Grade: check +

Act 9/10

Optimization 101: Squaring a Number

1. ✓

2. ✓

3. pow(x,2)'s : 4.45189 s

   x * x's : 0.395591 s ⟶ most efficient

4. squarecit is much more efficient than pow(x,2) and slightly less efficient compared to x*x.

   Overhead: ~.2 s

   This would be worthwhile if you had to square a substantially large function and not just 2 numbers.

5. the macro was almost as efficient as x*x the 1st run, and more efficient on the 2nd run. I'd say they are "tied" for most efficient.

6. Using the macro would be most efficient. It is just as efficient as x*x but it would be better at squaring large functions and not just numbers. It is also easier to read for a user.

7. All of the times are now in the 10^{-5} or 10^{-6} range. This is clearly the best way to run code efficiently.

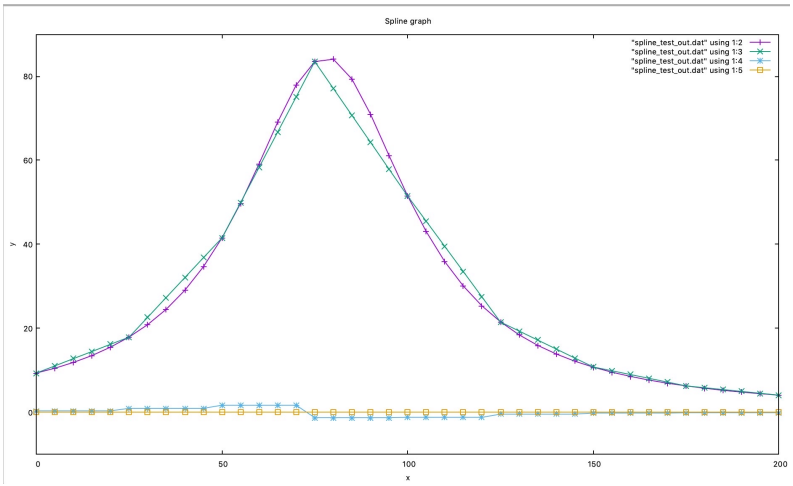
8. ✓

GSL Interpolation Routines

1) ✓

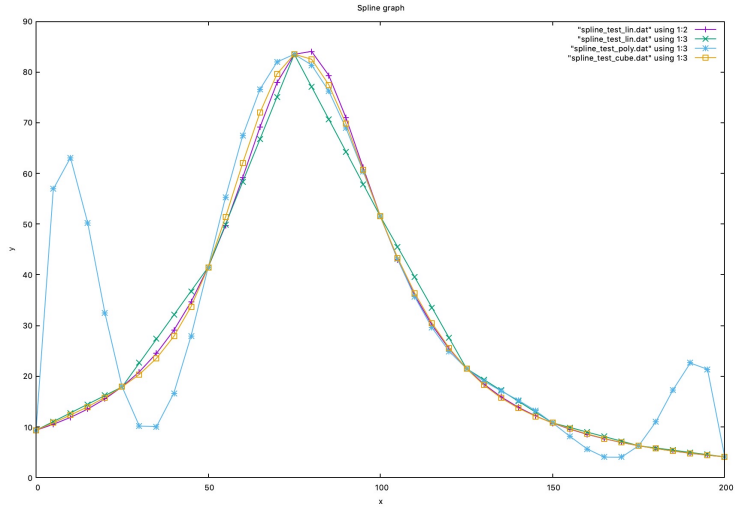
2) ✓

3)



4) I got "Illegal Spline Type!" → fixed that later.

5)



Cubic interpolation is closest to the "exact" value for y throughout the whole x-range. Linear does slightly worse than the cubic interp. Poly does an ok job near the peak, but a poor job near the end points.

Python Scripts for C++ Programs

1. ✓
2. I have tried
3. no
4. ✓

Cubic Splining

1. ✓
2. inline ✓
3. used the same r_max from an earlier session w/ the same spacing.

Command Line Mystery

Jeremy Bowers