Roots of equations, Random numbers and Integral transforms

Key words: polynomials, roots of polynomials, random numbers, Gaussian random numbers, integral transforms, Fourier transforms, Laplace transforms

7.1 Introduction

This chapter is an assorted collection of topics. We are devoting a total of four chapters on numerical methods and in this last chapter on numerical methods, we are considering the themes that could not be fitted into the earlier three chapters. As was mentioned in the chapters on matrices, we need to solve polynomial equations to gets the roots of these equations. These roots are the eigenvalues of the matrix. Random numbers, as the name indicates are numbers that are randomly distributed(with no conceivable pattern)within a given range of numbers, say between 0 and 1 as in the case of most calculators. These numbers find extensive use in studying processes called 'stochastic processes',which constitute a large number of processes in nature.Imagine two persons tossing a coin each a hundred times. One person may get a sequence HHTTTHTHTHHTTTHHHTTT…. and the other person may get a sequence TTHTHTTTTHHHTHHHHTHH…… If they repeat the experiment again, they are unlikely to get the same sequence. Such sequences of events are called a random sequence of events. The same argument holds for a series of numbers in a given range of randomnumbers. Processes such as diffusion of one type of molecules into another, picking one card out of a well shuffled set of cards (wherein the probability of picking one card is exactly equal to the probability of picking any other card) are called stochastic processes. Random numbers are also used in calculating multi-dimensional integrals as well as in a method for studying chemical equilibrium through

what is called the Metropolis Monte Carlo method. Towards the end of this chapter, we consider integral transforms. Examples are the Fourier transform (FT)and the Laplace transform(LT). These are especially included here as their applications are very special to chemistry. The former one is used in all FT NMR machines to convert the time domain data (such as the free induction decay signals) into the frequency domain. They are also used to convert the wave vector data (intensity versus the wave vector which is the reciprocal length) of X-ray diffraction patterns into spatial data (data in the x, y, z co-ordinate space) which is the information on crystal structures. The principal use of Laplace transforms is in solving differential equations.

7.2 ROOTS OF POLYNOMIAL EQUATIONS

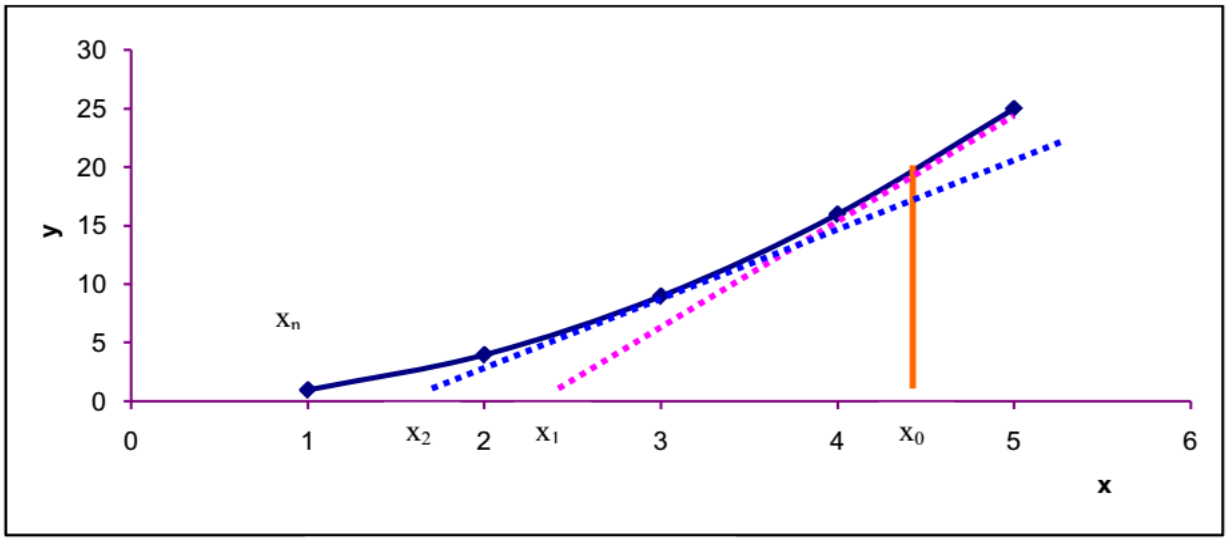


Fig1

Please use the above figure and not the one in the last version.

Consider the function $y = f(x)$ shown in fig 7.1. We want to find the value x_n at which $f(x_n) = 0$. This is a root of the function$f(x) = 0$. Since we do not know the value x at which $f(x) = 0$, we begin with an initial guess $= x_0$. If we approximate $f(x)$ by linear interpolation at x_0, then the slope (line) touches the x axis atx_1. Now consider $f(x_1)$ and the slope at x_1. This slope crosses the x-axis atx_2.

Repeat the procedure till you reach x_n at which the function f itself is zero. This is called the Newton Raphson method. The algorithm can be derived as follows

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)f'(x_i) + \cdots \quad (7.1)$$

Truncating the Taylor expansion at first order, x_{i+1} can be obtained from x_i by expecting $f(x_{i+1})$ to be zero.

$$x_{i+1} = x_i - f(x_i)/f'(x_i) \quad (7.2)$$

If $f(x_{i+1})$ is not zero, then repeat the procedure until $f(x_n) = 0$. There are several methods for finding roots of equations such as the successive bisection method, the inverse interpolation method and so on. Often the roots of equations could be complex and special care has to be taken for such cases.

An n^{th} order polynomial can be expressed in terms of its roots as follows

$$P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = 0$$

$$= (x - x_1)(x - x_2) \ldots\ldots (x - x_n) \quad (7.3)$$

It is seen by expanding the above equation that that $a_0 = \prod x_i$, and that all the other coefficients of the polynomial can also be expressed in terms of the rootsx_1, x_2, … x_n. Once a given root of the polynomial say x_1 is found, to find the remaining roots, we divide the polynomial by $(x - x_1)$ to get a polynomial of degree n-1 and repeat the procedure until the roots are obtained.There are other clever methods for finding roots such as taking out factor such as x*x + p*x +q.

The problem of dividing a polynomial by $(x - x_i)$ may be taken up as an exercise.

7.3. THE PROGRAM FOR THE NEWTON RAPHSON METHOD

```
c     program to solve equation by using Newton Raphson method
write (*,*)'enter intial guess'
```

```
write (*,*)'enter intial guess'
read (*, *)X0
write (*, *) 'enter tolerrence'
read (*, *) EPS
10 X1= X0 - F(X0)/F1(X0)
cthere will be in trouble if x0 equals zero and special care should be taken to
c     avoid dividing by zero.
if ((abs(X1 - X0)/abs(X0)) .lt. EPS) then
write (*, 1) X1
1   format ('solution=',F15.10)
stop
else
      X0 = X1
go to 10
end if
end
      FUNCTION F(X)
      X2 = X * X
c   F is the value of the function to be evaluated
      F = X2 - 25
return
end
      FUNCTION F1(X)
c   F is the value of the derivative of the function to be evaluated
F1 = 2 * X
return
end
```

Assignment: Execute this programme and modify the programme for other polynomials.

7.4. RANDOM NUMBERS

Random numbers result from such experiments as throwing coins (heads vs. tails), dice (numbers 1 to 6), picking a card from a well shuffled pack of playing cards, rotating a wheel/needle in a gambler's game, and it has results in a given range of outcomes.

Example 1: Start with number 01. Multiply by number 13. The result is 13. Multiply by 13 again and discard multiples of 100. The following two digit sequence is generated.

01 13 69 97 61 93 09 17 21 73 49 27 81 53 89 57 41 33 29 77 01 ……

After the number 77, the sequence 01 to 77 repeats again and again. These numbers are fairly uniformly distributed between 0 to 100 with no obvious preference in any region of the full range of 0 to 100. But there are some trends such as upward sequences of size 4, i.e. 01, 13, 69, 97 and 9, 17, 21 and 73 and downward sequence such as 89, 57, 41, 33, 29.

Instead of 01 and 13, if we tried 01 and 25, our sequence is 01, 25, 25, 25… which has hardly any fluctuation and is in no way random. We can generalize this to the following algorithm.

R_i is the i^{th} random number, R_{i-1} is the $(i-1)^{th}$.

$$R_i = (mul * R_{i-1} + add) * mod\ m \qquad (7.4)$$

Here *mul* is an integer multiplier, *add* is an integer and mod m is the remainder function. i.e. 'a mod b' divides a by b and gives the remainder as the result e.g. 169 mod 100 is 69 and 51 mod 13 is 12.

To get a good distribution of random numbers, we need take *add*, and *mul* as some large prime numbers. These values are obtained by trial and error.

e.g. $R_i = (25173 * R_{i-1} + 13849) * mod\ 65536 \qquad (7.5)$

This gives a much better "random sequence than the $R_i = (13 * R_{i-1} + 0) * mod\ 13$

It should be noted that unlike picking cards from a pack, these "random" sequences are extremely well defined once the starting number or seed R_0 is determined and hence these are often called "pseudo" random numbers.

7.5. GAUSSIAN RANDOM NUMBERS

We have considered so far random numbers that are uniformly distributed over a given range, say 0 to 1 or 0 to 65536. The second set can be trivially converted to the set between 0 and 1by dividing all the numbers by 65536. The random numbers in any range (-L, L) by transforming all the numbers by the formula (r – 0.5) x 2L.

In some applications we need random numbers distributed according to a specific distribution such as the Gaussian distribution shown in the following figure, fig. 7.1.

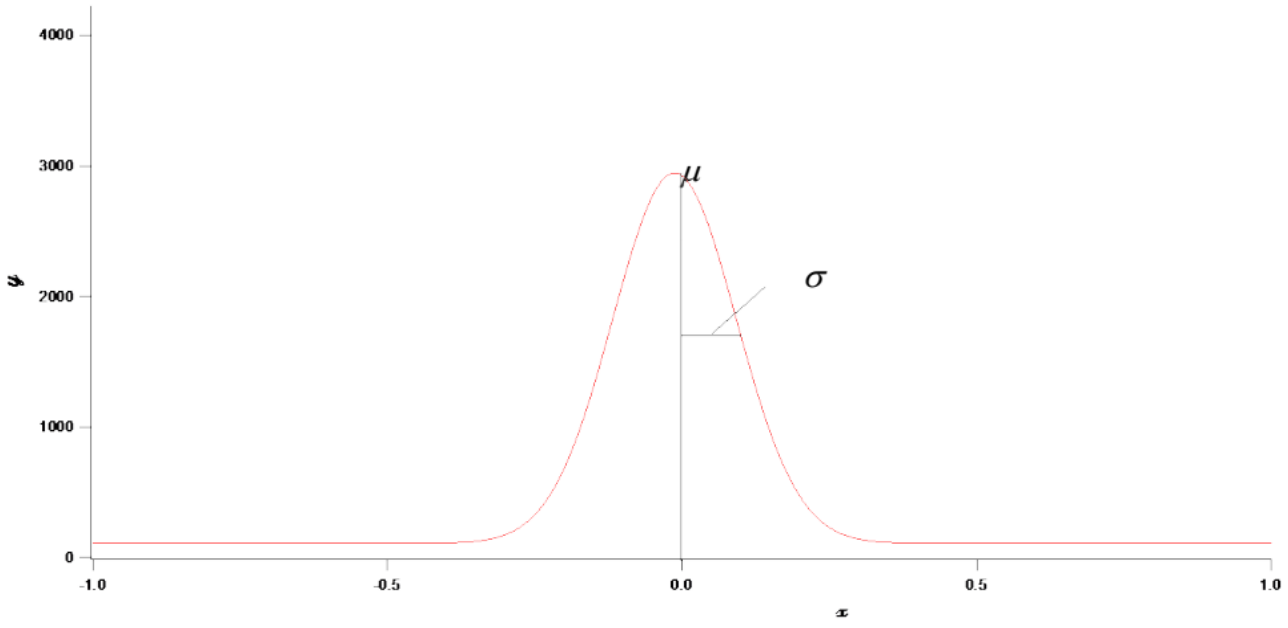


*Fig.*7.1. *A Gaussian Distribution with mean* μ *and* Variance σ^2

This is the well-known bell shaped curve. The mean of the distribution is μ and the variance is σ^2. The formula is

$$\frac{1}{\sqrt{2\pi\sigma}} \exp[-(x-\mu)^2/2\sigma^2] \quad (7.6)$$

A much simpler distribution is when $\sigma = 1$ and $\mu = 0$. We have the Gaussian distribution $p(y)dy$ as

$$p(y)dy = \frac{1}{\sqrt{2\pi}} e^{-y^2} dy \quad (7.7)$$

If we have two random numbers x_1 and x_2, both in the range of 0 to 1, we can obtain Gaussian distribution in y through the equations

$$y_1 = (-2\, lnx_1)^{\frac{1}{2}} \cos 2\pi x_2 \text{and} y_2 = (-2 \ln x_1)^{\frac{1}{2}} \sin 2\pi x_2 \quad (7.8)$$

Or equivalently

$$x_1 = e^{[-\frac{1}{2}(y_1{}^2+y_2{}^2)]} \text{and} x_2 = \frac{1}{2\pi} tan^{-1}(\frac{y_2}{y_1}) \quad (7.9)$$

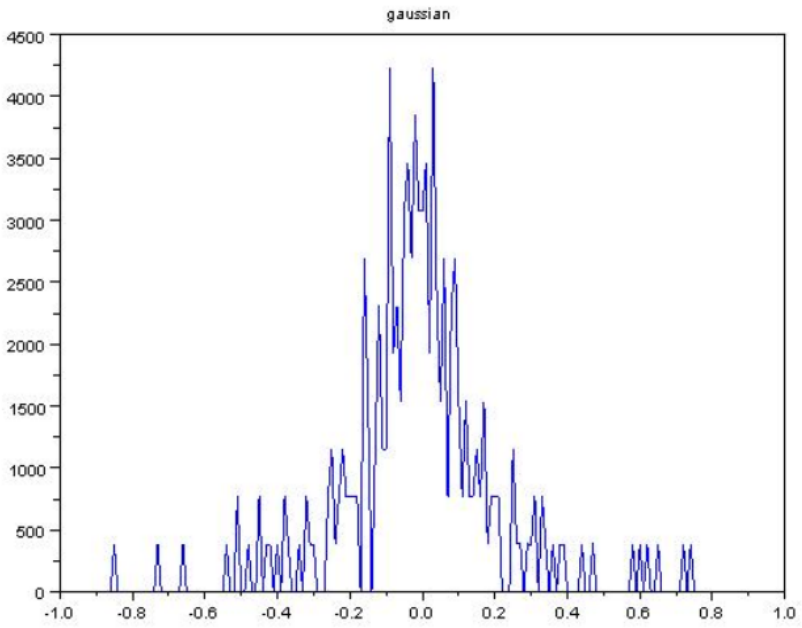
To avoid using trigonometric functions, we can obtain two numbers v_1 and v_2 which lie in a circle of unit radius about the origin and note that $R^2 = v_1{}^2 + v_2{}^2$ is a uniform random number and can be used in place of x_1 and the angle that the vector from the origin to (v_1, v_2) makes with the vector $(0, v_1)$ can be a random angle $2\pi x_2$. The advantage is that cos $2\pi x_2$ is now $\frac{v_1}{R}$ and sin $2\pi x_2$ is $\frac{v_2}{R}$.

```
program gauss
dimension a(201)
c       integer*8 (i-m)
c       determine thousand gaussian random numbers
c       this is an illustartive program and not accurate
iseed = 123
```

```
do 5 i = 1, 201
a(i) = 0
5  continue
c       gran1 = pran(iseed)
c       gran2 = pran(iseed)
do 10 i = 1, 10000
call gran (gran1, gran2, ncount,iseed)
c       write (*, *) gran1
ii= (gran1 * 10) - 100
if (abs(ii) .gt. 100) go to 10
a(ii) = a(ii) + 1
   10 continue
open (unit=11, file='x.txt')
open (unit=12, file='y.txt')
do 15 i = 1, 201
xx = (i - 101)* 0.01
write(11,*) xx
  15   write(12, *) a(i)
end
subroutine gran(gran1, gran2, ncount,iseed)
ncount=0
 10   x1 = pran(iseed)
x2 = pran(iseed)
w1 = (2 * x1 - 1.)
w2 = (2 * x2 - 1.)
rsq = w1 * * 2 + w2 ** 2
ncount = ncount + 1
if (ncount .gt. 1000) stop
if (rsq .ge. 1.0) go to 10
```

```
fac = sqrt (-2.0 * log (rsq))/rsq
      gran1 = w1 * fac
      gran2 = w2 * fac
return
end
functionpran(iseed)
mm = 16777216
ii= (65539 * iseed + 12345)
pr = mod(ii, mm)
pran = real(pr / mm)
iseed = ii
return
end
```

Note that in eq. (7.8) while x_1 and x_2 span the range 0 to 1, lnx1, covers the entire range of 0 $to -\infty$!!



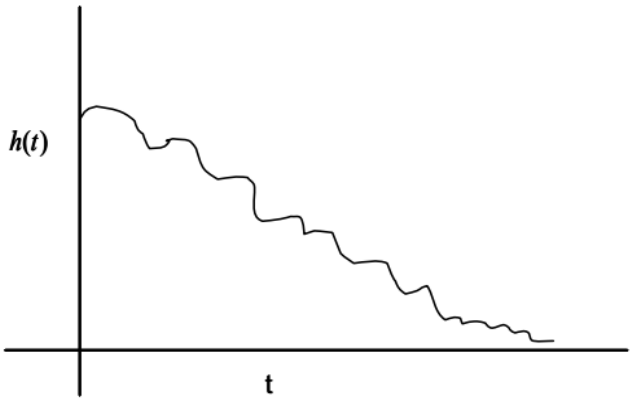
The above program can be used to obtain the Maxwell distribution of velocities of a gas at a given temperature (It follows a Gaussian distribution. The input to the programme will be the standard deviation of the distribution, σ,which is a function of temperature.

7.6 INTEGRAL TRANSFORMS

The above program can be used to obtain the Maxwell distribution of velocities of a gas at a given temperature (It follows a Gaussian distribution. The input to the programme will be the standard deviation of the distribution, σ ,which is a function of temperature.DELETE THis PART FROM THE EARLIER VERSION

Fourier transforms have had huge impact on fields ranging from astronomy and spectroscopy to medical imaging and seismology.

If we have a time dependent function $h(t)$ which is oscillatory as shown in fig. 7.2, we want to know which are the frequencies f that contribute predominantly to the function. This is given by the Fourier transform,



$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

Here f is the frequency in sec^{-1}. Often angular frequency w defined as $2\pi f$ is preferred to f and the integral transform looks a lot simpler.

$$H(w) = \int_{-\infty}^{\infty} h(t)e^{iwt}\,dt \qquad 7.10$$

The inverse transform obtains $h(t)$ in terms of$H(w)$, i.e.

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(w)e^{-iwt}\,dw \qquad 7.11$$

Numerical data is often available in discrete forms as in$\big(t_i, h(t_i)\big)$for $i = 0, \ldots . N-1$. If the interval in time is Δt, then $t_i = i\Delta t, i = 0, \ldots . N-1$. Since only N values of $h(t_i)$ are available, it is most convenient to seek the values of the transform also at N points as in

$$H(f_n) = \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_k}.\Delta t \qquad 7.12$$

$$H_n = (\Delta t)\sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \qquad 7.13$$

This is the discrete Fourier transform of N pointsh_k. If the spacing in t space between adjacent data points isΔt, then the spacing in the frequency space (spacings between adjacent points) Δf is given by$\Delta t.\Delta f = \frac{2\pi}{N}$

7.7 Sine and Cosine Transforms:

For real functions, it is often useful to use sine (for odd functions) and cosine (for even functions of t) transforms.

The sine transform uses sine functions only as a complete set of functions in the interval from 0 to 2π. The sine function is defined as

$$F_k = \sum_{j=0}^{N-1} f_i \sin\left(\frac{\pi jk}{N}\right) \qquad 7.14$$

Where $j = 0,1, \ldots . . N-1$and the data array is f_i with $f_0 = 0$

The inverse sine transform is

$$f_j = \frac{1}{N}\sum_{k=0}^{N-1} F_k \sin(\frac{\pi jk}{N}) \qquad 7.15$$

The cosine transform is obtained from the above functions by using the cosine function in place of the sine function and the requirement of $f_0 = 0$ is not necessary.

7.8 THE FAST FOURIER TRANSFORM (FFT)

The discrete Fourier transform (FT) described above requires a set of N × N operations to get the Fourier transform values of the function F_k from f_j. In an algorithm called the Fast Fourier transform, we need to use only $Nlog_2N$ operations in the place of N^2 operations. This is of great advantage when we need to do FTs a number of times during the execution of a program. For say N = 6, the time savings using the FFT is in the ratio of $\frac{N}{log_2 N}$ which is of the order of 1000 for N = 10^6. The FFT uses the DANIELSON and LANCZOS LEMMA that a FT of N points can be written as a sum of two FTs, one for odd number of points of the original N and the other for the even numbered. This argument can be reduced recursively by a factor of 2 during each iteration. If N = 2^{10}, this reduction can be done 10 times. The FFT uses a bit reversal algorithm to reduce the number of computations. We will use examples of FFT in the next chapter

7.9 PROGRAM FOR SINE AND COSINE TRANSFORMS

The simplest way to go about these transforms is to sum the terms in the various equations we have come across so far. If the values of the date for functions $f(t_n)$ are available at very closely spaced intervals, then the sum is sufficient. If the spacing is larger, then some methods of integration, such as Simpson's rule, can be used. It is important to check the accuracy of the program by calculating the inverse transform and check how closely the inverse transformed function matches with the original function.

7.10 LAPLACE TRANSFORMS:

As mentioned in the introduction, Laplace transforms are very useful in solving differential equations. The Laplace transform is defined by

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)dt \qquad 7.16$$

The integral is very easy to evaluate using the Trapezoidal or the Simpson's rule, if $f(t)$ is known at equally spaced points. One has to ensure that enough sampling points are taken so that $e^{-st}f(t)$ values are negligible outside the range considered. The inverse Laplace transform is defined as

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \frac{1}{2\pi i}\lim_{T\to\infty}\int_{\gamma-iT}^{\gamma+iT} e^{st}F(s)ds \qquad 7.17$$

Here γ is a real number. This is an integration in the complex plane and $\gamma > Re(Sp)$ where Sp refers to all the singularities of $F(s)$. If all the singularities of $F(s)$are in the left half of the complex plane, then γ can be chosen as zero as $Re(Sp) < 0$ for all values of P.

3-dimensional Fourier transforms(using polar coordinates and spherical symmetry here)

$$\tilde{f}(k) = 4\pi\int_0^L f(r)\sin(rk)(\frac{r}{k})dr \qquad 7.18$$

$$f(r) = \frac{1}{2\pi^2}\int_0^K \tilde{f}(k)\sin(rk)(\frac{k}{r})dk \qquad 7.19$$

Broyles Discrete representations

$$\tilde{f}(k) = \tilde{f}(m\hat{k}) = \left[\sum_{n=0}^{N-1} n\, f(n\hat{r})\sin(nm\hat{r}\hat{k})\right]\left(\frac{4\pi\hat{r}^2}{m\hat{k}}\right) \qquad 7.20$$

$$f(r) = f(n\hat{r}) = \frac{\hat{k}^2}{2\pi^2 n\hat{r}}\sum_{m=0}^{N-1} m\,\tilde{f}(m\hat{k})\sin(nm\hat{r}\hat{k}) \qquad 7.21$$

$$\hat{r}\hat{k} = \frac{2\pi}{(2N-1)}$$

Spacing in r space $r = (j-1)\hat{r}$; spacing in k space $k = (k-1)\hat{k}$

$$\tilde{f}(k) = e^{-k^2/4} \quad f(r) = \left(\frac{1}{\pi}\right)^{3/2} e^{-r^2} \qquad 7.22$$

```
program FT
c      this calculates 3 dimensional FT and inverse FT
dimension fr(2048),fk(2048),frift(2048),fkan(2048)
c      the function in r space is f(r) = exp (- r), spacing 0.005
open(unit = 12, file = 'ftoutput')
      n = 2048
      ntimes2 = 2 * n
spacer = 0.005
pi = 3.14159
spacek = 2.0 * pi / (ntimes2 - 1)
rhkh = spacer * spacek
c      determine f(r)
do  i = 1,n
cfr(i) = exp ( -(i-1)* spacer)
fr(i)=(1/(pi * * (1.5))) * exp(- ( i-1)* (i-1)* spacer**2)
write(12,*) fr(i),(i - 1) * spacer, (i-1) * spacek
end do
```

```
c       determine forward transform
do 100 k = 1,n
      kmin1 = k - 1
sum = 0.0
if (k.eq.1) then
factor = 4.0 * pi * spacer * spacer
do 99 j = 1,n
      jmin1 = j - 1
rr = jmin1 * spacer
sum = sum + jmin1 * fr(jmin1)
99  continue
sum=sum*factor
else
spk = (k - 1) * spacek
factor =  4.0 * pi * spacer * spacer / (kmin1 * spacek)
do 999  j = 1,n
      jmin1 = j - 1
sum = sum + jmin1 * fr(jmin1) * sin (kmin1 * jmin1 * rhkh)
999 continue
sum=sum*factor
endif
fk(k) = sum
100 continue
c       determine the inverse transform
do 200 kk = 1,n
      kmin1 = kk - 1
sum = 0.0
if (kk.eq.1) then
factor = spacek * spacek / (2. * pi * pi)
```

```
do  199 j = 1,n
      jmin1 = j - 1
rr = jmin1 * spacer
sum = sum + jmin1 * fk(jmin1)
199 continue
sum= sum* factor
else
spr = (kk - 1) * spacer
factor = spacek * spacek / (2. * pi * pi * kmin1 *spacer)
do  299j = 1,n
      jmin1 = j - 1
sum = sum + jmin1 * fk(jmin1) * sin (kmin1 * jmin1 * rhkh)
299 continue
sum= sum * factor
endif
frift(kk) = sum
200 continue
write(12,*) 'spacer, spacek =',spacer,spacek
do m = 1,n
fkan(m) = exp (-(m-1)**2 * (spacek ** 2)/4)
write(12,*) fr(m), fk(m),fkan(m),frift(m)
end do
end
```

7.11. SUMMARY

In this chapter, we have explored three topics in numerical methods which play a critical role in helping solve problems which are useful to chemistry. Solution of roots of equations find applications in all studies of Brownian motion and also studies concerning thermodynamic equilibrium which use the

metropolis Monte Carlo algorithm. This algorithm allows sampling states of a system according to the Boltzmann distribution of energies. Integral transforms play a major role in solving differential equations since these convert differential equations to algebraic equations which are easier to solve. After the solution is obtained in the transformed variable, we need to use the inverse integral transform to get the solution of the original problem. Fourier transforms are basic ingredients of all FT NMR machines and X-ray diffraction machines. The applications these numerical methods extend not only to physical and biological sciences, but to social sciences as well.

PROBLEMS

1. Write a program to find roots for the following equations using appropriate methods.

$$x^3 - 4x^2 - x + 4 = 0$$

$$x^4 + x^3 - x^2 - x + 1 = 0$$

2 Find a set of 1000 random numbers and obtain their mean and standard deviation.

3. Find the Fourier transform of a function whose analytic FT is known and compare the numerical and analytic FT.

4. Do the same for a Laplace transform.