♦ MoHPC ♠   *The Museum of HP Calculators*

HP Forum Archive 16

[Return to Index | Top of Index]

Random simulation (and stimulation?)

*Message #1 Posted by **Kiyoshi Akima** on 17 July 2006, 2:23 p.m.*

Most modern calculators have random number generators (RNGs) built in, and they are useful things. Of course, most calculator (and computer) RNGs aren't truly random, though there are hardware RNGs (usually with a Geiger counter or other external input).

Even software pseudo-RNGs can be useful. And not just for games. Much of probability theory is built upon the study of random events.

Consider a large flat surface ruled with parallel lines d apart. Onto this drop a needle of length $L<d$. When the needle comes to rest (assuming the needle is blunt so it neither sticks nor stands on end), it either intersects a line or it doesn't. What's the probability that it intersects a line (or touches one)?

Some of you may be familiar with this problem and know the solution. But try the micro-challenge below anyway. (I call it a micro-challenge because I don't specify the HP calculator to be used, the only limit on program size is the calculator's memory, the only limit on running time is your patience (and perhaps the calculator's battery life)), and as for accuracy, well, read on.

Take your favorite HP calculator and program it to simulate the experiment. Because the outcome of a single random experiment is not of much use other than deciding which side gets the ball first, make the program do it ten times and report the number of intersections. Run it more than once. Do you get the same result every time? If not, what result do you get the most often?

Enhance your program to run n trials (of ten tosses each) and keep track of the results. That is, record how many times it got 0 intersections, 1 intersection, and so on up to 10 intersections. Run the program for a suitable value of n (depending on your calculator's speed and your patience). What does the distribution of results look like? If you can compute the theoretical results, compare them with your impirical results.

I'll present my RPN and RPL programs at the end of the week, along with a fuller discussion of the problem and the results.

Extra Credit
On your HP calculator, evaluate the following integral, giving the value and an estimate of the error.

```
   /1 /1 /1 /1 /1 /1 /1 /1      dA dB dC dD dE eF dG DH
I = |  |  |  |  |  |  |  |   --------------------------------
```

```
/0 /0 /0 /0 /0 /0 /0 /0  1 + A + B + C + D + E + F + G + H
```

Something a little more precise than The minimum value of the function is 1/9 and the maximum value is 1. Thus 1/9 < I < 1.
(After part 3 of this series, you will know how to do this.)
(And that was a big hint right there.)

Extra Extra Credit
Tie in the original problem to the recent World Cup final.
(Answer later this week.)

Re: Random simulation (and stimulation?)

Message #2 Posted by Valentin Albillo on 18 July 2006, 2:19 p.m.,
in response to message #1 by Kiyoshi Akima

Hi, Kiyoshi Akima:

Kiyoshi Akima posted:

"Extra Credit

  On your HP calculator, evaluate the following integral, giving the value and an estimate of the error.
```
        /1 /1 /1 /1 /1 /1 /1 /1      dA dB dC dD dE eF dG DH
    I = |  |  |  |  |  |  |  |   ----------------------------------
        /0 /0 /0 /0 /0 /0 /0 /0  1 + A + B + C + D + E + F + G + H
```
  Something a little more precise than The minimum value of the function is 1/9 and the maximum value is 1. Thus 1/9 < I < 1.
  (After part 3 of this series, you will know how to do this.)"

   I think I pretty much know how to do this right now :-)

   Matter of fact this is just a straightforward application of Monte Carlo techniques, so to merely evaluate the 8-dimensional integral this 3-liner for the bare-bones HP-71B (93 bytes) will do:


```
    1 DESTROY ALL @ RANDOMIZE PI @ FOR K=1 TO 7 @ M=10^K @ S=0
    2 FOR N=1 TO M @ S=S+1/(1+RND+RND+RND+RND+RND+RND+RND) @ NEXT N
    3 I=S/M @ DISP K;M,I @ NEXT K


    >RUN
```

```
    #  Points        Integral
    -------------------------------
    1  10            .186416796277
    2  100           .206225523207
    3  1000          .206581885362
    4  10000         .20584278312
    5  100000        .205729644257
    6  1000000       .205804443536
    7  10000000      .20580445907
```

so your 8-dimensional integral evaluates to **I = 0.20580**, correct to all five digits plus or minus 1 ulp.

If you alwo want to compute a running estimate of the error, this slightly longer version will do, still at 3 lines:

```
1 DESTROY ALL @ RANDOMIZE PI @ FOR K=1 TO 7 @ M=10^K @ S=0 @ E=0
2 FOR N=1 TO M @ T=1/(1+RND+RND+RND+RND+RND+RND+RND+RND) @ S=S+T @ E=E+T*T
3 NEXT N @ I=S/M @ E=SQR((E/M-I*I)/M) @ DISP K;M,I,E @ NEXT K


>RUN
```

```
    #  Points        Integral           Estimated error
    -------------------------------------------------------
    1  10            .186416796277      7.87240479015E-3
    2  100           .206225523207      3.63497557268E-3
    3  1000          .206581885362      1.1943736672E-3
    4  10000         .20584278312       3.62667298278E-4
    5  100000        .205729644257      1.15685295773E-4
    6  1000000       .205804443536      3.66089195702E-5
    7  10000000      .20580445907       1.15814079667E-5
```

so we can now confidently state that **I = 0.205804 +- 0.000012** which arguably is "*a little more precise than 1/9 < I < 1*", as you requested.

As is typical of Monte Carlo methods, the error is proportional to 1/Sqrt(N), where N is the number of random multidimensional points used, so the error diminishes by a factor of Sqrt(10) = 3.16+ for each increase of an order of magnitude in the number of points used. This is evident in the above data, where each Estimated Error approximatey equals the previous one divided by 3.16+.

Faster convergence (order 1/N and better) may be easily obtained by a simple modification to the above 3-liners but for the purposes of computing "*a little more precise*" value, this will do.

Your first question is a simple case of B*****'s needle, which I'll leave for others to deal with, and as for the World Cup I couldn't care less ! :-)

Best regards from V.

Edited: 18 July 2006, 2:21 p.m.

Re: Random simulation (and stimulation?)

Message #3 Posted by Kiyoshi Akima on 24 July 2006, 12:54 p.m.,
in response to message #2 by Valentin Albillo

Yes, the Monte Carlo is the obvious approach. But it's not necessarily the best.

How long did it take for 1E7 points? Emu71 took about an hour on my machine and I estimate one to two weeks on a real 71B.

It's true that the error for the simple Monte Carlo is proportional to $1/\sqrt(N)$, and there are ways to improve this to $1/N$. However, there is a way to evaluate this integral with an error proportional to $1/N^2$. And this scheme does at least as well as a 1E8-point Monte Carlo with N<1000. This gives a calculator or a real-speed emulator a chance to get five or six digits in time measurable by a clock instead of a calendar.

BTW, what you call the "error estimate" is actually a one-standard-deviation bound on the error, not a rigorous bound. And since there's no guarantee that the errors follow a Gaussian distribution, it can only be used as a rough indication of probable error.

Re: Random simulation (and stimulation?)

Message #4 Posted by RDunn on 20 July 2006, 2:04 p.m.,
in response to message #1 by Kiyoshi Akima

For ten tosses of Buffon's Needle, the most likely outcome as given by the binomial formula is 7, which is a little surprising as 6 gives the best approximation to the expected probability, which is 2/pi.

Here's a first stab at Buffon's Needle for the HP 48GX:

```
<< 1 0
  { 0 0 0 0 0 0 0 0 0 0 } -> n h p o
  << i n START
       1 10 START
         IF RAND RAND pi * SIN <
           THEN 'h' INCR DROP END
       NEXT
       'o' h GET 1 + 'o' h ROT PUT
       h 'p' STO+
       1 'h' STO
     NEXT
```

```
   o p n - n 10 * /
  >>
>>
```

The program produces a list of the number of occurences of each of the eleven possible outcomes and the proportion of hits.

Here's the output of a run of 100 trials, which took about 45 seconds:

```
{ 0 0 0 2 8 14 24 28 14 7 3 }
.653
```

which gives 3.0627871363 as an estimate for pi.

Re: Random simulation (and stimulation?)

Message #5 Posted by Kiyoshi Akima on 24 July 2006, 12:27 p.m.,
in response to message #1 by Kiyoshi Akima

As has been mentioned, this is simply Buffon's Needle. The probability of a needle of length L intersecting one of a set of lines spaced d is 2L/(pi d). A fuller discussion, including programs in RPN, BASIC, and RPL, is available here in PDF format.

Re: Random simulation (and stimulation?)

Message #6 Posted by Rodger Rosenbaum on 28 July 2006, 1:42 a.m.,
in response to message #1 by Kiyoshi Akima

It has been more than a week now, and I'm eagerly awaiting your solution, especially if you're going to expound on:

"It's true that the error for the simple Monte Carlo is proportional to 1/sqrt(N), and there are ways to improve this to 1/N. However, there is a way to evaluate this integral with an error proportional to 1/N2. And this scheme does at least as well as a 1E8-point Monte Carlo with N<1000. This gives a calculator or a real-speed emulator a chance to get five or six digits in time measurable by a clock instead of a calendar."

Re: Random simulation (and stimulation?)

Message #7 Posted by Kiyoshi Akima on 28 July 2006, 12:28 p.m.,
in response to message #6 by Rodger Rosenbaum

As stated originally, the multi-dimensional integral will be covered in part 3. Due to the less than overwhelming amount of interest out there, I'll be taking it slowly. Watch for part 2 next week.

But I did promise an answer to the Extra Extra Credit question tying the problem into the recent World Cup. As RDunn pointed out, Comte du Buffon was French. In addition, the goalkeeper for the winning Italian side was named Buffon.

Re: Random simulation (and stimulation?)

Message #8 Posted by Rodger Rosenbaum on 28 July 2006, 7:08 p.m.,
in response to message #1 by Kiyoshi Akima

The "extra credit" integral can be integrated without any special tricks other than lots of algebra, to give a final result of:

(11760*Log[27] - 44914*Log[2] - 2352*Log[729/256] - 290829*Log[3] + 3525221*Log[4] + 451150*Log[5] - 15676416*Log[6] + 3136*Log[27/4] + 23059204*Log[7] - 16777216*Log[8] + 4782969*Log[9] - 107163*Log[256/27] + 1003520*Log[3125/256] + 7840*Log[16])/5040

which evaluates to 0.2057972212989439 (if I haven't made a typo). I don't think this is the technique you had in mind, but it does provide a benchmark to compare the accuracy of the Montecarlo results.

I am aware of the so-called quasi-Montecarlo technique which can give convergence of roughly 1/N, but I'll have to wait to see the trick for 1/N^2

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall