

Distribution of Round-Off Errors

According to Landau and Paez, *Computational Physics*, it is often reasonable to assume that round-off errors are distributed randomly. Here we do an empirical check of this assumption. Here's a simple C++ code for errors in $\pi\sqrt{i}$, for $i = 1$ to 10,000:

```
// file: random_round-off.cpp
//
// This program tests whether round-off errors are randomly distributed.
//
// Programmer: Dick Furnstahl  furnstahl.1@osu.edu
//
// Revision history:
//   03-Jan-2004  original version
//   16-Jan-2005  changed type casting to functional form
//
// Notes:
// * We want to look at the round-off error in a set of numbers that
//   don't have too many that are exactly represented.  Just looking
//   at 1 to 10000, for example, is not very good.  We chose to take
//   the sqrt and then multiply all by pi, just to scramble things.
// * We use the difference of double and single precision numbers
//
// To do:
// * Test other sets of numbers.
// * Plot a histogram of the errors, to check distribution
//
//*****//

// include files
#include <iostream>           // note that .h is omitted
#include <iomanip>            // note that .h is omitted
#include <fstream>           // note that .h is omitted
using namespace std;        // we need this when .h is omitted

//*****//

const double pi = M_PI;     // use the built-in definition of pi=3.14159...

int
main (void)
{
    float x_float = 0.;      // single-precision number
    double x_double = 0.;   // double-precision number

    // open an output file stream
    ofstream out ("random_round-off.out");

    // print out column headings
    out << "#i" << " relative round_off error " << endl;

    for (int i = 1; i < 10000; i++)
    {
        x_double = sqrt ( double(i) ) * pi;
        x_float  = sqrt ( float(i) ) * pi;

        // round_off is the **relative** error (keep the sign)
        double round_off = (x_double - x_float) / x_double;

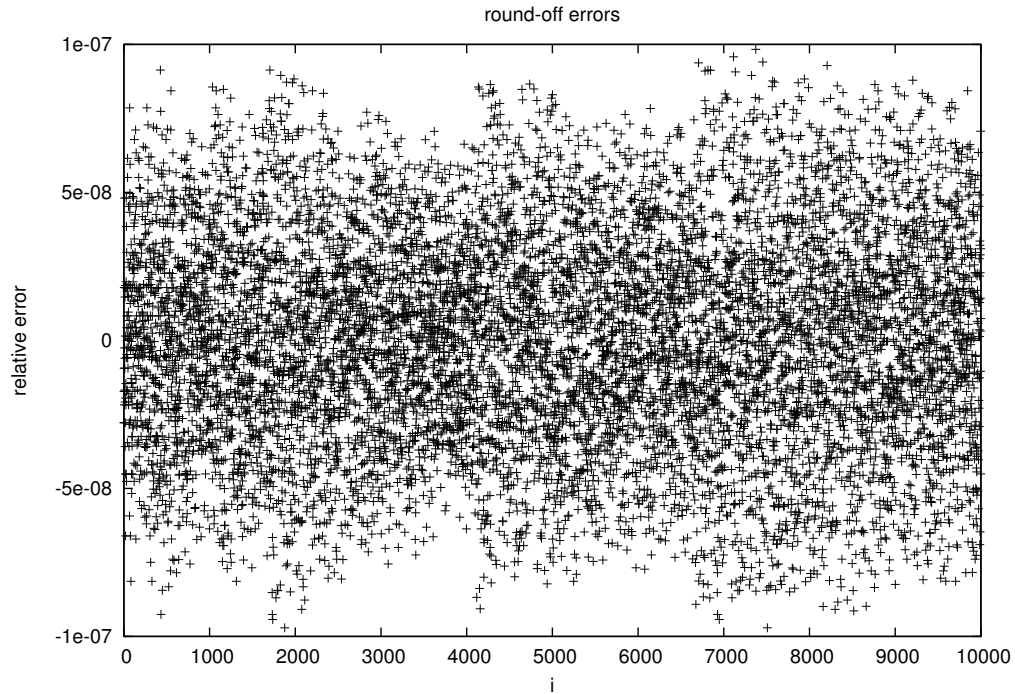
        out << " " << i << " " << scientific << round_off << endl;
    }

    // close the output file
    out.close ();

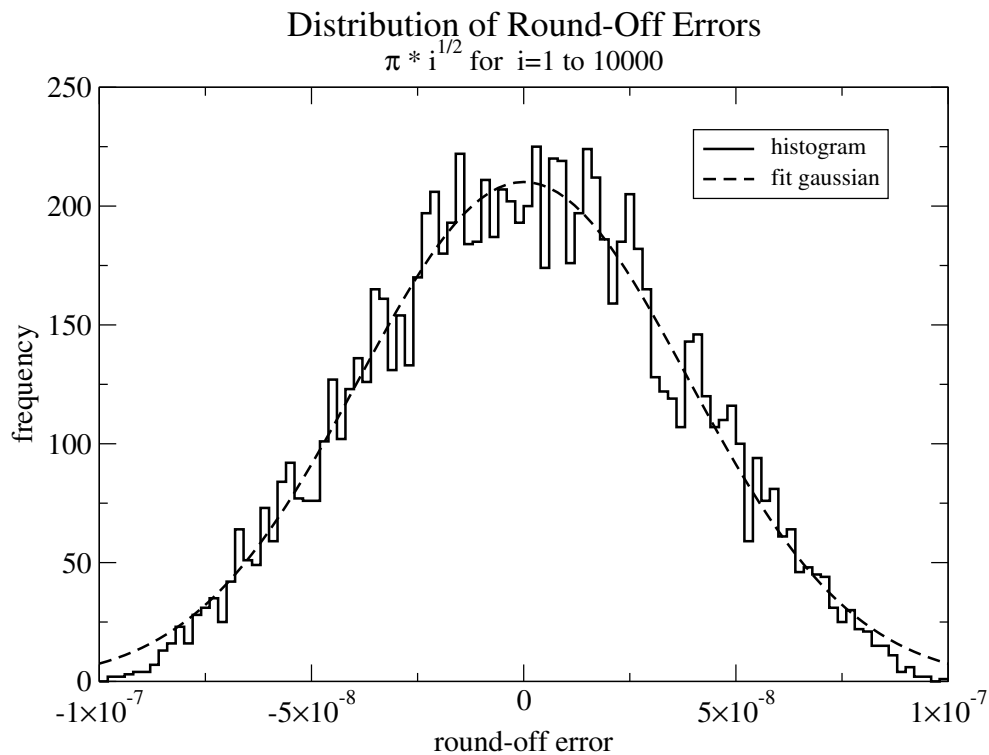
    return 0;
}

//*****//
```

We can make a simple plot of the relative error (second column) versus the index (first column) using gnuplot:



We see that the distribution of errors looks symmetric about zero and random by eye. Check the distribution by making a histogram with xmgrace:



Does this work for most sequences of round-off errors? Check out some examples!