HW3 Integration and Monte Carlo

Fei Ding

September 25, 2009

Problem1

To get the probability of a dart landing in a D-dimensional hypersphere that is embedded in a D-dimensional cube, the crucial step is to get the volume of the D-dimensional hypersphere of reaiud R, $V_D(R)$, which should be proportional to R^D, that is $V_D(R) = C_D R^D$.

In order to get the volume, I would evaluate the integral $\int e^{-\mathbf{X}^2}\, d^D\mathbf{x}$ in two different ways.

In D-dimensional space, the position vector $\mathbf{x}$ is denoted as$(x_1, ..., x_D)$ in Cartesian coordinates. The volume element

$$dV_D = d^D\mathbf{x} = \prod_{i=1}^{D} dx_i$$

so the integral can be written as the product of integral of one dimension as,

$$\int e^{-\mathbf{X}^2}\, d^D\mathbf{x} = \int \dots \int e^{-\sum_{i=1}^{D} x_i^2} \prod_{i=1}^{D} dx_i = \pi^{D/2}$$

since we know that $\int e^{-x^2}\, dx = \pi^{1/2}$.

On the other hand, $dV_D = DC_D R^{D-1} dR$, so the integral can be also written as,

$$\int e^{-\mathbf{X}^2}\, d^D\mathbf{x} = \int e^{-R^2}\, dV_D = \int_0^\infty e^{-R^2} DC_D R^{D-1}\, dR = DC_D \frac{1}{2}\Gamma(\frac{D}{2}) = (\frac{D}{2})!C_D$$

according to the equation

$$I_\nu = \int_0^\infty e^{-y^2} y^\nu\, dy = \frac{1}{2}\Gamma(\frac{\nu+1}{2})$$

. Compare these two expressions, we can get that $C_D = \pi^{D/2}/(\frac{D}{2})!$, so finally we get the expression of the volume of the D-dimensional sphere,

$$V_D(R) = \frac{\pi^{D/2}}{(D/2)!} R^D$$

.

So,

$$V_2(R) = \pi R^2, V_{10}(R) = \frac{\pi^5}{5!}R^{10}, V_{20}(R) = \frac{\pi^{10}}{10!}R^{20}$$

Problem2

The integral,

$$\begin{aligned} I &= P\int_0^{+\infty} \frac{f(k)}{g(k)-g(k_0)}\,dk \\ &= P\int_0^{k_0-\epsilon} \frac{f(k)}{g(k)-g(k_0)}\,dk + P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)}{g(k)-g(k_0)}\,dk + P\int_{k_0+\epsilon}^{+\infty} \frac{f(k)}{g(k)-g(k_0)}\,dk \end{aligned}$$

The principal integral is divided into three parts. It is easy to see that the first and the last terms do not have singularity. So the principle value integral symbol P can be removed. Thus, we only need to deal with the second term?

Consider the first order expansion of function $g(k)$,

$$g(k) = g(k_0) + g'(k_0)(k-k_0),$$

so the second term,

$$\begin{aligned} P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)}{g(k)-g(k_0)}\,dk &= P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)}{g'(k_0)(k-k_0)}\,dk \\ &= \frac{1}{g'(k_0)}P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)}{k-k_0}\,dk \end{aligned}$$

We know $P\int_{k_0-\epsilon}^{k_0+\epsilon}\frac{dk}{k-k_0} = 0$ because the curve of $\frac{1}{k-k_0}$ as a function of k has equal and opposite areas on both sides of the singular point k_0. So we can do a simple subtraction of zero integral to the second term without affecting the principal-value integration,

$$\begin{aligned} \frac{1}{g'(k_0)}P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)}{k-k_0}\,dk &= \frac{1}{g'(k_0)}P\int_{k_0-\epsilon}^{k_0+\epsilon} \frac{f(k)-f(k_0)}{k-k_0}\,dk \\ &= \frac{2\epsilon f'(k_0)}{g'(k0)} \end{aligned}$$

The last equation is from the first order expansion of function f(k) near k_0, $f(k) = f(k_0) + f'(k_0)(k-k_0)$

Problem3

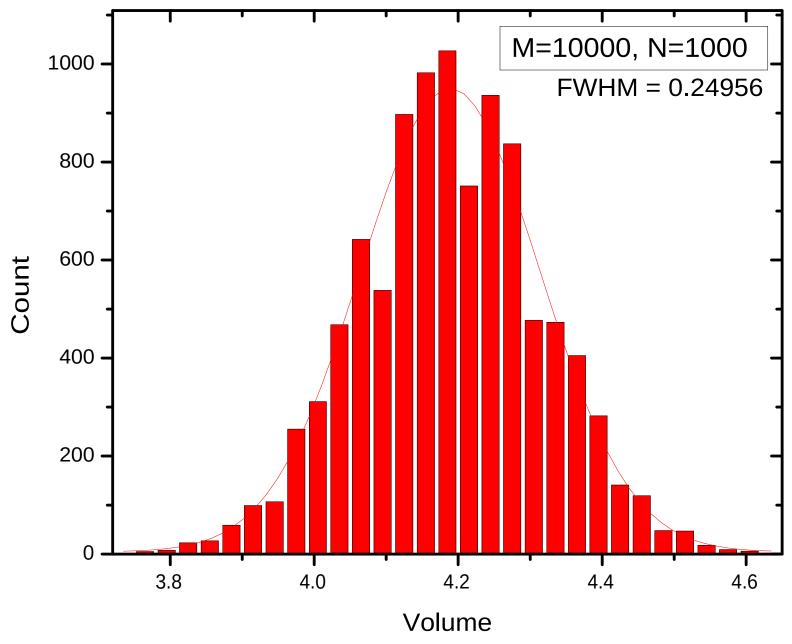
```
The Code:
program sphereVolume
IMPLICIT NONE
integer, parameter:: M=10000
integer:: seed, N, numIn
integer:: i, numTimes, numThrow
real(8):: x, y, z, r
real(8):: r0, cube, sphere, rm, sRange
real(8), dimension(M):: volume
open(unit=1, file='data1.txt') !if N=100,000, file='data2.txt'
volume=0.0
N=1000
r0=1.0
sRange=2*r0
rm=r0-sRange
cube=sRange**3
seed=918172
call srand(seed)
do numTimes=1, M
   numIn=0
   do numThrow=1, N
      x=(rand()*sRange)+rm
      y=(rand()*sRange)+rm
      z=(rand()*sRange)+rm
      r=sqrt(x**2+y**2+z**2)
      if (r<=r0) then
         numIn=numIn+1
      end if
   end do
   sphere=real(numIn)/N*cube
   volume(numTimes)=sphere
 write(1,*) volume(numTimes)
end do
close(1)
end program sphereVolume
```

I used Origin to get the two histograms, which can do statistical count automatically.

It can be seen from the figure that the histogram is closer to Gaussian distribution as the N increases. From the relationship $FWHM = 2\sigma\sqrt{2ln2} = 2.35\sigma$, the standard deviations of the histograms are 0.106 and 0.0107 for N=1000 and N=100,000, respectively.

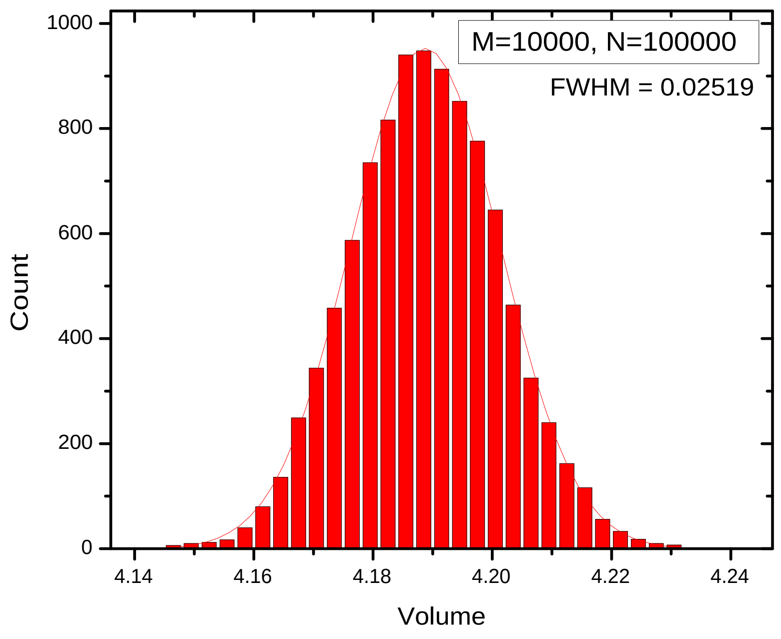
Problem 4

The code of this problem is based on the code provide on the course webpage

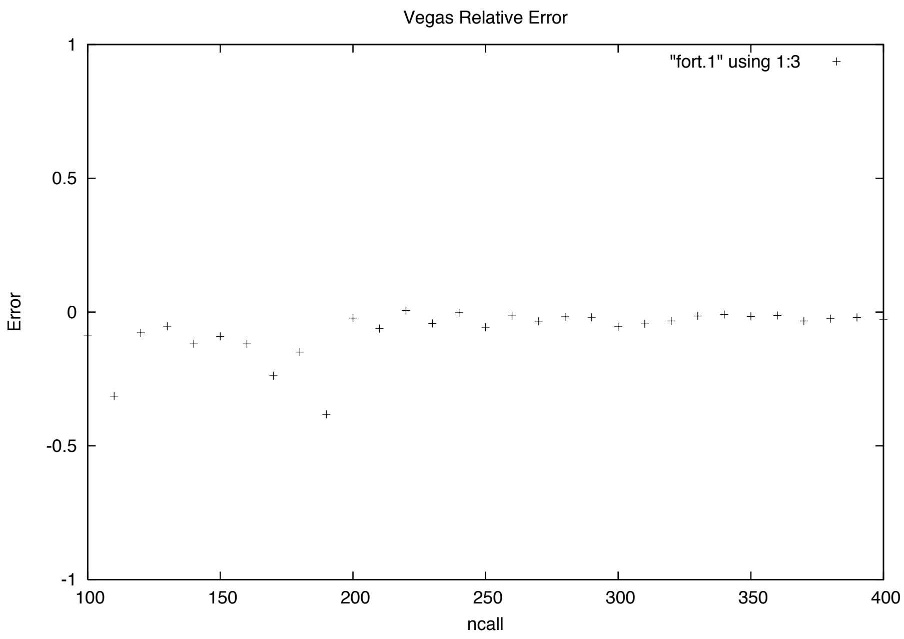


vegasEx.f and vegas.f, I made no change of vegas.f and some necessary changes to vegasEx.f. This problem was done in FORTRAN77 to be compatible with vegas.f.

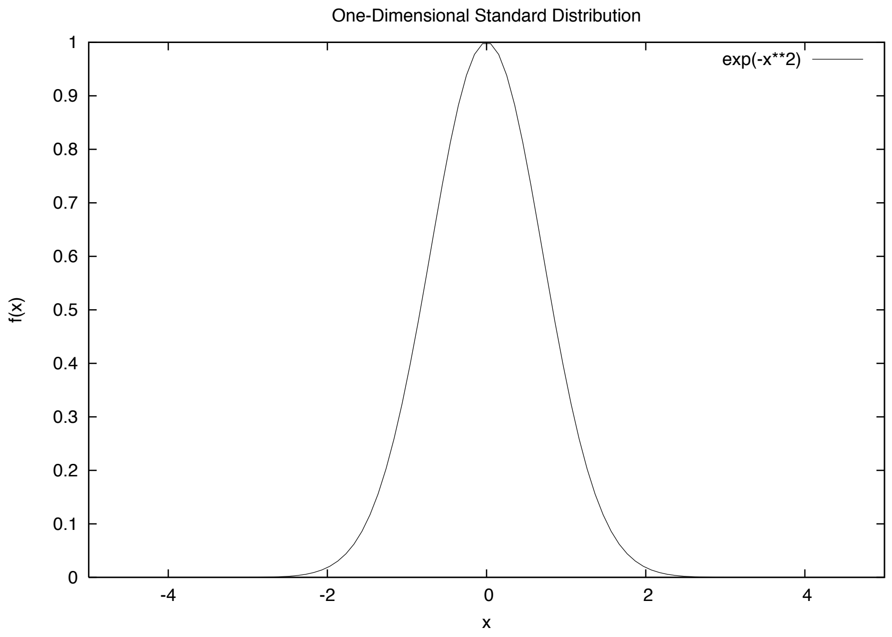
```
The code:
      program vegasIntegral
      implicit none
      real*8 reg(12),pi,result,analyresult,sdr,chi
      real*8 negInf, posInf,analysresult,error
      integer ndim,ncall,itmx,nprn,idum
      common /ranno/ idum
      external fcn
      pi = 4.d0*datan(1.d0)
      idum=9812371
      ndim=6
      itmx=10
      nprn=-1
      negInf=-2.0
      posInf=2.0
      reg(1) = negInf
```



```
reg(2) = negInf
reg(3) = negInf
reg(4) = negInf
reg(5) = negInf
reg(6) = negInf
reg(7) = posInf
reg(8) = posInf
reg(9) = posInf
reg(10)= posInf
reg(11)= posInf
reg(12)= posInf
analyresult=pi**3
do ncall=100, 400, 10
   call vegas(reg,ndim,fcn,0,ncall,itmx,nprn,result,sdr,chi)
   !reg: the integral limit
   !ndim: dimension
   !fcn: grand function
   !0: the initial value
   !ncall: number of throwing
   !itmax: number of interation
   !nprn: output requirement
   !result: the value of integral
   !sdr: standard deviation
   error=(result-analyresult)/analyresult
   write (1,*) ncall, result,error
end do
stop
end program vegasIntegral
INCLUDE 'vegas.f'
real*8 function fcn(p)
 implicit none
 real*8 p(6),pi
 pi = 4.d0*datan(1.d0)
 fcn = exp(-p(1)**2-p(2)**2-p(3)**2-p(4)**2-p(5)**2-p(6)**2)
 return
end
```



Instead of integral limit from negative infinity to positive limit, I set the limit to be -2 to 2. It seemed that VEGAS doesn't work well with large integral limit. While with [-2, 2] integral limit, VEGAS turns out to work quite well as shown in the figure. And as ncall increases, which is the number of thrown dots, the data looks better. The integrand function is six-dimensional standard normal distribution. Looking at the plot of one-dimensional standard normal distribution, it can be seen that the function decays very rapidly when exceeding



the range [-2, 2]. So it is reasonable for vegas to get a really small number when the range is expanded since most of the dots thrown in the decaying zone is difficult to get into the area under the rapidly decaying function.

After I discussed with my batch-mates, I saw that if I increase ncall to a much larger number, I can also expand the integral limit to [-10, 10].

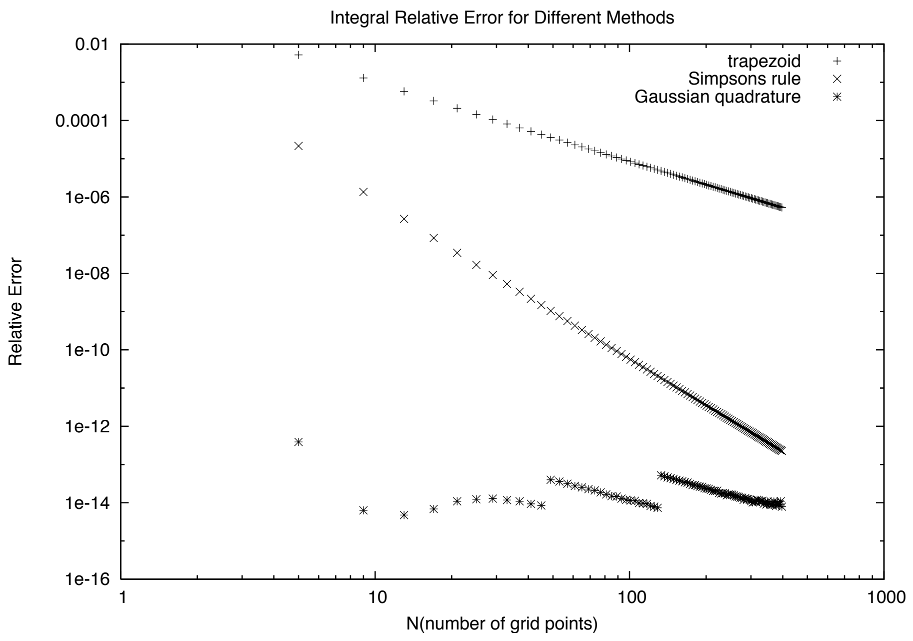
Problem 5

The main program called the subroutine GaussLegendre.f, I made some small changes to let it fit FORTRAN90 because I used the allocatable array in the main program.

```
program integral
IMPLICIT NONE
double precision:: a,b,h,analvalue,trapezoid,simpson,gauss,trapErr,simpErr,gausErr
double precision, allocatable, dimension(:) :: w,x
double precision:: fcn
integer:: N,i
a=0d0
b=1.d0
analValue=1.d0-exp(-1.d0)
do N=5, 400, 4
   ALLOCATE(w(N))
   ALLOCATE(x(N))
   h=(b-a)/(N-1)
!Trapezoid Method
  w(1)=h/2.d0
  w(N)=h/2.d0
  do i=2, N-1
     w(i)=h
  end do
  do i=1, N
     x(i)=a+(i-1)*h
  end do
  trapezoid=0.d0
  do i=1,N
     trapezoid=trapezoid+w(i)*fcn(x(i))
  end do
  trapErr=DABS(trapezoid-analValue)/analValue
!Simpson's rule
  w(1)=h/3.d0
  w(N)=h/3.d0
  do i=2, N-1, 2
     w(i)=4.d0*h/3.d0
  end do
  do i=3, N-1, 2
     w(i)=2.d0*h/3.d0
  end do
  simpson=0.d0
  do i=1,N
     simpson=simpson+w(i)*fcn(x(i))
  end do
  simpErr=DABS(simpson-analValue)/analValue
!Gaussioan quadrature methods
  CALL GaussLegendre(x,w,N,a,b)
  gauss=0.d0
  do i=1,N
     gauss=gauss+w(i)*fcn(x(i))
  end do
  gausErr=DABS(gauss-analValue)/analValue
  IF(ALLOCATED(w)) DEALLOCATE(w)
  IF(ALLOCATED(x)) DEALLOCATE(x)
  write(1,*) N,trapErr,simpErr,gausErr
end do
end program
INCLUDE 'GaussLegendre.f'

double precision function fcn(x)
IMPLICIT NONE
double precision:: x
```

```
fcn=exp(-x)
return
end function fcn
```



From the relative error figure, we can see that all of the three methods have a trend that the error decreases as N increases. While Gaussian quadrature turns out to be the most accurate one, the Simpson's rule takes the second position, the trapezoid method is the least accurate one, which is as expected because the integrand is smooth function. A strange phenomenon is that the Gaussian quadrature has non-continuing relative error. At some specific points the error would jump up and then decrease again to some points. I think it may have something to do with the property of Legendre polynomials.