Monte Carlo Integration

Project #12

Computational Physics Laboratory

[due April 10, 2008]

Complete the coding of the MonteCarloIntegrator class given below. A copy of this base source code is located at *~crede/MonteCarlo/MCIntegrator.cc*. This class should implement both Monte Carlo Sampling and Monte Carlo Mean Value Methods. Develop the class definition and main program source with plenty of documenting comments including doxygen-style comment blocks (-> Thursday's class). Use the given main program source to compile and build a program. Run your program to numerically calculate the value of pi. Use your program to calculate pi varying the number of Monte Carlo points as: 10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8. Use *doxygen* to generate html documentation for the project (-> Thursday's class).

Make a copy of your MonteCarloIntegrator class to create a new class definition which implement multi-dimensional integration via Monte Carlo mean value techniques. Evaluate the below integral using for 10, 100, 1000, 10,000, ..., and 1,000,0000 random Monte Carlo points. Show that the error in the integral decreases like $1/\sqrt{N}$ (N is the number of MC points) by calculating the integral for each of the values of N above 10 times and plotting the different results.

$$\int_0^{\pi}\int_0^{\pi} \sin(2*x*y)\,dx\,dy$$

Record your work and report your results on your computational physics website. Create a html page for the exercise. Create a link from your main project web page to this html page. This html page should include the following heading information: exercise title, exercise number, your name, & today's date. The main content of this page should include the following:

- a short description of the exercise
- a table of numerical results for pi vs number of MC points.
- a plot showing that the error in the above integral decreases like $1/\sqrt{N}$
- a link to program source code files
- a link to the doxygen generated html documentation (Homework)
- images (not links) of your plots from root.

```
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <iostream>
using namespace std;



class MonteCarloIntegrator {
private:
  unsigned long  iNumberOfPoints;
  double iLeftLimit, iRightLimit, iYLimit;
  double iResult;
  double (*iIntegrandFunction) (double);

public:
  MonteCarloIntegrator(double aIntegrandFunction(double), double aLeftLimit,
              double aRightLimit, unsigned long aNumberOfPoints );
  double integrateByMeanValue();
   double integrateBySamples();
  unsigned long numberOfPoints() ;
  double leftLimit() ;
  double rightLimit() ;
  double yLimit() ;
  double result() ;
  void setNumberOfPoints( unsigned long aNumberOfPoints );
  void setLeftLimit( double aLeftLimit );
  void setRightLimit( double aRightLimit );
  void setYLimit( double aYLimit );
  void setResult( double aResult );
};
```

```
//////////////////////////////////// MAIN PROGRAM ////////////////////////////////////////

double halfCircle(double aX ){
  double radius = 1.0;
  return sqrt( ( radius )*( radius ) - aX * aX );
}


int main(){
  double leftLimit = -1.0, rightLimit = 1.0;
  unsigned long numberOfPoints = 10000000;

  MonteCarloIntegrator MCI( halfCircle, leftLimit, rightLimit, numberOfPoints  );


  cerr.precision(8);
  cerr << "Pi by Mean Value: \t" << 2.0 * MCI.integrateByMeanValue( ) << endl;
  cerr << "Pi by Sampling: \t" << 2.0 * MCI.integrateBySamples( ) << endl;
  cerr << "Pi from M_PI: \t\t" << M_PI << endl;
}
////////////////////////////////////////////// END OF MAIN ////////////////////////////////
```