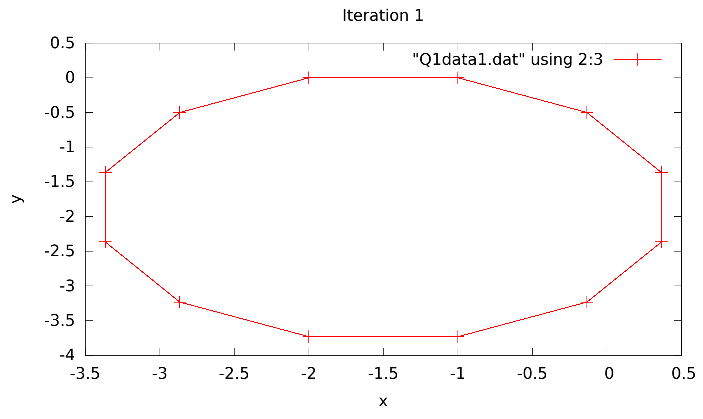
PHYS3071 Assignment 1

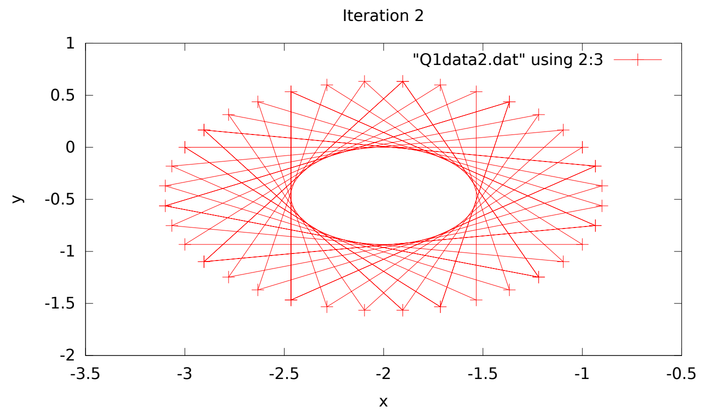
Ryan White
s4499039

August 18, 2022

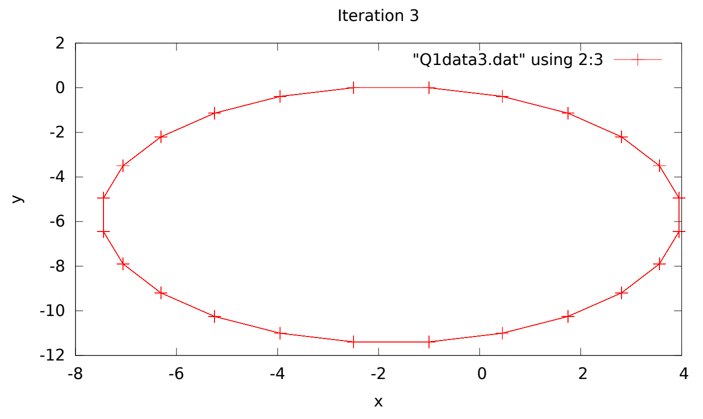
1 Question 1



(a) 100 Iterations, 30° Angle Increment, 1cm Movement



(b) 50 Iterations, 130° Angle Increment, 2cm Movement



(c) 200 Iterations, 15° Angle Increment, 1.5cm Movement

Figure 1: Depending on the user-defined parameters, the trail of the ball movement could show vastly different shapes.

The main goal of this question was to plot the trail of a ball rolled by some distance at some angle to the left (from the ball's forward facing point of view), N times. The ball was first rolled 1cm to the left of the origin, such that its position was $(x, y) = (-1, 0)$. To calculate the next position on each iteration, simple trigonometric functions (sines and cosines) were used; which function was used exactly depended on the magnitude of the angle with respect to the positive x-axis. After a basic function was created which satisfied parts 'a.' and 'b.' of the question requirements, the program was modified to be human readable and to accept user input. An example of the program output (with user input) is shown below:

```
This is a program that models the position of a
ball (with respect to the origin) after it has
been rolled to the left by some angle and some
distance.
On each step, the ball is rolled some angle, and
some distance.  The program will output the final
position, and the closest point to the origin
(and the step).
Enter the number of steps to iterate over:  200
Enter the angle to increment each time:  15
Enter the distance the ball will travel each
increment:  1.5
The ball was closest to the origin at step 191,
with a distance of 0.593483
The final position of the ball was (x, y) =
(-7.44682, -6.44682).
Thank you for using the program!
```

The main `.cpp` file ran the Question 1 code three times to produce a few different outputs according to user input. Some examples of possible user inputs and their resultant output are shown to the left. These graphs were plotted using GNUplot, using saved data files (with the extensions `.dat`) from each of the code runs.

2 Question 2

To begin, the supplied data was plotted using GNUplot, shown in Figure 2.

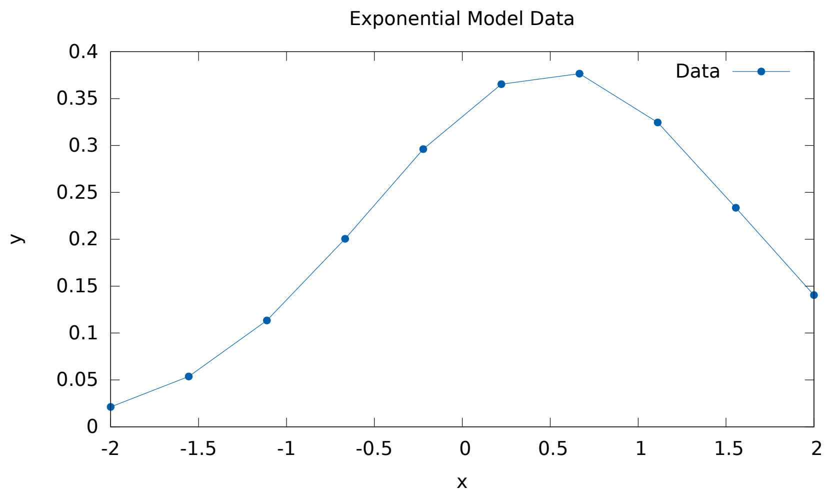


Figure 2: Normal Distribution Data

Using the cubic interpolation code previously bug-fixed (from a supplied version) from Workshop 3, the data was interpolated using 100 points. The cubic interpolation worked by first estimating the double derivatives of the underlying function of the data points, and then using them in a polynomial-like function to estimate the y-values (given some x-values) between defined data points. The result from this is shown by the blue points in the top subfigure of Figure 3 (approximately where the red points are).

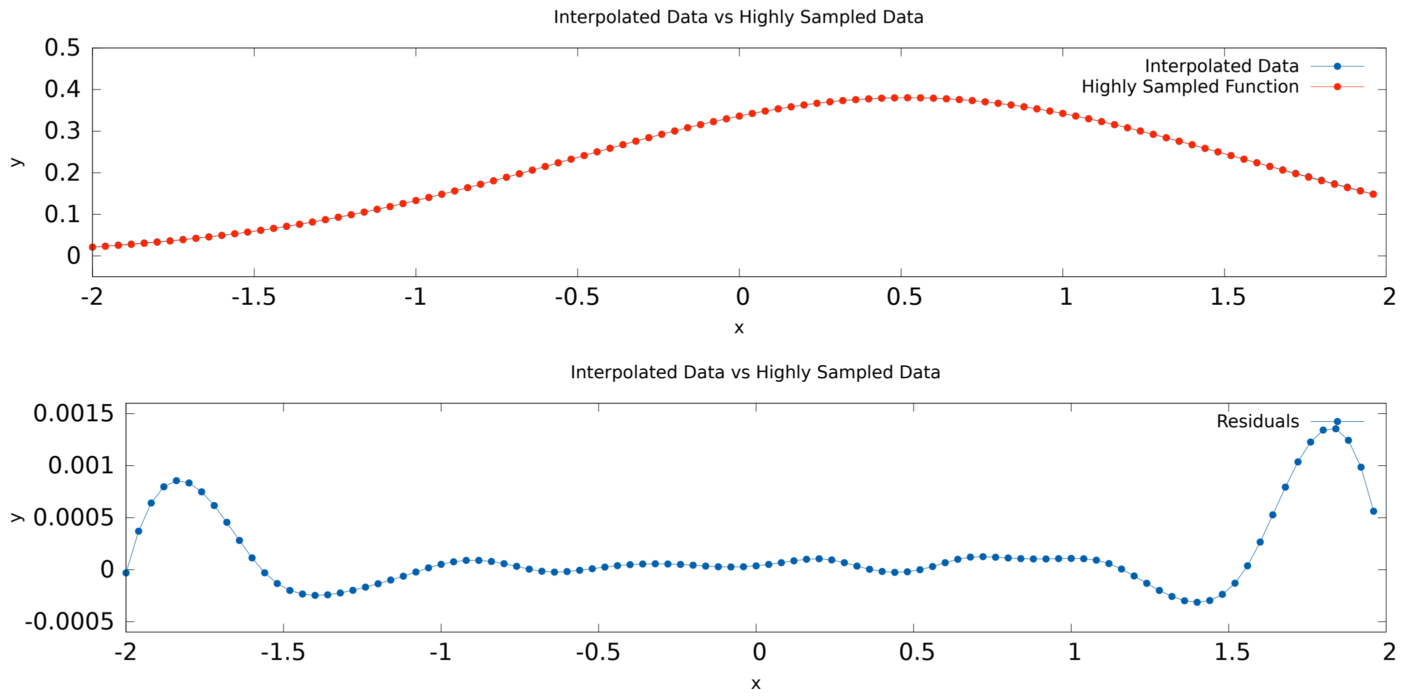


Figure 3: Top: the interpolated and highly sampled data plotted on the same axis. It's difficult to discern the difference, and the residuals (bottom) show that they are in excellent agreement across the whole domain. Some inconsistency is seen near the boundaries of the domain, which is likely due to inaccuracy of the inferred double derivatives of the data (as a result of the interpolation function). With that said, even at its worst it is still in good agreement with the theoretical model.

Due to the symmetry of the normal distribution, the mean value of the function occurs at the peak of the distribution. As such, it was easy to estimate the mean of the underlying distribution in the data by finding where the peak of the interpolated data was: $x_{\text{peak}} = \mu$. The standard deviation wasn't quite as simple, but by no means difficult. Since the exponential term in the normal distribution is just 1 when at the mean, the y-value at the peak of the distribution is equal to the coefficient term in the normal distribution:

$$y_{\text{peak}} = \frac{1}{\sqrt{2\pi\sigma^2}} \tag{1}$$

This is simple to rearrange, giving $\sigma = 1/(y_{\text{peak}}\sqrt{2\pi})$. With both the mean and the standard deviation estimated, a more highly sampled distribution of data (shown in red in Figure 3) was created and plotted against the interpolated data (blue). By subtracted the y-value of the model from that of the interpolated data, the residuals could be calculated.

The cubic interpolation is clearly an incredibly powerful tool (as is evident by the magnitude of the residuals), especially when there are a *reasonable* number of data points to begin with in the base data.