Monte Carlo Integration

Science and Dice

by Tom Robey, Bob Robey and Nick Bennett

I can't believe that God plays dice with the universe.
- Albert Einstein

Supercomputing Challenge Kickoff 2009-2010
October 25-26, 2009

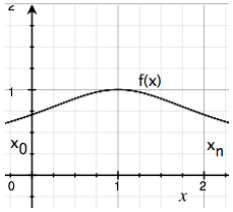
Monte Carlo Integration

- One dimensional integration (e.g. Area under a curve)
  - Newton-Cotes rules
  - Gauss-Legendre rules
  - Adaptive Integration
- Higher dimensional integration
  - Curse of dimensionality
  - Monte Carlo
  - Advanced Monte Carlo
  - Adaptive Sampling
  - Example

One-dimensional Integration

Find the area under a curve

$$I(f) = \int_{x_0}^{x_n} f(x)dx$$



We divide the x-axis into equal intervals of width, $h=(x_n-x_0)/n$. By approximating the area for each interval an overall approximation for $I(f)$ can be obtained

$$I_n(f) = \sum_{i=0}^{n-1} \sum_{j=1}^{N} w_j f(x_j)$$

The last summation represents **quadrature rules** which are covered next. There are many quadrature rules which have different weights, w_j, number of function evaluations, N, and locations, x_j. In the following, the dimension s=1.

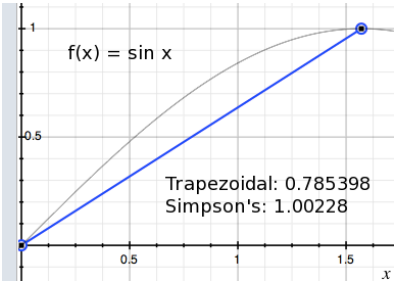
Convergence Rate

- The error of a method with a **convergence rate** p is "order p" and is written as $O(n^{-p})$
- If the error is $O(n^{-p})$ then doubling the number of intervals:
  - p=1: error is halved
  - p=2: error decreases by one-fourth
  - p=3: error decreases by one-sixth
- The error can be fit by a curve $y = Cn^{-p}$, where C is a constant
- For intervals of equal width, $O(n^{-p}) = O(h^p)$

Newton-Cotes Rules

Trapezoidal Rule (blue line):

$$I_i(f) = \left(\frac{x_{i+1}-x_i}{2}\right)[f(x_i)+f(x_{i+1})], i=0,\ldots,n-1$$



$O(h^2)$ which means cutting the interval in half cuts the error by four.

Simpson's Rule:

$$I_i(f) = \left(\frac{x_{i+1}-x_i}{6}\right)\left[f(x_i)+4f\left(\frac{x_i+x_{i+1}}{2}\right)+f(x_{i+1})\right], i=0,\ldots,n-1$$

$O(h^4)$ which means cutting the interval in half does what to the error?

Higher order methods (greater p) require a greater amount of smoothness. Smoothness is expressed as the number of derivatives.

Gauss-Legendre Rules

For 2 points on [-1,1]:

$$I_n(f) = f\left(-\frac{\sqrt{3}}{3}\right)+f\left(\frac{\sqrt{3}}{3}\right)$$

$O(h^4) = O(n^{-1/4})$

For 3 points on [-1,1]:

$$I_n(f) = \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right)+\frac{8}{9}f(0)+\frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right)$$

$O(h^6) = O(n^{-1/6})$

Convergence

- By **convergence** we mean that our approximation approaches the actual integral (area) in some sense.
  - *h*-convergence is when the maximum interval size *h*->0.
  - *p*-convergence is when the order *p*->∞. This requires the function to be smooth.

Adaptive Integration

In practice, for some intervals the integration is easy and other intervals are more difficult to get an accurate approximation. Why not just concentrate on the hard intervals?

- Compute the approximate area for each interval and an estimate of the error
- If the error estimate is higher than acceptable, divide the interval in half and go to the first step. Exit if the error estimate is acceptable.

Implementing this is a classic case of recursion.

Algorithms are based on quadrature rules but are specialized for adaptive integration (e.g. Romberg Integration)

Multi-dimensional Integration

A simple example of one-dimensional integration is finding the area under a curve. For two dimensions, a simple example is finding the volume under a surface. Higher dimensions are harder to visualize, but an example for three dimensions is finding the total volumes under a group of surfaces.

- **Curse of dimensionality**: A quadrature rule that is $O(n^{-p})$ in one-dimension requires n^s quadrants in s dimensions and is $O(n^{-p/s})$

Monte Carlo Methods

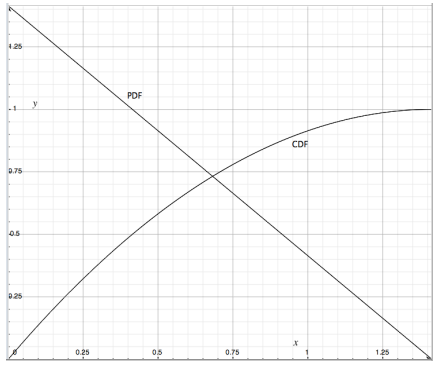
- Simulation using random numbers - most common application is integration

$$I_n(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}$$

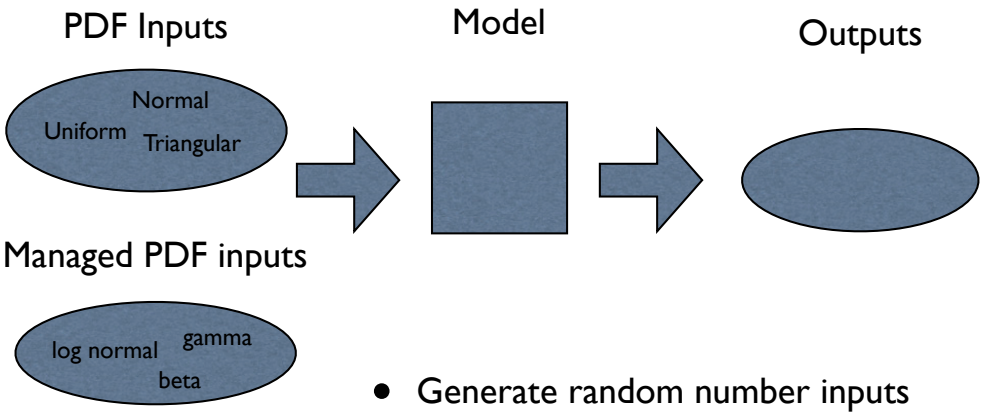
- Convergence uses probability and the **Law of Large Numbers**: $O(n^{-1/2})$
- Convergence rate does not require smoothness in function
- Monte Carlo is both simple and general

Probability Distribution Function

- Area under probability distribution function (PDF) is 1
- Cumulative distribution function (CDF) is the integral of the PDF. What is its maximum? In practice, monotonically increasing.



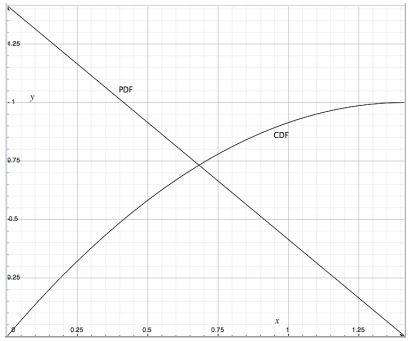
Monte Carlo Process



- Generate random number inputs
- Run model with inputs and get results
- Organize the results (hidden integral)

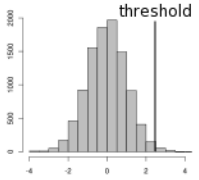
Sampling Distributions

- Start with random number generator for a uniform distribution [0,1]. How do you generate other random numbers?
- Using the CDF, enter on the y-axis which ranges from [0,1] with a random number from the uniform distribution. Go horizontally to the CDF and then go down to the x-axis. This is a random number from that PDF.



Threshold Example (Uncertainty Quantification)

- Simple and common way to organize results
- Set a threshold and count all the results that exceed that threshold and divide by total number of runs
- Equivalent to integration where function is 1 when the result is over the threshold and 0 otherwise
- The average and standard deviation of the results are also approximation of integrals

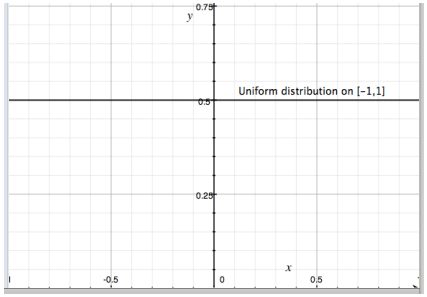


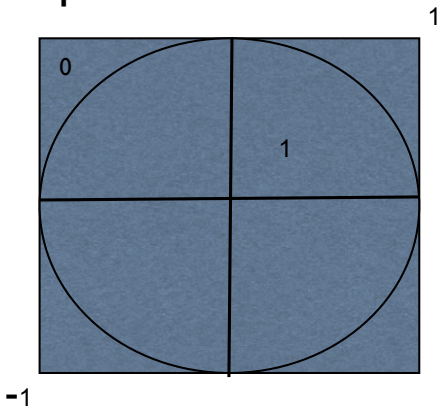
Input and Output Studies

- Sensitivity analysis - look at results that exceeded your threshold. Were some of the inputs from the tails?
- Managed input parameters - were some of the input variables due to management policies? What happens if the management policy is changed?
- Threshold variation - what happens if the threshold is changed?

π Example

The two input variables are uniform distributions on [-1,1]. The output when the sum of the squares of the inputs are less than 1 is 1, and is otherwise 0.





$$I_n(f) = \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i)}{0.5 \cdot 0.5}$$

Advanced Monte Carlo Methods

- Is there a way to combine the advantage of Monte Carlo methods and the higher convergence rates of one-dimensional integration methods?
  - Use quasi-random numbers to spread out the random numbers throughout the domains.
  - Divide up domains to spread out random numbers

Quasi-Monte Carlo

- Uses numbers generated from a special sequence (e.g. Halton sequence).
- Rate of convergence is

$$O\left(\frac{(\log n)^s}{n}\right)$$

Stratified Monte Carlo Sampling

- Divide the domain into several non-overlapping regions
- $O(n^{-(1+2/s)/2})$ for functions with a bounded derivative and $O(n^{-(1+1/s)/2})$ for piecewise continuous functions.
- Stratified sampling can increase the convergence rate in low-dimensional domains but has little effect in high-dimensional domains.

Adaptive (Sequential) Sampling

- Choose random numbers where the function is changing rapidly
- Importance sampling: choose random numbers where the function is larger

Monte Carlo Example

Integrate the three dimensional integral:

$$\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \frac{1}{(2\pi)^3} \frac{1}{(1-\cos(x)\cdot\cos(y)\cdot\cos(z))}\, dx\, dy\, dz$$

The exact answer is 1.393203929685676859...

- Use regular Monte Carlo and Stratified Sampling. The function is piecewise continuous so Stratified Sampling should converge at $O(n^{-2/3})$ while regular Monte Carlo converges at $O(n^{-1/2})$.
- Plot the error on a log-log graph for various numbers of samples from 100 to 1e9.

MonteCarlo.c

```c
#include <stdlib.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_monte.h>
#include <gsl/gsl_monte_plain.h>
#include <gsl/gsl_monte_miser.h>

/* Computation of the integral,

 I = int (dx dy dz)/(2pi)^3  1/(1-cos(x)cos(y)cos(z))

 over (-pi,-pi,-pi) to (+pi, +pi, +pi).  The exact answer
 is Gamma(1/4)^4/(4 pi^3).  This example is taken from
 C.Itzykson, J.M.Drouffe, "Statistical Field Theory -
 Volume 1", Section 1.1, p21, which cites the original
 paper M.L.Glasser, I.J.Zucker, Proc.Natl.Acad.Sci.USA 74
 1800 (1977) */

/* For simplicity we compute the integral over the region
 (0,0,0) -> (pi,pi,pi) and multiply by 8 */

double exact = 1.393203929685676859184246260325;

double
g (double *k, size_t dim, void *params)
{
     double A = 1.0 / (M_PI * M_PI * M_PI);
     return A / (1.0 - cos (k[0]) * cos (k[1]) * cos (k[2]));
}

void
display_results (double result, double error)
{
     printf ("result = % .6f\n", result);
     printf ("sigma  = % .6f\n", error);
     printf ("error  = % .6f = %.2g sigma\n", result - exact,
                  fabs (result - exact) / error);
}

int
main (void)
{
     int i;
     double res, err, avgerr;

     double xl[3] = { 0, 0, 0 };
     double xu[3] = { M_PI, M_PI, M_PI };

     const gsl_rng_type *T;
     gsl_rng *r;

     gsl_monte_function G = { &g, 3, 0 };

     size_t calls = 100;

     gsl_rng_env_setup ();

     T = gsl_rng_default;
     r = gsl_rng_alloc (T);

     printf("exact = % .6f\n", exact);
     printf("calls = %d\n\n", (int) calls);

     printf("Plain\n");
     {
            gsl_monte_plain_state *s = gsl_monte_plain_alloc (3);
            gsl_monte_plain_integrate (&G, xl, xu, 3, calls, r, s,
                                                    &res, &err);
            gsl_monte_plain_free (s);

            display_results (res, err);
     }

     printf("\nMiser\n");
     {
            gsl_monte_miser_state *s = gsl_monte_miser_alloc (3);
            gsl_monte_miser_integrate (&G, xl, xu, 3, calls, r, s,
                                                    &res, &err);
            gsl_monte_miser_free (s);

            display_results (res, err);
     }

     gsl_rng_free (r);

     return 0;
}
```

Running from the Command Line

Copy MonteCarlo.c to a directory. Then type the following:

gcc -o MonteCarlo MonteCarlo.c -Wall -I/opt/local/include -L/opt/local/lib -lgsl

To run the program

./MonteCarlo

Using Xcode IDE
(see Appendix for other platforms)

1. Open XCode (/Developer/Applications/ and drag Xcode.app to your Dashboard. Then open)
2. File/New Project...
3. In the "New Project" Assistant, expand the "Command Line Utility" group.
4. Select "Standard Tool"
5. Click "Next"
6. Give a project name (Montecarlo) and directory, then click "Finish".
7. Press Cmd-Shift-R to open the Console window. Output will appear there.

Project->Add to Project and add MonteCarlo.c. Delete main.c.

Edit Project->Edit Project Settings

Under Search Paths:
  Add to User Header Search Paths: /opt/local/include
  Add to Library Search Paths: /opt/local/lib

Under Linking add to Other Linker Flags: -lgsl

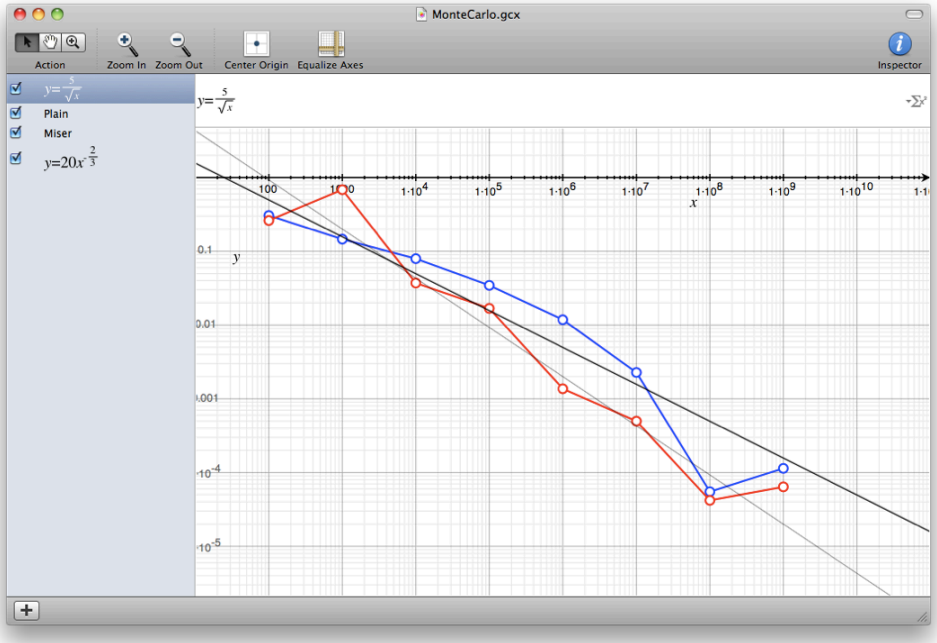
Set the Architecture to Native Architecture

Click the "Build and Go" toolbar button.

Graphing

Graph the function by going to Applications->Utilities->Grapher and select 2D plot log-log. Under Format->Axes & Grids set the abscissa to 10^1 to 10^{11}.

Click the "+" and add two New Point Sets to plot the errors versus the number of calls. Add the functions representing the order of convergence of the two methods.



The convergence lines are approximately fit by eye.

Appendices

Windows Setup

Install Wascana by running the Wascana installer
Install libraries and gnu software by running gnu installers for gsl-1.8
Add to system path by opening up control panel, search for system environment and adding to the end of the path
  ;C:\Program Files (x86)\GnuWin32\bin;C:\Program Files (x86)\Wascana\mingw\bin
When setting up the compiling in Eclipse
  Add to GCC C Compiler:Directories
    "C:\Program Files (x86)\GnuWin32\include"
  Add to GCC C Compiler:Symbols
    GSL_DLL
  Add to MinGW C Linker:Libraries
    Libraries
      gsl
      gslcblas
    Library search path
      "C:\Program Files (x86)\GnuWin32\lib"

Mac OS X Instructions

Install Xcode development tools. These should be on your OS disk or register and download at

http://developer.apple.com/technology/xcode.html

Install MacPorts (choose the version for your OS):

http://www.macports.org/install.php

Make the following directory and cd to it:

/opt/local/bin/portslocation/dports/gsl

Then install the Gnu Scientific Library

sudo port install gsl

Linux Setup (64 bit Ubuntu)

1) Install Eclipse by running the eclipse-cpp-galileo-SR1-linux-gtk-x86_64 installer.
2) Install Java by running the jdk-6u16-nb-6_7_1-linux-ml.sh installer
3) Install compilers and supporting tools with the package manager
• build-essential
4) Install libraries and gnu software with the package manager
• libgsl0ldbl and libgsl0-dev
5) When setting up the compiling in Eclipse
• Add to GCC C Linker:Libraries
gsl
gslcblas