# Computational Chemistry & Classical Molecular Dynamics

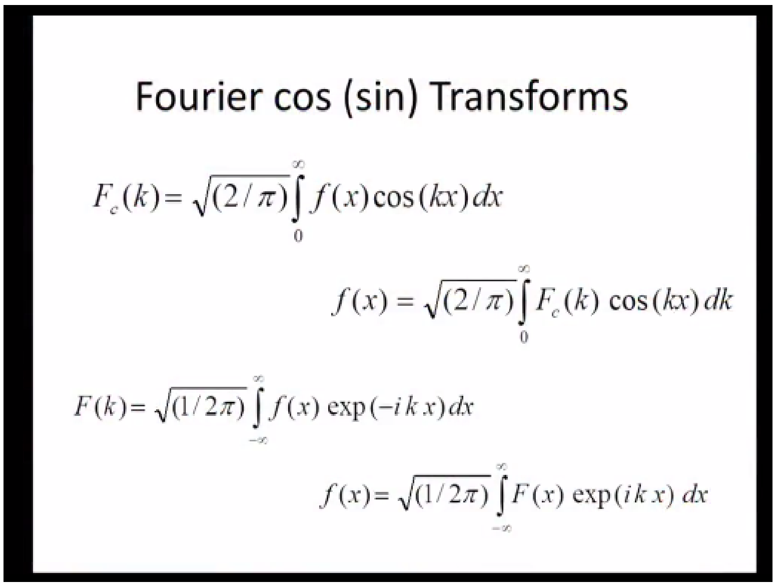
**Prof. B. L. Tembe**
**Department of Chemistry**
**Indian Institute of Technology – Bombay**

## Lecture – 28
## Scilab-6: Integral Transforms; Introduction to Molecular Dynamics (MD)

Welcome again, so this will be our last session on Scilab, so the last session we want to do Fourier transforms and Laplace transforms, these transforms are very useful in not only mathematics, but in a lot of science and engineering. In fact, you may have heard that there are now FT NMR spectrometers and FT IR spectrometers.

So the Fourier transforms are extremely powerful to convert data from say time domain to frequency domain or from your spacial variables to your momentum variable. There are very good and they have huge number of applications.

**(Refer Slide Time: 00:54)**



### Fourier cos (sin) Transforms

$$F_c(k) = \sqrt{(2/\pi)}\int_0^\infty f(x)\cos(kx)\,dx$$

$$f(x) = \sqrt{(2/\pi)}\int_0^\infty F_c(k)\,\cos(kx)\,dk$$

$$F(k) = \sqrt{(1/2\pi)}\int_{-\infty}^\infty f(x)\,\exp(-i\,k\,x)\,dx$$

$$f(x) = \sqrt{(1/2\pi)}\int_{-\infty}^\infty F(x)\,\exp(i\,k\,x)\,dx$$

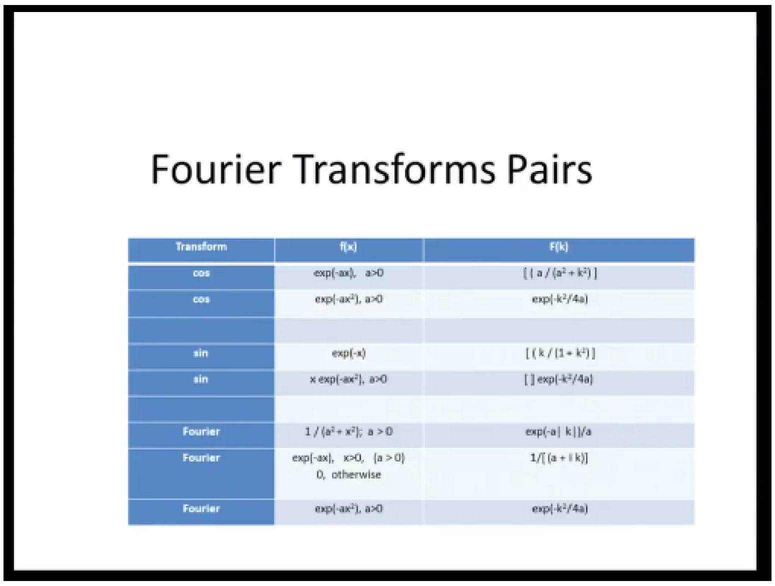
So basically these transforms are nothing but integrals, so I want to define these transforms and discuss how to calculate these using the programming including the Scilab programming. So let us see the definition of Fourier cos and sin transform. So cos transform Fck is defined as square root of 2/pi integration 0 to infinity fx cos kx dx what this does, suppose you have a function fx, it multiplies that function by cos kx and integrate some 0 to infinity to give a result which depends only on k now.

When x is integrated out the result will be a function only of k. So there is a cos transform. So Fc of k is the cos transform of fx, now fx is the inverse transform of Fc. So you have a forward for your transform going from fx to Fc of k and you have in inverse transform to go from Fc of k * fx, these are inverse transform. So really speaking when you do the inverse transform you should get the original function fx. So this is the cos transform. So in place of cos if you put the sin it will be a sin transform okay.

So you have a cos transform and a sin transform, but the more general is the exponential, remember e to the i kx, I can write it as cos kx + i sin kx. So these exponential e to the i kx is the sum of cos and sin. So this exponential transform is really a sum of cos and sin transform. So how is that defined f of k, now this is the Fourier transform, not the cos transform. Fourier transform is 1/2pi – infinity to infinity fx e to the i kx dx.

So this may forward transform and when I do the inverse transform the sin is change here, you see I had –i kx so here will be + i kx so this is my inverse transform. So for many functions fx these transforms have been calculated and you will find several tables of Fourier cos and sin transforms.

**(Refer Slide Time: 03:17)**



## Fourier Transforms Pairs

| Transform | f(x) | F(k) |
|---|---|---|
| cos | exp(-ax), a>0 | [ ( a / (a$^2$ + k$^2$) ] |
| cos | exp(-ax$^2$), a>0 | exp(-k$^2$/4a) |
| | | |
| sin | exp(-x) | [ ( k / (1 + k$^2$) ] |
| sin | x exp(-ax$^2$), a>0 | [ ] exp(-k$^2$/4a) |
| | | |
| Fourier | 1 / (a$^2$ + x$^2$); a > 0 | exp(-a\| k \|)/a |
| Fourier | exp(-ax), x>0, (a > 0)<br>0, otherwise | 1/[ (a + i k)] |
| Fourier | exp(-ax$^2$), a>0 | exp(-k$^2$/4a) |

I will just give one example here okay, in this example I have given several examples of Fourier Transform Pairs. What is the meaning of Transform Pair? If I take the function fx take a Fourier transform you will get f of k. When I do the inverse transform I get the original function. So exponential of –ax, the cost transform is a/a square + k square, the inverse transform is e to the –ax.

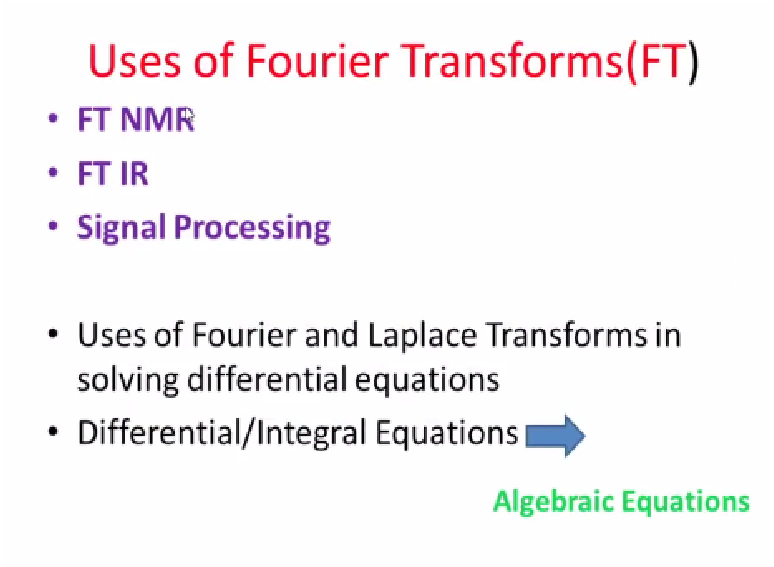
So suppose I take a Gaussian function, one of the interesting thing is that the Fourier transform of a Gaussian is also a gaussian function, remember I have e to the –ax square a > 0, it is Fourier transform is e to the –k square /4a, when you inverse transform e to the –k square/4a I will get my original function e to the –ax square. Now this is sin transform, sin transforms are a little different because sin is an odd function and cos is an even function so these are different.

So finally there is a list of Fourier transform okay I have taken Fourier transforms of 3 functions, there inverse transform. So in literature you will find a large number of data points. Now before I go into some applications let me go back and suppose now I want to evaluate this if I want to evaluate this using the programing technique now we are all fairly good at evaluating integrate.

So how would I evaluate, I just discretize fx into a large number of points suppose I have 1000 values of fx multiplied by cos x and I can integrate using trapezoidal rule or Simpson's rule, but the main point is when x becomes large, if fx becomes very large then this will not converge. So what is the meaning of convergence? The integral should give you a finite value, okay.

So for convergence there is a requirement that at large values of x these fx should go to 0, if it does not go to 0, you will have a divergent integral. So you can do these Fourier transform only for certain specific functions, okay, so that should be borne in mind.

**(Refer Slide Time: 05:29)**



## Uses of Fourier Transforms(FT)

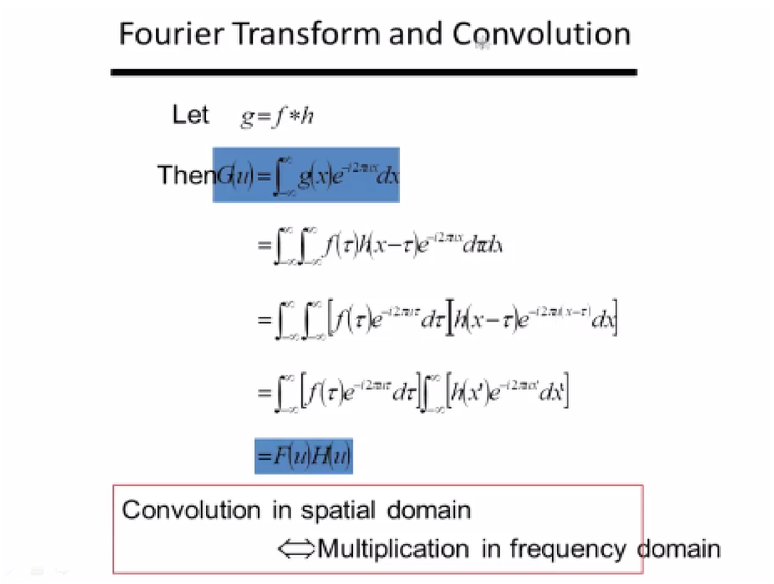
- FT NMR
- FT IR
- Signal Processing

- Uses of Fourier and Laplace Transforms in solving differential equations
- Differential/Integral Equations ➡

Algebraic Equations

So now let me go to some applications, what are some very useful applications, as I mentioned FT NMR, FT IR, the Fourier Transforms have great use in signal processing, so electrical engineers, I can almost say that you cannot study electrical engineering if you do not do Fourier Transforms, this maybe an over statement, but fairly true, so now these transforms are useful in solving differential integral equations as well.

**(Refer Slide Time: 05:58)**



## Fourier Transform and Convolution

Let $g = f * h$

Then $G(u) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi ux} dx$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau) h(x-\tau) e^{-i2\pi ux} d\tau dx$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau\right]\left[h(x-\tau) e^{-i2\pi u(x-\tau)} dx\right]$$

$$= \int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau\right]\int_{-\infty}^{\infty} \left[h(x') e^{-i2\pi ux'} dx'\right]$$

$$= F(u)H(u)$$

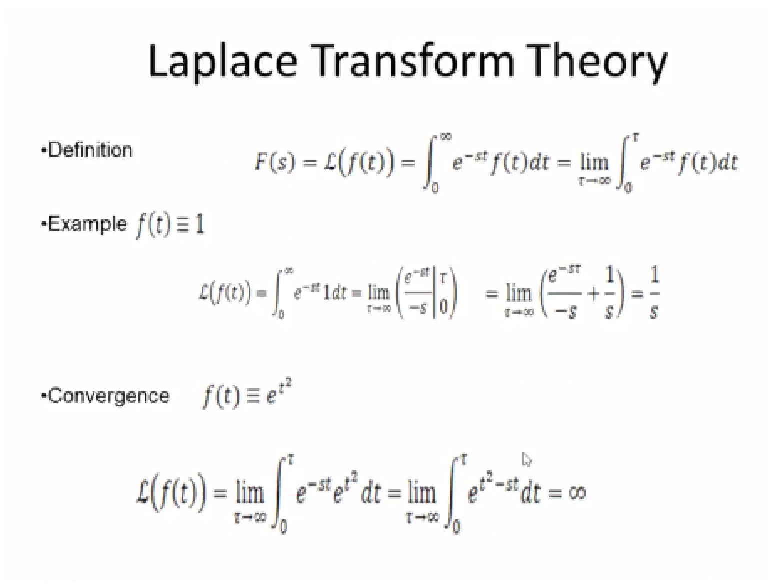
Convolution in spatial domain
$\Leftrightarrow$ Multiplication in frequency domain

So I will give you one example, so what this slide shows you it tells you the Fourier transform of a convolution. So G is the convolution function, f star h, this convolution means, look at this particular line. Convolution function is, you have a function f of t you multiply by x – t and integrate okay, so that you have only one way. This is called a convolution. Whenever you have f of t and x – t integrate with respect to d tau so it is a convolution function.

So what we are showing here the Fourier transform of a convolution, this Gu is a Fourier transform of a convolution okay. So the Fourier transform of a convolution is nothing but the product of the Fourier transforms of G and H. So this is very useful because whenever you have equations involving convolutions you can convert them into products of Fourier transform by taking a Fourier transform.

And when I take now an inverse transform I will get the inverse transform of the convolution. So this is very useful okay, so convolution in the spacial domain will give you multiplication in the frequency domain, here u is the frequency. So a very complicated problem in the spacial domain gives you a very simple multiplicative problem in the frequency domain and hence they are very useful in solving these kind of equations. So I will now next consider Laplace transform.

**(Refer Slide Time: 07:33)**



## Laplace Transform Theory

•Definition $F(s) = \mathcal{L}(f(t)) = \int_0^\infty e^{-st} f(t)dt = \lim_{\tau\to\infty} \int_0^\tau e^{-st} f(t)dt$

•Example $f(t) \equiv 1$

$$\mathcal{L}(f(t)) = \int_0^\infty e^{-st} 1 dt = \lim_{\tau\to\infty} \left( \frac{e^{-st}}{-s} \Big|_0^\tau \right) = \lim_{\tau\to\infty} \left( \frac{e^{-s\tau}}{-s} + \frac{1}{s} \right) = \frac{1}{s}$$

•Convergence $f(t) \equiv e^{t^2}$

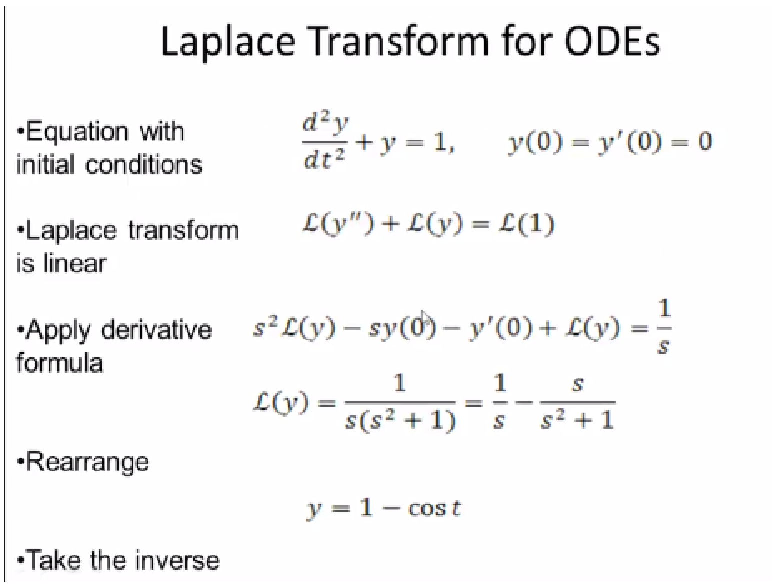
$$\mathcal{L}(f(t)) = \lim_{\tau\to\infty} \int_0^\tau e^{-st} e^{t^2} dt = \lim_{\tau\to\infty} \int_0^\tau e^{t^2 - st} dt = \infty$$

Many of you may have been familiar Laplace transform of function of t is nothing but 0 to infinity, e to the –st f of t dt. The Laplace transform has a multiplicative factor e to the –st, for Fourier transform it was e to the –ik. So Fourier transform has a complex exponential. So these are real function now, this is the Laplace transform of a function is given by 0 to infinity e to the –st f of tdt.

So this is an example, suppose my ft is a constant function, if the function is constant, the Laplace transform of that constant is 1/s. Now instead of a constant function, if my function is e to the –the square. So if I take the Laplace transform of e to the –t square, it will go to infinity because e to the –st and e to the t square, e to the t square is a very rapidly increasing function.

So I will have a divergent Laplace transform. So just that I had Fourier Transform Pairs I can have Laplace Transform Pairs and I will show you one application of use of this Laplace transform.

**(Refer Slide Time: 08:40)**



## Laplace Transform for ODEs

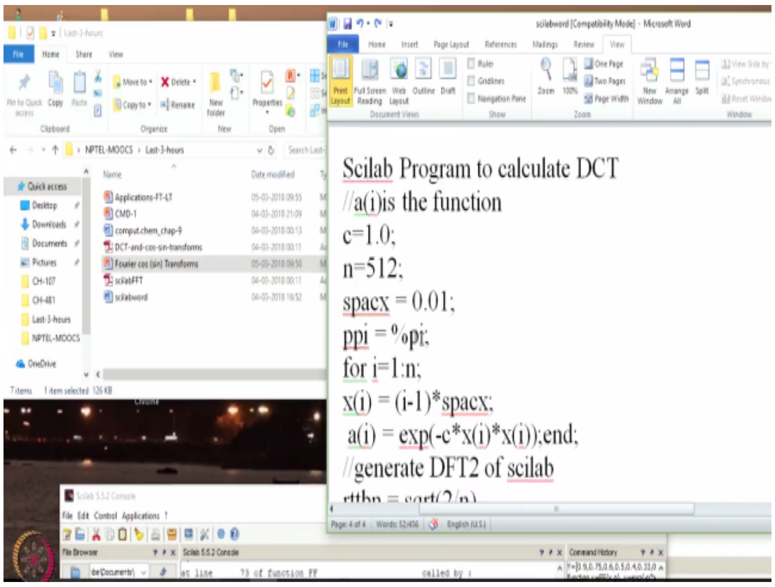
- Equation with initial conditions $\frac{d^2y}{dt^2} + y = 1, \quad y(0) = y'(0) = 0$
- Laplace transform is linear $\mathcal{L}(y'') + \mathcal{L}(y) = \mathcal{L}(1)$
- Apply derivative formula $s^2\mathcal{L}(y) - sy(0) - y'(0) + \mathcal{L}(y) = \frac{1}{s}$
  $\mathcal{L}(y) = \frac{1}{s(s^2+1)} = \frac{1}{s} - \frac{s}{s^2+1}$
- Rearrange
  $y = 1 - \cos t$
- Take the inverse

Suppose you want to solve this differential equation with some initial condition is the second order derivative d2y/dt square + y = 1 y0 = 0, y prime 0 = 0, I want to solve this. How do I solve it. Take the Laplace transform of this equation, so Laplace transform of y double prime second derivative, Laplace transform of y = Laplace transform of 1 we already saw in the last slide that Laplace transform of 1 was 1/s.

Now there is a formula for Laplace transform of the derivative. The Laplace transform of the derivative is s square Ly – sy0 – y prime 0 okay. So substitute y0 and y prime 0, so you will see that your Laplace transform of y is 1/s – s/s square +1. So when you take an inverse transform you will see that your y = 1-cos t. So that is how equation which was very complicated is a derivative, it is now written as an algebraic equation.

Now algebraic equation you can invert to get your final solution of y from the Laplace transform of y, which was is you invert it you get the function, which is the solution you got. So this way you can use Laplace transform as well as Fourier transform, but our present aim now is to really use Scilab to calculate this Fourier transforms okay.

**(Refer Slide Time: 10:21)**



```
Scilab Program to calculate DCT
//a(i)is the function
c=1.0;
n=512;
spacx = 0.01;
ppi = %pi;
for i=1:n;
x(i) = (i-1)*spacx;
 a(i) = exp(-c*x(i)*x(i));end;
//generate DFT2 of scilab
```

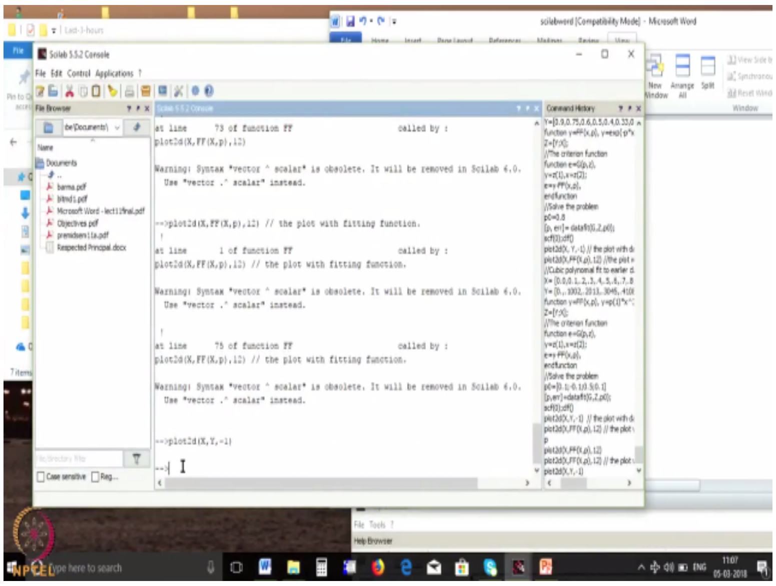
So here is my program. So I am going to calculate what is called a DCT, DCT is discrete cost transform. So what we discussed so far was continuous. What is the meaning of continuous? It takes all values between 0 and infinity. For a discrete Fourier transform they will take only discrete points. See we cannot do an infinite integral using a computer so what I will do, I will just illustrate calculation of a discrete cost transform of a function. So what is my function now. So let me define a function.

So this is what I am going to define, ai is my function okay, my constant is 1, so I want to determine 512 points of the function a. So what is my function, it is defined here c = 1, n = 512, spacx this is the spacing, in the x axis it is 0.01, ppi is percentage pi, if you recall we said that percentage pi is nothing but the value of pi. So that I have equated to ppi. So I do not call it pi because pi there will be confusion.

Already percentage pi is the 3.14, so I will define a new variable and equate it to pi. Now for i going from 1:n remember this colon, not comma, okay, for i going from 1:n my Xi is i-1 * spacx, so 0.01 is my spacing that means the first value is point, i – 1 will be 0. So 0 * spacx is 0. My first value of xi is 0, second value is 0.01, third value 0.02, fourth value 0.03 this way I determine xi values for 512 points.

And my a function now is nothing but e to the –c*xi square. This is the gaussian function, I want to take a discrete cost transform of gaussian function. So what I will do I will copy paste all these things, I will copy paste okay these functions and go to Scilab.

**(Refer Slide Time: 12:38)**

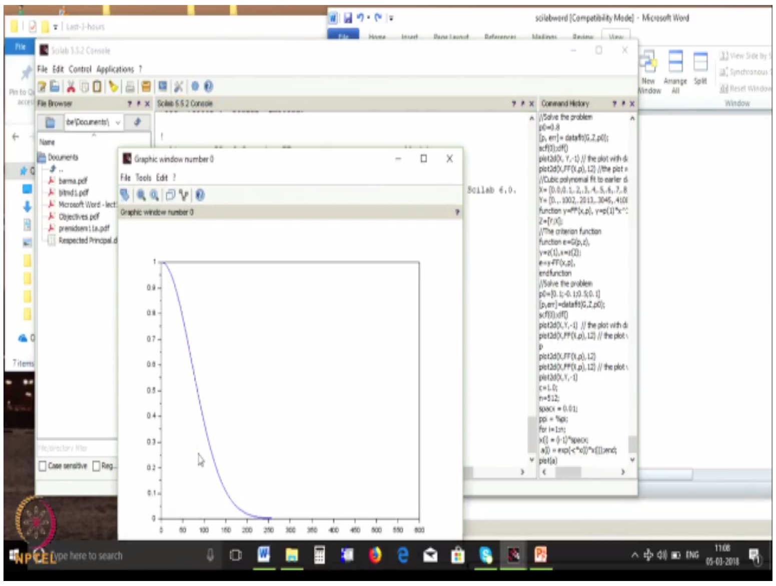


```
at line      73 of function FF                        called by :
plot2d(X,FF(X,p),12)

Warning: Syntax "vector ^ scalar" is obsolete. It will be removed in Scilab 6.0.
  Use "vector .^ scalar" instead.


-->plot2d(X,FF(X,p),12) // the plot with fitting function.
 !
at line       1 of function FF                        called by :
plot2d(X,FF(X,p),12) // the plot with fitting function.

Warning: Syntax "vector ^ scalar" is obsolete. It will be removed in Scilab 6.0.
  Use "vector .^ scalar" instead.

 !
at line      75 of function FF                        called by :
plot2d(X,FF(X,p),12) // the plot with fitting function.

Warning: Syntax "vector ^ scalar" is obsolete. It will be removed in Scilab 6.0.
  Use "vector .^ scalar" instead.


-->plot2d(X,Y,-1)

-->
```

```
Y=[1.0,0.75,0.6,0.5,0.4,0.33,0
function y=FF(x,p), y=exp(-p*x
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
endfunction
//Solve the problem
p0=0.8
[p, err]= datafit(G,Z,p0);
scf(0);clf()
plot2d(X, Y,-1) // the plot with d
plot2d(X,FF(X,p),12) //the plot w
//Cubic polynomial fit to earlier d
X= [0.0,0.1,.2,.3,.4,.5,.6,.7,.8
Y= [0.,.1002,.2013,.3045,.4108
function y=FF(x,p), y=p(1)*x^
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
endfunction
//Solve the problem
p0=[0.1;-0.1;0.5;0.1]
[p,err]=datafit(G,Z,p0);
scf(0);clf()
plot2d(X,Y,-1) // the plot with d
plot2d(X,FF(X,p),12) // the plot
p
plot2d(X,FF(X,p),12)
plot2d(X,FF(X,p),12) // the plot
plot2d(X,Y,-1)
```

Okay, so I will just copy paste, okay, I have executed, now I can plot this function A, okay, so let us say, plot * (a) okay so it will plot.

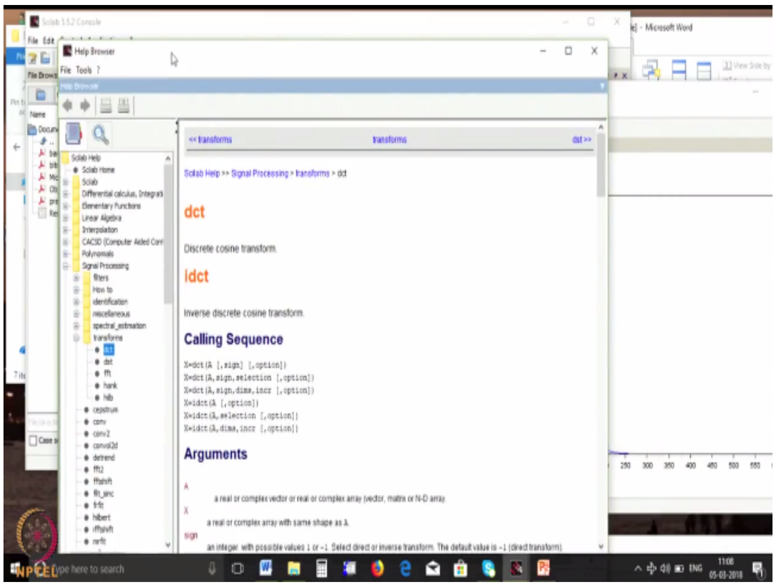
**(Refer Slide Time: 12:55)**



```
//Solve the problem
p0=0.8
[p, err]= datafit(G,Z,p0);
scf(0);clf()
plot2d(X, Y,-1) // the plot with d
plot2d(X,FF(X,p),12) //the plot w
//Cubic polynomial fit to earlier d
X= [0.0,0.1,.2,.3,.4,.5,.6,.7,.8
Y= [0.,.1002,.2013,.3045,.4108
function y=FF(x,p), y=p(1)*x^
Z=[Y;X];
//The criterion function
function e=G(p,z),
y=z(1),x=z(2);
e=y-FF(x,p),
endfunction
//Solve the problem
p0=[0.1;-0.1;0.5;0.1]
[p,err]=datafit(G,Z,p0);
scf(0);clf()
plot2d(X,Y,-1) // the plot with d
plot2d(X,FF(X,p),12) // the plot
p
plot2d(X,FF(X,p),12)
plot2d(X,FF(X,p),12) // the plot
plot2d(X,Y,-1)
c=1.0;
n=512;
spacx = 0.01;
ppi = %pi;
for i=1:n;
x(i) = (i-1)*spacx;
a(i) = exp(-c*x(i)*x(i));end;
plot(a)
```

So it has plotted this Gaussian function, it has plotted this Gaussian function which is e to the – x square. Now I want to take the Fourier transform of this. So if I want to take a discrete cost transform I need to know what is my discrete cost transform, so how will I find out, one way would be to go to the help file, remember, Scilab, one great thing in Scilab is it gives you a help browser.

**(Refer Slide Time: 13:24)**


<< transforms transforms dst >>

Scilab Help >> Signal Processing > transforms > dct

dct

Discrete cosine transform

idct

Inverse discrete cosine transform

Calling Sequence

```
X=dct(A [,sign] [,option])
X=dct(A,sign,selection [,option])
X=dct(A,sign,dims,incr [,option])
X=idct(A [,option])
X=idct(A,selection [,option])
X=idct(A,dims,incr [,option])
```

Arguments

A

a real or complex vector or real or complex array (vector, matrix or N-D array

X

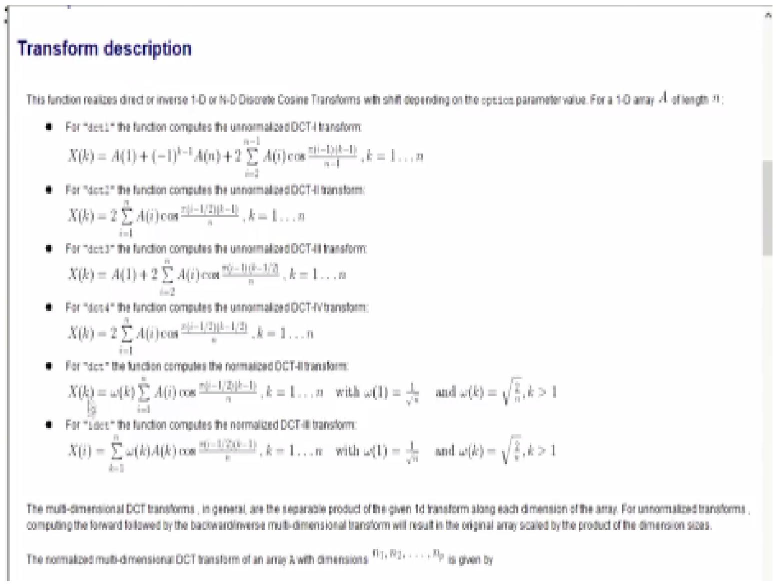
a real or complex array with same shape as A.

sign

an integer with possible values 1 or -1. Select direct or inverse transform. The default value is -1 (direct transform).

So there is a help browser okay, so this is the help browser, so I just typed help in Scilab, DCT is the discrete cost transform. So this is the sequence now. So this help tells you how to do the discrete cost transform.

**(Refer Slide Time: 13:43)**


Transform description

This function realizes direct or inverse 1-D or N-D Discrete Cosine Transforms with shift depending on the option parameter value. For a 1-D array $A$ of length $n$:

- For "dct1" the function computes the unnormalized DCT-I transform:
  $X(k) = A(1) + (-1)^{k-1}A(n) + 2\sum_{i=2}^{n-1} A(i)\cos\frac{\pi(i-1)(k-1)}{n-1}, k = 1\ldots n$
- For "dct2" the function computes the unnormalized DCT-II transform:
  $X(k) = 2\sum_{i=1}^{n} A(i)\cos\frac{\pi(i-1/2)(k-1)}{n}, k = 1\ldots n$
- For "dct3" the function computes the unnormalized DCT-III transform:
  $X(k) = A(1) + 2\sum_{i=2}^{n} A(i)\cos\frac{\pi(i-1)(k-1/2)}{n}, k = 1\ldots n$
- For "dct4" the function computes the unnormalized DCT-IV transform:
  $X(k) = 2\sum_{i=1}^{n} A(i)\cos\frac{\pi(i-1/2)(k-1/2)}{n}, k = 1\ldots n$
- For "dct" the function computes the normalized DCT-II transform:
  $X(k) = \omega(k)\sum_{i=1}^{n} A(i)\cos\frac{\pi(i-1/2)(k-1)}{n}, k = 1\ldots n \quad \text{with } \omega(1) = \frac{1}{\sqrt{n}} \quad \text{and } \omega(k) = \sqrt{\frac{2}{n}}, k > 1$
- For "idct" the function computes the normalized DCT-III transform:
  $X(i) = \sum_{k=1}^{n} \omega(k)A(k)\cos\frac{\pi(i-1/2)(k-1)}{n}, k = 1\ldots n \quad \text{with } \omega(1) = \frac{1}{\sqrt{n}} \quad \text{and } \omega(k) = \sqrt{\frac{2}{n}}, k > 1$

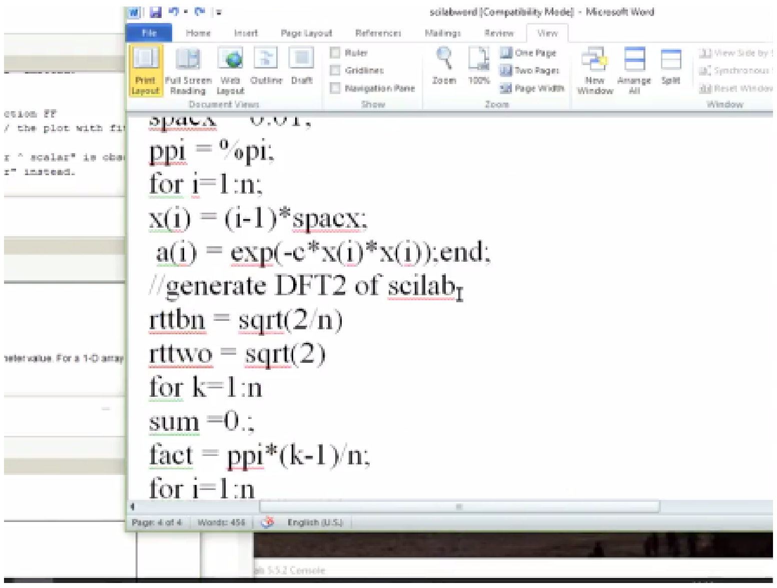
The multi-dimensional DCT transforms , in general, are the separable product of the given 1d transform along each dimension of the array. For unnormalized transforms , computing the forward followed by the backward/inverse multi-dimensional transform will result in the original array scaled by the product of the dimension sizes.

The normalized multi-dimensional DCT transform of an array A with dimensions $n_1, n_2, \ldots, n_p$ is given by

So you will see that this is the description okay, there are several ways of doing discrete cost transform DCT-I, DCT-II DCT-III, DCT-IV, so many are there, so what we want to do. We want to calculate this DCT, discrete cost transform, the algorithm is this, I hope you can see it well, let me make it a little larger. The algorithm is the transformed variable is k okay. So I get xk = omega k * ai * cos pi * i-1/2 * k-1/n k going from 1 to n.

This omega 1 is 1/route n for first value and for other values it is 2/n square root okay, so this is my discrete cost transform, I can do it in 2 different ways, what are the ways I can do. I can do the discrete cost transform through my comments.

**(Refer Slide Time: 14:48)**



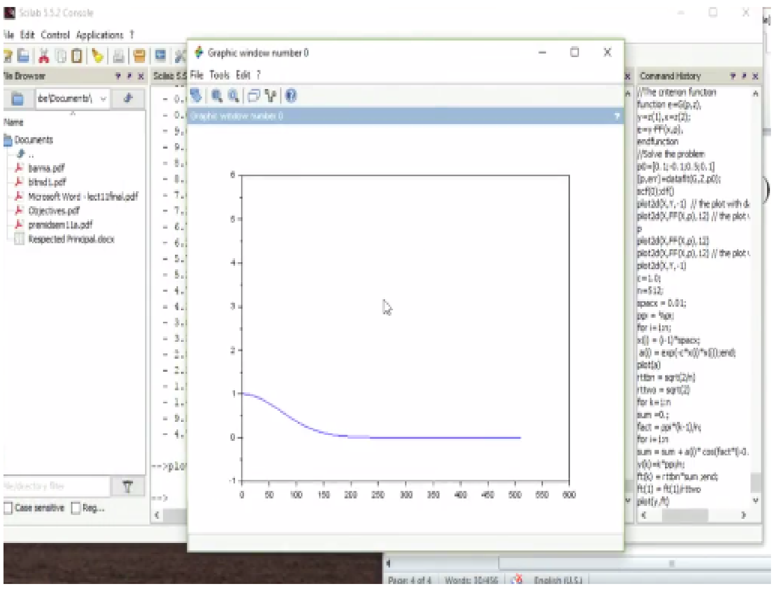
```
ppi = %pi;
for i=1:n;
x(i) = (i-1)*spacx;
 a(i) = exp(-c*x(i)*x(i));end;
//generate DFT2 of scilab
rttbn = sqrt(2/n)
rttwo = sqrt(2)
for k=1:n
sum =0.;
fact = ppi*(k-1)/n;
for i=1:n
```

You see these are what I have done here. I have written my own program to calculate the discrete Fourier transform remember there was a square root of 2/n, square root of 2 i going from 1 to n, sum = 0. This factor is pi * k-1/n and then the integrant was ai * cos of i-0.5 * k-1/n*pi. So I have exactly calculated this ft by my own program I did not use the Scilab discrete for your transform. I use exactly my own program.

So what I will do, I will copy these commands. So I have calculated a. So I will copy paste all my own, what I have done, I have written a Scilab program exactly like the Scilab algorithm. Remember this was my Scilab algorithm. So I shall use the Scilab algorithm as well as my own expression of the same algorithm and show you the calculations okay. So now I do not need this help command anymore.

So I shall close the help command, so I will go here I will paste, I made a mistake I should paste it. So I copy paste it okay, so it gave some warning that is okay. So it has calculated let us see okay.

**(Refer Slide Time: 16:31)**

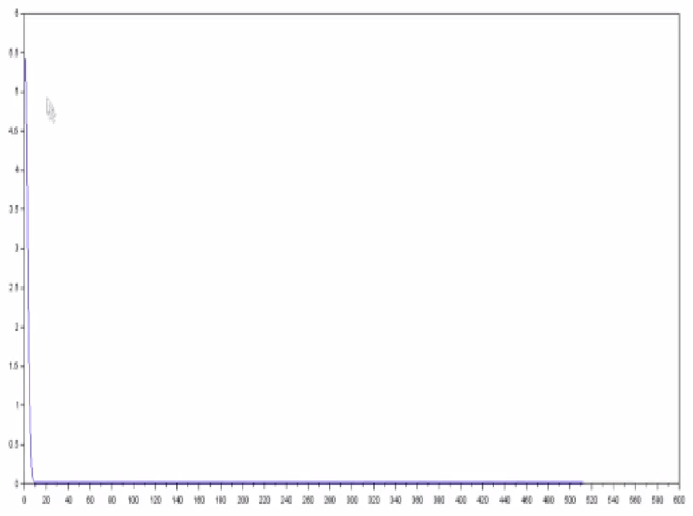


```
//The criterion function
function e=G(p,z)
y=z(1),x=z(2);
e=y-FF(x,p),
endfunction
//Solve the problem
p0=[0.1;-0.1;0.5;0.1]
[p,err]=datafit(G,Z,p0);
scf(0);clf()
plot2d(X,Y,-1) // the plot with d
plot2d(X,FF(X,p),12) // the plot
p
plot2d(X,FF(X,p),12)
plot2d(X,FF(X,p),12) // the plot
plot2d(X,Y,-1)
c=1.0;
n=512;
spacx = 0.01;
ppi = %pi;
for i=1:n;
x(i) = (i-1)*spacx;
 a(i) = exp(-c*x(i)*x(i));end;
plot(a)
rtbn = sqrt(2/n)
rtwo = sqrt(2)
for k=1:n
sum =0.;
fact = ppi*(k-1)/n;
for i=1:n
sum = sum + a(i)* cos(fact*(i-0.
y(k)=k*ppi/n;
ft(k) = rtbn*sum ;end;
ft(1) = ft(1)/rtwo
plot(y,ft)
```

So this is my exponential original function so I want to now see how the Scilab, I shall knock it out. So I shall plot, I will just plot ft, so let us see okay. So this is my Fourier transform, Fourier transform calculated using my program. So let me also calculate this Fourier transform, discrete cost transform using the Scilab, okay. So let me close it. My function was a, so I will say now b=dct is the direct cost transfer (a).

So ft was the Fourier transform I calculated using exactly the algorithm in the Scilab, now I will calculate using the discrete a,1. So I am using this ,1 because it is a forward transform, remember there is a forward transform and a reverse transform. So my b should be the discrete cost transform away forward direction. So let me calculate. So it has calculated, now plot b so you will see this is the discrete cost transform calculated using the Scilab.

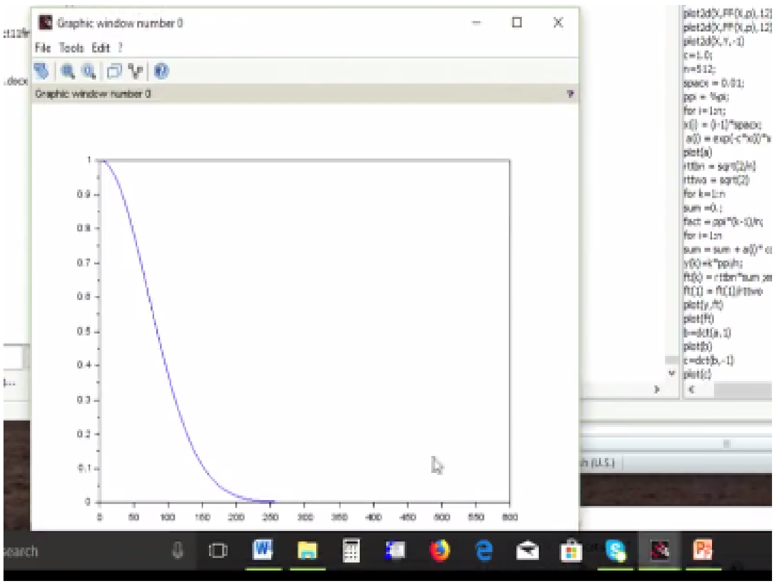
**(Refer Slide Time: 17:54)**



This is exactly what we calculated as ft our ft and the Scilab b are identical because they use the same algorithm. So now what I want to do. I want to calculate the inverse transform. So what is an inverse transform. So I had a and I have got b which is the forward transform. Now let me say c = dct of b,-1. So -1 is my inverse transform okay. So I have a forward transform which is b inverse transform is c.

C is the direct discrete cost transform of p inverse -1. So let me do the c okay, so I calculated c. Now c was the inverse transform so let us say plot, let me plot c.

**(Refer Slide Time: 18:54)**



```
plot2d(X,FF(X,p),12)
plot2d(X,FF(X,p),12)
plot2d(X,Y,-1)
c=1.0;
n=512;
spacx = 0.01;
ppi = %pi;
for i=1:n;
x(i) = (i-1)*spacx;
 a(i) = exp(-c*x(i)*
plot(a)
rtttbn = sqrt(2/n)
rttwo = sqrt(2)
for k=1:n
sum =0.;
fact = ppi*(k-1)/n;
for i=1:n
sum = sum + a(i)* c
y(k)=k*ppi/n;
ft(k) = rttbn*sum
ft(1) = ft(1)/rttwo
plot(y,ft)
plot(ft)
b=dct(a,1)
plot(b)
c=dct(b,-1)
plot(c)
```

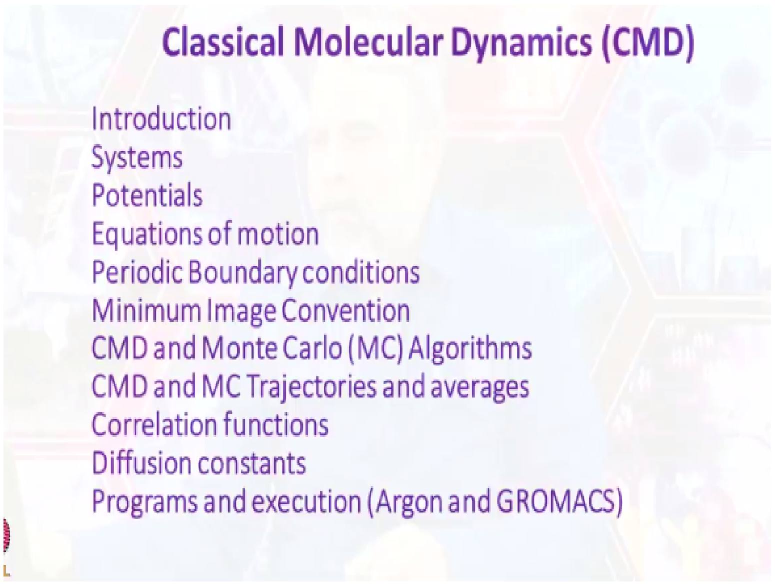
So we will see that this is my c which is really an exponential function okay. So my forward transform was b, my inverse transform is c. So you see that I got this original function, c is the inverse transform with the forward transform, I am getting exactly the same as the original. On the same graph let me also plot a. So I will now plot a on top of c, okay, so let us see, okay, so there has been no difference okay.

There has been no difference, so what did I do. I started with a function a, which was a Gaussian function, took the forward cost transform, took the inverse cost transform and I got the original function. So what I have done so far is to discuss Fourier transforms as well as Laplace transform, told you how it is so easy to calculate Fourier transforms using Scilab. It is just to define the function and use DCT for direct cost transform.

And for sin transform maybe it is DST, so there are other ways of doing it, there are fast transform, fast Fourier transform, FFT again 1 for forward transform, -1 for reverse transform. So you can in your exercises you can see the table that I had given, I had given you a table of forward transforms and then practise using this Scilab software. So this now concludes my discussion on application of Scilab what I will do now.

I will start discussion on molecular dynamic simulations, so this is our main application. I will close all my word file, my main thing here was how to apply computational chemistry to actual chemical calculations or problems in chemistry. So what I want to do, I will begin now discussion on classical molecular dynamics calculations. So this is what we are going to do for the next 3 or 4 hours using our own program.

**(Refer Slide Time: 21:14)**



## Classical Molecular Dynamics (CMD)

Introduction
Systems
Potentials
Equations of motion
Periodic Boundary conditions
Minimum Image Convention
CMD and Monte Carlo (MC) Algorithms
CMD and MC Trajectories and averages
Correlation functions
Diffusion constants
Programs and execution (Argon and GROMACS)

Then after sometime we shall use GROMACS program. So I shall begin now classical molecular dynamic simulation, this is going to be my outline till the end of this course. Up to now we have done lots of numerical methods, wrote Fortran programs and executed those programs. We also did elementary programming before we did the numerical methods. Then I used Scilab to illustrate most of the things that I demonstrated using Fortran programming.

The only thing we did not to was to write a program for finding the diagonal values of a matrix. We just had a very simple program for diagonalization, what I will urge you, you go to that numerical recipes, look at some standard routines for matrix diagonalization and try to use it in your own Fortran program. Of course diagonalization using Scilab lab is very easy, it is just a one line command, lambda.

You know that lambda, b = b diagonal your matrix a. So now I begin classical molecular dynamics, so what is that outline, I will first introduce what is classical molecular dynamics then consider all the systems that can be studied okay and basically we want to know how this classical molecular dynamics can be used to do several useful calculations in chemistry. This classical molecular dynamics very strongly depends on the intermolecular potentials.

So it depends on the intermolecular potentials and once we have the intermolecular potentials I will be solving the equations of motion okay, normally we will be using classical molecular dynamics that means I am going to use classical equations of motions which are Newton’s laws or equivalent forms such as Lagrange A equations, Hamilton's equations and so on.

So in classical molecular dynamics a very important concept is a periodic boundary conditions. The reason we use periodic boundary conditions, the number of particles you can use in classical molecular dynamics is going to be limited into 10s, 100s, 1000s or 10,000s. You cannot use infinite particles because if you use infinite particles your memory requirement would be infinite and you can do very little.

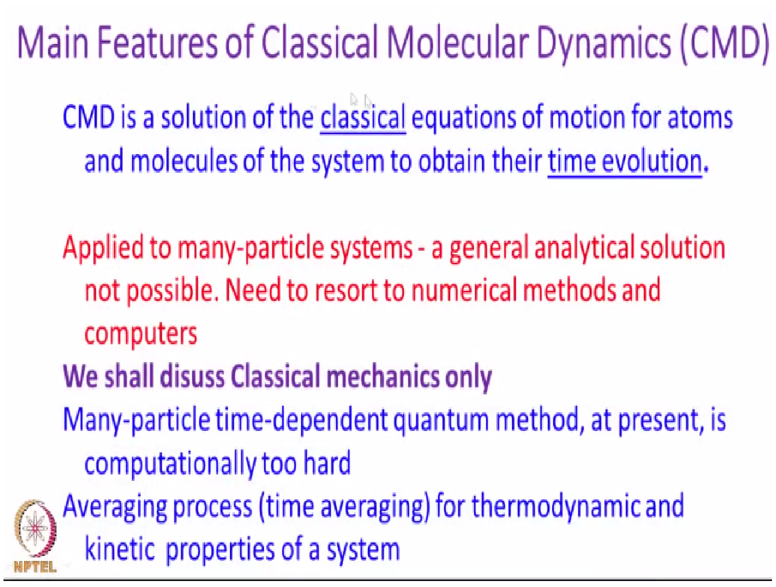
So you will use a finite number such as 100 or 1000. So finite set will not represent a liquid, it will only represent a box or a droplet. So to make your calculation agree with bulk liquids I have to use periodic boundary conditions we will describe that in sufficient detail then in addition to periodic boundary condition there is what is called minimum image convention we will discuss this in a later slide.

Then in addition to classical molecular dynamics we will also be doing Monte Carlo simulations, Monte Carlo algorithms. These Monte Carlo Algorithms use the random numbers which we have already discussed. We discussed 2 types of random numbers, linear random numbers or uniform random numbers between 0 and 1. We also consider random numbers which are Gaussian in nature.

So both those random numbers algorithms are used in Monte Carlo calculations then using these algorithms what we do is we generate a trajectory. I will describe what we mean by molecular dynamics trajectory then after you do molecular dynamic simulations what you are going to calculate these are called correlation functions. There are different kinds of correlation functions which are of use in chemistry.

They are spacial correlation functions and time correlation functions, then we will calculate diffusion constant, diffusion constant tell you how the particles in a liquid diffuse then after describing all the theoretical frame work we will execute 2 sets of programs one is our own program written in Fortran and then we will also describe a public domain software GROMACS, which is a very powerful software and you can apply to many systems of interest okay. So now let us see what are the main features of classical molecular dynamics.

**(Refer Slide Time: 25:26)**



## Main Features of Classical Molecular Dynamics (CMD)

CMD is a solution of the classical equations of motion for atoms and molecules of the system to obtain their time evolution.

Applied to many-particle systems - a general analytical solution not possible. Need to resort to numerical methods and computers

**We shall disuss Classical mechanics only**

Many-particle time-dependent quantum method, at present, is computationally too hard

Averaging process (time averaging) for thermodynamic and kinetic properties of a system

The main features of a classical molecular dynamics are the classical molecular dynamics tries to solve classical equation of motion. You want to solve this equations of motion for atoms and molecules of the system. So take a system containing atoms and molecules. You solve the equations of motion to obtain the time evolution, where going to evolve the system as a function of time.

So that is why it is called dynamics, dynamics because how things move with the function of time and we want to apply this to many particle systems okay, so what is a many particle system, many particle system is anything more than 2 particles is many particles, okay, so once you have more than 2 particles say 3 particles, 4, 5, a general analytical solution is not possible.

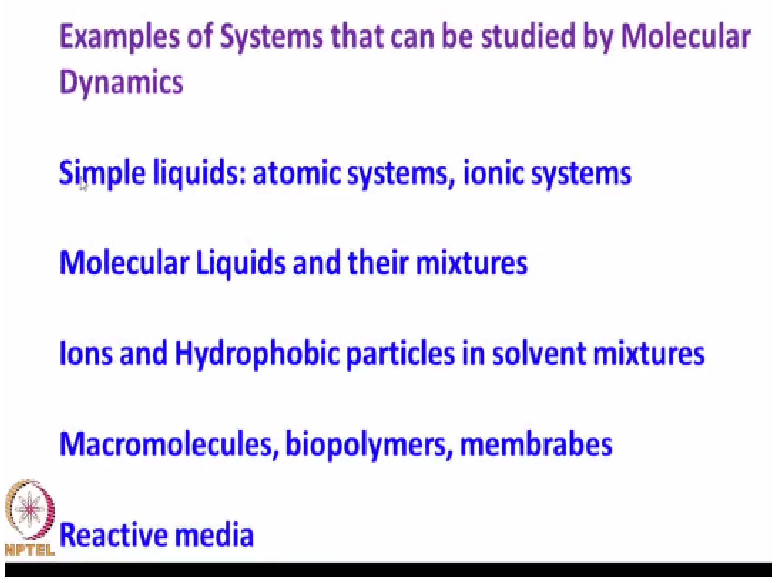
You do not have an exact solution with a formula okay, just as you have formula for 2 particle motion like hydrogen atom or an earth and sun moving together, one is a classical system, one is a quantum system. So for 2 particles there is an exact solution, for 3 or more there is no exact solution. So you have to use numerical methods. So that is how whatever we did in our course all the numerical method that we used during the course will now be extremely useful in solving these problems of real interest.

As I mentioned again we will be discussing only classical mechanics because if you want to discuss many particle time dependent quantum method it is very difficult. The time dependent quantum method involving the Schrodinger equation is much more difficult than classical molecular dynamics and classical molecular dynamics gives you fairly accurate results for certain systems. So we will also see which system we can study and what are our algorithms for classical molecular dynamics.

And the main thing would be after you do classical molecular dynamics you study the time evolution, you will average over all the results of your simulation okay, it is called time averaging, you average to get thermodynamic and kinetic properties of the system. So the goal is your thermodynamics and kinetics. Your technique is molecular dynamics and in the next lecture we shall let us see okay.

I will just, before I conclude let me give you examples of systems that can be studied using molecular dynamics.

**(Refer Slide Time: 27:46)**



Examples of Systems that can be studied by Molecular Dynamics

Simple liquids: atomic systems, ionic systems

Molecular Liquids and their mixtures

Ions and Hydrophobic particles in solvent mixtures

Macromolecules, biopolymers, membrabes

Reactive media

One set is simple liquids atomic systems, ionic systems, molecular liquids and their mixtures, ions and hydrophobic particles in solvent mixtures, macromolecules, biopolymers, membranes, reactive media. So all the systems can be studied using molecular dynamics. So in the next lecture I will be discussing several algorithms that are useful in molecular dynamics.

So to summarize what we did today, we used Scilab to do Fourier and Laplace transforms. Laplace transform I will leave as an exercise. Go to the Scilab website, look at the Laplace transform, how to use Laplace transform. Fourier transforms I have already told you how to do it using your own program as well as using Scilab, I introduced classical molecular dynamics and in our remaining lectures in the course.

We will be discussing several aspects of molecular dynamics and Monte Carlo simulations; I will close here. Thank you.