Multiple Integrals and Probability: A Numerical Exploration

March 30, 2012

The homework questions are due in class on Monday 9 April

1 Probability

Definition 1.1. *$f : U \subset \mathbb{R}^2 \to \mathbb{R}_+$ is a* **probability density function** *if*

$$\int\int_U f \mathrm{d}A = 1$$

Definition 1.2. *If f is a probability density function which takes the set $U \subset \mathbb{R}^2$, then the probability of events in the set $W \subset U$ occurring is*

$$P(W) = \int\int_W f \mathrm{d}A.$$

Example 1.1. *The joint density for it to snow x inches tomorrow and for Kelly to win y dollar in the lottery tomorrow is given by*

$$f = \frac{c}{(1+x)(100+y)}$$

for

$$x, y \in [0, 100] \times [0, 100]$$

and $f = 0$ otherwise. Find c.

Definition 1.3. *Suppose X is a random variable with probability density function $f_1(x)$ and Y is a random variable with a probability density function $f_2(y)$. Then X and Y are* **independent random variables** *if their joint density function is*

$$f(x, y) = f_1(x) f_2(y).$$

Example 1.2. *The probability it will snow tomorrow and the probability Kelly will win the lottery tomorrow are independent random variables.*

Definition 1.4. *If $f(x, y)$ is a probability density function for the random variables X and Y, the* **X mean** *is*

$$\mu_1 = \bar{X} = \int\int x f \mathrm{d}A$$

and the **Y mean** *is*

$$\mu_2 = \bar{Y} = \int\int y f \mathrm{d}A.$$

Remark 1.1. *The X mean and the Y mean are the expected values of X and Y.*

Definition 1.5. *If $f(x, y)$ is a probability density function for the random variables X and Y, the* **X variance** *is*

$$\sigma_1^2 = \overline{(X - \bar{X})^2} = \int\int (x - \bar{X})^2 f \mathrm{d}A$$

and the **Y variance** *is*

$$\sigma_2^2 = \overline{(Y - \bar{Y})^2} = \int\int (y - \bar{Y})^2 f \mathrm{d}A.$$

Definition 1.6. *The standard deviation is defined to be the square root of the variance.*

Example 1.3. *Find an expression for the probability that it will snow more than 1.1 times the expected snowfall and also that Kelly will win more than 1.2 times the expected amount in the lottery.*

Homework question 1: A class is graded on a curve. It is assumed that the class is a representative sample of the population, the probability density function for the numerical score x is given by

$$f(x) = C \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

For simplicity we assume that x can take on the values $-\infty$ and ∞, though in actual fact the exam is scored from 0 to 100.

a) Determine C using results from your previous homework.

b) Suppose there are 240 students in the class and the mean and standard deviation for the class is not reported. As an enterprising student, you poll 60 of your fellow students (we shall suppose they are selected randomly). You find that the mean for these 60 students is 55% and the standard deviation is 10%. Use the Student's t distribution `http://en.wikipedia.org/wiki/Student%27s_t-distribution` to estimate the 90% confidence interval for the actual sample mean. Make a sketch of the t-distribution probability density function and shade the region which corresponds to the 90% confidence interval for the sample mean.[1]

Remark Fortunately, all the students are hard working, so the possibility of a negative score, although possible, is extremely low, and so we neglect it to make the above computation easier.

2 Riemann Integration

Recall that we can approximate integrals by Riemann sums. There are many integrals one cannot evaluate analytically, but for which a numerical answer is required. In this section, we shall explore a simple way of doing this on a computer. Suppose we want to find

$$I2d = \int_0^1 \int_0^4 x^2 + 2y^2 \mathrm{d}y\mathrm{d}x$$

If we do this analytically we find

$$I2d = 44.$$

Let us suppose we have forgotten how to integrate, and so we do this numerically. We can do so using the following Matlab code:

```
% A program to approximate an integral

clear all; format compact; format short;

nx=1000;     % number of points in x
xend=1;     % last discretization point
xstart=0;   % first discretization point
```

[1]The Student's t distribution is implemented in many numerical packages such as Maple, Mathematica, Matlab, R, Sage etc., so if you need to use to obtain numerical results, it is helpful to use on of these packages.

```
dx=(xend-xstart)/(nx-1);    % size of each x sub-interval

ny=4000;     % number of points in y
yend=4;     % last discretization point
ystart=0;   % first discretization point
dy=(yend-ystart)/(ny-1);    % size of each y sub-interval

% create vectors with points for x and y
for i=1:nx
    x(i)=xstart+(i-1)*dx;
end
for j=1:ny
    y(j)=ystart+(j-1)*dy;
end

% Approximate the integral by a sum
I2d=0;
for i=1:nx
    for j=1:ny
        I2d=I2d+(x(i)^2+2*y(j)^2)*dy*dx;
    end
end
% print out final answer
I2d
```

We can do something similar in three dimensions. Suppose we want to calculate

$$I3d = \int_0^1 \int_0^1 \int_0^4 x^2 + 2y^2 + 3z^2 \mathrm{d}z\mathrm{d}y\mathrm{d}x.$$

Analytically we find that

$$I3d = 68$$

Homework question 2:

a) Modify the Matlab code to perform the three dimensional integral.

b) Try and determine how the accuracy of either the two or three dimensional method varies as the number of subintervals is changed.

3 Monte Carlo Integration

[2] It is possible to extend the above integration schemes to higher and higher dimensional integrals. This can become computationally intensive and an alternate method of integration based on probability is often used. The method we will discuss is called the *Monte Carlo method*. The idea behind it is based on the concept of the *average value* of a function, which you learned in single-variable calculus. Recall that for a continuous function $f(x)$, the **average value** $\bar{f}$ of f over an interval $[a, b]$ is defined as

$$\bar{f} \;=\; \frac{1}{b-a}\int_a^b f(x)\,dx \;. \tag{1}$$

The quantity $b - a$ is the length of the interval $[a, b]$, which can be thought of as the "volume" of the interval. Applying the same reasoning to functions of two or three variables, we define the **average value** of $f(x, y)$ over a region R to be

$$\bar{f} \;=\; \frac{1}{A(R)}\iint_R f(x,y)\,dA \;, \tag{2}$$

where $A(R)$ is the area of the region R, and we define the **average value** of $f(x, y, z)$ over a solid S to be

$$\bar{f} \;=\; \frac{1}{V(S)}\iiint_S f(x,y,z)\,dV \;, \tag{3}$$

where $V(S)$ is the volume of the solid S. Thus, for example, we have

$$\iint_R f(x,y)\,dA \;=\; A(R)\bar{f} \;. \tag{4}$$

The average value of $f(x, y)$ over R can be thought of as representing the sum of all the values of f divided by the number of points in R. Unfortunately there are an infinite number (in fact, *uncountably* many) points in any region, i.e. they can not be listed in a discrete sequence. But what if we took a very large number N of *random* points in the region R (which can be generated by a computer) and then

[2] This section is taken from Chapter 3 of Vector Calculus by M. Corral which is available at `http://www.mecmath.net/` and where Sage programs for doing Monte Carlo integration can be found.

took the average of the values of f for those points, and used that average as the value of $\bar{f}$? This is exactly what the Monte Carlo method does. So in formula (4) the approximation we get is

$$\iint_R f(x,y)\,dA \;\approx\; A(R)\bar{f} \pm A(R)\sqrt{\frac{\overline{f^2} - (\bar{f})^2}{N}}\;, \tag{5}$$

where

$$\bar{f} \;=\; \frac{\sum_{i=1}^{N} f(x_i,y_i)}{N} \quad \text{and} \quad \overline{f^2} \;=\; \frac{\sum_{i=1}^{N} (f(x_i,y_i))^2}{N}\;, \tag{6}$$

with the sums taken over the N random points (x_1, y_1), ..., (x_N, y_N). The $\pm$ "error term" in formula (5) does not really provide hard bounds on the approximation. It represents a single *standard deviation* from the *expected* value of the integral. That is, it provides a *likely* bound on the error. Due to its use of random points, the Monte Carlo method is an example of a *probabilistic* method (as opposed to *deterministic* methods such as the Riemann sum approximation method, which use a specific formula for generating points).

For example, we can use formula (5) to approximate the volume V under the surface $z = x^2 + 2y^2$ over the rectangle $R = [0,1] \times [0,4]$. Recall that the actual volume is 44. Below is a Matlab code that calculates the volume using Monte Carlo integration

```
% A program to approximate an integral using the Monte Carlos method

% This program can be made much faster by using Matlab's matrix and vector
% operations, however to allow easy translation to other languages we have
% made it as simple as possible.

Numpoints=512;   % number of random points

I2d=0; % Initialize value
I2dsquare=0; % initial variance
for n=1:Numpoints
    % generate random number drawn from a uniform distribution on (0,1)
    x=rand(1);
    y=rand(1)*4;
    I2d=I2d+x^2+2*y^2;
```

```
    I2dsquare=I2dsquare+(x^2+2*y^2)^2;
end
% we sclae the integral by the total area and divide by the number of
% points used
I2d=I2d*4/Numpoints
% we also output an estimated error
I2dsquare=I2dsquare*4/Numpoints;
EstimError=4*sqrt( (I2d^2-I2dsquare)/Numpoints)
```

The results of running this program with various numbers of random points are shown below:

```
N = 16: 41.3026 +/- 30.9791
N = 256: 47.1855 +/- 9.0386
N = 4096: 43.4527 +/- 2.0280
N = 65536: 44.0026 +/- 0.5151
```

As you can see, the approximation is fairly good. As $N \to \infty$, it can be shown that the Monte Carlo approximation converges to the actual volume (on the order of $O(\sqrt{N})$, in computational complexity terminology).

In the above example the region R was a rectangle. To use the Monte Carlo method for a nonrectangular (bounded) region R, only a slight modification is needed. Pick a rectangle $\tilde{R}$ that encloses R, and generate random points in that rectangle as before. Then use those points in the calculation of $\bar{f}$ only if they are inside R. There is no need to calculate the area of R for formula (5) in this case, since the exclusion of points not inside R allows you to use the area of the rectangle $\tilde{R}$ instead, similar to before.

For instance, one can show that the volume under the surface $z = 1$ over the nonrectangular region $R = \{(x, y) : 0 \leq x^2 + y^2 \leq 1\}$ is π. Since the rectangle $\tilde{R} = [-1, 1] \times [-1, 1]$ contains R, we can use a similar program to the one we used, the largest change being a check to see if $y^2 + x^3 \leq 1$ for a random point (x, y) in $[-1, 1] \times [-1, 1]$. A Matlab code listing which demonstrates this is below:

```
% This program can be made much faster by using Matlab's matrix and vector
% operations, however to allow easy translation to other languages we have
% made it as simple as possible.

Numpoints=65536;   % number of random points

I2d=0; % Initialize value
```

```
I2dsquare=0; % initial variance
for n=1:Numpoints
    % generate random number drawn from a uniform distribution on (0,1) and
    % scale this to (-1,1)
    x=2*rand(1)-1;
    y=2*rand(1) -1;
    if ((x^2+y^2) <1)
        I2d=I2d+1;
        I2dsquare=I2dsquare+1;
    end
end
% we scale the integral by the total area and divide by the number of
% points used
I2d=I2d*4/Numpoints
% we also output an estimated error
I2dsquare=I2dsquare*4/Numpoints;
EstimError=4*sqrt( (I2d^2-I2dsquare)/Numpoints)
```

The results of running the program with various numbers of random points are shown below:

```
N = 16: 3.5000 +/- 2.9580
N = 256: 3.2031 +/- 0.6641
N = 4096: 3.1689 +/- 0.1639
N = 65536: 3.1493 +/- 0.0407
```

To use the Monte Carlo method to evaluate triple integrals, you will need to generate random triples (x, y, z) in a parallelepiped, instead of random pairs (x, y) in a rectangle, and use the volume of the parallelepiped instead of the area of a rectangle in formula (5). For a more detailed discussion of numerical integration methods, please take a further course in mathematics such as Math 371, Math 471 or Math 472.

Homework Question 3

a) Write a program that uses the Monte Carlo method to approximate the double integral $\iint_R e^{xy}\, dA$, where $R = [0, 1] \times [0, 1]$. Show the program output for $N = 10$, 100, 1000, 10000, 100000 and 1000000 random points.

b) Write a program that uses the Monte Carlo method to approximate the triple integral $\iiint_S e^{xyz}\,dV$, where $S = [0,1] \times [0,1] \times [0,1]$. Show the program output for $N = 10$, 100, 1000, 10000, 100000 and 1000000 random points.

c) Use the Monte Carlo method to approximate the volume of a sphere of radius 1.

4 Parallel Monte Carlo Integration

As you may have noticed, the algorithms are simple, but can require very many grid points to become accurate. It is therefore useful to run these algorithms on a parallel computer. We will demonstrate a parallel monte Carlo calculation of π. Before we can do this, we need to learn how to use a parallel computer. We shall use Trestles located at the San Diego supercomputing center. Information on this computer is available at: `http://www.sdsc.edu/us/resources/trestles/`

4.1 MPI

Before we can do Monte Carlo integration, we need to learn a little about parallel programming. A copy of the current standard MPI standard can be found at `http://www.mpi-forum.org/`. It allows for parallelization of Fortran, C and C++ programs. There are newer parallel programming languages such as Co-Array Fortran (CAF) and Unified Parallel C (UPC) which allow the programmer to view memory as a single addressable space even on a distributed memory machine. However, computer hardware limitations imply that most of the programming concepts used when writing MPI programs will be required to write programs in CAF and UPC. Compiler technology for these languages is also not as well developed as compiler technology for older languages such as Fortran and C, so at the present time, Fortran and C dominate high performance computing. An introduction to the essential concepts required for writing and using MPI programs can be found at `http://www.shodor.org/refdesk/Resources/Tutorials/`. More information on MPI can be found in Gropp, Lusk and Skjellum[3], Gropp, Lusk and Thakur[4] and at `https://computing.llnl.gov/tutorials/mpi/`. There are many resources available online, however once the basic concepts have been mastered, what is most

[3] Using MPI MIT Press(1999)

[4] Using MPI 2 MIT Press (1999)

useful is an index of MPI commands, usually a search engine will give you sources of listings, however we have found the following sites useful:

- `http://publib.boulder.ibm.com/infocenter/zos/v1r13/index.jsp?topic=%2Fcom.ibm.zos.r13.fomp200%2Fipezps00172.htm`
- `http://www.open-mpi.org/doc/v1.4/`

Homework Question 4

a) What does MPI stand for?

b) Please read the tutorials at `http://www.shodor.org/refdesk/Resources/Tutorials/BasicMPI/` and at `https://computing.llnl.gov/tutorials/mpi/`, then explain what the following commands do:

- `USE mpi` or `INCLUDE 'mpif.h'`
- `MPI_INIT`
- `MPI_COMM_SIZE`
- `MPI_COMM_RANK`
- `MPI_FINALIZE`

c) What is the version number of the current MPI standard?

d) Try to understand the Hello World program in listing 7. Run the program in listing 7 on 32 and 64 MPI processes[5]. Put the output of each run in your solutions, the output will be in a file of the form
`job_output`
An example makefile to compile this on Trestles is in listing 8. An example submission script is in listing 9. On Trestles, there is a maximum of 32 cores per node, so if more than 32 MPI processes are required, one needs to change the number of nodes as well. The total number of cores required is equal to the number of nodes multiplied by the number of processes per node. To change the number of MPI processes that the program will run on from 32 to 64, change
`nodes=1:ppn=32`
to

[5]One can run this program on many more than 64 processes, however, the output becomes quite excessive

`nodes=2:ppn=32`
and also change the submission script from
`mpirun_rsh -np 32 -hostfile $PBS_NODEFILE helloworld`
to
`mpirun_rsh -np 64 -hostfile $PBS_NODEFILE helloworld`.

```
!-----------------------------------------------------------------------
!
!
! PURPOSE
!
! This program uses MPI to print hello world from all available
! processes
!
! .. Parameters ..
!
! .. Scalars ..
!  myid       = process id
!  numprocs     = total number of MPI processes
!  ierr       = error code
!
! .. Arrays ..
!
! .. Vectors ..
!
! REFERENCES
! http:// en.wikipedia.org/wiki/OpenMP
!
! ACKNOWLEDGEMENTS
! The program below was modified from one available at the internet
! address in the references. This internet address was last checked
! on 30 December 2011
!
! ACCURACY
!
! ERROR INDICATORS AND WARNINGS
!
! FURTHER COMMENTS
!
!-----------------------------------------------------------------------
! External routines required
!
! External libraries required
! MPI library

PROGRAM hello90
USE MPI
INTEGER(kind=4) :: myid, numprocs, ierr

CALL MPI_INIT(ierr)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)

PRINT*, 'Hello World from process', myid
CALL MPI_BARRIER(MPI_COMM_WORLD,ierr)
IF ( myid == 0 ) THEN
  PRINT*, 'There are ', numprocs, ' MPI processes'
END IF
CALL MPI_FINALIZE(ierr)
END PROGRAM
```

Listing 1: A Fortran program which demonstrates parallelizm using MPI.

```
#define the complier
COMPILER = mpif90
# compilation settings, optimization, precision, parallelization
```

```
  FLAGS = -O0

# libraries
LIBS =
# source list for main program
SOURCES =  helloworld.f90

test: $(SOURCES)
    ${COMPILER} -o helloworld $(FLAGS) $(SOURCES)

clean:
  rm *.o

clobber:
  rm helloworld
```

Listing 2: An example makefile for compiling the helloworld program in listing 7.

```
#!/bin/bash
# the queue to be used.
#PBS -q normal
# specify your project allocation
#PBS -A mia122
# number of nodes and number of processors per node requested
#PBS -l nodes=1:ppn=32
# requested Wall-clock time.
#PBS -l walltime=00:05:00
# name of the standard out file to be "output-file".
#PBS -o job_output
# name of the job
#PBS -N MPI_Hello
# Email address to send a notification to, change "youremail" appropriately
#PBS -M youremail@umich.edu
# send a notification for job abort, begin and end
#PBS -m abe
#PBS -V
cd $PBS_O_WORKDIR #change to the working directory
mpirun_rsh -np 32 -hostfile $PBS_NODEFILE helloworld
```

Listing 3: An example submission script for use on Trestles.

We now examine a Fortran program for calculating π. These programs are taken from `http://chpc.wustl.edu/mpi-fortran.html`, where further explanation can be found. The original source of these programs appears to be Using MPI by Gropp, Lusk and Skjellum.

Serial

```
!-----------------------------------------------------------------------
!
!
! PURPOSE
!
! This program use a monte carlo method to calculate pi
!
! .. Parameters ..
!  npts       = total number of Monte Carlo points
!  xmin       = lower bound for integration region
!  xmax             = upper bound for integration region
! .. Scalars ..
!  i                = loop counter
!  f        = average value from summation
!  sum              = total sum
```

```
!  randnum          = random number generated from (0,1) uniform
!                     distribution
!  x                = current Monte Carlo location
! .. Arrays ..
!
! .. Vectors ..
!
! REFERENCES
! http://chpc.wustl.edu/mpi-fortran.html
! Gropp, Lusk and Skjellum, "Using MPI" MIT press (1999)
!
! ACKNOWLEDGEMENTS
! The program below was modified from one available at the internet
! address in the references. This internet address was last checked
! on 30 March 2012
!
! ACCURACY
!
! ERROR INDICATORS AND WARNINGS
!
! FURTHER COMMENTS
!
!-----------------------------------------------------------------------
! External routines required
!
! External libraries required
! None
PROGRAM monte_carlo
  IMPLICIT NONE

  INTEGER(kind=8), PARAMETER       :: npts = 1e10
  REAL(kind=8), PARAMETER      :: xmin=0.0d0,xmax=1.0d0
  INTEGER(kind=8)          :: i
  REAL(kind=8)             :: f,sum, randnum,x

  DO i=1,npts
    CALL random_number(randnum)
    x = (xmax-xmin)*randnum + xmin
    sum = sum + 4.0d0/(1.0d0 + x**2)
  END DO
  f = sum/npts
  PRINT*,'PI calculated with ',npts,' points = ',f

  STOP
  END
```

Listing 4: A Fortran program which demonstrates parallelizm using MPI.

```
#define the complier
COMPILER = mpif90
# compilation settings, optimization, precision, parallelization
  FLAGS = -O0

# libraries
LIBS =
# source list for main program
SOURCES =  montecarloserial.f90

test: $(SOURCES)
    ${COMPILER} -o montecarloserial $(FLAGS) $(SOURCES)

clean:
  rm *.o

clobber:
  rm  montecarloserial
```

Listing 5: An example makefile for compiling the helloworld program in listing 7.

```
#!/bin/bash
# the queue to be used.
#PBS -q shared
# specify your project allocation
#PBS -A mia122
# number of nodes and number of processors per node requested
```

```
#PBS -l nodes=1:ppn=1
# requested Wall-clock time.
#PBS -l walltime=00:05:00
# name of the standard out file to be "output-file".
#PBS -o job_output
# name of the job
#PBS -N MCserial
# Email address to send a notification to, change "youremail" appropriately
#PBS -M youremail@umich.edu
# send a notification for job abort, begin and end
#PBS -m abe
#PBS -V
cd $PBS_O_WORKDIR #change to the working directory
mpirun_rsh -np 1 -hostfile $PBS_NODEFILE  montecarloserial
```

Listing 6: An example submission script for use on Trestles.

Parallel

```
!-----------------------------------------------------------------------
!
!
! PURPOSE
!
! This program uses MPI to do a parallel monte carlo calculation of pi
!
! .. Parameters ..
!  npts        = total number of Monte Carlo points
!  xmin        = lower bound for integration region
!  xmax             = upper bound for integration region
! .. Scalars ..
!  mynpts      = this processes number of Monte Carlo points
!  myid        = process id
!  nprocs      = total number of MPI processes
!  ierr        = error code
!  i                = loop counter
!  f         = average value from summation
!  sum              = total sum
!  mysum            = sum on this process
!  randnum          = random number generated from (0,1) uniform
!                     distribution
!  x                = current Monte Carlo location
!  start       = simulation start time
!  finish      = simulation end time
! .. Arrays ..
!
! .. Vectors ..
!
! REFERENCES
! http://chpc.wustl.edu/mpi-fortran.html
! Gropp, Lusk and Skjellum, "Using MPI" MIT press (1999)
!
! ACKNOWLEDGEMENTS
! The program below was modified from one available at the internet
! address in the references. This internet address was last checked
! on 30 March 2012
!
! ACCURACY
!
! ERROR INDICATORS AND WARNINGS
!
! FURTHER COMMENTS
!
!-----------------------------------------------------------------------
! External routines required
!
! External libraries required
! MPI library
  PROGRAM monte_carlo_mpi
  USE MPI
  IMPLICIT NONE
```

```
    INTEGER(kind=8), PARAMETER  :: npts = 1e10
    REAL(kind=8), PARAMETER    :: xmin=0.0d0,xmax=1.0d0
    INTEGER(kind=8)         :: mynpts
    INTEGER(kind=4)            :: ierr, myid, nprocs
    INTEGER(kind=8)            :: i
    REAL(kind=8)             :: f,sum,mysum,randnum
    REAL(kind=8)             :: x, start, finish

    ! Initialize MPI
    CALL MPI_INIT(ierr)
    CALL MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)
    CALL MPI_COMM_SIZE(MPI_COMM_WORLD, nprocs, ierr)
    start=MPI_WTIME()

  ! Calculate the number of points each MPI process needs to generate
    IF (myid .eq. 0) THEN
      mynpts = npts - (nprocs-1)*(npts/nprocs)
    ELSE
      mynpts = npts/nprocs
    ENDIF

    ! set initial sum to zero
    mysum = 0.0d0
  ! use loop on local process to generate portion of Monte Carlo integral
    DO i=1,mynpts
      CALL random_number(randnum)
      x = (xmax-xmin)*randnum + xmin
      mysum = mysum + 4.0d0/(1.0d0 + x**2)
    ENDDO

  ! Do a reduction and sum the results from all processes
    CALL MPI_REDUCE(mysum,sum,1,MPI_DOUBLE_PRECISION,MPI_SUM,&
          0,MPI_COMM_WORLD,ierr)
    finish=MPI_WTIME()

    ! Get one process to output the result and running time
    IF (myid .eq. 0) THEN
         f = sum/npts
         PRINT*,'PI calculated with ',npts,' points = ',f
         PRINT*,'Program took ', finish-start, ' for Time stepping'
    ENDIF

    CALL MPI_FINALIZE(ierr)

    STOP
    END PROGRAM
```

Listing 7: A Fortran program which demonstrates parallelizm using MPI.

```
#define the complier
COMPILER = mpif90
# compilation settings, optimization, precision, parallelization
  FLAGS = -O0

# libraries
LIBS =
# source list for main program
SOURCES =  montecarloparallel.f90

test: $(SOURCES)
    ${COMPILER} -o montecarloparallel $(FLAGS) $(SOURCES)

clean:
  rm *.o

clobber:
  rm  montecarloparallel
```

Listing 8: An example makefile for compiling the helloworld program in listing 7.

```
#!/bin/bash
# the queue to be used.
#PBS -q normal
# specify your project allocation
```

```
#PBS -A mia122
# number of nodes and number of processors per node requested
#PBS -l nodes=1:ppn=32
# requested Wall-clock time.
#PBS -l walltime=00:05:00
# name of the standard out file to be "output-file".
#PBS -o job_output
# name of the job, you may want to change this so it is unique to you
#PBS -N MPI_MCPARALLEL
# Email address to send a notification to, change "youremail" appropriately
#PBS -M youremail@umich.edu
# send a notification for job abort, begin and end
#PBS -m abe
#PBS -V

# change to the job submission directory
cd $PBS_O_WORKDIR
# Run the job
mpirun_rsh -np 32 -hostfile $PBS_NODEFILE  montecarloparallel
```

Listing 9: An example submission script for use on Trestles.

Homework Question 5

a) Explain why using Monte Carlo to evaluate

$$\int_0^1 \frac{1}{1+x^2}\mathrm{d}x$$

allows you to find π and, in your own words, explain what the serial and parallel programs do.

b) Find the time it takes to run the Parallel Monte Carlo program on 32, 64, 128, 256 and 512 cores.

Bonus Questions

a) Use a parallel Monte Carlo integration program to evaluate

$$\iint x^2 + y^6 + \exp(xy)\cos(y\exp(x))\mathrm{d}A$$

over the unit circle.

b) Use a parallel Monte Carlo integration program to approximate the volume of the ellipsoid $\frac{x^2}{9} + \frac{y^2}{4} + \frac{z^2}{1} = 1$.

c) Write parallel programs to find the volume of the 4 dimensional sphere

$$1 \geq \sum_{i=1}^{4} x_i^2.$$

Try both Monte Carlo and Riemann sum techniques.