COMP114 - Assessment 1
Semester 2 – 2011
Monte Carlo Estimates

Assessment Information

Assessment Number	1
Contribution to Overall Mark	10%
Submission deadline	**Friday 4th March, 16.00**

Relevant Learning Outcomes for this Assessment

1	Awareness of different areas of CS for which experimental methods are relevant.
2	Ability to distinguish use of experiments for testing and evaluation.
5	Ability objectively to assess, analyse, and present experimental results.

Assessment Description – Overview

One of the earliest uses of experimental methods in computing contexts was to obtain estimates of the numerical value of quantities that were hard to compute by direct techniques. One widely used application of this type being to estimate the area of a given region. The basic idea applied would be to "guess" random points in a square (of area 1), count the number of these that fell inside the region whose area was to be determined, and then use as an estimate the number of guesses "inside" the region divided by the *total* number of guesses. The purpose of this assignment is to examine the effectiveness of this approach in some selected example cases.

Assessment Description – Details

Suppose we want to estimate the area of the shaded region in Figure 1 (assuming we do not know the formula for the area of a a circle of radius r to be $\pi \times r^2$). We could do this by choosing random $\langle x, y\rangle$ coordinates with $0 \leq x, y \leq 1$

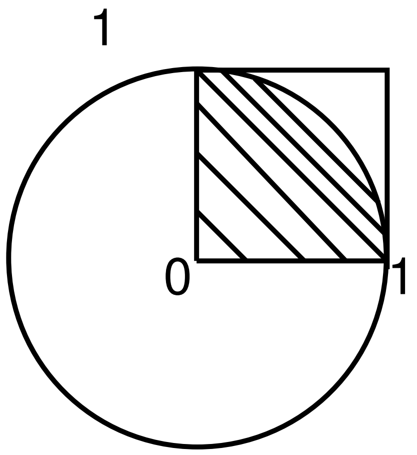

Figure 1: Simple numerical estimation example.

and counting the number of times the chosen co-ordinate fell within the shaded region, i.e. the number of times when y was at most $\sqrt{1 - x^2}$.

The informal idea underlying so-called *Monte Carlo* methods is the following: if we throw darts randomly at the square area so that a dart has an equal chance of landing anywhere within the square then the actual number that fall within the shaded region will (as a proportion of the total number thrown) be roughly the area of the shaded section: in principle, the more darts thrown the better this estimate will be as an approximation of the actual area we are trying to calculate.

The class **RealFunction** which may be found on the module web pages in `http://www.csc.liv.ac.uk/~ped/COMP114/Assessments11/RealFunction.java` describes a general class of real-valued single argument functions, of the form

$$f(x) = (\sum_{i=0}^{k} a_i \; x^i)^q$$

Here $a_0, \ldots, a_k$, and q are (positive or negative) *reals* (i.e. **double** values), k is a positive integer (**int**) value.

The class specification has:

public class RealFunction

Class Fields

Constructors and Methods

The class has a single constructor,

RealFunction(**double**[] a, **int** k, **double** q)

Type	Description
private double[] coefficients	The values $\{a_0, a_1, \ldots, a_k\}$
private int degree	The value k
private double power	The valkue q

and single instance method

public double ComputeValue(**double** x)

The statement F = **new** RealFunction(a,k,q) will create an instance (F) of the RealFunction class (given an array of $k+1$ double values, an integer k, and a double value q) with the method $F.ComputeValue(x)$ returning the **double** value $(\sum_{i=0}^{k}\ a[i] * x^i)^q$.

Assessment Requirements

MonteCarlo Class Implementation

Using the *RealFunction* class just described, write a Java class – *MonteCarlo* – which does the following:

1. Reads the values below as input:
   a. A **double** value – *mult*.
   b. A **double** value – q.
   c. An **int** value – k (which *must be* ≥ 0)
   d. A collection of $k+1$ **double** values, into a **double**[] a.
   e. An **int** value – $NumberofTests$
2. Creates an instance, F, of the *RealFunction* class

$$F = \textbf{new} RealFunction(a, k+1, q)$$

3. Initiates an **int** counter *hits* to 0.
4. Performs the following calculations $NumberofTests$ times:
   a. Generate a random **double** x with $0 \leq x < 1$.
   b. Generate a random **double** y with $0 \leq y < 1$.
   c. If $y \leq F.ComputeValue(x)$ then *hits* is incremented by 1.
   d. Outputs the **double** value $mult*(double)hits/(double)(NumberofTests)$

Experimental evaluation

a. Using your implementation of the class MonteCarlo described above, evaluate its performance with the following input data:

   1. $mult = 4.0$; $k = 2$; $q = 0.5$; $a[0] = 1.0$; $a[1] = 0.0$; $a[2] = -1.0$ with $NumberofTests = 100;\ 1000;\ 10,000;\ 100,000;\ 1,000,000.$
   2. $mult = 1.0$; $k = 1$; $q = -1.0$ $a[0] = 1.0$; $a[1] = 1.0$ with $NumberofTests = 100;\ 1000;\ 10,000;\ 100,000;\ 1,000,000.$

b. The data described in experiment (1) corresponds to calculating the area of a circle with radius 1, i.e. obtaining an approximation to the value π.

   1. Recalling that $\pi = 3.1415926\ldots$... what is noticeable about the accuracy of the approximations compared to the number of tests used to obtain them?
   2. Another approximation to π is given by computing

   $$4.0 * \sum_{i=0}^{NumberofTests} \frac{(-1)^i}{2 * i + 1}$$

   How well do the approximations produced by the MonteCarlo approach compare to the value given by this approximation? (in order to answer this you will have implement a short program which computes the summmation given above).

   3. The value estimated in (2) can also be approximated using

   $$\sum_{i=0}^{NumberofTests} \frac{(-1)^i}{i + 1}$$

   Is the behaviour of the MonteCarlo estimate in (2) similar to that of (1) when compared with this series, whose value is approximately $0.69314718056\ldots$?

What should be Submitted

a. The *source code* of your implementation of the class $MonteCarlo$.

b. Source code of the programs for your implementations in b2 and b3 (note that code for b3 requires only very minor changes to the code you wrote for b2) of the experimental evaluation.

c. Tables of output organised as *five* columns in the form

d. A *brief* (couple of sentences) summary giving your observations in respect of (b1), (b2) and (b3) above. In particular, given the behaviour that you ought to observe from running the experiments in (b2) and (b3) do you think there are cases where MonteCarlo approaches would be useful?

e. A completed and signed *Declaration On Plagiarism and Collusion Form.*

Number of Tests	Expt 1 Value	Exp 2 value	Value in (b2)	Value in (b3)
100				
1000				
10,000				
100,000				
1,000,000				

How the work should be submitted

Items (a–e) should handed in to the *Student Office*. A cover sheet should indicate all of the following information:

a. The Assessment *number*.

b. Your **name** and University **e-mail address**

c. Your lab group.

d. The name of the *demonstrator/tutor* responsible for this group.

e. Your degree programme, e.g. G400 Computer Science, G500 Computer Information Systems.