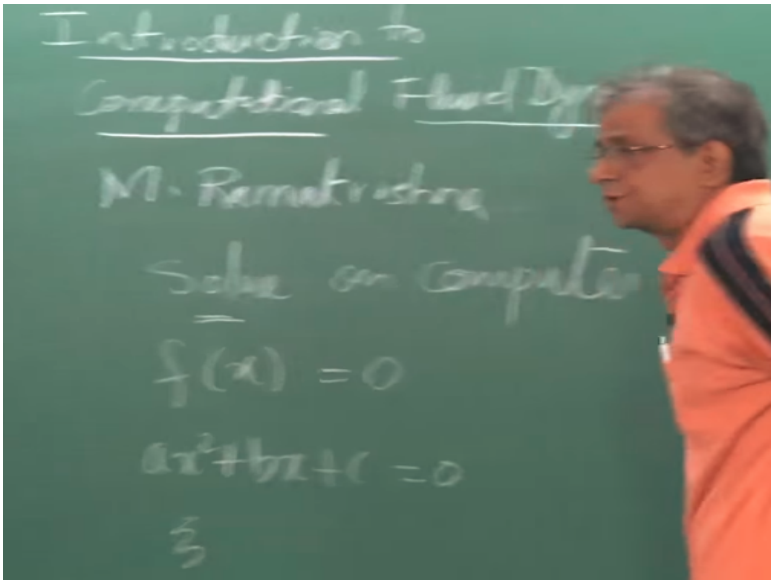
Introduction to Computational Fluid Dynamics
Prof. M. Ramakrishna
Department of Aerospace Engineering
Indian Institute of Technology – Madras

Lecture – 01
Introduction, Why and How We Need Computers

So welcome to the class. The course is introduction to CFD.

(Refer Slide Time: 00:15)



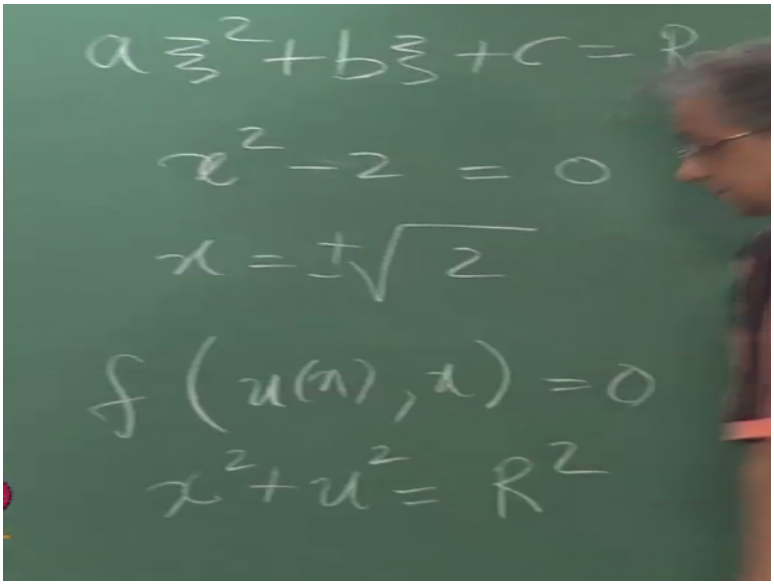
Introduction to Computational Fluid Dynamics. My name is Ramakrishna. So the course is introductory in nature. So, basically what I am going to do is, I am going to talk about fundamental concepts in computational fluid dynamics. I am not going to talk about advance topics. I am not going to cover a lot of schemes and so on. I am going to look more at a few of the simple schemes, why they work, how they work.

So but before we start off let us look at the title and figure out little more about what we are going to talk about in this class. I have told you what I meant by introduction. The course also title has computational and fluid dynamics in it. It implies that, we are going to study differential equations that come from that governs fluid dynamics problem using computers that of course is a very general title.

In reality what we will do is we will look at differential equations and representing the differential equations somehow on computers and solving the resulting equations that we get. If you say using computers in fluid dynamics you could also interpret it as being you could take gather data using experimental techniques for example, and you could post process that data that would really fall into the category of computational fluid dynamics.

So having said this, what does this involve? What is the nature of solving things on the computer? So in order to understand the question, solve on computer we have to look at what we mean by solve say an equation. So you are familiar with this. If I give you an equation of the form f(x) = 0 in algebraic equation for example it could be ax square + bx + c = 0 you know that you could find out if some x or some psi is a solution by substituting that psi into the equations. This is what we do and to see whether the psi actually satisfies the equation.

(Refer Slide Time: 03:24)

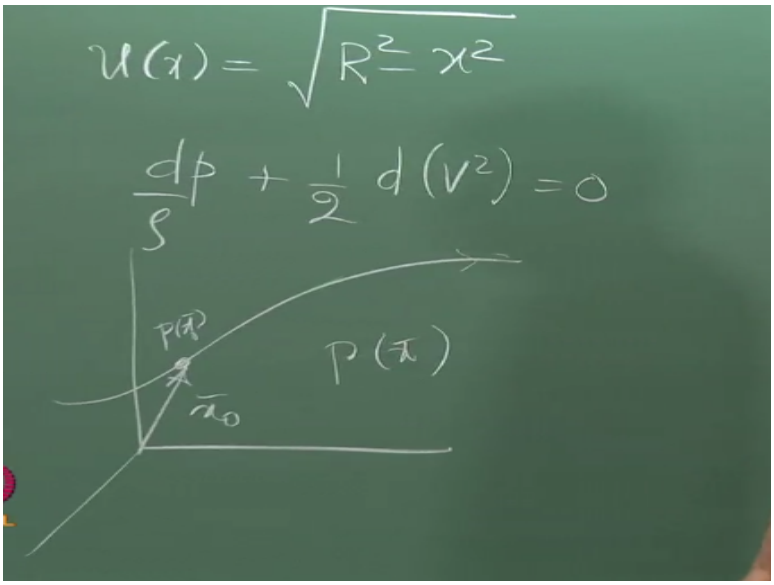


So if you substitute into the equation if you substitute psi you get a psi square + b psi + c. If it happens to be 0 you have a solution. If it happens to be not 0 then it leaves something called a residue. So if there is a residue left over you do not have a solution. We expect the right hand side to the 0, you are left with a residue you do not have a solution. So, the mechanism that we have to find out whether something is a solution or not is to take the equation that we are trying to solve and substitute.

We have this, we have solved algebraic equations in this case you know how to find as a solution of this quadratic equation and particular if I say x square = or x square - 2 = 0 you know that you could come up with a solution which is x = square root of there are 2 possible solutions. Am I clear? Now what happens if the solution is not a number it is not always a number. Take a situation where you have f of u of x, x = 0 and you have encountered situations like this.

One possibility is that you have x square + u square = r square and you could be asked to solve for u in terms of x.

(Refer Slide Time: 05:07)



And you know one such solution is u of x is R square - x square square root is of course a negative solution to it also. So what we have here is a solution to an equation which happens to be a function it is not a number as we had in this case and this is the kind of equations that we are going to looking at in this course. That is, you may have seen situations where you either use bisection method or Newton method to find roots to this polynomial.

Or you try to solve equations of this form. In this case, you have an equation in which the solution actually is a function. It is not a number. So we have this issue that if I am going to solve fluid dynamics problems on computers that I necessarily need to represent functions somehow on computers. We have to address that issue. So the other situations where we have something like this.

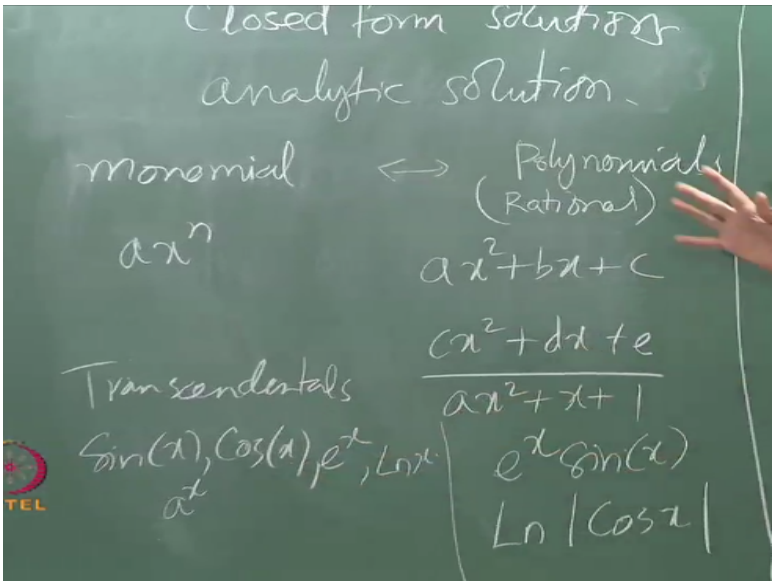
You could for example look at dp/rho + 1/2 d of v square = 0 this is you are familiar with. This is called the Bernoulli's equation in differential form. It is possible that in 3 dimensions along the stream line that someone has given you at some location x that someone has give you the pressure and you have the velocity field everywhere so you have pressure at x at this point x0 and you would like to integrate this equation, density is a constant.

You would like to integrate this equation along this trajectory and what is the result which you get. The result that you get is p as a function of the position vector x again a function. So in fluid mechanics this is just an application to fluid mechanics. This is of course also an equation whose solution is a function. This is just an application to fluid mechanics. What I am trying to say is that we have whole series of problems in which the solution to the equations that we seek are functions and numbers.

So the first thing that we have to look at is how do we go about representing these functions on the computer or in particular, how do we represent anything at all on the computer. So we are going to look at computer representation to start with, but before we get there you can ask the question why should I use computers at all? Right one simple answer is in this day and age what else do you expect.

Everything is computerized we might well use a computer, but there is a more serious reason for why we are going to use computers here. If you have if you think about solutions to differential equations that you have learnt before or solutions to other equations like this where they are like functions, you talk in terms of close form solutions or analytic solutions.

(Refer Slide Time: 08:24)



You would have heard this expression called closed form solutions or analytic solutions not to be confused with anything that you learnt in complex variables. Closed form solutions. So basically meaning that it is not an infinite series or something of that sort that you have something that is the finite combination of what that is the question finite combination of something. So what are all the functions? I mean if you think about it what are the functions that we know?

If you sit down and enumerate all the possible functions that we know in the sense that we can write, we know monomials that is something of the form ax power n. We can use combinations of monomials using our standard algebraic operations to come up with polynomials or rational polynomials possibly rational I will put that in brackets. What I mean by that is you could have ax square as I have indicated before bx + c combination taken by using summation.

Or you can have cx square + dx + e/ax square + x + 1 whatever I am just making this up fine. So you can make up functions, you can cook up functions using combinations of these, but I am not going to take an infinite sector, infinite sum that is what I mean by closed form solutions. I am not going to take an infinite sum I am just going to take combinations of these types so you can create function like this. What are the other kinds that we know?

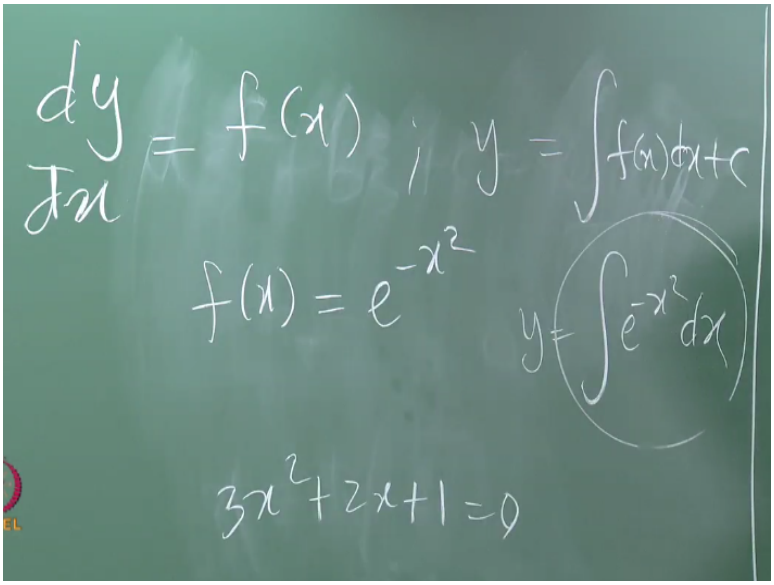
We know transcendentals so that could be sin of x, cosine of x, exponent of x, log x and so on, a power x I mean a raise tot eh poser x. we really quickly run out of and of course the

combinations again you could get tangent of x which is sin x/cos x or you can see e power x sin x. you can take combinations. You can take ln of mod cos x. see there are combinations that you can take to construct more complex function, but this is all we have.

So if you are going to turn around and you are going to say that I have a solution to some function and I want a closed form solution this is only choice that I have, I have nothing else that is left, I have no other combination here that is going to work so if it is not possible, the question is if it is not possible to represent my solution in terms of closed form solutions what do I do? So how does that happen?

How do we interpret that situation that I mean why does that happen at all? So there are situations that you are encountered.

(Refer Slide Time: 11:39)



So for instants if I have dy/dx = fx. So this is a simple differential equation. I mean this can be just integrated and you know get y an indefinite integral you can write it in terms of an indefinite integral fx dx + constant that is one way to do this. You can write it in terms of an indefinite integral. The problem is that if f(x) = e power - x square then they are in difficulty then you have a problem, because unfortunately e power - x square does not have a closed form solution.

So as a consequence what that means is e power - x square cannot be expressed in terms of any of these basic functions that we have. So to repeat there are functions we have this set of (()) (12:42) functions that we have accessed to we can create an enormous number of enormous set of functions with these functions taking linear combinations of these functions. However, there are still functions which cannot be represented in terms of these basic functions.

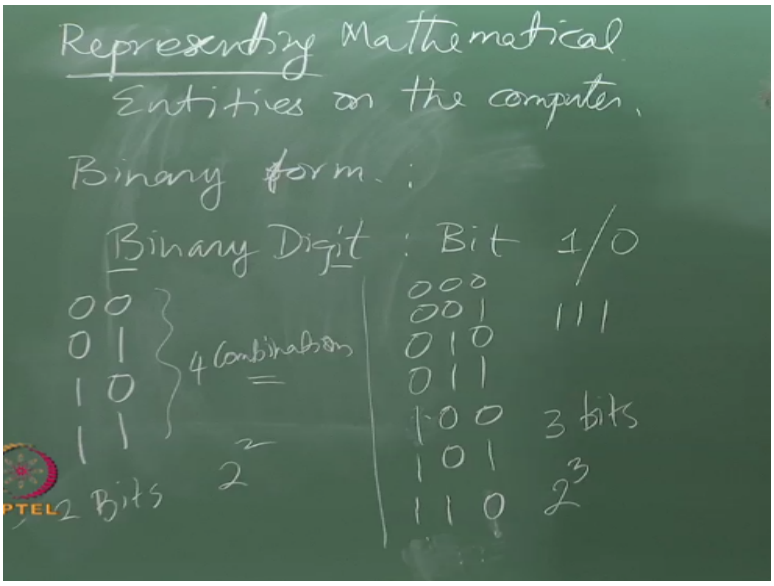
So they do not have a closed form. So this is what I mean is y = integral e power - x square dx unfortunately this integral does not have a closed form representation. e power - x square is okay, do not get me around, but integral of e power - x square dx does not have a closed form representation and therein lies a problem. So we have no choice we have to use computers we have to go through this process of somehow representing functions if not in terms of closed form solutions on the computer in some fashion.

The second reason why I want to use a computer is if you look at what I said earlier right in the beginning so if you have 3 x square + 2 x + 1 and you are looking for a solution one thing that I can do is I can substitute I can guess values and I can substitute those values here so that to check whether the residue is 0 or not. So you can do a blind search essentially. You can keep substituting values and trying to figure out what the solution is.

Of course you it has been proven that beyond the quartic we cannot get even a closed off solution for this in the sense that you cannot get an expression that will get a solution to a quartic equation this is I mean beyond a Quintic equation and so on. This is a quadratic equation. So you necessarily have to guess or come up with an algorithm, a mechanism to generate guesses automatically which will get you a solution to the equation that you see.

And the process of automation which computers of course a very good at this kind of track work but what do you need some kind of organization or something of that sort so that you structure the search can be structured and an algorithm can be developed. So we have now reduced our requirement to representing things on the computer this thing is a generic term.

(Refer Slide Time: 15:05)



So I will say representing we have now come to what I said earlier in the class representing mathematical entities on the computers and we will follow this in a sequence. I will make up a history. We will follow this in a sequence which is very similar to what we do in calculus. We will start first by constructing the real line. So towards that end we asked the question how do you represent anything on the computer?

What is the basic nature of how things are represented on the computer? I am not going to dwell on computer architecture, I am not going to dwell on other high performance related issues that are there that one would have to that you know if you need by we need be we can visit at a later time, but the only thing that I want you to know is that on a computer I will just recollect most of you have already encountered this.

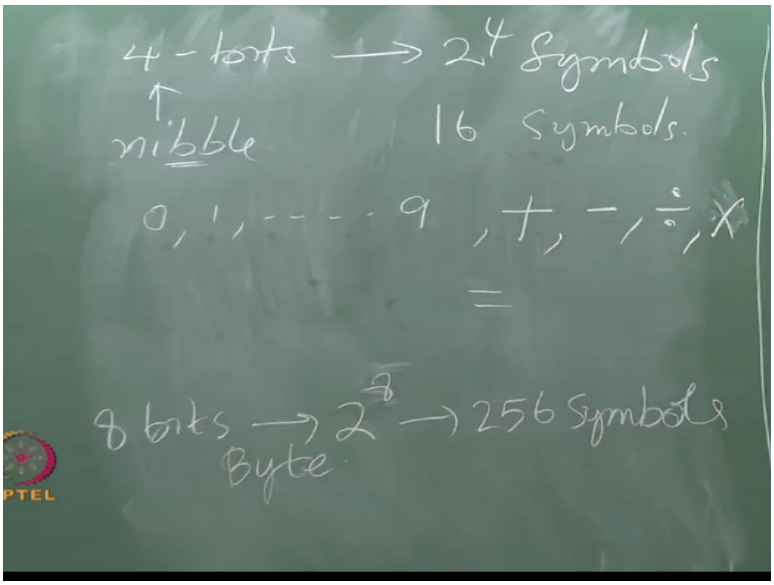
Numbers are represented in a binary form which means that a binary digit or a bit coming from a binary digit can either be 1 or a 0 can be a 1 can take 2 case 1 or a 0 normally they will give you examples of off and on and so on. It can be a 1 or a 0. The important thing is that it has 2 states 1 or a 0. So if I use 2 binary digits 1 next to the other just like we do our regular decimal numbers then the 2 binary digits can represent 4 combinations.

So 2 bits so to speak end up with 4 combinations. So these 4 combinations can be used to represent 4 symbols of some kind and you can decide what symbols that you want to use. The

number of bits that we use then seems to relate to the number of combinations that we have so 3 of course 3 bits will give us 0 0 0 I will just do this 1 more time 0 0 1, 0 1 0, 0 1 1, and 1 0 0, 1 0 1, 1 1 1. Have I got everything? 1, 2, 3, 4, 5, 6, 7 and I missed something. 0 1, 1 0, 1 1, 1 1 0.

Thank you very much, that is very important so I will write that a little further up okay 1 1 0 very important and 1 1 1 that is 8 of them. So it is 3 bits. You are going to get in fact it seems 2 power 3 where this is 2 power 2. So we have something here. So if I use 4 bits I get 16 and I am not going to work out the details of 4 bits obviously I am not able to write 3 bits or no resistance risking 4 bits so but now you can actually write them you, you just have to be systematic and little organized but you can write it out.

(Refer Slide Time: 18:38)



So if there are 4 bits, so 4 bits corresponds to 2 power 4 symbols which are 16 symbols. So there is a reason why I put 4 bits. 4 bits incidentally in computer parlance is called a nibble simply this is just for fun it is called a nibble because 8 bits are called a byte. So it is called a nibble there are 16 symbols. So 16 symbols if you can imagine 0, 1 to 9 those are 10 symbols then you can use so 10 of the symbols can be used to represent 0 through 9.

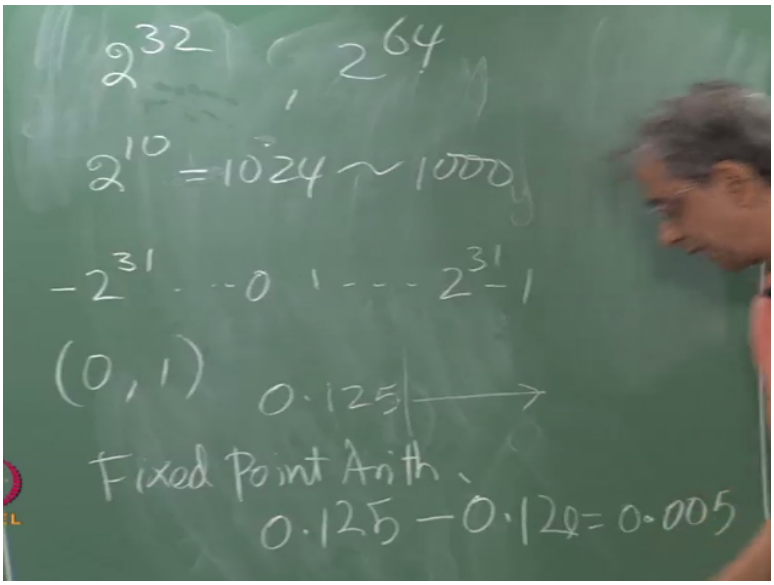
Which of course leaves 6 more symbols with which it is possible for you to represent + that by represent I mean you interpret in that fashion - revision and * that still leaves you 2 more may be an equal sign and something else. So in actuality now you can just imagine that if I use 4 bits that

is 16 symbols I can interpret what I mean by interpret just coming back here what I mean by interpret is I can say for instants 0 0.

This represents the number 0, 0 1 represents the number 1 and so on. So each of these symbols when I have 8 combinations I can interpret it as something just like when I write 1 the chalk dust 1, just like when I write the chalk dust 1 you interpret it as 1. You give it the meaning 1. So in that sense these 16 symbols for each of these 16 symbols you can actually attribute some number associated with those symbols and other operations.

And you on the whole were able to build a 4 function calculator using a 4-bit computer if possible it is just 4 bits like the 4 bit representation. It is actually possible for us to do this. So now what do we do in real life so of course just for completion 8 bits gives me 2 power 8 256 symbols is called a byte and most of the computers that you use these days either have are 32 bit computer are increasingly 64 bit computers.

(Refer Slide Time: 21:10)



So you have 2 power 32 or 2 power 64 which are enormous number of combinations, enormous number of symbols so 2 power 32 will give you of the order of 4 billion symbols basically. So 3 useful things to learn the powers of 10 just as an aside. So 2 power 10 is 1024 which is approximately like 1000 so 2 power 32 is 4 * 1 billion right it is in the order of 4 billion and it is very easy to quickly estimate what these values are?

So we have 2 power 32 symbols so what we can do now is we can use these symbols first to count just like we went from 0 to 9 you can actually go from 1 to 4 billion plus combinations or you can use them to represent integers which basically means that you take 2 power 31 going from - 2 power 31 up to of course there is 0 which is neither negative nor positive. So 0 1 and going up to 2 power 31 - 1.

So it is possible for us to interpret them assign the symbol if one of them being - 2 power 31 and the other one to be 2 power 31 and you can represent basically integers using the 32 bits, but integers 32 bits it is a very large number 4 billion, but it is not in the full set of countable numbers, it is not the full set of integers so it is very large however large it is it is not even the set of integers.

So here I am I started of missing I want to represent mathematical entities on the computer. I want to start with building the real line is what I promised, but I am not even able to represent integers, the whole set of integers. So there lies the problem. So from here how do I build the real line. So the real line is greater difficulty. I mean real lines consists of rational and irrationals. Rationales are of course countable set so you can map them to integers.

Irrationals are an uncountable infinity. So there is no hope of representing irrationals at all. So you could start the argument by saying that thanks to West Ross that I can represent any irrational as closely as I want by a rational, but that is still does not have so we throw away all the rationals. They are left with rationals. They are left with rational numbers and what we have is that we have to look at basically may be representing rationals in terms of scaling a problem between 0 and 1.

We scale a problem all our problems between the interval 0 1. Then it is possible that we represent all our computations basically work as a fraction on the interval 0 1 what I mean by that is that we only have the mantissa which is somehow managed to scale the whole problem the problem that you are working on so that the solution always lies between 0 and 1. So it is possible for us to do that.

You can use fix point arithmetic of course if you can scale it you can always add a few decimals here. So it is always possible for you to use 6-point arithmetic so as you say 0.125 or something of that sort. So it is always your possibility to use fixed point arithmetic. So what you basically do is you have the decimal point and you fix number of you use the 32 bits to represent the mantissa.

So as you can see what I have really done is a cheat what I have really done is I have taken these integers that I have got I am just interpreting the integers as a mantissa. I am just basically saying that it goes from 0 to 1 so instead of mapping it to - 2 power 31 to 2 power 31 - 1 I am mapping it to 0 up to 1 and representing the mantissa as basically as that integer and all the arithmetic that takes place is the integer arithmetic.

So this looks like a good thing. This is called fixed point arithmetic. So what is wrong with fixed point arithmetic? Why that should we do fixed point arithmetic. Well, fixed point arithmetic there are 2 issues that could crop up. One is that if you do add numbers temporarily we may have numbers that exceed 1 you may have to develop algorithms to take care of that so you have 2 numbers that are added up they exceed 1 then you turn around.

And say that look I will take temporarily take because I know my answer is finally going to be between 0 and 1. I will store that excess still it goes away that is one way to look at it, but you have to do special things clearly you have to do special things and there are times there are known instances where in our space program the French space program there are known instances where the over flows are actually caused failures.

So using fixed point arithmetic there it is very fast because it is effectively equivalent to integer arithmetic, but overflows that can be a serious problem. There is another problem which is worse. So you have just say you are carrying only 3 decimal places. So what is the problem? What is the issue? What is the big deal? So whatever is here beyond is whatever actual numbers whatever the actual decimal representation that you have.

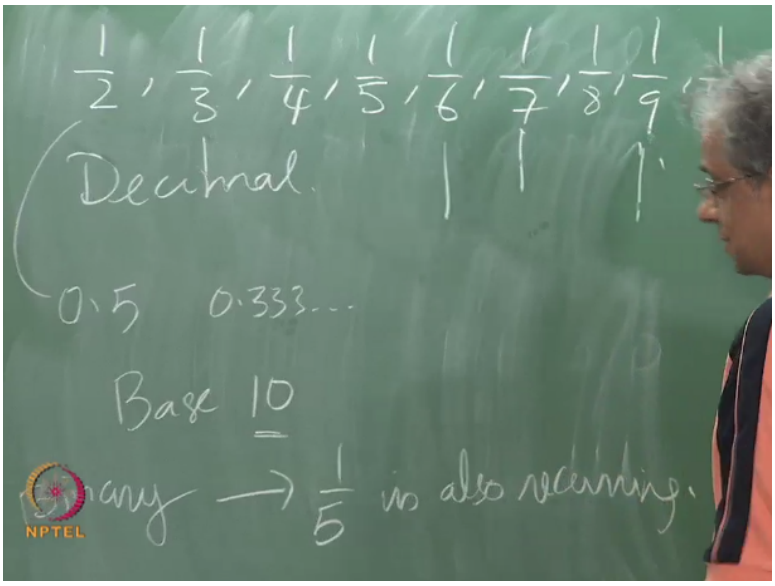
We will get to the issue of binary representation and so on. Whatever you have beyond these 3 digits that you have put up here are actually been thrown away. You are just essentially truncated in because you have an ability only to represent 3 places. So this position here is suspect. You may have rounded off, you may have rounded down, you may have truncated so this last number 5 is suspect.

So if I start performing arithmetics so for example if it turns out that from 0.125 I am actually going to subtract 0.12 as a course as a part of my computation that I am performing this leaves me that 0.005 and it leaves me with a number that I started off by saying a suspect that is I have 0.120 0 is also suspect, but I am left with something that is not as good as what I started off with I lost my most significant digits.

So anytime that you do subtraction this is going to happen. This loss is going to happen anytime that you do to subtraction. So there are 2 extremes 1 extreme is when you are performing additions then it is possible that you somehow overflow that fixed point container that you have. The other extreme is that you have loss of significance in the sense that you subtract out.

Somehow due to this operation you knock of the significant digits that you have the digits that you have whole meaning and you are left with digits that are suspect. So if you say that I have only 3 places in which I can represent this I have a problem and you can say well why should have to truncate, why should I have to reduce or I always have to truncate is it possible not to truncate.

(Refer Slide Time: 28:32)



You look at the issue, look at these numbers. I have 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9 and of course 1/10. I want to take it up until 10 in decimal. Look at this in decimal and ask yourself the question which one of these fractions that I am representing here has a finite or a closed I would say representation and which one of them has what we call recurring representations. So this would be of course 0.5 whereas this would be 0.333 recurring.

This would be 0.25, this would be 0.2, no problem 0.1666 is recurring. So this would be recurring, this would be recurring, that would be recurring, that would be recurring. So what does it that differentiates the recurring ones in the nonrecurring ones. What differentials it is that these are all primes like and the base 10 of our number system is a composition of 2 primes 2 and 5 and as long as you have fractions that are made up of 2 and 5 they have a close representation.

If they are primes that are not 2 or 5 then you have problem that is the key. So this is 3 times 3 which is a problem, 1/7 is a problem, 1/6 is a problem which is 2 times 3 so though the 2 works the 3 does not work and therein lies the problem. So you cannot get finite number of positions in decimal for 1/3. So now by going to binary on computers we have done something worse. On binary you cannot even represent 1/5 this is also recurring.

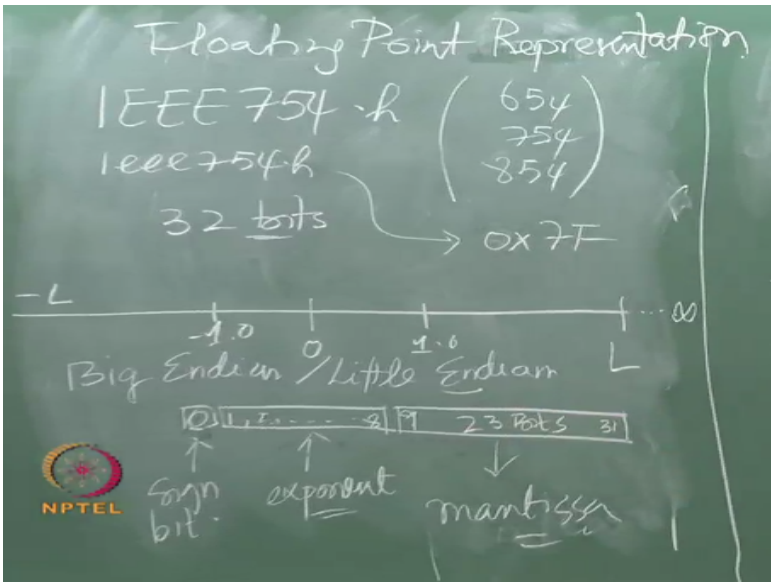
This has unfortunate consequences because being I would always say decimal I hold my hands up being of because of 10 digits we have gone for a base 10 representation and as a consequence

we are always going to take when we say give me a set of points or give me a set of intervals we variable it in 110 and the tragedy is that one tenth cannot be represented exactly on a computer it is given.

We never think to take one eighth or one fourth or one sixteenth or whatever we tend to take one tenth. So back here so what it means coming back here what it means is you have no choice. So if you are representing one third it is going to be 0.333 and you know there are an infinite number of 3s here and you had to cut it, you had to truncate, you had to truncate the series you have no choice.

So you have an error associated with this and as a consequence depending on what is happened here this digit as I said would be suspect. It is not as large as it should be or it is not as small as it should be. So fixed point arithmetic therefore has an issue. Fixed point arithmetic very clearly has an issue. How do we fix this well we go to something called floating point arithmetic as many varieties of floating point arithmetic that have existed, as computer manufacturers, once upon a time.

(Refer Slide Time: 32:10)



But ever since floating point representation I said representation. So nowadays, we would find we would use I triple E 654 754 you can go look this up on popular versus you may go look for this I triple E 754.h of course the header file most probably will be of the form I triple E 754.h.

so the associated standards were 654, 754, and the latest one 854. You can go find out the details about these.

But the whole idea is that we ran into problems earlier simply because the decimal point was fixed. So if I allow the decimal point to move it is possible that something like this I may have ended up with more digits I do not know. So that is what we are looking at. So what we will do is the 32 bits we will assign these 32 bits in a fashion that we allow us to how should I put this that will allow us a larger dynamic range we are not going to restrict ourselves to 0 to 1.

So what we will take is we will take the real line and I will show you what I mean. So I will assign we will pick the origin that is very clear. We will pick one we know what we want to do there. We know that we cannot pick all the points on the real line that is not possible so I have a largest number that I can represent which is L I may choose to satisfy a symbol for infinity and we can repeat the same thing on the other side.

It is possible for us to have a - L there and the - infinity at the other end. So we have 32 bits and we need to pick these numbers. So, one other things of course as I said, we typically in and around between 1 and 0, we want to cluster of lot of numbers. So we want a certain amount of density here. So we will divide up these 32 bits in the following fashion. We will assign 1 bit. I will write this is bit 0. We will call this the sign bit.

So this tells me whether the number is positive or negative. So this tells me the number is positive or negative. The second thing that we do is we assign the next 8 bits, these are 8 bits this goes from 1, 2, up to 8 bits the next 8 bits for an exponent. We assign the next 8 bits to an exponent. So what is the size of the exponent, it can go to 2 power 8 so that is 256. So clearly just like we did with integers you could have negative exponents and positive exponents.

You can bias the exponents so that the negative numbers and positive numbers. If you look up this file you will see that for the 32 bit representations they will actually have a they would define a bias in terms of the hexadecimal I do not why I am writing this right now but in terms of

hexadecimal they will define the bias so that is half way through, half of it is negative, half of it is positive of course there is 0 in between and then the remaining.

How many are left 23 bits are left going from bit 9, bit 9 remember we started to count at 0 up to 31 the remaining 23 bits are used to represent the mantissa. The remaining 23 bits are used to represent the mantissa is that fine. So this is a standard that is used to represent floating point numbers. What I want to point out here this is a particular set, particular type of floating point representation. It corresponds to this standard.

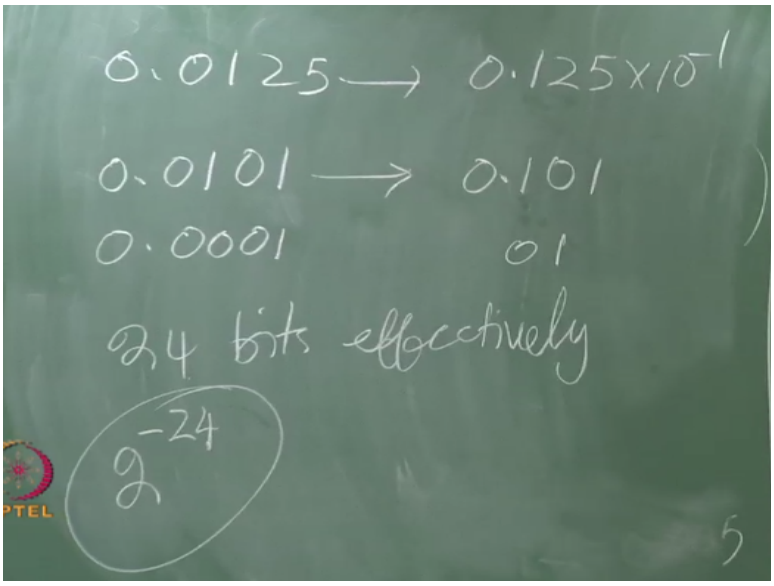
There are little more to it. If you got look it up and you can check this out in any resource that you want given on the net. You go look up the meaning of Big Endian and Little Endian because there is a humorous story is associated with why this nomenclature Big Endian and Little Endian came but you can go look up Endian as END Endian so there is a reason why these numbers are chosen this way.

So the standard setting up of the standards of course there is a little thought that is put into all of these. So if you perform a comparison on a computer and you want to know if one number is larger than the other number the first thing that you can do is just check the sign, but you just need to check the first bit, if the first bit 1 is positive, the other negative then you know the positive number is > the negative number just checking the first bit you will immediately know.

The second thing that you can do is you can check the exponent if the signs are the same, if 1 exponent is larger than the second exponent then, you know that is larger. The exponents are the same and this is the same then you go and check the mantissa to see whether that we will perform a comparison. So in that sense the order of representation is very important.

The sequence is very important because it makes comparisons and so on very easy. So you would like to do now is I will just give you a little detail on this tiny detail on just to show how clever we are. I will give you tiny detail on this. So what we do in floating point arithmetic is that make sure that we eliminate all leading zeros.

(Refer Slide Time: 38:34)



So because I have an exponent now so the decimal equivalent would be if I had 0.0125 then the decimal equivalent of that would be I would convert it to 0.125 * 10 power - 1. decimal equivalent of that would be I would shift so my exponent what I would store in my exponent this is 1 and what I would store in my mantissa would be 125 and I do not have to store this 0 and point because I know that 0 point is there.

But choosing binary has an advantage having chosen binary there is an advantage and what do I mean by that I mean the symbols can be either 1 or 0 so if I have if I say that I have a binary number 0.0101 something of this order and I want to do a shift I could have actually chosen this because this is actually a power but it does not matter. I can shift so I would get binary 0101 so I have shifted 1 place and therefore my exponent has appropriately changed and my leading term is 1.

The idea is that my leading term will always be 1 because it is binary there is no other option. In decimal, the leading term can be 1 or a 2 or 3 or a 4 or whatever it is, but in binary the leading term will always be 1. So there is no reason why I should store the 1 what I am trying to say is in this case there is no reason why I should have this 1 here. Why should I have the 1 I know it is going to be a 1. So I can actually get rid of this 1 and just store 0 1.

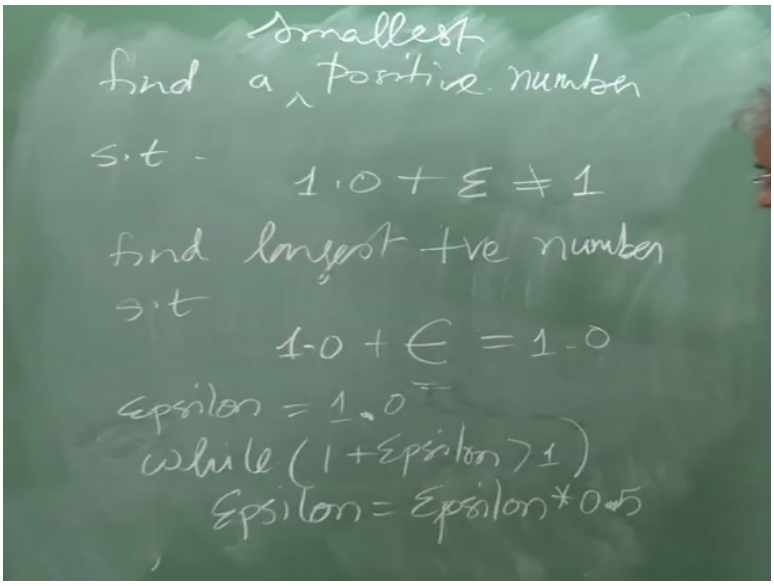
I do not put the decimal point. What I mean is that 1 is implied if it was the difference between this and this would be that this would go all the way the 1 will shift all the way and of course I am not going to store it and therefore I will store 0s. So what I am trying to say is this last the leading 1 that I have does not need to be stored because the leading term they are always going to 1. So effectively we have it looks like 24 bits.

You can see where the 24 bit you can think about where the 24 bit came from. 24 bits effectively. It is not that we got it free it may look like we got it free. They are being clever but we just managed to get everything that we can get. So the smallest number that we can represent smallest combination or the number of combinations that we have is 2 power 24, but you can imagine that if I get rid of if I shift everything 2 power - 24 is what I can discriminate. What do I mean by that? Well we can ask the question.

We can ourselves a question. I have these numbers here. This is the number line that I have. The number line that I have I am representing certain numbers here so there are 2 power 24 combinations in this from 0 to 1 that is what we have basically shown 2 power 24 is large but it is only 16 million of the order of 16 slightly over 16 million.

So it is not really I mean it is not definitely the continuum there. It is not all the rationals there either. So what I have actually done when I say floating point representation and I am looking at this is my real line what I have actually done what I have managed to do? What is that I am doing? To answer that question, I am going to ask myself.

(Refer Slide Time: 42:14)



Is it possible for me to find a positive number, and we get a positive number you can always generalize it later positive number so find what is the nature of this positive number? So it should be this smallest positive number such that since I have taken one 1 + this number epsilon is != 1. I want to find this smallest positive number such that 1 + epsilon != 1 and sort of get normally the way I would like to do it is.

I would like to find the largest or the flip side of this is fine may be this will make a little more sense and you can think about it largest positive number I will say positive number such that I will use a different epsilon do not get confused 1 + I will use that epsilon = 1. So first question is, is it possible for us to find an epsilon so that 1 + epsilon = 1 this is actually possible. Well on our number line it turns out it is possible.

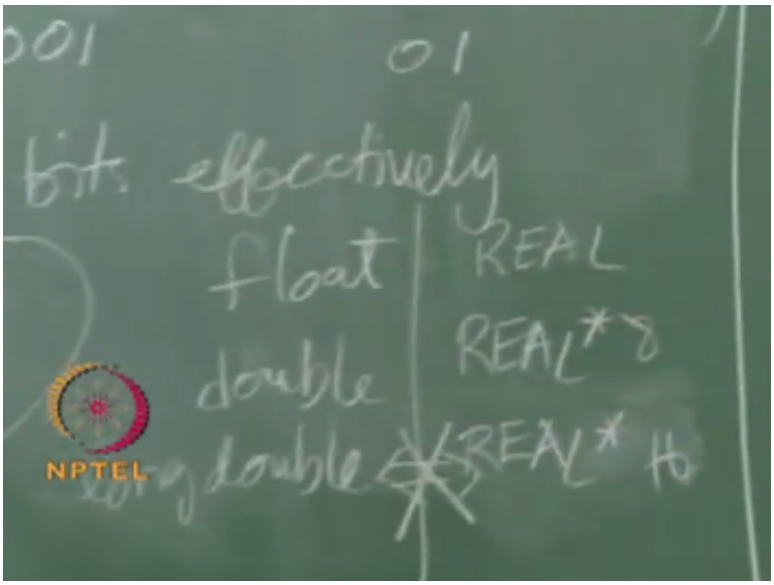
I would suggest that you write a program to find out what this value is and such exist. You may be under the impression that a mathematics of course if you write this you will assert that epsilon = 0, but in our case this is not really true I mean 1 + epsilon = 1 it is possible that you may be able to find an epsilon positive numbers such that 1 + epsilon = 1. A simple code that you can write for this is for example, say while so you can set let us set epsilon.

Simple I am just going to write pseudo code epsilon = 1.0 and I will ask the question so the code while 1 + epsilon. You can say > 1 or != 1 > 1 epsilon = epsilon * 0.5 you have it. We make it

one half so we are hunting. So it is going to go through this and as long as 1 + epsilon is > 1 right this process is going to keep going till you get to a point that suddenly 1 + epsilon is not > 1. Epsilon is positive so you can ask the question how is that possible?

Well it is possible print out epsilon and see, what is that you get. So then you have to ask the question you look at how the while loop works and what is that you are actually measuring and try to figure out are you getting this, are you getting this or you are getting something else which are the epsilons you are getting. So the issue is that is it possible that this actually works and what is the number that you get. I suggest that you try this out or 3 data types for those of you who are in C.

(Refer Slide Time: 45:34)



Please do this for float on this program for double and if possible on your complier long double. For those of you who are using Fortran what I mean by that is real, real * 8 and real * 16 is possible. These do not correspond. We will start 16 if possible and please let me know what is that you get. So that is basically if something that you can try out just run this on your favourite computer and with whatever language that you want.

If you have a scripting language also it is fine just try it out and see what is it that you get within this? So what we manage to do in today's classes I have basically shown you that we need to in order to solve differential equations that govern the fluid flow problems we need to find the

solutions or functions we need to able to represent functions on the computer. We also need to represent the differential equations that we are going to solve on the computer.

So that is basically what we have set out to do and towards that direction we have started off with like we do in calculus and mathematics trying to represent the real line and the real line we started with integers, we could not represent all the integers of course there is no hope of really representing the whole of the real line we have thrown away the irrationals we have represented some of the rationals.

We have used the I triple E standard to do that representation. We are now trying to find out what exactly it is that we are doing in representing numbers in this fashion. Once we understand this, we will then in the next class go on with representing other mathematical entities like possibly vectors and matrices and then see if we cannot represent functions.