Monte Carlo Method for Definite Integration

http://www.mathcs.emory.edu/ccs/ccs215/monte/node13.htm

Numerical integration techniques, such as the Trapezoidal rule and Simpson's rule, are commonly utilized for the purpose of approximating the value of a definite integral. However, these methods of approximation will guarantee a particular rate of convergence only when the integrand has continuous derivatives to a certain order. Before it can be applied, a numerical integration technique also requires a certain amount of knowledge about the function, such as the location of its domain, and where any discontinuities may exist. If a function fails to satisfy the requirements associated with a given numerical integration technique, or if it behaves in such a manner that one cannot beforehand discern a sufficient amount of information about the function in order to circumvent any problems that may arise with such an approach, then we can often apply a Monte Carlo method to obtain an approximation of the integral.

```
Clear["Global`*"];
```

```
ClearAll[x, y]
```

```
t1 = Array[x, {10}];
lims = Table[{i, 0, 1}, {i, t1}];
```

Knowing that the (##) represents the sequence of arguments supplied to a pure function and (## n) represents the sequence of arguments supplied to a pure function, starting with the nth argument (f[x, ##, y, ##] & [a, b, c, d]-> f[x, a, b, c, d, y, a, b, c, d], or f[##2] &[a, b, c, d] -> f[b, c, d])

```
NIntegrate[Exp[-t1.t1]/Sqrt[t1.t1], ##, Method → "AdaptiveMonteCarlo"] &@@
 lims
```

```
0.0347335
```

The number of samples used to evaluate $\int_0^1\int_0^1 \frac{1}{\sqrt{x+y}}\,dy\,dx$ for different relative tolerances:

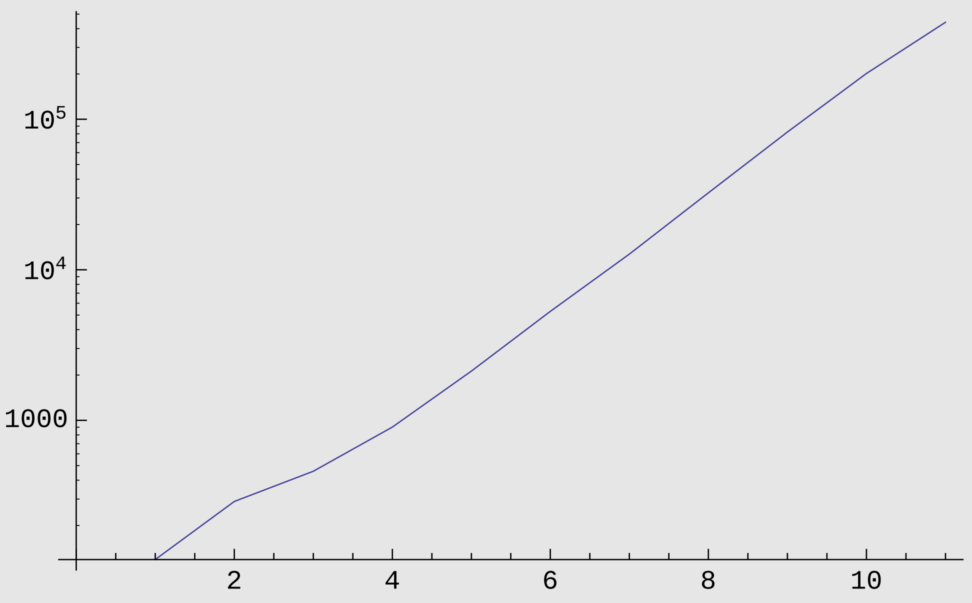
```
sampleCount[pg_] :=
  Module[{k = 0},
   NIntegrate[1 / Sqrt[x + y], {x, 0, 1}, {y, 0, 1}, PrecisionGoal → pg,
    EvaluationMonitor :→ k++];
   k
  ];
```

The number of samples needed typically increases exponentially with the `PrecisionGoal`:

```
Table[sampleCount[pg], {pg, 2, 12}]
```

```
{119, 289, 459, 901, 2125, 5287, 12716, 32436, 82348, 201518, 440963}
```

```
ListLogPlot[%, Joined → True]
```



It can be time-consuming to compute functions symbolically:

```
f[x_] := Nest[Sin[# + Sin[2 #]] &, x, 20]
```

```
NIntegrate[f[x], {x, 0, 1}] // Timing
```

```
{1.109, 0.947747}
```

Restricting the function definition avoids symbolic evaluation:

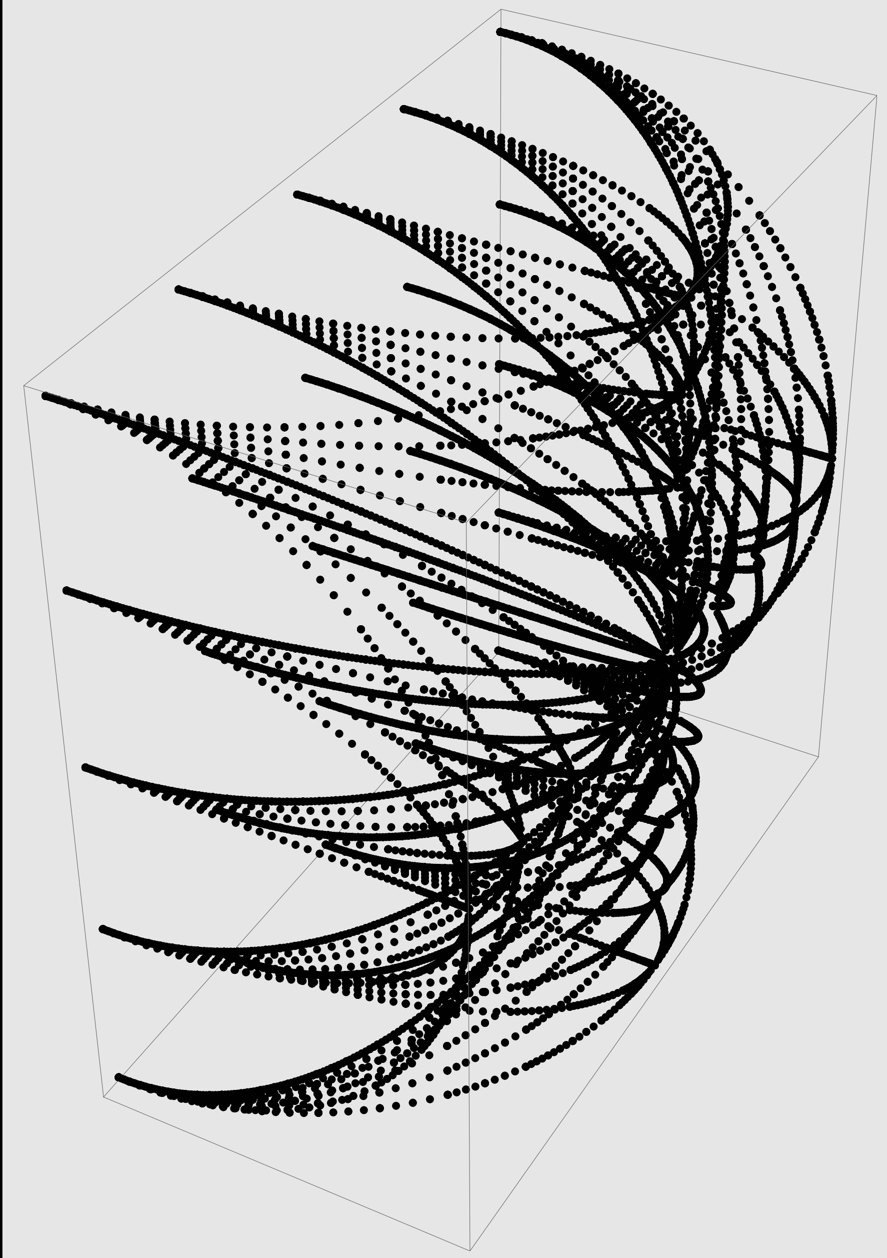
```
g[x_?NumericQ] := Nest[Sin[# + Sin[2 #]] &, x, 20]
```

```
NIntegrate[g[x], {x, 0, 1}] // Timing
```

```
{0., 0.947747}
```

```
points =
  Reap[NIntegrate[Boole[(x1)^2 + (x2 - 1)^2 < 1] + Boole[(x1)^2 + (x3 - 1)^2 < 1],
      {x1, 0, 1}, {x2, 0, 2}, {x3, 0, 2},
      Method → {"GlobalAdaptive",
        Method → {"TrapezoidalRule", "Points" → 3},
        "SingularityHandler" → None}, PrecisionGoal → 4,
      EvaluationMonitor :→ Sow[{x1, x2, x3}]]][[2, 1]];
```

```
Graphics3D[Point[points]]
```



There are two methods of Monte Carlo integration that we shall be discussing: the hit or miss method and the sample mean method.

Let y = f(x) be a function that is bounded and non-negative on the interval [a,b]. Then there must be some real number c, c > 0, such

that
$f(x) \leq c \leq 0$ for all x in [a,b]. Now, the definite integral :

$$\int_a^b f(x)\, dx$$

represents the area of the region beneath the graph of y = f(x) and above the x-axis on [a,b]. Let us denote this region as R. Then this region R is contained entirely within the rectangle bounded by the lines x = a, x = b, y = 0, and y = c, and which has an area of c(b - a). Let A be the area of region R--the value of this definite integral. Then the fraction of the rectangle that the region R represents is given by A/c(b - a). Suppose that we randomly select a point (x1, y1) from within this rectangle (including its boundary). Now, to randomly select (x1,y1) from within the rectangle implies that we randomly select x1 from the interval [a,b] and y1 from the interval [0,c]. The probability that this point will lie within region R will then be p = A/c(b-a). This means that the point (x1,y1) has probability p that it will satisfy the relation $y1 \leq f(x1)$.

We shall consider is the approximation of the area of the region between the x - axis and the graph of the curve y = 1 / x on the interval
[1, 2].

This implies that we are finding an approximation to $\int_1^2 1/x\, dx = \ln(2)$
On the interval [1,2], the curve y = 1/x satisfies the bounds $0 \leq 1/x \leq 1$=>the region is contained within the rectangle bounded by the lines x = 1, x = 2, y = 0, and y = 1, having an area of 1 square unit. The ratio ln(2)/1=ln(2) is then approximated by randomly choosing points (x,y) such that $1 \leq x \leq 2$ and $0 \leq y \leq 1$and then by finding the ratio of the number that satisfy $y(x) \leq 1/x$ to the total.

```
p1[x_] := 1 / x;
nmbr := 1
Do[event = 0;
 Do[u = Random[];
  v = Random[];
  x = 1 + u;
  y = v;
  If[y ≤ p1[x], event = event + 1], {i, 1, nmbr}];
 Print["nr.= ", nmbr, "  average = ", event * 1. / nmbr];
 nmbr = 2 * nmbr, {j, 1, 22}]
```

```
nr.= 1  average = 1.

nr.= 2  average = 1.

nr.= 4  average = 0.25

nr.= 8  average = 0.625

nr.= 16  average = 0.6875

nr.= 32  average = 0.53125

nr.= 64  average = 0.5625

nr.= 128  average = 0.671875

nr.= 256  average = 0.695313

nr.= 512  average = 0.660156

nr.= 1024  average = 0.716797

nr.= 2048  average = 0.687012

nr.= 4096  average = 0.708496

nr.= 8192  average = 0.694336

nr.= 16 384  average = 0.68811

nr.= 32 768  average = 0.695709

nr.= 65 536  average = 0.690002

nr.= 131 072  average = 0.694077

nr.= 262 144  average = 0.694744

nr.= 524 288  average = 0.692966

nr.= 1 048 576  average = 0.692958

nr.= 2 097 152  average = 0.692826
```

```
NIntegrate[1 / x, {x, 1, 2}]
```

```
0.693147
```

Let y = f(x) be a function that is bounded on the interval [a,b]. Then the mean or average value of f(x) on [a,b] is given by

et y = f(x) be a function that is bounded on the interval [a,b]. Then the mean or average value of f(x) on [a,b] is given by

$$\mathbf{fmed}=\frac{1}{b-a}\int_a^b f(x)\,\mathrm{dx}$$

Now suppose that we randomly select n values,x1,x2,...xn, from the interval [a,b]. Then we would expect to be able to approximate the average value of f(x) on [a,b] by averaging the values of f(x) at the xi:

$$\mathrm{fmed}\approx\frac{1}{n}\sum_{i=1}^{n} f(\mathrm{xi})$$

Thus we see that

$$\int_a^b f(x)\,\mathrm{dx}\approx\frac{b-a}{\pi}\sum_{i=1}^{n} f(\mathrm{xi})$$

This then yields a method of approximating the definite integral of f(x) on [a,b], known as the sample mean method of Monte Carlo integration.

Let us once again consider how to approximate the area of the region above the x-axis and beneath the graph of the function y = f(x), where

$f(x)=\mathrm{Sin}(50x)\mathrm{Cos}(120x)+1$

on the interval $[0,\pi]$. In this case we shall utilize the sample mean method of Monte Carlo integration to find the value of the definite integral

$$\int_0^{\pi}[\mathrm{Sin}(50\,x)\,\mathrm{Cos}(120\,x)+1]\,dx$$

We incorporate the sample mean method of Monte Carlo integration to obtain a sequence of approximations of the average value of this function on $[0,\pi]$ by means of the Mathematica program

```
p2[x_] := IntegerPart[Sin[50 * x] * Cos[120 * x] + 1]
a = 0.0;
b = Pi;
nmbr := 1
Do[smm = 0.0;
 Do[x = a + Random[] * (b - a);
  y = p2[x];
  smm = smm + y, {i, 1, nmbr}];
 Print["nr. = ", nmbr, " average =   ", smm / nmbr];
 nmbr = 2 * nmbr, {j, 1, 22}]
```

```
nr. = 1 average =   1.
nr. = 2 average =   0.5
nr. = 4 average =   0.5
nr. = 8 average =   0.5
nr. = 16 average =   0.4375
nr. = 32 average =   0.4375
nr. = 64 average =   0.453125
nr. = 128 average =   0.46875
nr. = 256 average =   0.53125
nr. = 512 average =   0.488281
nr. = 1024 average =   0.518555
nr. = 2048 average =   0.50293
nr. = 4096 average =   0.494873
nr. = 8192 average =   0.497803
nr. = 16 384 average =   0.500183
nr. = 32 768 average =   0.501221
nr. = 65 536 average =   0.5009
nr. = 131 072 average =   0.499077
nr. = 262 144 average =   0.49894
nr. = 524 288 average =   0.501053
nr. = 1 048 576 average =   0.499733
nr. = 2 097 152 average =   0.499512
```

From this data it appears that the average is approximately 0.500, which yields a value of 3.142*0.500=1.571 for the area of the region

```
NIntegrate[Sin[50 * x] * Cos[120 * x] + 1, {x, 0, π}]
```

```
3.14159
```

Monte Carlo Method for Definite Integration

Steven Keltner and Scott Sloan University of Missouri-Rolla

The program evaluates definite integrals by creating a "box" with dimensions defined by the limits of integration and the maximum and minimum values of the function on the selected interval. The computer randomly generates the coordinates of a point within the box. The function is evaluated at the randomly selected x value. If the random point falls within the area of the integral, it is counted by Mathematica. After performing the procedure for the specified number of points (iterations), the computer calculates the final result by multiplying the total area of the box by the percentage of points that fell within the bounds of the integral. Of course, many functions have regions that fall below the line y=0. If the program encounters a point that is within the area of the integral but below the x-axis, it subtracts one from the total number of points counted.

One would expect that the more points generated for each evaluation of the integral, the more accurate the answer will be. Since random numbers are used, however, there will always be variation from one determination to the next. Because of the lack of precision, the program is designed to perform many calculations with a varying number of iterations. Doing so makes it possible to "zero in on" the actual answer.

The first part of the program defines the parameters :

```
h[x_] = Exp[x]^2 - x + 1;  (*you can use any function = ??*)
x0 = -2; (*begining of the interval*)
xm = 2; (*ending of the interval*)
stepsize = 2000 ; (*step size*)
it0 = 1000 ; (*smallest # iterations*)
itm = 2000 ; (*largest # iterations*)
trials = 100 ;  (*repeats for same # iterations*)
l1 = {}; (*empty list*)
l2 = {} ;
(*empty list*)
```

The second step is to determine the maximum and minimum value of the function on the specified interval. The program uses these results as boundaries between which random y values are selected. The values are also necessary to determine the area of the "box."

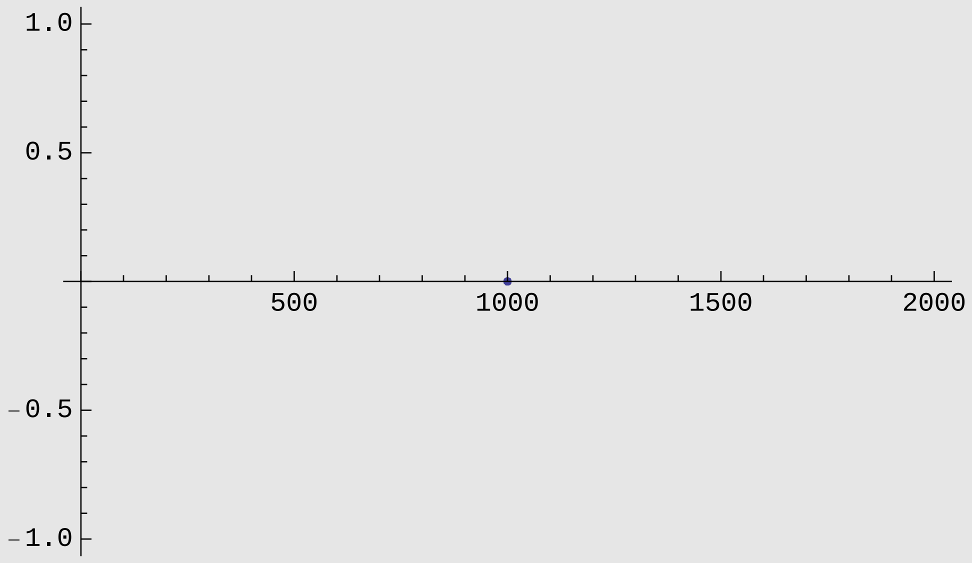
```
y0 = Min[Table[N[h[x]], {x, x0, xm, 0.01}]];
ym = Max[Table[N[h[x]], {x, x0, xm, 0.01}]];
```

The third step of the program generates random coordinates and determines how each point should be counted. If the random y value **r** is less than or equal to the function evaluated at the random x value **s**, it is counted.

```
Do[Do[Counter = 0;
Do[r = Random[Real, {y0, ym}];
s = Random[Real, {x0, xm}];
If[r <= h[s] && r >= 0, Counter = Counter + 1, Counter = Counter];
If[r >= h[s] && r < 0, Counter = Counter - 1, Counter = Counter],
      {i, 1, k}];
     AppendTo[l2, N[(Counter / k) * ((xm - x0) * (ym - y0))]];
AppendTo[l1, k], {j, 1, trials}];
  , {k, it0, itm, stepsize}];
```

In the forth step after transposing the data, the information is plotted:

```
data = Transpose[{l1, l2}];
ListPlot[data, PlotRange -> All]
```


1.0
0.5
−0.5
−1.0
500
1000
1500
2000