Computational Chemistry & Classical Molecular Dynamics
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology- Bombay

Lecture – 03
Programming Techniques-1: Evaluating the Sine Function

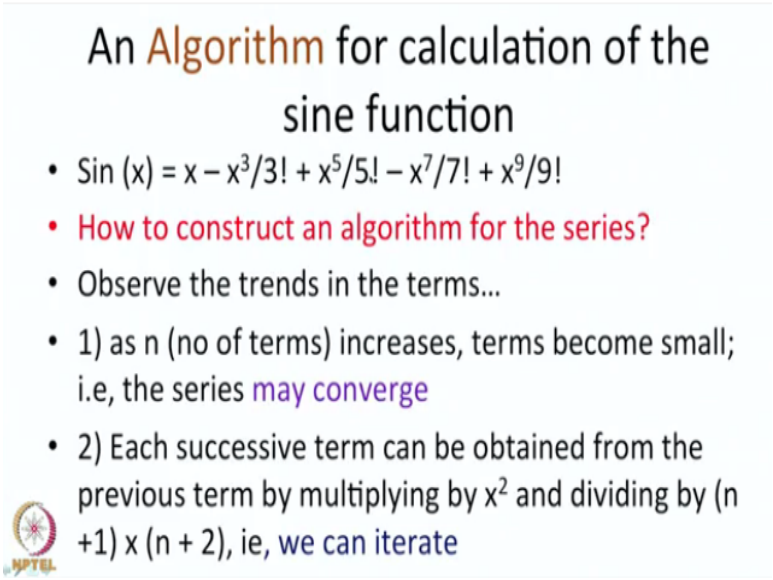
Hello once again, we will resume from where we left last time, what we discussed last time where many aspects of computational chemistry, one aspect was a programming language. A programming language is different from normal English because it has its own rules for example, in Fortran you start each (()) (00:38) column, first 5 columns are for line numbers, then equal to sign in a programming language is not a mathematical equal to.

But a = b means that whatever is the value of b, you replace a by that value of b, it is not equivalence but it is replacing whatever is on the right into the value whatever is on the left of the equation sign, so we proceeded along that and wrote some simple programs, we had one program where a do loop was there. What is a do loop? Do loop is there is a set of commands say 4 or 5 commands and these commands are repeatedly evaluated until the end of the do loop parameter is there.

For example, when I say do I going from 1 to n, so I start with a value 1 goes on repeating the loop with i = 2, i = 3 until i = n and then, once i is n next time you go out of the do loop, so we used some of these things, then our next task was what is a flowchart, we said that flowchart is a representation of an algorithm where all the tasks are written in a pictorial manner and the flowchart tells you how to go from one point of the picture to another picture.

So, what is an algorithm then; algorithm is an unambiguous statement of tasks to be performed, unambiguous tasks because if there is any ambiguity, you cannot use the algorithm, so we started with an evaluation of a sine function, you know that a sine function can be expanded in a power series.

(Refer Slide Time: 02:22)



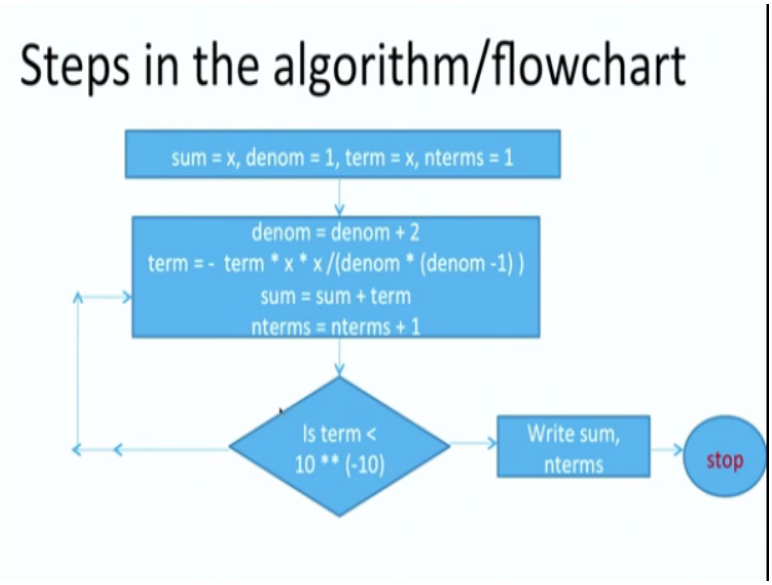
This is the power series, this is also a Taylor series, we will have more to say about this series as we go along because these Taylor series you may look up a little bit on the websites what a Taylor series is, it is expanding a function in different powers of x and each coefficient of that series is related to the derivative, so in particular sine x has this particular expansion, x – x cube/ 3 factorials + x 5th/ 5 factorial, alternate terms are changing sign and the new term has 2 powers higher than the previous term.

See for example, x to the third power, next one is x to the fifth power and the denominator has a factorial which is 2 numbers > the previous number, so we already discussed that this is a convergent series, the word convergence is very crucial in computation because convergence means a sum of terms reaches a finite value. Let us take this example of the series 1 + 1/2 + 1/3rd + 1/4th + 1/5th, this called a harmonic series.

So, the series goes on and on and on, the question is whether it gives you a finite number or not, there was a lot of confusion in the history whether this series is convergent or not, even great mathematicians thought that this series gives you a finite number but one can easily show, I will leave it to you this time as an exercise, you can show that this series is divergent, the sum is infinity.

So, when you compute you have to be very careful whether you get a finite number or infinite number, if it is an infinite number your computer will not help you, so now we come to this sine series, this is a convergent series, so we discuss the algorithm as well as a program for it.

(Refer Slide Time: 04:20)



Now, I want to write this as a flowchart okay, so now this is the flowchart, what is the job of a flowchart; it tells you how the control in the process flows, so what we have seen here is the first line defines all the terms, this is a sum of terms, so I want to start initiating the sum. So, for example I say, sum = x and denominator; denominator is what is dividing each of the terms, I start with denominator = 1 and term = x and n terms = 1.

So, this is my initial statement or these are all defined, so once these objects are defined, my task is now to compute the next term in the series remember, we said there is a recursion relation, what is the recursion relation? In a recursion relation, a new term is obtained in terms of the previous terms in this case, the new term is obtained in terms of the previous terms in a very simple manner that is you divide by x square, change the sign.

And then divide by denominator + denominator + 1 that is the new denominator is > the old denominator by 2 units, so now let us see so, when I want the next term I define denominator = denominator + 2, then the new term is - the old term multiplied by x square divided by denominator times denominator – 1. Now, you see that this denominator already is 2 units > the old denominator.

In particular, if the first term had a denominator 1, the second term top value is 3, so I will be dividing by 3 * 2, so the first term was x, so the next term will be –x, this term is x * x square, so that is -x cubed divided by 3 * 2, so the second time this term is -x cube by 3 factorial, so this new value of term, I add to the sum, sum = sum + term, so the this value of sum is x – x cube/ 3.

Again, look at this statement in mathematics, if you say sum = sum + term, so sum will cancel on both sides okay, so term will be 0, in mathematics if you write a = a + b that means your b is 0 because otherwise, you cannot have an equation but in a computer, what this sum = sum + term means is that take the old value of sum which was x, take the term which is - x cube / 3 and the sum of these 2 are put in the new value of this sum.

So, this sum now is a sum of 2 terms new value, then n terms = n terms + 1, what this does is; it increments the number of terms by 1. at the starting point, number of terms was 1, so when you do once this whole set of operations; 4 operations, your n terms has become 2, so once you do that then you come to the next line. Now, this is a diamond shaped line, so what this line asks is term < 10 power – 10.

That is, I am asking the question is the value of term which I calculated very, very small, if it is much smaller than a very small number, then I stop my calculation but if it is not smaller, then I have to redo the calculation, I have to do, I have to redo the whole series of steps. So, actually in this diagram, I should say on this diagram, I should have said if term is less, you go here, if term is greater, then you go to the left.

So, going to the left is if the term is greater, going to the right if the term is less and another important thing actually rather than just the value of the term, I should take the absolute value of term because any negative number is always going to be < 10 raised to 10 - 10, so since I know that the second term is already negative unless, I take the absolute value of term, the program will immediately stop as soon as you get a negative number, so you cannot do that.

Therefore, instead of this is term < this, it should be is absolute value of term < this number, so suppose that value is > 10 – 10, I go back, so I start again, what do I do; denominator is denominator + 2, last time the denominator was 3, now, it is 5, now what happens to the new term, this term = old value of term which was - x cube / 3 factorial, now that - x cube / 3 factorial multiplying by x square and a – sign, so the numerator becomes x to the fifth power with a positive sign.

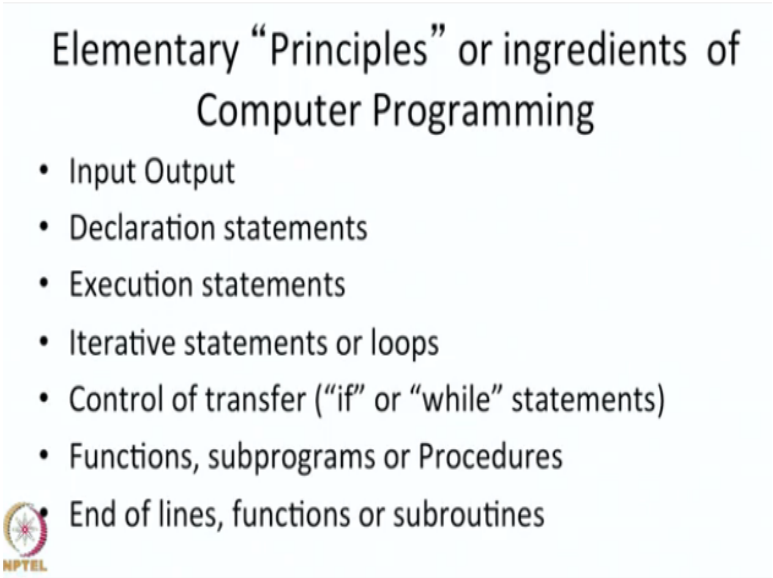
And the denominator already, there was a denominator with a 3 factorial, now what do I do; I multiply by 5 times 4 because denominator is 5 now, so 5 times 4, so the new value of term now

is x to the fifth divided by 5 factorial, so I add the third term here, the new value of sum is x - x cube / 3 factorial + x5 / 5 factorial that is my new value of sum, the number of terms increases by 1.

Then again, I go to this question, I ask again is the term value < this 10 -10, again this is not term but it is an absolute value of term. So, when I repeat this process in the next fourth step, this term will have 4 terms; x - x cube/ 3 factorial + x5/ 5 factorial - x to the 7/ 7 factorial, so when I add those 4 terms, then again as the question is the term that is the latest, term is a value of the last thing that is added.

So, whenever that term becomes < 10 -10, I stop this calculation, so this is a flow chart; flow chart is nothing but representation of the algorithm in a pictorial way, arrows tell you which is the direction of flow, so all you have to add here on the right side, you should say yes, then it will go here, if the answer is no, you will go here, if the term is not < 10 – 10, you compute this whole thing again.

(Refer Slide Time: 11:02)



So, this is my flow chart now, let us go to the next slide, so now what we will do; let us now analyse what are all the different ingredients of computer programs. So far what we have done; we had considered execution statements, what was an execution statement; sum = sum + term, any statement that evaluates the right side and puts it on the left side that is an executable statement.

For example, that do 10, i going from 1 to n, this was also an executable statement because that statement tells the program do this thing until the value of i = n, so now that executable statement also it was an iterative statement, the do loop is not only an executable statement but it also tells the program to repeat something for a given number of terms, so these were the 3 things we have already considered a do statement, execution statement.

Now, declaration statement we will have more to say because we already said that in Fortran, all variables that begin with i, j, k, l, m, n are automatically integer variables, others like those which begin with a, b, c, d, e, f, g or x, y, z, these are all real variables, so there is a built in system in Fortran which considers some objects as real and some objects as integer but you can always define things the way you want by a declaration statement.

So, as we go along, we will have more and more declaration statement now, input and output; this is a very essential part of the program, any program has to have some input, in our last calculation, x was your input and output was the value of sine x, in every program there will be inputs there will be outputs, so that is part of the programming setup declaration statement, execution statement now, iterative statement.

And the next type of statement is called a control or transfer, remember we had an if statement in the last program, so if statement what it does; it compares 2 objects and if the condition is satisfied it goes in one direction and if the condition is not satisfied it goes to some other part of the program. So an, if statement allows you to transfer the control from one part of the program to other.

Because unless you can control transfer, you cannot go back and forth because a program is not like a train journey, where you only go in one direction, you can always come back and execute the statements, so control is again important. Then as we go along, we will have other aspects in programming which are called functions or sub programs or procedures, so these are like separate set of sub programs, which help the main program to do the calculations.

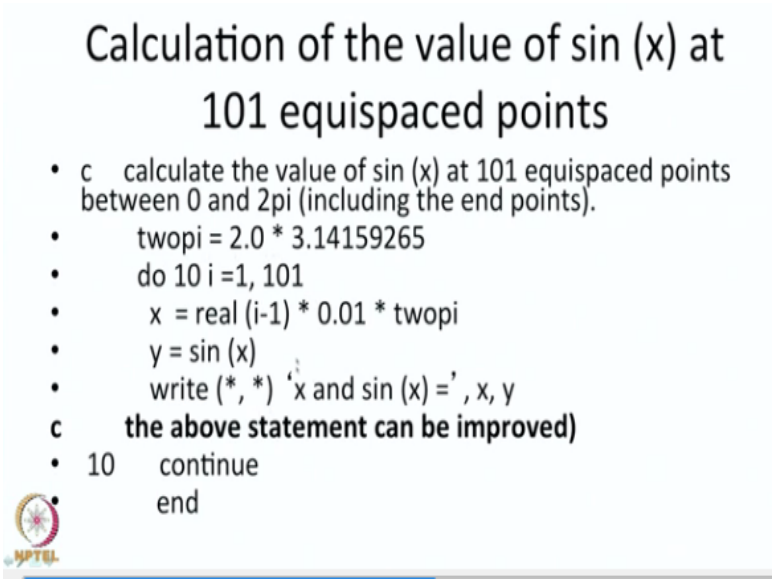
For example, remember in our program, let us say y = sqrt into bracket x, so what this is doing; it is saying that the square root of x is calculated and its value is put in the variable y, so this sqrt was a function, so is the case for a sine function, cos function, so many functions you have

already come across in your mathematics, these are all functions and your Fortran program also has statements which makes the left hand side take the value of the function.

Then, finally every program has a beginning but we did not say anywhere that this is the beginning of a program, it is understood, the first line is the beginning of the program but remember at the end, there was a stop or an end statement, so that end statement tells that the program has ended. If there is no end statement, the program will just get stuck; it does not know how to come out.

So, end statements are also needed, so these objects like 1, 2, 3, 4, 5, 6, 7, these 7 aspects you may treat them as certain elementary principles or ingredients of programming, so whenever you write a program, you have to pay attention to these 7 elements.

(Refer Slide Time: 15:15)



So, now what we want to do; we have already calculated sin x using our program, so now I want to calculate the value of sin x at 101 equispaced points, so this particular program is a lot easier than the old program, where all I am going to do is to evaluate a particular function for a given number of points and then print that or write that on the screen. Now, why is this important?

Remember, suppose you want to plot a function, when you want to plot a function, what is the normal procedure; you have values of x on one side, value of the function on the right hand side and using a graph paper, you will plot, so the computer allows you to plot things on the screen, so even before you plot, you have to calculate all those values. So, in this case this is a very

simple program, it helps you to calculate the value of sin x at 110 equispaced points between 0 and 2pi okay, including the endpoints.

So that means, you will also include 0 and you will also include 2pi okay, now remember the first character here is a c that means, it is a comment, we told you already whenever the first character in a line is a comment, it is not an executable statement, it is just a comment. Now, see the second line, it starts with between, this is not correct because there should be c here, so how will you change it, put a c before b and shift b one column to the left.

So, the first 2 lines are just comments in the program and what these comments tell you; it tells you what the program is going to do because if this line was not there; the first 2 lines you will not know after some time what the program will do unless, you go through the whole program and every time when you open a file, you do not want to look at the entire program to find out what it is doing, just want to see the first and second line, then you want to know the whole thing.

So, the third line; so we want to calculate sin function from 0 to 2pi, so 2pi is a variable, it is = 2 times 3.1419265, this is the value of pi up to 1, 2, 3, 4, 5, 6, 7, 8, this value of pi is accurate up to 8 digits, so I am calculating 2pi calling it 2pi, 2 times this value of pi, so this 8 digits I have used 8 digits though this is called a single precision. Many computers works at different levels of precision, in a single precision you have 8 digits after the decimal.

In a double precision, you will have 16 digits after the decimal; in quadruple precision you will have 32 digits after the decimal. Now, you will wonder why I need so many precise values because suppose, you do calculations 100's and 100's and 1000's and 1000's of times unless your digits are up to double precision, the errors will start accumulating. For example, if this value of 2 pi, I multiply this by several numbers.

If all those numbers have some error in the last digit, when you do 10 or 20 operations, the errors start shifting from the last digit to the previous digit, if you do 10,000 operations, the because you will be have a round off error, truncation error all kinds of errors are there, so your last digits become more and more questionable therefore, for many precise calculations like say a satellite launch or doing some detailed computational chemistry, you will need more than single precision that is called double precision.

When we do actual programming, we will come up; come across different levels of precision, so now 2pi is defined, so I want to calculate 101 points, so what is the do statement or a loop statement for 101 points? Do 10, i going from 1 to 101, this says that i is a variable, integer variable because i, j, k, l, m, n are integer variables, it takes the first value of 1, then it goes up to 101 points.

So, these are do statement, what is the meaning of this 10; 10 is a line number you see at the end there is a line number, the line number is given between column 1 to column 5, so this statement says that do all the operations up to line number 10 starting with i = 1 up to 101, so do this iterative calculation 101 times, so these my do statement, so the next statement defines the value of x.

Now, x = real of i - 1 multiplied by 0.01 times 2pi, so here we have done something interesting, i is an integer, i - 1 is again an integer which is 0, so but this is 0 as an integer but computer distinguishes between 0 as an integer and 0.0 which is a real number, so you want to convert an integer into a real number and this is the statement real into bracket some object, so that will give you the real part of that variable.

Suppose, I say 2 is an integer, real into bracket 2 will give me 2.0, so now sin function you will be calculating for all real values of x, so now this x is real i – 1, first time it is 0 * 0.01 * 2 pi, so it means first time the value of x is just 0, then it calculates y, what is y? y is sin x, so x is a variable, sin is a function now, this sin x is the fortran function not the function you calculated using the algorithm.

The algorithm we discussed just a few minutes ago was your own algorithm to calculate sin x, the computer has its own algorithm which may be very much like your own algorithm but it may be different from your algorithm, this also tells you that in any calculations, there can be many, many ways of calculating the sin function. So, you may not realize the importance that there could be several algorithms to calculate a sin function.

But now, consider an integral, when you integrate a function from lower limit to upper limit, there are many, many ways of doing those integrals, so we will consider different algorithms to do integrals and some algorithms will be more accurate than some other algorithms okay, so

coming back to our problem at hand, y = sin x, the first time x is 0, so sin 0 is 0, so next line is write *, *, in quotation x and sin x = quotation complete.

So as we said earlier anything that is between quotation is just printed on the screen as it is; x, y, so it will write x and sin x = then x and y, x was 0 and y was again 0, so the first value will be 0 and 0, okay. Now, I have said here it is a comment now, the above statement can be improved, this statement you can improve, think of how you can improve it, we can discuss in the future or through problems.

And then 10 continue; 10 continue is the last statement of my do loop, so it will do all calculations up to this 10 and then go back now, i started with 1, so when you go back the first time i will be 2, so when i = 2 now, I state x = real i – 1, so in this case now 2 -1 is 1, so it is real of 1; 1.0, so 1.0 is nothing but 1, so the first value now, the second value is .01 * 2pi, you repeat the whole procedure, calculate the sin, print the values on the screen, go back, next time i = 3, calculate, print.

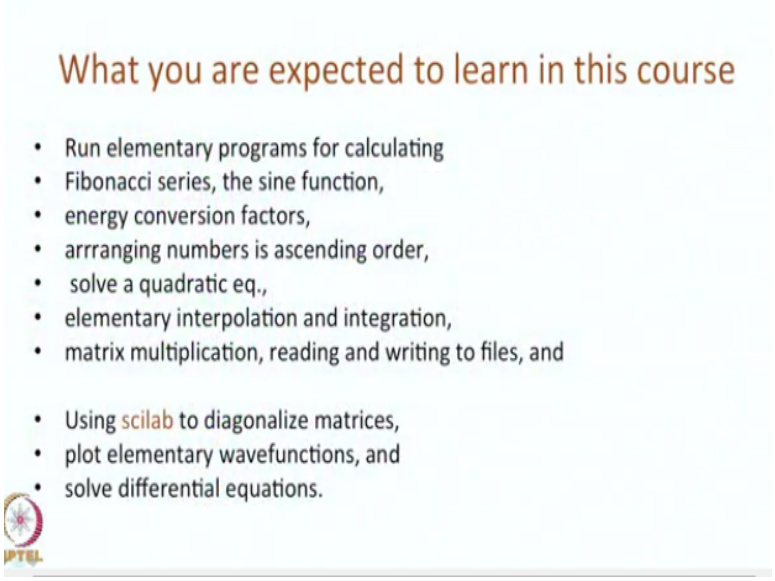
The last time i will be 101, when i is 101; 101 – 1 is 100, so real of 100 is 100.0; 100.0 * .01 is exactly 1; 1.0, so the last value of x is 2pi, calculate x is 2 pi sin 2pi, so it will write x and sin of that x and now it goes to 10 continue, i is already 101, so if you go next time it knows that i has reached already the last value, so it will not calculate again, it will go to the next line. So, after it calculates 101 values, the program will just end.

So, what will be written on your screen; these 100 and values of; 101 values of x and y again, let us go back to what is this *, *? *, * means write something on the screen, write on the screen okay, so this is a simple program to calculate the value of your sin function at 101 equidistance point, why is it that I have 101 rather than 100 because if I started with .01, .02, .03 up to 1.0, then I will have already 100, so I will not be able to include the 0th point.

So, to have the 0th point, what I have done is; I have included that 101 okay, so for example suppose, you need 10 numbers from 1 to 10, from 1 to 10 there are 10 numbers but if I want to include 0; 0, 1, 2, 3, 4 up to 10, there will be 11 numbers, so having this 101, allows you to calculate the values for all those in that range. So, now before I conclude today's lecture, let me now comment on what are the other things you will be doing now.

So, far you have already seen some 2 or 3 simple programs, from tomorrow onwards what we will be doing; we will actually execute this program because now you have enough knowledge of different statements in programs, line statements, do loops, beginning, end, *, *, you have known all these things, so what we are expected to do here; run elementary programs for calculating, what are the things we will calculate?

(Refer Slide Time: 25:45)



Fibonacci series, the sin function, we have already discussed the sin function today, energy conversion factors, we discussed it last time, arranging numbers in an ascending order, it is a very interesting program we will consider it very shortly, solving a quadratic equation, you may think that what is the big deal; quadratic equation has only 2 roots, I know how to do that but when you do through a computer, you have to be very careful of writing all the steps.

Because you know, you could have negative numbers and when you try to take the square root of a negative number, you will have a problem, so computer does not know whether a number is real or imaginary, so you have to take care of that then, we will also consider elementary interpolation and integration, we will consider matrix multiplication and then we will consider reading and writing to files.

So far what we have done is just to write something on the screen and read something from the screen that is completely alright, if you have to just write 1 or 2 lines but if you have to read lots and lots of data say, 100 data, 1000 data, 20,000 data points you cannot probably sit on the computer and give this 101 data points, 1000 data points, you will be tired and you may make mistakes.

So, it is best to give the data in a file, so we will consider that reading and writing to files, then we may consider some more things like Fourier transform which are useful in chemistry, we will consider interpolation, I have already mentioned, we will also consider matrix operations; matrix operations are very crucial, so since some of these programs are difficult to write in a very short time, we will use scilab which is a public domain software.

So, scilab will do many of these operations in almost a fraction of a second, so we will use scilab to diagonalize matrices, to integrate, to do many, many things which we cannot do easily by writing our own programs, then we will plot wave functions of atoms and molecules because after all in chemistry, we are supposed to learn about atoms and molecules, so we will plot wave functions, we will solve differential equations.

Then, after all this is over, then we will go to a gromacs software to do molecular dynamic simulations of liquids and solutions that will be our final application and I will conclude this time and next time, we will do a couple of more programs then actually, show them on the screen, okay thank you, we will conclude today lecture.