# CSE 280: Assignment 2

Evan Ott
UT EID: eao466

November 7, 2016

## 2.0

```
tar -xf gsl-1.16.tar.gz
cd gsl-1.16
mkdir builds
cd builds
mkdir gcc0 gcc3 icc0 icc3

cd gcc0
module unload intel
module load gcc
../../configure CC=gcc CFLAGS=-O0
(time make -j1) 2> serial.txt
make clean
(time make -j16) 2> parallel.txt
make check -k 2>&1 | tee check_results.txt

cd ../gcc3
../../configure CC=gcc CFLAGS=-O3
(time make -j1) 2> serial.txt
make clean
(time make -j16) 2> parallel.txt
make check -k 2>&1 | tee check_results.txt

cd ../icc0
module unload gcc
module load intel
../../configure CC=icc CFLAGS=-O0
(time make -j1) 2> serial.txt
make clean
(time make -j16) 2> parallel.txt
make check -k 2>&1 | tee check_results.txt

cd ../icc3
../../configure CC=icc CFLAGS=-O3
(time make -j1) 2> serial.txt
make clean
(time make -j16) 2> parallel.txt
make check -k 2>&1 | tee check_results.txt
```

I found the results for the table by using `cat` on the `serial.txt` and `parallel.txt` files and using a one-liner for pass/fail:

```
grep "# PASS: " check_results.txt | \
awk '{print substr($0, 10)}' | awk '{ sum += $1 } END { print sum }'
```

| **Configuration** | Compilation Times | | Regression Tests | |
|---|---|---|---|---|
| | Serial (secs) | Parallel (secs) | Passes | Failures |
| gcc/noOpts | 145.767 | 35.637 | 48 | 0 |
| gcc/O3 | 221.200 | 54.408 | 48 | 0 |
| icc/noOpt | 334.042 | 60.797 | 48 | 0 |
| icc/O3 | 450.186 | 90.841 | 44 | 4 |

Conclusions:

- It was faster to build optimized GNU rather than optimized Intel, by around a factor of 2: (2.04 for serial, 1.67 for parallel).
- Non-optimized GNU was 4.09x faster to compile in parallel than in serial.
- The only regression failures occured using the Intel compiler using `-O3`, which isn't surprising given the "magic" that Intel does at that level of optimization. I ran

  ```
  grep -A4 "# FAIL:  1" icc3.check_results.txt | grep "See " | awk '{print substr($0, 5)}'
  ```

  to identify the directories where failures occured (in this case, I got lucky because each time there were any failures in a directory, there was exactly one failure, so I used `grep` for exactly that condition then looked a few lines down to find the directory. In this case, the failing directories were (including a note from the GSL documentation):

  - `linalg` – "functions for solving linear systems"
  - `specfunc` – "GSL special function library" for families of functions like Bessel, elliptic integrals, etc.
  - `poly` – "functions for evaluating and solving polynomials"
  - `ode-initval2` – "functions for solving ordinary differential equation (ODE) initial value problems"

## 2.2

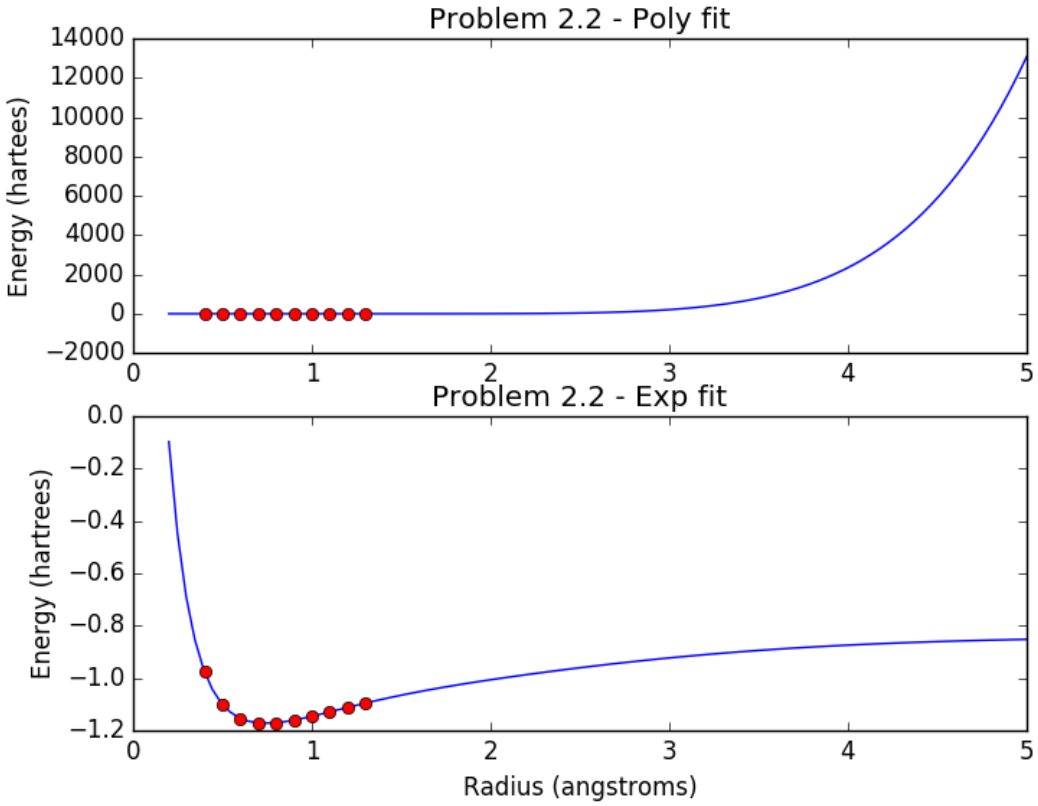


Figure 1: Showing the semi-anticipated behavior that a polynomial fit will get increasingly bad for something that should have asymptotic behavior, whereas the exponential fit is extremely reasonable and in line with typical "chemistry curves."

## 2.3

1. Excluding today, how many days are there left in this month?

   ```
   # bc -l <<< "‘cal | tail -c4 | head -c2‘ - ‘date +%d‘"
   23
   ```

2. Here's a way to look at the average and spread of round-trip-time to `google.com`, formatted nicely.

   ```
   # ping -i0.2 -c5 google.com | tail -n1 | cut -d"/" -f5,7 --output-delimiter=" +/- "
   6.236 +/- 0.147 ms
   ```

3. Here's a way to find the PID and user for the top five processes (using your default sort order).

   ```
   # top -n1 -b | grep -a5 "PID" | tail -n5 | sed -e 's/^[ \t]*//' | cut -d" " -f1-2
   18168 xchma
   18167 xchma
   27308 evanott
   27321 shawnlin
   25814 taslima
   ```

4. Generate a random word from the built-in dictionary

   ```
   # cat /usr/share/dict/words | sed "‘shuf -i 1-\‘cat /usr/share/dict/words | wc -l \‘ -n 1‘q;d"
   hindbrain
   ```

5. Finally, you find that your annoying co-worker leaves their workstation logged in, but unattended, then this is a reasonably fast way to make sure they don't do it again. NOTE: PLEASE do not actually run this command. Seriously. I ran it on my Mac and even after deleting the file (through the file manager, rather than command line – big mistake), it kept writing, to the tune of a 90GB or so file. I had to reboot my machine to go back and delete it.

   ```
   # (yes > ~/logout_next_time.txt &); clear
   ```

   No output. Alternatively, one can use something like `top` instead of `yes`. Again, don't actually run it unless you *actually* want to foobar someone (or yourself).