Labs are to help you improve your skills and graded on “participation”. For full credit you must turn in at the end of each lab copies of source codes, output data and plots made during the lab period to demonstrate a “serious effort.”

In the first homework assignment you wrote and tested a simple routine using *double precision* arithmetic to evaluate the sin function through the series

$$\sin(x) \approx \sin_N(x) = \sum_{l=0}^{N} (-1)^l \frac{x^{2l+1}}{(2l+1)!} = \sum_{l=0}^{N} A_l.$$

In that exercise you were asked to form each term in the series ‘from scratch’; that is, explicitly combining together the factors $(-1)^l$, $x^{2l+1}$, and $(2l+1)!$ in each term of the sum. The goal of today’s exercise is to write a modified version of that routine that is more computationally efficient and to compare the execution time of your new routine to your original routine.

a) Take your original routine and adapt it so that it computes $K = 1,000$ values of $\sin(x)$ spread uniformly over the interval $[-\pi \leq x \leq \pi]$. You should have found in your homework exercise that you need about 13 terms ($N = 13$) in order to have a solution converged close to double precision machine accuracy. So, let $N = 13$ in your series. Make a plot of the computed $\sin_N(x)$ to test that you have done this correctly. (You do not need to print this plot, just view it as a reality check.)

b) Use the appropriate cpu timing function for your language of choice (see note below) to compute the execution time required for the calculation in part a). You will need to call this function in your code just before and just after you compute all the values. The time difference is your “execution time.”

c) Copy the lines of code used in part a) along with your timing structure and paste them at the end of your code in preparation for modifying them according to the following prescription. When you get to part e) you will end up computing all the values twice; once by each algorithm.

d) Given your working routine from part a) you want now to compute the terms in the series for $\sin_N(x)$ more efficiently by not repeating unnecessary operations. In particular, note that given $A_{l-1}$, then $A_l = -\{x^2/[(2l+1)(2l)]\} A_{l-1}$. Thus, starting with $A_0 = x$, you can build all of your other terms recursively with little computational effort. Modify your original routine so that it constructs the series in this fashion. Test it by comparing plots of $\sin_N(x)$ computed this way and as in part a).

e) Now apply your cpu timing structure to compare the execution times for the two methods to compute $K = 100,000$ values of $\sin_N(x)$. What is the ratio of the execution times? **Note:** You may want to run the code several times for this comparison, since timing measures can be influenced by other activity in the system. It’s best to compute the average time per value.

f) For fun you might want to compare your routine’s times against the library sin function. That one is hardwired into most modern processors to compute trig functions very quickly. On the cselab machines the time per library sin(x) value should be a bit less than 30 nsec.

Simple pseudocode for an execution time test:

```plaintext
Call cpu_time(t1)
   Code to be timed
Call cpu_time(t2)
   Time = t2 – t1
```
Note: In Fortran there are two calls you can use to accomplish timings. One is “call cpu_time(t0)”, as shown in the pseudo code. This call returns a time value, t0, in seconds. The other is “call system_clock(t0)”, which returns a time value in milliseconds. In C++ the equivalent operation is “gettimeofday(&t0,0);”. Sample F90 and C++ routines using these are: ftime.f90 and ctime.cpp. These are linked to the syllabus as ‘lab file1’ and’ lab file2’.