# PHY 407 Lab 2

Natalie Price-Jones, 999091021

natalie.price.jones@mail.utoronto.ca

19 September 2014

## 1 Question 1

### 1.a Part b)

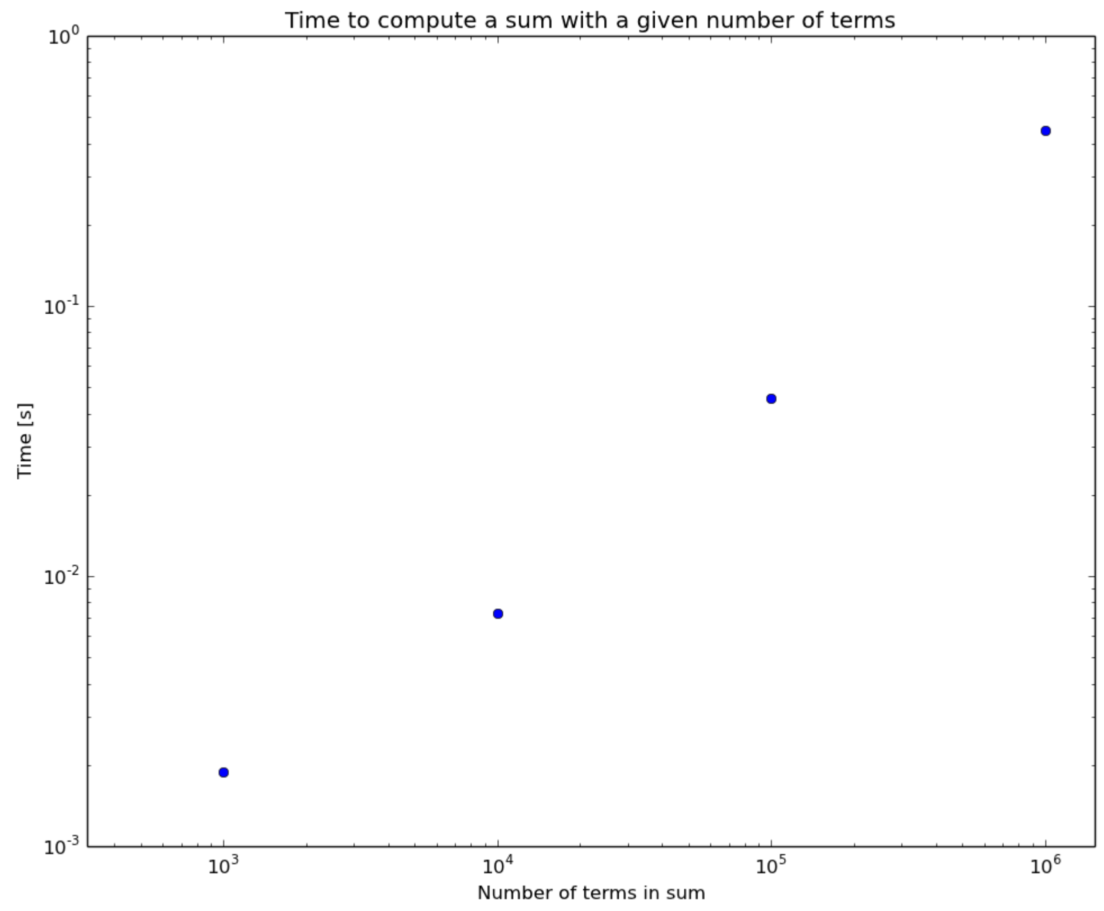

Figure 1: Time needed to compute a sum with a set number of terms. Both axes have been logscaled.

### 1.b Part c)

The for loop that cycles through each term in the sum is given below:

```
[1] for n in range(terms):
[2]     E = n + 0.5
[3]     weight = exp(-beta*E)
[4]     S += weight*E
[5]     Z += weight
```

We can explicitly count the number of floating point operations (flops) for each iteration of this loop. On line [2], we have 1 flop (addition). On line [3], there are 3 flops (two multiplications and an exponential). On line [4], there are 2 flops (addition and multiplication). On line [5], there is 1 flop (addition). Therefore our total tally is 7 flops per iteration of the loop.

### 1.c Part d)

We can calculate the number of flops per second by first taking the number of terms in the sum, multiplied by the number of flops per iteration (i.e. per term). This gives the total of number of flops for each sum. Then we divide this total by the time it took to compute the sum.

$$flop/s = (flop/term) * (nterms)/time$$

In order to compare this to the June 2014 top supercomputer, we then divided this flop/s by the number of cores in my laptop (two cores). The value varied, depending on the number of terms in the sum, but the averages over 10 repetitions of the program averaged to around $1 \times 10^{6.74}$ flop/s/core (Figure 1 had an average value of $1 \times 10^{6.7452}$ flop/s/core).

The top supercomputer for June 2014 according to www.top500.org could calculate $1 \times 10^{10}$ flop/s/core. This separates it from my laptop by about three and half orders of magnitude in speed.

## 2 Question 2

### 2.a Part b)

The result of running the Python file called lab2_q2.py is 4.40042666667. This is the calculated value of the integral of $x^4 - 2x + 1$ from 0 to 2 by dividing it into $N = 10$ slices and approximating it with Simpson's Rule.

We can calculate the fractional error ($\epsilon$) in this result ($I_{calc}$) by comparing it with the known true result ($I_{true} = 4.4$).

$$\begin{aligned}\epsilon &= (I_{calc} - I_{true})/I_{true}\\ \epsilon &= (4.40042666667 - 4.4)/4.4\\ \epsilon &= 9.70 \times 10^{-5}\end{aligned}$$

So the fractional error in the result when the integral is approximated with Simpson's Rule is $9.70 \times 10^{-5}$

## 2.b Part c)

| Number of Slices | Trapezoidal Rule | Fractional Error | Simpson's Rule | Fractional Error |
|---|---|---|---|---|
| 10 | 4.50656 | $2.4\times10^{-2}$ | 4.40042666667 | $9.7\times10^{-5}$ |
| 100 | 4.401066656 | $2.4\times10^{-4}$ | 4.40000004267 | $9.7\times10^{-9}$ |
| 1000 | 4.40001066667 | $2.4\times10^{-6}$ | 4.4 | $9.7\times10^{-13}$ |

Table 1: Fractional errors for trapezoidal rule and Simpson's rule for increasing number of divisions in the region of interest. Fractional error represents the deviation from the true value of 4.4.

It's easy to see that for the same number of slices, Simpson's rule gives vastly more accurate results.

## 3 Question 3

It is possible to calculate the error in an integral approximated with the trapezoidal rule with the following equation.

$$\epsilon = \frac{1}{3}|I_2 - I_1| \tag{1}$$

In Equation 1, $\epsilon$ is the error and $I_2$ is the evaluation of the integral with double the number of slices of those used to evaluate $I_1$. For the integral approximated with the trapezoid rule for $N = 10$ then $N = 20$ slices in the file lab2_q3.py, this equation yields $\epsilon = 0.026633333333333137$. However, the actual deviation from the true value, $|I_2 - 4.4| = 0.026660000000000572$ does not match the estimation of the error given by Equation 1. This is because Equation 1 is an approximation of the error. It quite obviously gives the right order of magnitude, but fails to represent precisely the error involved in the integral approximation.

## 4 Question 4

### 4.a Part c)

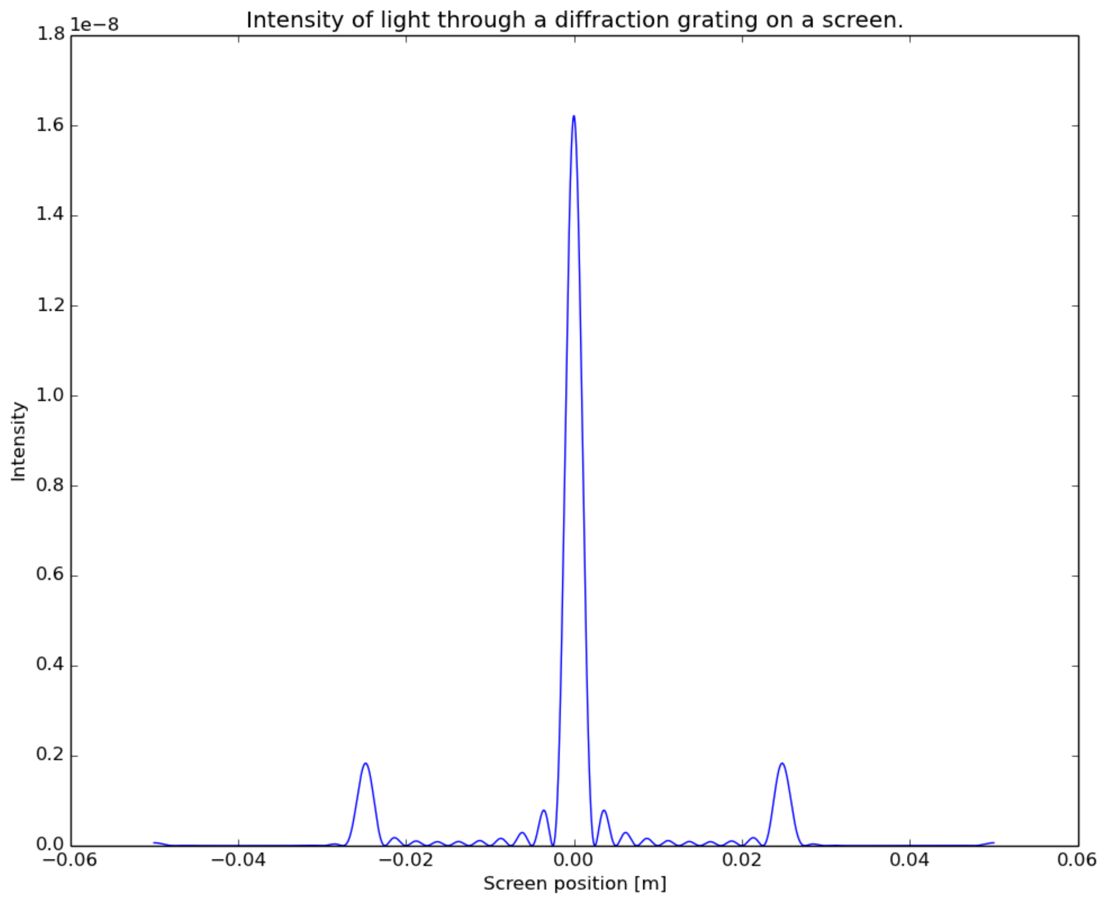

Figure 2: Intensity of light across a screen in one dimension for a transmission function given by $q(u) = sin^2(\alpha u)$.

### 4.b Part d)

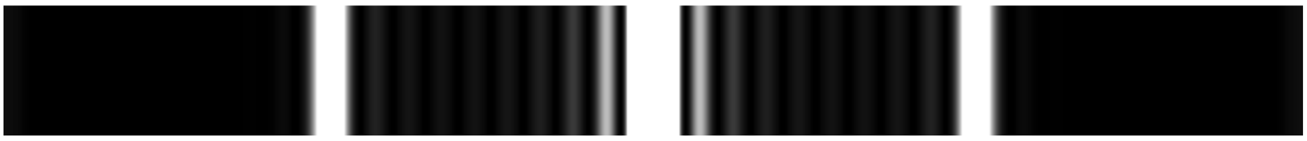

Figure 3: Intensity of light across a screen in two dimensions for a transmission function given by $q(u) = sin^2(\alpha u)$.

## 4.c Part e)

### 4.c.1 Part i)

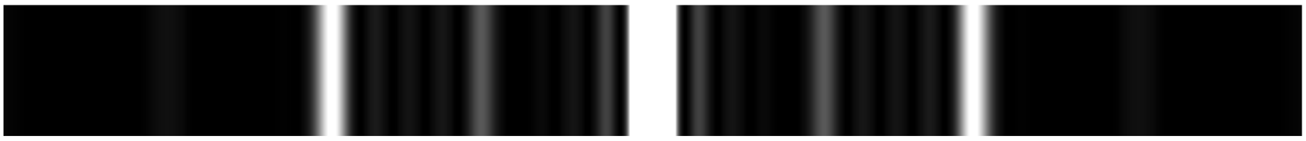

Figure 4: Intensity of light across a screen in two dimensions for a transmission function given by $q(u) = sin^2(\alpha u) sin^2(\beta u)$.

### 4.c.2 Part ii)

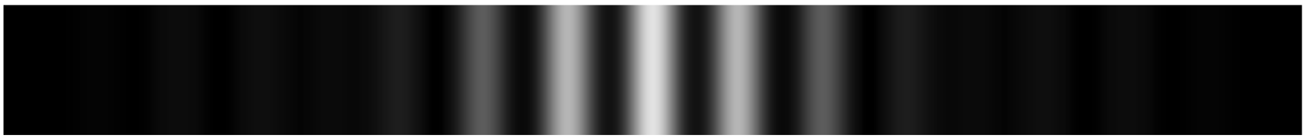

Figure 5: Intensity of light across a screen in two dimensions for a transmission function for square slits, one $10\mu$m slit $60\mu$m from a $20\mu$m slit.