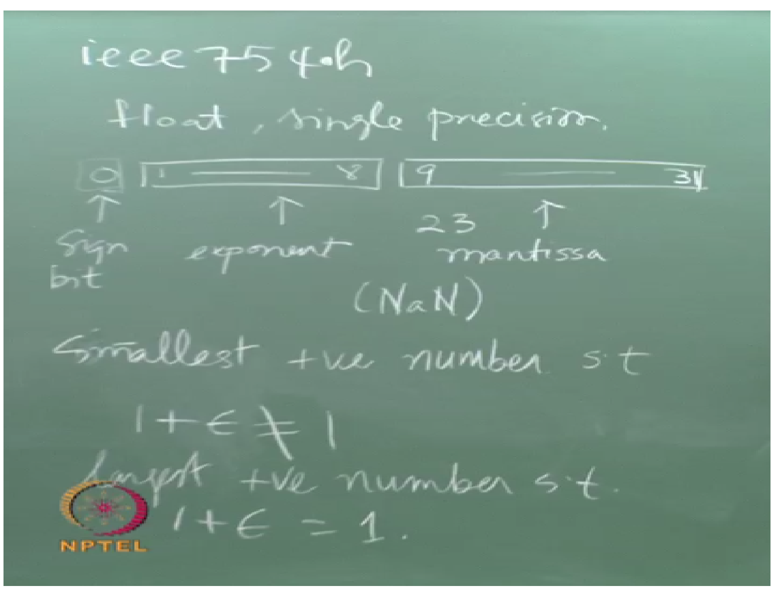
Introduction to Computational Fluid Dynamics
Prof. M. Ramakrishna
Department of Aerospace Engineering
Indian Institute of Technology – Madras

Lecture – 02
Representing Arrays and Functions on Computers

So this morning we will continue with representation of numbers. Let me just get back to where we were, what we did in the last class. We saw that computational fluid dynamics that we are going to solve differential equations that come from fluid mechanics. The solutions to such equations are functions and these functions in ordered for us to be able to solve these equations both the differential equations.

And the functions had to be represented on a computer that is the idea. So to that end they decided to represent various mathematical entities on a computer. We started the way that you have learnt calculus that we started by saying that let us try to represent to the real line on the computer and what we have basically done now is we have tried fixed point. We have done integers. We have finally come down to floating point. I have asked you to look up by triple E 754 standard. I gave you what is used for what is the format that is used for single position.

(Refer Slide Time: 01:27)

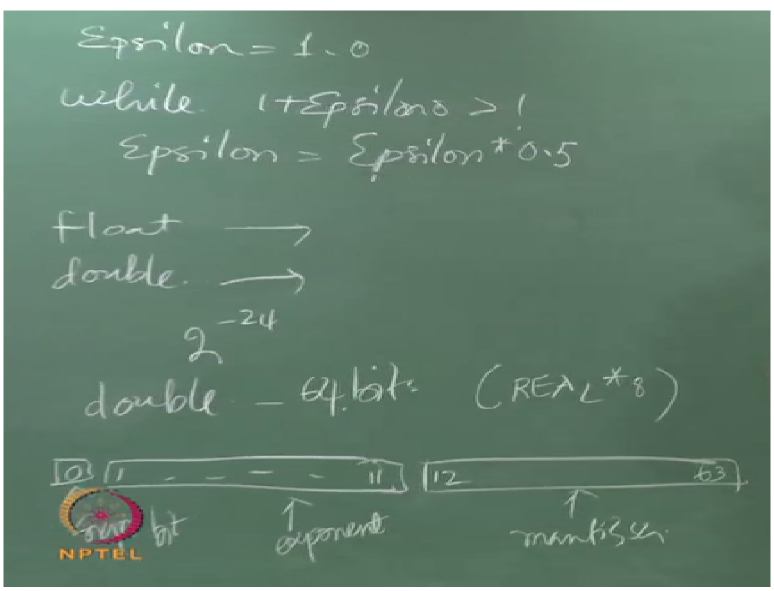


Single position for a float or in fact I asked you to look up this file on your computer, locate where it is a look up this file on your computer, and just to recollect for float or I use the word now a single precision as suppose to double precision. So we have indicated that the first bit was the sign bit. We used the next 8 bits 1 through 8. We used them for the exponent. Again I remind you that please go look up Big Endian and Little Endian.

Then the remaining how many are there 23 bits going from 9 numbers till 31. Please remember we started the count at 0. Remaining bits are typically used for the mantissa. It is possible that if you come across something called a NaN not a number we will see that may be there are 22 bits and we will look up the standard we will find out about it, but I am not going to really talk about this now so they have 23 bits which are the mantissa.

To this end I had asked you to try out an experiment which I hope that you guys have tried which is to ask the question is there a positive what is the smallest positive epsilon, smallest positive number such that, that is epsilon 1 + epsilon smallest positive number such that 1 + epsilon is different from 1. I also said that there is an alternate definition. You can also look for the largest positive number such that 1 + epsilon = 1. Is that fine? So that is the possibility. So, we have written the piece of symbol code for this. I had asked you to try this out.

(Refer Slide Time: 03:50)



So you let epsilon in fact I think yesterday I had used different symbol for this I had used that. Epsilon = 1 as a start and you can say while 1 + epsilon this is a candidate epsilon is > 1 it is != 1 or you can also say != 1. Divide epsilon by 2. This is something that I had suggest that you do try it out and you know if you have tried it out you will see that you will have a variety of experiences that you guys would have had.

Some of you possibly for float double and long double. You may have got different answer for these, but depending on your experience some of you may have got the same answer for both of them. So you have to ponder why epsilon sometimes comes out the same for both of these and what does it mean with respect to? What is the accuracy with which we can calculate make our calculations? Is that fine? So this is single precision.

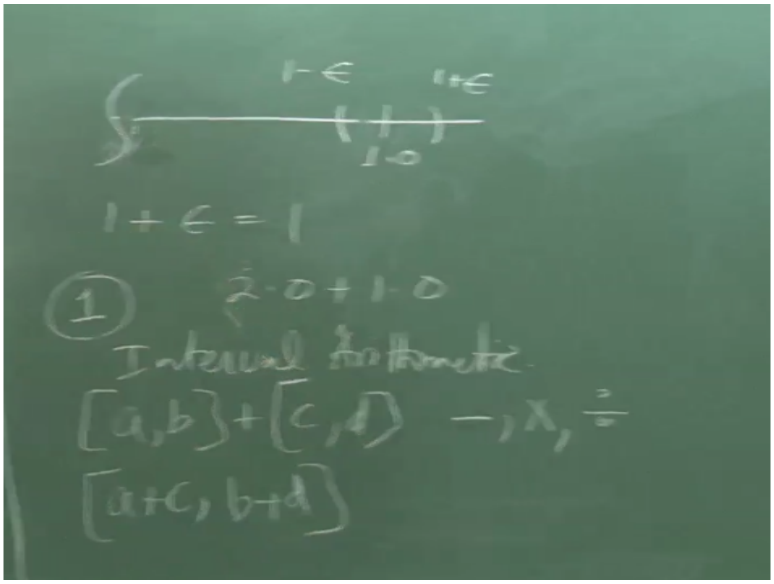
You try to relate this to the size of the mantissa that we have so that is what we are really measuring here when we do this epsilon what we are essentially doing is we are asking the question with respect to 1 what can we resolve and what we can resolve is given by the size of the mantissa. So you please check the epsilon that you get and compare it to 2 raise to the power - 24 and I have explained to you why it is 24 yesterday.

And see what is the relationship between what you have printed out and what you have got here. So what about what is the next deal? So, before I go on with the discussions since I have got that up there. Let me just tell you that there is a double which use a 64 bits. This is double precision and Fortran or real star 8 depending on what you are used to or you say REAL * 8 double precision is much longer to type.

So we would have liked it if the 32 bits that you added here had been added directly to the mantissa unfortunately that has not happened. So what happens in this standard is that you have the sign bit just like you had in the single precision and the 11 bits 1 through 11 are set aside for the exponent and the remaining how many would there be 52 bits? The remaining 52 bits 12 to 63 are set aside for the mantissa.

So this would be double precision is that fine. So now that I have told you this let step back and ask ourselves the question if we are performing computations using single precision or double precision and associated with it there is an epsilon what are we actually doing. What is the computation that we are actually performing? So let us go back to the real line that we had yesterday and zoom in on around what is that we are actually is doing.

(Refer Slide Time: 07:17)



So this is 0.0, this is 1.0 well in fact I will not indicate 0.0. 0.0 Or somewhere there I mean it is far off. I am going to really zoom in and this is basically epsilon. So this is - epsilon, 1 - epsilon and this is 1 + epsilon and what we are essentially saying here is any number here basically maps into 1 that is essentially what we are saying. That is if you give me an epsilon, if you give me a sufficient epsilon if epsilon is sufficiently small any number that could actually fit in here.

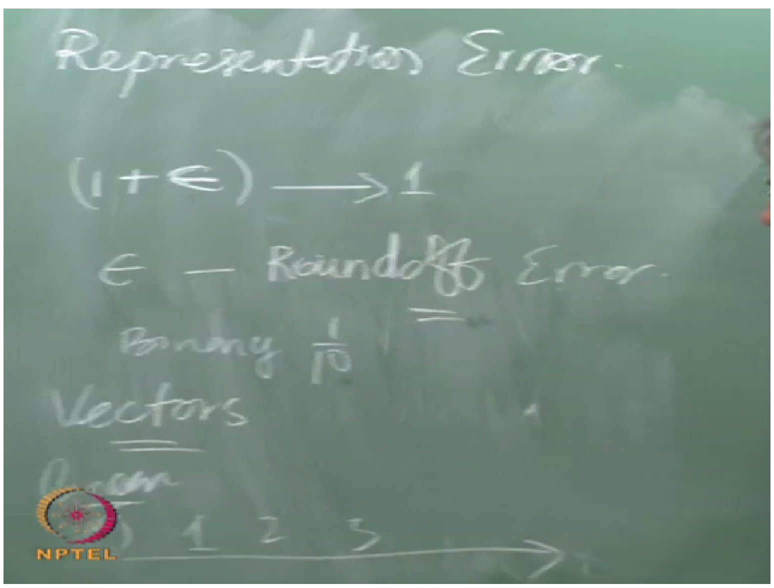
Ends up being 1 that is essentially what it says that when you say 1 + epsilon is 1 what you are effectively saying is you give me this epsilon and it is going to turn out that when I added to 1 I am still going to get 1. So any epsilon, this neighborhood is actually it is like a mush factor that I have mush area that I have here. The 1.0 that I use on the computer suppose to 1 that I use when I talk mathematics so 1.0 that I am talking about here actually represents an interval.

This represents this chalk this represents the idea of 1. This chalk dust here represents the whole interval 1 - epsilon to 1 + epsilon. So when we are adding 2 numbers together that is when we add say 2.0 + 1.0. Actually what we are doing is we adding 2 intervals. There is an interval associated with this number 2. There is an interval associated with this number 1 and we are actually adding these 2 numbers together, is that fine.

So basically the arithmetic that we are doing is not the standard mathematical arithmetic that we do, but what I would say is the interval arithmetic. There is a lot of interest in this interval arithmetic. So you have to ask the question if you have a, b + c, d what would that turn out to be. If you had a, b + c, d what would that turn out to be? So a, b + c, d will turn out to be a + c, b + d. In a similar fashion, if I were going to go with the subtraction instead of addition.

So you try the mathematical operations. Please try the operations -, *, and /. With intervals instead of actual numbers and that gives you ideas to what is the nature of the arithmetic that we are doing. So if you say that if I have 1.0 you now have an uncertainty as to actually what the original number was. You do not know whether it was 1 or something else that became 1. So we now introduce for the first time an error associated with. We now introduce for the first time the error associated with this representation.

(Refer Slide Time: 10:37)



I will call the general error since we are talking about representations on a computer. We will call the general error representation error. The representation error is the difference between the mathematical entity and its representation on the computer. So that is a very general expression. So if you have any mathematical entity and you want to represent it on the computer the difference between the mathematical entity.

And its representation on the computer would be representation error. So in this case if you had some number that was actually of the order of 1 + epsilon it would be represented by 1. That is what we are saying. It would be represented by 1 and the error that you have it is of the order of epsilon is in fact epsilon in this case if it were 1 + epsilon. So this error for numbers is called round off error. So it is a particular type of representation error called round off error.

So round-off error is a problem that we have seen. Now first we know when I say that I am representing numbers I am actually representing intervals that when I perform arithmetic I am actually doing arithmetic of intervals and if I say that I add 2 numbers a + b to c + d it is very likely that the result is actually more uncertain possible more uncertain then what we started off with.

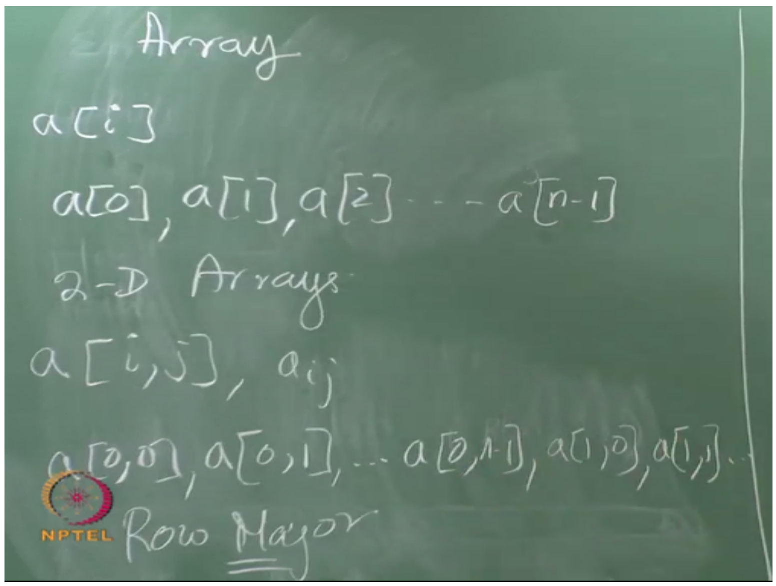
So there is this uncertainty associated with these numbers that we have and if I just take a number just represent the number then there is an error associated with it which is a round off error and in last class I already indicated that on the binary system even 1/10 is recurring because 1/2 though it can be represented exactly 1/10 has a problem because 1/5th is a recurring number and if you truncate it to 23 bits in the mantissa.

Then somewhere along the line you are going to chop the end off. So we are going to end up with what is called round off error. So I hope that is quite clear. So we have this thing now. So we have known that we cannot represent the real line exactly and we know that we have made a selection or what are the points on the real line that we are going to represent, however it seems that associated with that decision there is a connected round off error.

And we have to deal with that connected round off error. So this is as far as representing numbers goes on the computer. Let us see what else we can do. What are the mathematical entities we are interested? We are interested in vectors for instances. So if you want to represent vectors as I said I am not going to say much about computer architecture and so on however it is important for us to know that computer memory in our mind the model that we have off computer memory is linear that is.

You have the zeroth locational memory, first location, second location, third location. It is computer memory occurs in a linear fashion it comes one after the other in a linear fashion. So if I wanted to store a vector, vectors and matrices of course is what I told you that I would do in the beginning of yesterday's class, but however vectors and matrices are associated with various operations and properties. In reality what I am going to do is I am going to represent arrays, I will come them arrays just to distinguish from vectors and matrices right.

(Refer Slide Time: 14:17)



Though I promise vectors and matrices what I am actually going to give you is an array and when I say a vector what I mean is a 1 dimensional array. So 1 dimensional array ai has 1 subscript so I do not care whether it is a column or a row here right now, but because memory is linear what I will do is I will represent, I will store on the computer we will store or our complier will store.

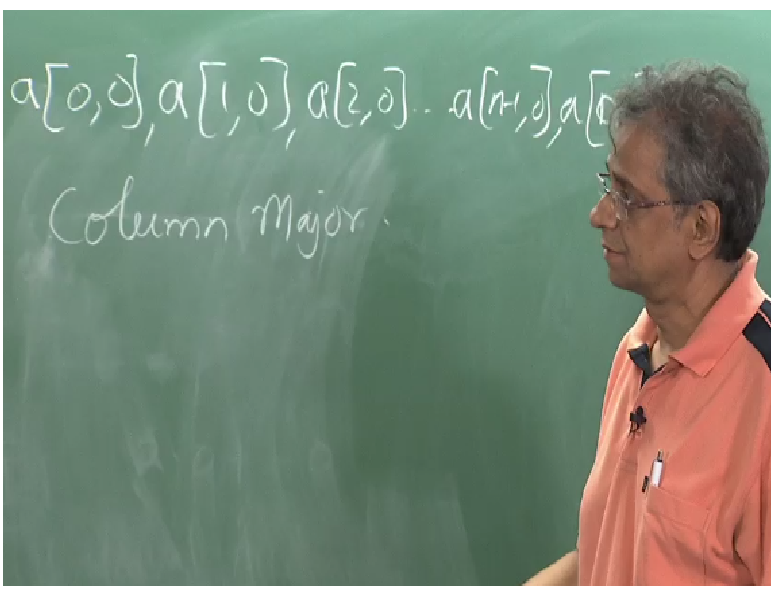
The arrangement is made automatically so if there are n of these since I am using the c notation of starting the count at 0, a 0 through an - 1 will be stored in the computer memory linearly 1 after the other is that fine. So that is the quick way of doing vectors. So but as I said there is an algebra associated with it which has to be done by you it is not going to be something that is done automatically on most computers.

What about 2 dimensional arrays? So you want matrices, but I will talk about 2D arrays. So how do we do 2D arrays? 2D arrays have 2 subscripts. So they have a I will say i, j just for or if you want a i,j so in a matrix that would be row, column and the way you would store it as you would store it either row wise or column wise in the linear memory what I mean by that is you would store a 0,0 a 0, 1 same that is the first row second column.

And so on till a is 0 n - 1 and then move on the next row which is a 1 0, a 1,1 and so on. So the idea is that in the linear fashion I will store the first row then I will store the second row and then I will go ahead and store the third row. So the model that we have of the computer memory is linear and what we will basically do is I will store the individual rows 1 after the other. This process this is how arrays in c for example lot of languages are stored.

This process is called a row major. That is the standard and as one can see if you can store it row wise you can also store it column wise so if you store it instead of the first row as we have done here and then go on to the second row instead of which if you do the first column and then move on to the second column you would get a column major operation is that fine.

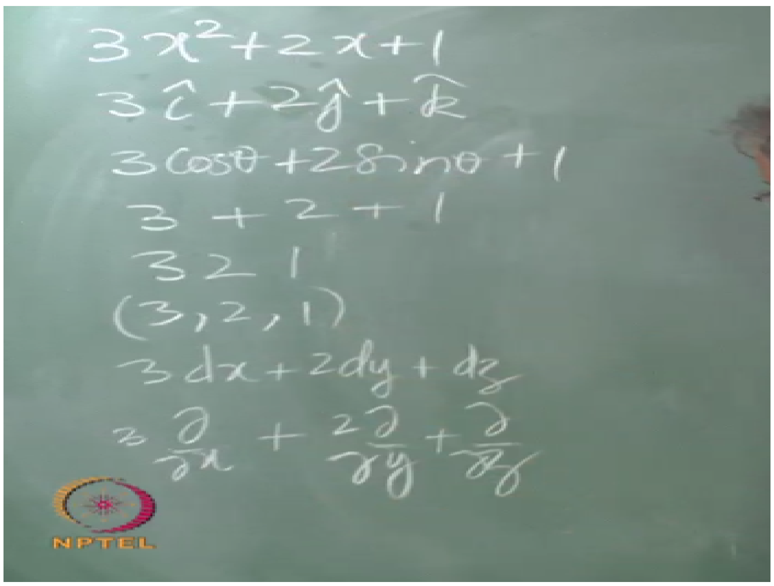
(Refer Slide Time: 17:04)



So that would be something like a 0, 0 a 1, 0 and you will notice that the column is kept fixed a 2, 0 and so on till you got to a n - 1, 0 and then you go to the next column a 0, 1, then you go to the next column and you sweep through the next column you go column by column this is called column major. The responsibility to implement either matrix algebra or vector algebra either lies with you or somebody else has done it in the form of a library.

And you should necessarily use those libraries if you want the associated algebra. So what you have shown now is integers, fixed point, floating point, floating point of various resolutions so to speak then we have arrays, vectors supposedly, then we have matrices or 2-D arrays. If you have the associated algebra you can do matrix algebra so the only thing that we have now is we have to look at functions and how functions can be represented if possible. That is the deal. I am going to put up a bunch of things now. You please see what they have in common.

(Refer Slide Time: 18:30)

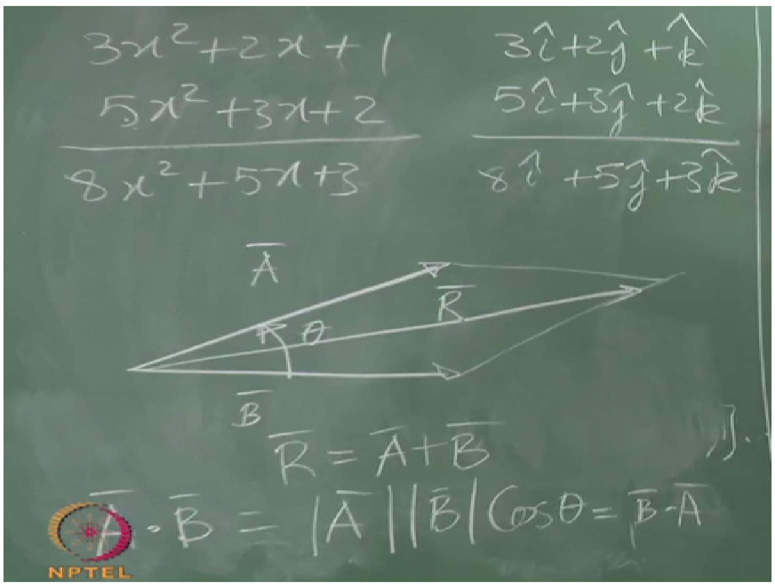


So I write a bunch of things 3 x square + 1. I will put this up right 3 I + 2 j + k, 3 cos theta + 2 sin theta + 1, 3 + 2 + 1, what else can I add, 321, 3, 2, 1 and just for the fun of it 3dx + 2dy + dz or even 3 dou/dou x I am not going to really talk about these, but I am just doing it just for the fun of it so that you get ideas to. So what do all these expressions have in common? What is that they have in common? What is different?

So you can look at this say that is a polynomial, this seems to be a vector and this is possibly terms in a Fourier series I do not know that may be a clue. This is just the addition I mean that is 6. So if you pay attention to this except for this in all of these expressions that I have written here you think that I am quite crazy if I actually added the 3 to the 2 so if you ask the question ask yourself the question what role does the I here.

Play your mathematics background will tell you about the i is a unit vector that is along the x axis and so on, but in reality if you ask the question here at this expression, this expression would add the 3 to 2. In this expression you do not add the 3 to 2 simply because the i and j are there. So the i, j, k here sort of prevent k from adding the 3 to the 2 and that role is played as well by 3, 2, 1. So in that sense, these are quite similar and if I take the combinations.

(Refer Slide Time: 20:38)



3 x square + 2x + 1 and I add to that 5 x square + 3x + 2 that really is no different from 3i + 2j + k and make sure I get the same numbers 5i + 3j + 2K. So if I add these up in deed I add these respective components here so I will get 8x square + 5x + 3 whereas here I get 8i + 5j + 3k. What is the difference? There is no difference. So basically the summation seems to work just the same as this.

So this looks like just like this is some point in 3 dimensions you would easily identify this as a point in 3 dimensions. This looks like similar to a point in 3 dimensions. So it looks like we can actually represent a function deal with a function as though it is a point in space. Is that fine? So we can deal with this function as though there are point in space. Of course, I should point out that if x were = 1 coming back here if x square = 1.

Then in deed you will get 3 + 2 + 1 and x square = 10 you would actually get 321. So there is a connection between these 2. This is the reason why I have written this. I have already mentioned Fourier series and you can as think about this and what it means? whether it makes any sense to you. It is not something that I want to get into right now, but I just I thought just throw that out there if you had a course in differential equations or linear algebra they make more sensitive.

So now we have a big thing. What it basically says is that by representing and taking an array see this is the 1 dimensional array? This is the big deal that we have done here. This is a 1 dimensional array so I can use this 1 dimensional array to represent this vector or to represent this function or to represent that function. So that is the great thing that we have achieved here. So it is a simple comparison.

But what it tells me is that it is possible for me to assemble various components and it also looks like that if I take a function now it is actually a point and some kind of a space, some kind of a function space. So we are dealing with computers and of course even if you have to do a manipulation ourselves. one of the things that we would like to do is we would like to organize any space that we have in a systematic fashion so that we could do searches, comparisons and so on.

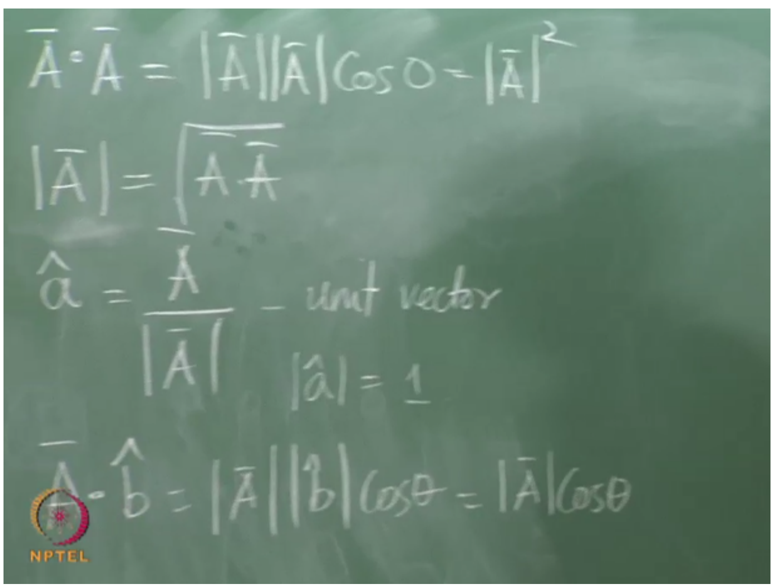
So that is there we are headed that is our objective. So let me then show you some systematic way by which we do it since I started off with vector algebra, I will show you how we do it with vector algebra and then may be in the next class it is possible we will start off with functions and how to represent functions. That is fine. So how do we do vector algebra. Where does it starts? So how what was the definition of a vector, the earliest definition of vector that you have seen.

The earliest definition of vector that you have seen is most likely a directed line segment. So if I have a vector A you most probably saw the vector A as a directed line segment. So the arrow head indicates the direction. The length of the line typically indicates the magnitude of the vector so it is a very geometrical definition. So if I have 2 directed line segments A and B then we have defined the sum of A and B using what you know already as a parallelogram law.

So the resultant R, R = A + B is given by the parallelogram law. So that is all very nice. We can do the geometry, but the whole point is to get to some kind of an algebra so that we can do the manipulation on paper rather than drawing figures and we will see how we do this. Then we can get to manipulating things on the computers so that we do not have to draw for that, that is the idea. So how do we get there?

There is another critical thing that we define, which is the dot product. A.B if you go back to your vector algebra the first time that you learnt is magnitude A, this is definition magnitude B. cosine of the included cosine of theta that angle is theta and incidentally this is = B.A if you just look at this multiplication sorry about that B.A. So A.B is the same as B.A. How does this help? What does this give up? Well this does something really neat.

(Refer Slide Time: 25:40)



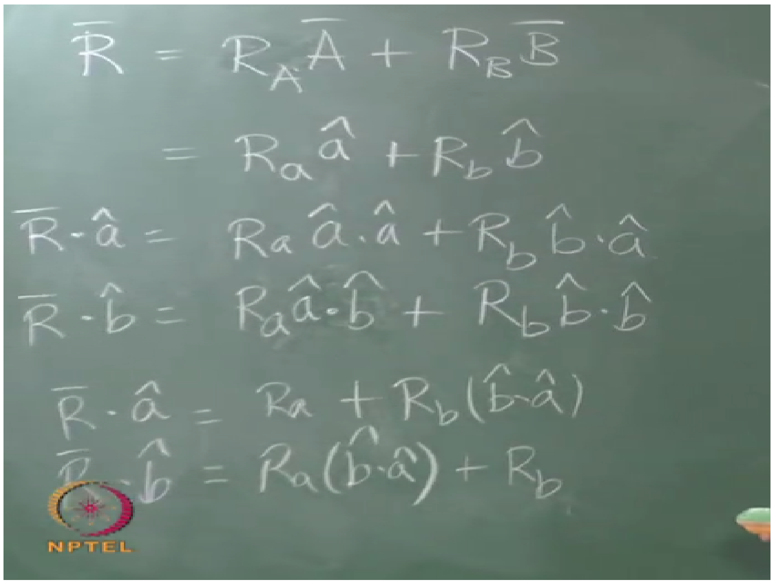
The first thing that I can do is, I can say that I can ask a question what is A.A and A.A is magnitude A * magnitude A cos of 0 the angle between A and itself which is magnitude A square. So this is neat. Now what we basically have is an expression for magnitude A, which is A.A square root. So previously we were measuring lengths at some point we will actually have to measure length so you have to get that magnitude somehow.

But right now as I long as you had the chalk dust you say it is A in the magnitude of A. That is the algebra part is given by square root of A.A and of the consequence there are need things that you can do. You can define a unit vector when I use the cap to indicate that it is a unit vector as A divided by magnitude A and as a consequence you can have something called a unit vector that is along A, but has magnitude 1.

So clearly magnitude of the unit vector, magnitude of A = 1 is that fine. So what have we got so far? So using this dot product let us try to interpret what is A.B. What happens if I take A. b hat where b hat is a unit vector along b when that is magnitude A magnitude b cos of theta. This is all a repetition of stuff that you guys know so which is magnitude A cos theta which is nothing but the projection of A on to B.

So this is a geometrical interpretation that you are aware of, but what I really want to do and this is the great thing is if I can start off with R equals A + B I get R from A and B. is it possible for me given R to write them in terms to write R in terms of A and B can I decompose it into A and B. Is it possible for me to decompose it into components, just like we have done here. As I said this is a (()) (27:55) of all the vector algebra that you know. I am going through this process simply because I propose to repeat this process for functions. So you just bare with me for that.

(Refer Slide Time: 28:07)



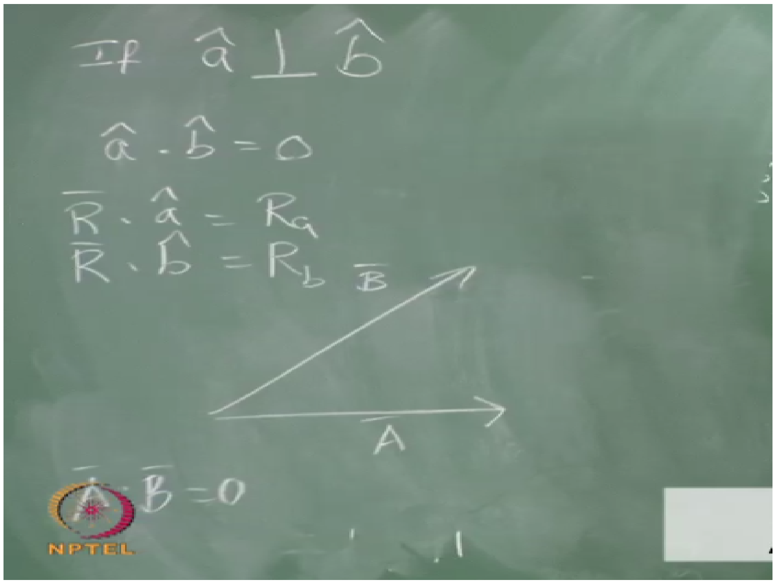
So the question is if I have R or if I have a P, if I have R so is it possible for me to write R as some R sub A A + R sub B B. This is a component along A and this is a component along B fine or is it possible for me to write it in terms of unit vectors. Note the case, this is lower case Raa + Rbb. So I will start with the second case first because that little easier. So what do we do? the only operations that we have a vector sums and dot product.

You have defined the dot product it is very clear that you can take the dot product here. So let us take the dot product of R and A. So R.A gives me Ra a.a + Rb b.a and R.b I think the dot product with b gives me Rb b.b. + I am sorry I am going mad Ra a.b + Rb b.b. So those of you have fluid mechanics with me you will recognize a.a, b.a, a.b, b.b. as metric tensor for the rest of you it does not really matter.

So basically what we have had here is what you have is if a.a is a unit vector this becomes 1. If this, they are unit vectors, but we do not know, what is the angle between them, so we are left with this. So in fact R. a this can be rewritten as Ra + Rb * b.a and please remember b.a and a.b are the same. They are just said that as part of the definition so this is Ra * a.b or b.a it does not matter + Rb.

So what we have is we have a system of equations that we need to solve for Ra and Rb and once you solve that system of equation presumably we can go back and do this representation. So that looks like a neat possibility, but of course you know that there is 1 condition where a.b or b.a is 0 and that is when they are orthogonal to each other. So if a and b are orthogonal to each other if you go back to the definition I will write.

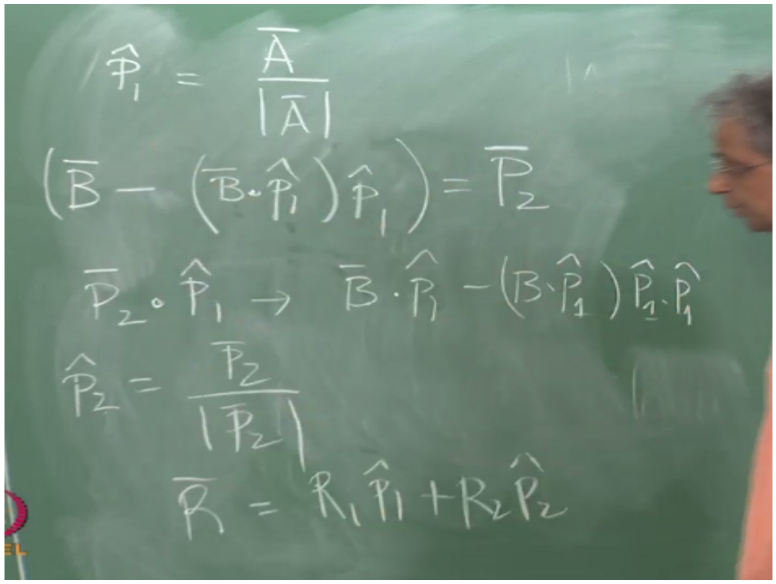
(Refer Slide Time: 30:57)



If a is orthogonal to b and in fact they are orthonormal if a is orthogonal to b. A.b = 0 so that it gives me R.a is Ra and R.b is Rb which of course tells you why we spent so much time trying to get an orthogonal coordinate system thus you get along in CFD you will see that it is done very often we desperately try to get orthogonal coordinate systems. So that is fine. So you have a situation here where a and b are orthogonal so it is Ra and Rb. What is they are not orthogonal what if the initial set not orthogonal.

So if we constrain ourselves to the board and somebody gives us 2A and Bs which are in this fashion. They are not orthogonal to each other that is you take A.B and you discover that A.B is not orthogonal is not 0 meaning that A.B are not orthogonal. I and D are not orthogonal how what do we do. So clearly of course they cannot be you do not want them to be collinear, but if they are not orthogonal and you will see why you do not want them to collinear. What you can do is you can choose the first vector so we can use P as we want.

(Refer Slide Time: 32:27)



So I will use P hat 1. This is the first vector. This vector is going to form the basis for my representation. So I want to look for an orthogonal basis. So there if a and b are orthogonal a and b form an orthogonal basis I want to try to get an orthogonal basis from here. So I will set P1 hat to in fact A/mod A may be wondering why did not? I just choose A hat, but I will show you why. What do we do now?

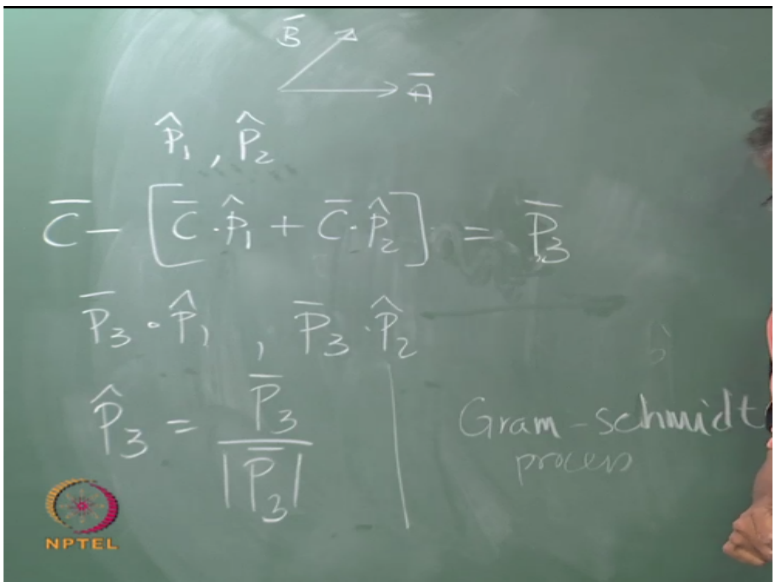
So how do I get if B is not orthogonal to A how do I get something that is orthogonal to P1 hat. So what I need to do is from B I need to take out the component that is along A. So if I find B.P1 the question is what is B.P1 is the component along P1 of B. This multiplied by P1 gives me the component basically in vector form. In fact, this would be the component along P1 so this is in the vector along P1 which is the part of B that is along P1.

And if I subtract from B, if I subtract this out right I knock this part out what I am left with is a P2 and this is a vector right not a unit vector, a vector. So the question is what is P2.P1? If you look at P2.P1, P2.P1 gives me from here just take a look P2.P1 gives me B.P1 - B.P1, P1.P1 which is 1 and (()) (34:35) these 2 will cancel giving me that P2 is in fact orthogonal to P1 hat. Therefore, my P2 hat is nothing but P2/magnitude of P2 is that fine.

So now given A and B that were not orthogonal to start with what we have managed to do is find the P1 hat and P2 hat that are orthogonal to each other and from here on we can ask the question given an R is it possible for us to find an R1 along P1 because R2 along P2 so that R1P1 + R2P2 = R, the vector R. Is that okay? Right I think what we have done right now is we have come up with the mechanism by which we are able to generate this in 2 dimensions.

What do we do in 3 dimensions what if I add 1 more dimension. Right what if I had a C that was not in the plane made up of A and B that is not in the plane of a black board, but C actually comes out of the black board so what is that we do in that case and it is not orthogonal to the black board. So what I had.

(Refer Slide Time: 35:53)



I have a B, A clearly not perpendicular to each other and I have C that is not orthogonal to the black board, but that is a angle. So what is that I am going to do in this case? So I already have I have got my P1 hat, I have got my P2 hat so from C, I need to find out that component of C that is along P2 and find out that component of C that is along P1 and subtract them out so it is the same process. So I can take C.P1 hat + C.P2 hat.

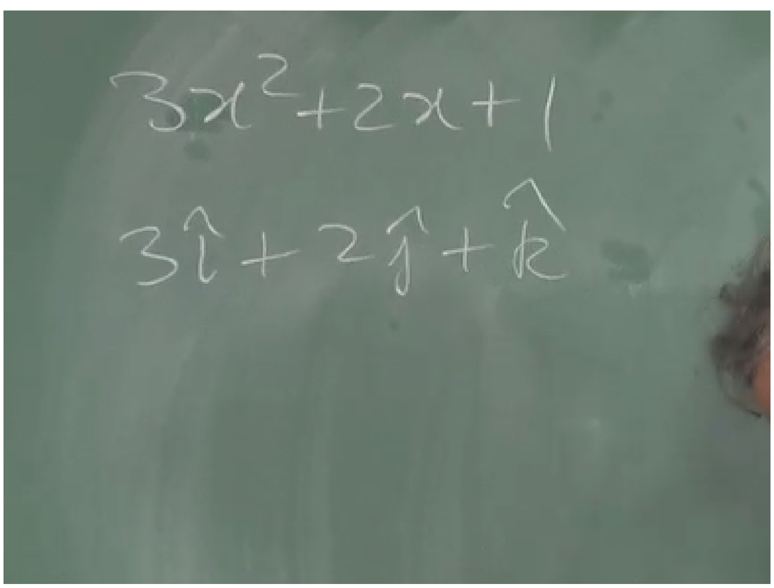
So this would be that part of C that is in the plane. So if I have a vector that is sticking out of the board then C.P1 hat and C.P2 hat is a shadow that is cast on the board. So this is the perpendicular part component that is on the board. So this combination right if you call it P3, did I call it P3 last time, what did I call it. I did not call it anything. So I will not call it anything. So I do not want to make a mistake and if subtract this combination out of C.

I will call this result P3 and you have to ask ourselves the question what is P3.P1 hat and what is P3.P2 hat and if there in mind that P1 hat and P2 hat are orthogonal to each other? So clearly if you go through this process I will let you do it for yourselves clearly P1 hat and P2 hat are orthogonal to P3 and from P3 now we can come up with the P3 hat which is P3/modulus of P3 and you also understand why it is that I have actually come up with numbers instead of alphabet.

So it is actually possible for us that if we have multiple dimensions. I have not restricted to 3 physical dimensions that we are talking about, but if you have n multiple dimensions it is actually possible for us now to systematically go through each one to get an orthogonal set. This is not really numerically if you are dealing with actual vectors great way to do it, but this is called a Gram-Schmidt process and as I said so it does not matter.

It is not restricted to 2 dimensions or 3 dimensions. You can go to as many dimensions as you want. So clearly if you go to the fourth one then you can knock out the components that would correspond to the first 3 and that results in something that is orthogonal to each one of them so it a neat process. So this now leads us to a situation where we are able to what should I say, we are able to construct vectors of any kind so if you want what should I say.

(Refer Slide Time: 39:31)



If you want something like 3x square + 2x + 1 like that earlier and we have 3i + 2j + k we just need to ask ourselves a question well I have got I know how I have done it here, how is that we are going to go about doing it here. So this systematic organization now I am going to do it for functions we will go through a series of class of functions.

In next class, I will basically start with Box functions. So what I will do here is I will stop at this point and get back to Box function and so on. So in the next class what we will do is we will try to represent these functions as a we show representations of these functions of various kinds and what is the relevant accuracy that goes with it?