -11 & 4 & 1 & 0 \\ 4 & -10 & 4 & 1 \\ 1 & 4 & -10 & 4 \\ 0 & 1 & 4 & -11 \end{bmatrix}$$

Basically, we have to do this calculation that is, we have to subtract p_3 minus i and multiply matrix a multiplying matrix a and a , b and minus 1 b and minus 1 is a and then multiply by this matrix with this matrix identify matrix multiply by p_3 and subtract okay and that is a new matrix b and minus 2 and being 4 will be it will be b_2 that is what see here so the b_2 should be b_1 by this then that is implemented here in this form.

Okay, so I have kind also use the element such we learned the program in here, so we have just contracted that trace of the matrix and which we divided by n minus k plus 1 is the i , okay that is 1 right and right row okay now just printed that out you so that within printed out and now we go into if k is greater than for 1 that is case, then we will go here.

Okay and then this multiplied matrices d and s so this is subroutine as the function to which we pass the matrix to matrices d and s this is matrix what matrices two matrices d and s and its order, okay 2 matrices are same order and order of the matrix which that pass that which and this function return sub product ok that what this particular function is doing okay so again notice that I pass 2 functions 2 matrices 2 arrays 2 dimensional arrays in fact, this two dimensional arrays of pointers right now and then two dimensional array.

(Refer Slide Time: 29:33)



```

{ x=x+s[j][j];
y[k]=power(-1,n+1)*x/(n-k+1);
printf("%d %f %f \n",k,x,y[k]);

if(k>=1)
{
c=multiply(D,s,n);
printf(" \n");

for(m=0;m<=n;m++)
{
for(j=0;j<=n;j++)
{
s[m][j]=(c)[m][j];
}
}
}
73,15 57%
```

I just past that is to and then I get to return what I get another matrix c we saw how do we do that so this is the to be function which has to return has the tow dimensional array, so this uses both such in to learned to the lecturer for this, how to pass two dimensional array to a function and how get a two dimensional array back from that function. Ok so both are here, so now this as the two dimensional array that has declare properly that is what we have done here.

(Refer Slide Time: 30:58)

```

{ x=0.0;
  for(j=0;j<=n;j++)
  { x=x+s[j][j];}
  y[k]=power(-1,n+1)*x/(n-k+1);
  printf("%d %f %f \n",k,x,y[k]);

  if(k>=1)
  {
    s=multiply(D,s,n);
    printf(" \n");

    for(m=0;m<=n;m++)
    {
      for(j=0;j<=n;j++)

```

Okay now, this been declare in declare has two dimensional pointers c, c will be a product okay the product come out has c and that is it two dimensional array of pointers two dimensional array of 4 and this multiply this function has to return that two dimensional array so that also dimensional has defined appropriately, that is returning a floating point number such float type is float and is array two dimensional array, so it is as two dimensional of the 4 so this way to defined a function which returns two dimensional array.

(Refer Slide Time: 21:12)

Solution: $n=4$ so the first step is $[B]_1 = [A]$ and the coefficient $p_1 = \text{trace}[A] = 8$

Then from the recurrence relation we have

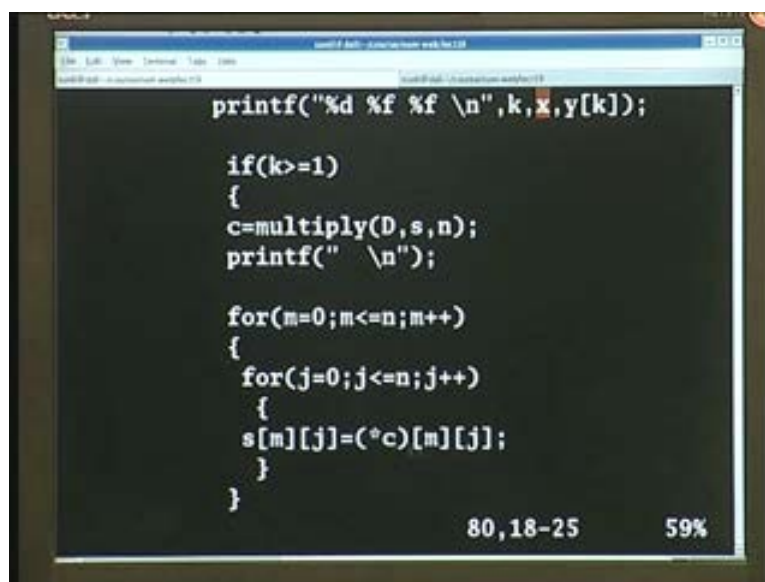
$$B_1 = [A](B_1 - p_1 I)$$
$$= \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

Okay please note that, we look and that earlier I am just showing that any implementation are use of that routine here. So I just pass these matrices d and s to this function, okay and returned to two dimensional array. So we have basically doing this multiplication here, so we doing the multiplication of the matrix d and s. Okay that is what we looking at, so that is this part ok we are constructed d and now I looking b_3 minus p_3 into i minus a. So first I am doing a into b_3 , okay that is what a into s because its remember s is storing the b_3 . Okay, so that is what is done here and we get the product.

okay then I store the product into the new s now so now b_3 is being now replace by the product of the d into s, that is matrix d and b_3 , ok that mean replace by the product, ok s was the b_3 matrix remember and that is now replace by the products, s_0 I am just equating all the elements, to the elements of the matrix c, so elements of matrix c being two dimensional array of pointers is read out like that, this in brackets in archery c and m j and m j elements and so that this gives matrix s the values of the this out this points the value of the m j the element of the matrix c right. So that is been so we equating that s, so s was b_3 to start with in this loop as s gets new value that is d and s, so now we construct the second product that is been need to construct to p_3 into I, okay that has been done here.

Okay, so then I going again and then do the product d and i that is matrix a and i identity matrix right, the identity matrix now remember, it was identity matrix to start with, ok so now the idea is that all the diagonal elements of the identity matrix, identity matrix now the replace by y of k, okay so now y of k is remember was the p_3 right. Okay I replace so y of k, this is been as is p_3 , k is 3 here. Okay start with k equal to n, so k was 3, okay so now $k_3 p_3$ is now y of 3. Okay, so now that is p three into i right is now simply replace now diagonal elements of this matrix by that element remember, I constant into a matrix into identity matrix so the diagonal elements goes into value p_3 .

(Refer Slide Time: 33:45)



```
printf("%d %f %f \n",k,x,y[k]);

if(k>=1)
{
c=multiply(D,s,n);
printf(" \n");

for(m=0;m<=n;m++)
{
for(j=0;j<=n;j++)
{
s[m][j]=(c)[m][j];
}
}
}
```

80, 18-25 59%

Okay then, I subtract this matrices okay is two matrices had subtracted that this is the product of d and b₃ and this product of now, just saw so d and identity matrix multiply by p₃. So that is, subtract and then we have the matrix b₂. Okay now this looks tedious here but now once we have returns one round and then rest of thinks follows right. So that is now the same think, so we have constructed b₂ and now we go back one we have done that matrix b₂ which is print out the matrix b₂ here.

(Refer Slide Time: 34:08)

$$= \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} -6 & -1 & 0 & 0 \\ -1 & -6 & -1 & 0 \\ 0 & -1 & -6 & -1 \\ 0 & 0 & -1 & -6 \end{bmatrix}$$

$$= \begin{bmatrix} -11 & 4 & 1 & 0 \\ 4 & -10 & 4 & 1 \\ 1 & 4 & -10 & 4 \\ 0 & 1 & 4 & -11 \end{bmatrix}$$

(Refer Slide Time: 34:26)

```

./[sunil@dali lect19]$ ./a.out |more
The matrix
2.000000 -1.000000 0.000000 0.000000
-1.000000 2.000000 -1.000000 0.000000
0.000000 -1.000000 2.000000 -1.000000
0.000000 0.000000 -1.000000 2.000000
3 8.000000 8.000000

0 -11.000000 4.000000 1.000000 0.000000
1 4.000000 -10.000000 4.000000 1.000000
2 1.000000 4.000000 -10.000000 4.000000
3 0.000000 1.000000 4.000000 -11.000000

2 -42.000000 -21.000000
--More--

```

Okay that is I want to do here. So just print out the matrix b₂ here, okay and we can see that is the matrix just print out. So that is 114, 14, 104 minus 104, 114 minus 104 and 14

minus 11. So that is a matrix b2 which here got and then where compute the trace of this, the trace of this is 42 that is a trace that is what we put it here. So n minus 2, it is so this pn minus 2 and so the pn minus 2 and then once we have construct this matrix simply going back from here this loop n is here okay.

(Refer Slide Time: 35:22)

```

{ I[i][i]=y[k];}
c=multiply(D,I,n);
for(m=0;m<=n;m++)
{
for(j=0;j<=n;j++)
{
s[m][j]=s[m][j]-(*c)[m][j];
}
}

printf(" %d %f %f %f %f \n",
m, s[m][0],s[m][1],s[m][2],s[m][3]);
}
printf(" \n");
}
1
93,18 72%

```

So we back here right, we go back here in this loop with the k loop and we will here, so we will we going back into this loop. Okay up to we print of this matrix b₂ matrix now b₃ gone to b₂ all s is replace by b₂, b₂ okay it was b₂ before now, 4 now become b₂ and then we go back this loop n is here and go backs and compute and trace okay.

(Refer Slide Time: 36:53)

this gives $trace(B_1) = -42$ and we get

$$B = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} -11 & 4 & 1 & 0 \\ 4 & -10 & -4 & 1 \\ 1 & 4 & -10 & 4 \\ 0 & 1 & -4 & -11 \end{bmatrix} \begin{bmatrix} -21 & 0 & 0 & 0 \\ 0 & -21 & 0 & 0 \\ 0 & 0 & -21 & 0 \\ 0 & 0 & 0 & -21 \end{bmatrix}$$

$$= \begin{bmatrix} 16 & -3 & -2 & -1 \\ -3 & 14 & -4 & -2 \\ -2 & -4 & 14 & -3 \\ -1 & -2 & -3 & 16 \end{bmatrix}$$

Now, once we have the trace now we have to divide it by $n - k + 1$ that will be now two, okay now a replace divide by 2 because i in this program is 2. So the p_2 is the trace of b_2 divide by 2, so that is what to going to be get here. So that will be that is what to be printed out here. So that will be the trace is 42 and the divided by 2 that is 21 the next coefficients is 21, okay and then we can repeat this process and we can, we will continue to construct b_1 okay b_1 will be again the product minus p now, p_2 into i p_2 is 21 so p_2 into i , okay and then we keep we keep doing this until we get all the coefficients in this form.

Okay that is what to be going to do and we will get all the coefficients here, okay so that is what to program this so this program simple repeat the loop will reach b_0 , okay and will construct all the coefficients one by one okay so that is what will be we will doing because of the coefficients store the 1 and then we have 20 and then will have 5 okay and that is for it ends. Now notice that one more point, then we reach b_0 right in which b_0 we have all the diagonal elements has minus 5 and every think has 0 right. So we can see that the b_0 matrix divided by p_0 is it identity matrix, so that is what I said you will use this that is what I am sterling you that we can use the last coefficients that when it reaches b_0 .

Okay which as simply b_1 minus p_1 into a that was b_0 right, that b_1 minus p_1 i into a that is b_0 now, we saw that b_0 divided by p_0 is identity matrix right. So that means what, that means b_1 minus p_1 i is, if i b_1 minus p_1 i divided by p_0 is inverse of a because this has to be identity matrix this has to be inverse of a. So we can see that a b_1 minus p_1 i is p_0 is inverse of a. So this process can also be used to compute the inverse of the matrix, right inverse of this matrix of this form, okay so we can construct that that two inverse of the matrix a we constructed not by using adjoin a by determinant a into a but by using this techniques of fadeev and laveera, okay so that what to be we have seen here.

(Refer Slide Time: 38:45)

The trace is now given by $\text{trace}[B_1] = 60$ and
 $B_1 = A(B_0 - p_1 I)$

$$B_1 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} -4 & -3 & -2 & -1 \\ -3 & -6 & -4 & -2 \\ -2 & -4 & -6 & -3 \\ -1 & -2 & -3 & -4 \end{bmatrix} = \begin{bmatrix} -5 & 0 & 0 & 0 \\ 0 & -5 & 0 & 0 \\ 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix}$$

This clearly agree with the relation

$$[A]^{-1} = \frac{1}{p_k} [I B_k - p_k I]$$

The characteristic polynomial is

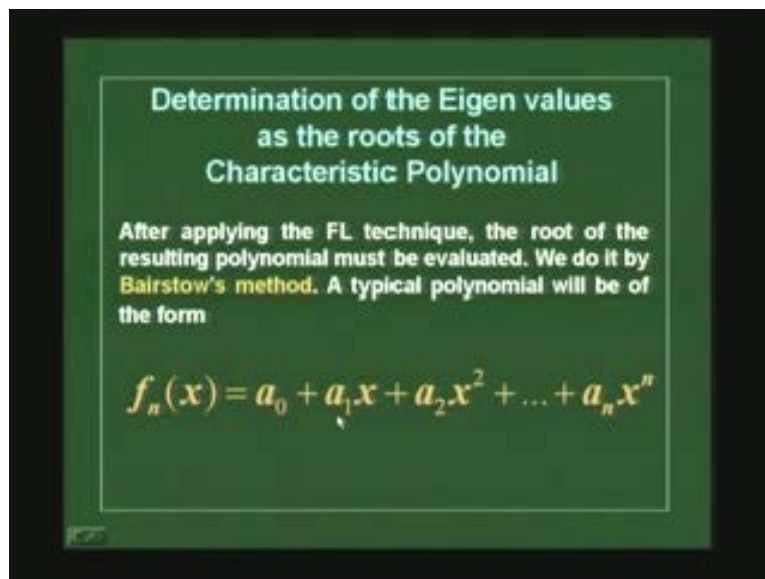
$$\lambda^4 + 8\lambda^3 - 21\lambda^2 + 20\lambda - 5 = 0$$

So now, we have the characteristic polynomial and the coefficients of lambda 4 as 1, 8 minus 21, 20 and minus 5. Okay, so we have this coefficients, we have polynomial and now what we have to do is just go add and solve, okay now that is how solve non trivial exercise because n is the very large, we will have go to will have to large polynomial and then we have to find out all the roots of the polynomial this polynomial n is the order of polynomial will have n roots and we are to determinant all the roots.

So now why will use one techniques, if that all this we will use one technique that is all the coefficients of this polynomial are real numbers then the roots has to the come its root should be either real or pairs of complex sequential consequential. Okay that is what going to use, we want to use that only look at that special case were we have all the coefficients of this polynomial has real. So and we will try to solve this equations by using, what is calls the Bairstows method, that is what to be the second part of the polynomial has for the eigen determination which is the looking at now.

Okay so now we saw that typical polynomial can be return in the in the form, we know that write a typical polynomial can be return as f_n of x is the a_0 plus $a_1 x$ up to an of x power n okay we are looking at k is the x is first our eigen values is call is equal lambda add this for coefficients call as p etcetera ok so that is same. Okay now, the roots as come spars, okay so since, we know the roots come as spars and spars of because of coefficients are real and root comes spars of come to consequential always come on spars okay, so we can use that we can make use of that have the say that the polynomial since in the roots come in spars.

(Refer Slide Time: 40:20)



Okay the polynomial should be always divisible by a quadratic polynomial of this time x square minus r_x minus x. So what we saying is we take this polynomial we want to find out want to roots of the polynomial but we know that, roots come in spars, okay I can say that this polynomial should be divisible by acrobatic polynomial. So I constructed

arbitrary polynomial of this form $x^2 - rx - s$. I constructed the quadratic polynomial, I divide original polynomial f_n order n polynomial by this and then I will get polynomial of order $n - 2$ right and if I just take arbitrary quadratic polynomial of this form with sum coefficients are r and s and if I divide my polynomial order of n by that okay I will get for a polynomial of order $n - 2$ and I will get a remainder.

(Refer Slide Time: 42:31)

In order to permit the evaluation of complex roots , we divide the polynomial by a quadratic factor, $x^2 - rx - s$. This would result in a new polynomial

$$f_{n-2}(x) = b_n x^{n-2} + b_{n-1} x^{n-3} + \dots + b_2$$

with a remainder $R = b_1(x - r) + b_0$

(Refer Slide Time: 43:02)

A simple recurrence relation can be used to perform the division by the quadratic factor:

$$b_n = a_n$$

$$b_{n-1} = a_{n-1} + rb_n$$

$$b_i = a_i + rb_{i+1} + sb_{i+2} \quad (\text{for } i = n-2 \text{ to } 0)$$

If the coefficients of the original polynomial are real, complex roots occur in conjugate pairs.

Okay the remainder which is the b_1 into x minus r plus b_0 the first order remainder. So that is what go to remain, okay so you have, if a polynomial under the remainder and then what we need to do is determined this b and b and minus 1 up to b_2 right and b_1 and b_0 would

be minder so know this is exact divisor, ok now, okay that is if square minus r_x by n says this quadratic polynomial is exactly divisor of the n th order of polynomial which we have and then this part of the would be 0, r would be 0 right okay so r will be 0, so what we would do is determinant this r_n and s such that it is no remainder is left right.

So we need this d_1 d_0 to go to 0, we have to choose r and s such that d_1 and d_0 is equal so that is the whole idea behind this method, so and because we know how to construct the polynomial, if you know the coefficients of this polynomial. Okay that is, a is original polynomial coefficients it return as a you know the all the coefficient is polynomial and then if I choose i and r and s I know what the values b_1 b_n minus 1 and b_2 or I can I can write down this equation for this okay and I can then, I can also get b_1 and b_0 from that and then make sure I can tune my r and s is that goes to 0, so that is what we going to do. Okay here is simple regression relation for again for that coefficients, so we have b_n as the a_n okay b_n minus 1 has a_n minus 1 plus r b_n and b and b_i in general as a_i plus r b_i plus 1 plus s b_i plus 2.

Okay so now, remember what is, what b_n or b_n is r the coefficients of the polynomial which obtained after dividing the original polynomial by the quadratic polynomial that is we go back again that and then please take note of this there is we have this polynomial which had a is 0 has a coefficients of the x_0 term and a_1 is coefficients x to the power 1 term a_2 has coefficients of x to the power of a_2 term are in general, a_n was the coefficient of x to the power n term right and then divided by this quadratic polynomial and then we have b_n as the coefficient of n minus 2 power term.

Okay, so that is what please note that now we return has the notation of that b_n is now the coefficient of the n minus 2 power term. Now we will divide this further, okay now we will choose this as our original polynomial this will become a_n minus etcetera minus 2, a_n minus 1 etcetera and 1 divide again by quadratic and then would becomes the coefficient of the highest power become n minus 4 will continue that.

So the notation with be now the coefficient of the x_n minus 2 term, so once we have loop and then we continue loop and determinant all the coefficient, all the roots right, so ok so that is, what writing here. So we are writing b_n is equal to a_n that is the coefficients of the x the power n into is the same has the coefficients of the x power of n minus term obtained after dividing the n th order polynomial by quadratic polynomial and then n minus 1 to term which the coefficients of n minus 3 term that is b_n minus 1 is the coefficients of the x the power of the n minus 3 term in the polynomial obtain after dividing that polynomial, that is equal to a_n minus 1 plus r times b_n r is remember again is the coefficients of the linear term which we use to divide the quadratic polynomial, we use to divide the polynomial okay so please note down this.

(Refer Slide Time: 45:18)

In order to permit the evaluation of complex roots ,
we divide the polynomial by a quadratic factor,
 $x^2 - rx - s$. This would result in a new polynomial

$$f_{n-2}(x) = b_n x^{n-2} + b_{n-1} x^{n-3} + \dots + b_2$$

with a remainder $R = b_1(x-r) + b_0$

And then we have, if so we can construct this this series of the series of that, so for we example b let see we have n th order of the polynomial so n equal to 4, so b_4, a_4, b_3 is a b_3 plus r times b_4 etcetera and we can keep on and then n minus two onwards established use this equations for n and n minus 1, we will use this n minus 2 onwards we will uses, ok so again, so we assume the solutions is come to complex complicate spars, so the solution would be now obtained now by the simply so what to be done the theorem done the then we will right polynomial f n order s has polynomial f n minus 2 multiply by x square minus r, x minus s that is equal to 0.

(Refer Slide Time: 45:46)

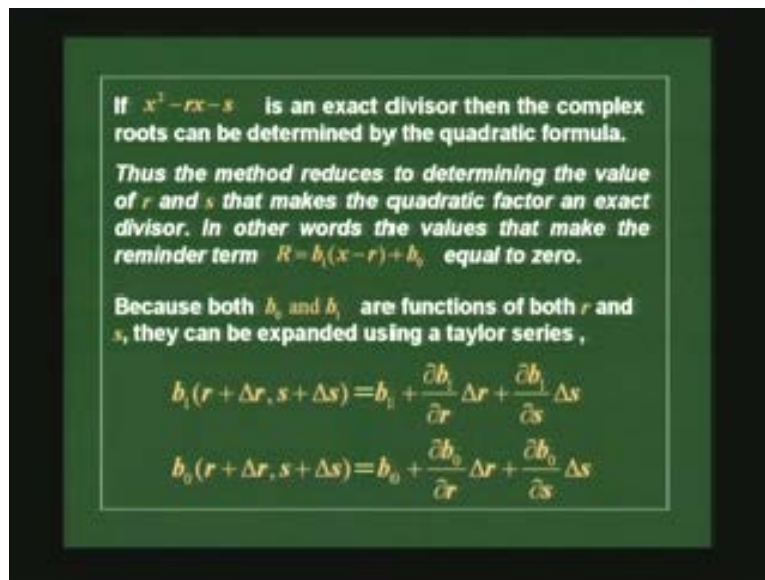
A simple recurrence relation can be used to
perform the division by the quadratic factor:

$$b_n = a_n$$
$$b_{n-1} = a_{n-1} + rb_n$$
$$b_i = a_i + rb_{i+1} + sb_{i+2} \quad (\text{for } i = n-2 \text{ to } 0)$$

If the coefficients of the original polynomial are real,
complex roots occur in conjugate pairs.

So the solution will be obtained by $x^2 - rx - s = 0$ and we know how to solve this quadratic equations, so we get two roots, so we got two roots solutions to the n th order polynomial equal to 0. So we took the n th order of polynomial and factorized it is n minus order of polynomial and quadratic polynomial. So we have polynomial of order n multiply by quadratic polynomial equal to 0 and then, we say that quadratic polynomial that have m is equal to 0. Okay and that gives a solutions the two roots and we can continue this till this polynomial becomes order 1 and order 2 depending upon the original polynomial was even or odd. Okay so that is what we have, ok we will get more idea of the actually implemented in this in a code.

(Refer Slide Time: 50:06)



So we have all the root, ok now once we have the roots, okay so we have the b_n is okay now what have to what be saying is that is this exact divisors right then, the remainder that is $b_1 x$ minus r plus 0 should be 0. So we just choose the arbiters r n is divided the n th order polynomial because the n minus 2 n th order polynomial with the coefficients as b_1 and b_2 have we go to the remainder which is $b_1 b_0$ right, but this is exactly remainder is 0.

So we have to choose r and s such that b_1 and b_0 here, again will go back to the kind of iterative scheme which have to used I, in the case of a fetching a non-linear equations fetching a non-linear function to a set of data of points. So we will use same techniques see here, so that is we will say that is b_1 of r plus delta r as us a b_1 and b_2 should be equal to 0, okay so now what we do is will expand this polynomial. Okay so we will expansion polynomial in using a Taylor serious is okay and then we will write b_1 right, then we will expansion coefficients of b_0 and b_1 which should 1 which should be actually 0, if the exactly divisors for which use arbitrary r and s values they won be 0 right so but that be same that be our choice is near perfect ok almost correct it we are we have very clos to b_1 , b_0 , b_1 and b_0 would be small okay then I would expand b_1 and b_0 around the r and s values which we are chosen okay and get in new b_1 and b_0 .

A new b_1 and b_0 that is, new b_1 and b_0 closed to r and s which we are chosen r plus Δr and s plus Δs should be, return as b_1 plus Δb and Δr Δs Δb Δw and Δs and similarly, for b_0 . So I remember the idea is that, if our choosing the correct r and s even and b should be 0 okay but of not 0, I should be able to go in round the value chosen r and s , okay such that I go to b_1 and b_0 equal to 0, so that is what I am do here. So I have, I am just going around that, I am changing r by Δr and s by Δs , so a new values for r plus Δr and s plus Δs and what is mean new b value is there and that is given by this the tailors expansion and what is my new b_0 values there that is given by this tailors expansion.

(Refer Slide Time: 50:28)

The changes Δr and Δs , can be estimated by setting the RHS of above equations to zero.

$$\frac{\partial b_1}{\partial r} \Delta r + \frac{\partial b_1}{\partial s} \Delta s = -b_1$$

$$\frac{\partial b_0}{\partial r} \Delta r + \frac{\partial b_0}{\partial s} \Delta s = -b_0 \quad (1)$$

Okay, so if the Δr and Δs is correctly chosen the left hand side should go to 0 that means I have the equations that b_1 plus Δb and Δr Δs Δb Δw and Δs equal to 0. So then, that is what I would have, okay so I have this equations and I can solve this equations, solve this equation because Δb and Δr , if I can determined Δb and Δr and Δb and Δs and Δb_0 by Δr and Δb_0 by Δs . Okay I can determined the new Δr and Δs values and I can go to new values of r and s . So the solution will be obtained by x^2 minus r , x minus s equal to 0 and we know how to solve this quadratic equations, so we get two roots, so we got two roots solutions to the n th order polynomial equal to 0.

(Refer Slide Time: 51:15)

A simple recurrence relation can be used to perform the division by the quadratic factor:

$$b_n = a_n$$
$$b_{n-1} = a_{n-1} + rb_n$$
$$b_i = a_i + rb_{i+1} + sb_{i+2} \quad (\text{for } i = n-2 \text{ to } 0)$$

If the coefficients of the original polynomial are real, complex roots occur in conjugate pairs.

Okay that is the idea and I can determined this because I know, I know how to write that because I know how the b is dependant r and s. Okay, I have this equations okay since I have the equations I can determined I know how does the coefficients for example depend on r and s so I so b_1 will a 0 plus r b_2 plus $s b_3$ right, so del b_1 by del r will be b_2 and del b_1 by del s is b_3 and similarly del b_0 by del r will be b_1 and del b_0 by del s and del b_2 and that is what can use so I can use that and substitute here in this equations, okay I can that set of linear equations, okay and I can solve it.

(Refer Slide Time: 51:57)

$$c_n = b_n$$
$$c_{n-1} = b_{n-1} + rc_n$$
$$c_i = b_i + rc_{i+1} + sc_{i+2}$$

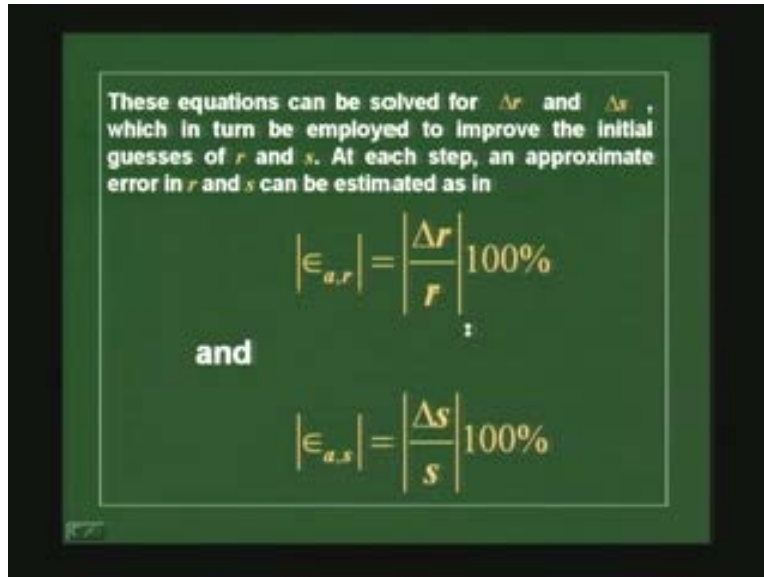
for $i = n-2$ to 1

After using the above obtained values in equation set (1) we get

$$c_2 \Delta r + c_3 \Delta s = -b_1$$
$$c_1 \Delta r + c_2 \Delta s = -b_0$$

Okay, so I have this equations okay and then I can just like that from here so I return that special case this is equal to 0 and then I can use the equations and differentiate with respect r and s and I can get this coefficients here. So since the coefficients of this column, all the column c but then I can determine this coefficients are from this equations which had and then I get that left hand side and solve this equations for delta r and delta s because note this one b₀, so that is the idea.

(Refer Slide Time: 52:32)

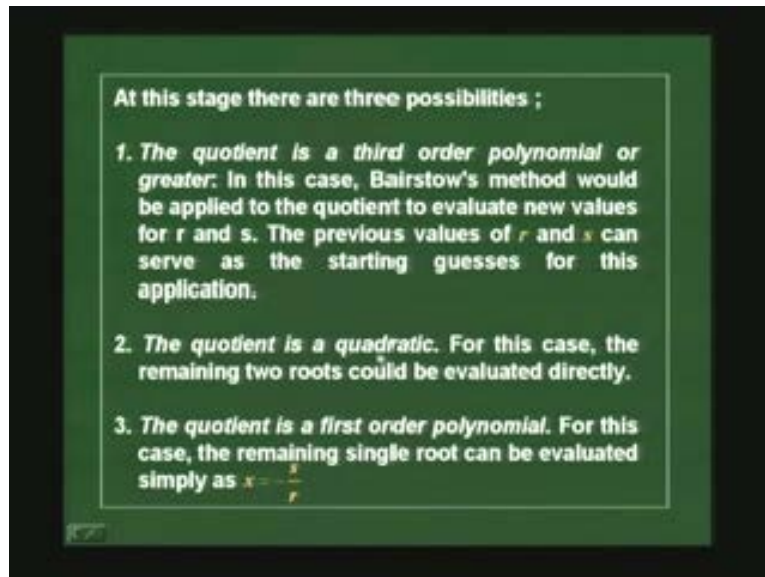


Okay so then, we will get new delta r and delta s, okay and then good compare what is the delta r and s, now we obtained, okay related to the old r, ok new r_1 b r plus delta r will whole r will be whole new s will be, s plus delta s. So the relative error one iteration step will be delta r by r into 100 occurs. So percentage error and then we could put some tolerance for this.

Okay whenever, we goes below sum value would stop and then that way can produce all the coefficients, so we get the solution r then r and s so one way the coefficients of the quadratic equations. So we have then x minus rs plus sorry the x minus rx plus s equal to 0 as equation and then from that we can determinant the first two rows and then we have n minus 2 row, 2 order polynomial left, and then n minus 2 order polynomial again divide it by another quadratic polynomial right, and then we will go to n minus 4 order polynomial into a quadratic polynomial and that from again get 2 rows.

So we can continue this determinant all the roots, okay so that, what we then once a done 1 loop that is three possibility left the coefficients is the third order polynomial, so that is if you started with, the continue with loop right at the finally, we ends up with this, that the last think would be that will be reach third order polynomial if you started with a if you started with odd number, then we will end up with third order of polynomial.

(Refer Slide Time: 54:31)



So we can do it one more round, how then we can get the values of r and s for that of the quotient is a quadratic, okay if we started with even order of polynomial and then enter with a quadratic equations okay and then we do not need to divide it further right, and we have that means the last equation we get is a product of two quadratic equations and we can easily determined the roots, all the roots the quotient is a first order of polynomial that is, one of this can okay right the end of the third order of first of first order of polynomial and then remaining single root is the simply x equal to minus s by r .

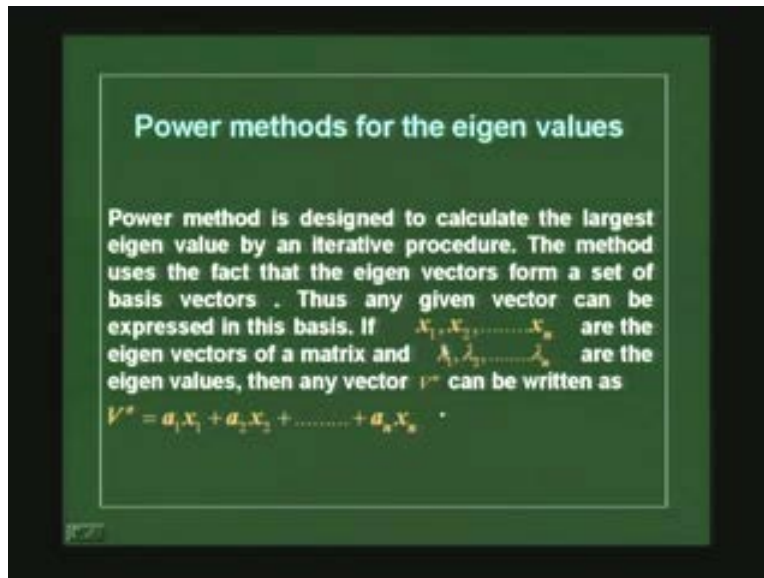
So in the case of fourth order polynomial which we discuss, we does need one division right because once we have divided one round, okay we have only quadratic polynomial left, ok and then that means, we write the fourth order, we use to Bartow's method and we have fourth order polynomial which we have discussing. We can write that as product two quadratic polynomial means, we have all the solutions that is what something was would be looking at, we will implemented that in a code and we will actually evaluating the coefficient.

This is what the only way of the determining the eigen values, there are other methods to determine the eigen values also eigen function also, we will discuss some of those methods briefly not in so much details and briefly we will use, we will see for the example of the what is call on power method put it determining eigen values and eigen functions and this method actually uses that fact that the eigen vectors of form what is call complete bases that is if you matrix and if you those all the eigen vectors of the matrix, they form a complete bases, that is any other vector we can express okay, in terms as a linear combination of the this vector each of this x_1, x_2, x_n .

I am mentioning here for an n th order matrix is a vector by the column vector x_1 is a column vector, x_2 is a column vector etcetera and they form complete base with eigen

values $\lambda_1, \lambda_2, \lambda_3$ etcetera, if you given, any arbitrary vector V , okay I can right that as combination, linear combination function of this of this set of matrices, the set column vectors x_1, x_2, x_3, x_n and so and now, it is take a matrix and multiply matrix decide matrix a multiply this vector by there. Okay and then we will get in equation like this is in, new vector we want would be matrix A time b_0 plus a_1 into because express b_0 as $a_1 x_1$ plus $a_2 x_2$ to $a_n x_n$.

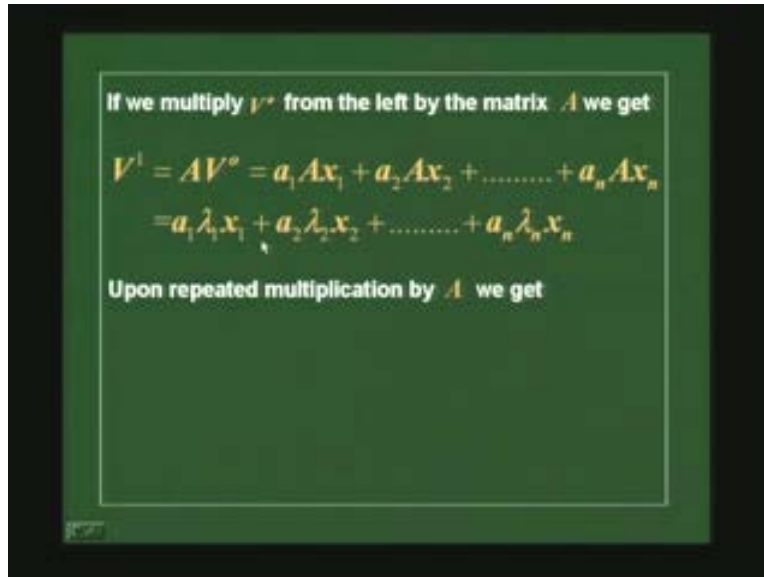
(Refer Slide Time: 56:14)



So when you multiply by A then will you get in equation of this form but we know that x_1, x_2, x_3 are eigen vectors of the matrix A , that means $A x_1 = \lambda_1 x_1$ and $A x_2 = \lambda_2 x_2$ and like it has $\lambda_2 x_2$ etcetera. So I will get equation of this form and right hand side so had an arbitrary vector we know that which has express in terms of eigen vectors, then I multiply that matrix A who is eigen vector to find we and find and we got a matrix of this form.

So now this what see this the repeatedly multiply this matrix, okay this vector by this matrix A , okay and then we will go to λ_1^n, λ_2^n etcetera and then if this λ_1 is ordered correctly, the only term which would left would be the highest power λ_1^n .

(Refer Slide Time: 57:40)



If we multiply V^o from the left by the matrix A we get

$$V^1 = AV^o = a_1Ax_1 + a_2Ax_2 + \dots + a_nAx_n$$
$$= a_1\lambda_1x_1 + a_2\lambda_2x_2 + \dots + a_n\lambda_nx_n$$

Upon repeated multiplication by A we get

So that way we can get the vector the simply taken as arbitrary vector multiplying repeatedly by matrix A we can get the eigen vector corresponding to the largest eigen values and that is call power method and we will see the use of that are implementation of that in the, in the next lecture.