Elizabeth Bradley

November 10th, 2010

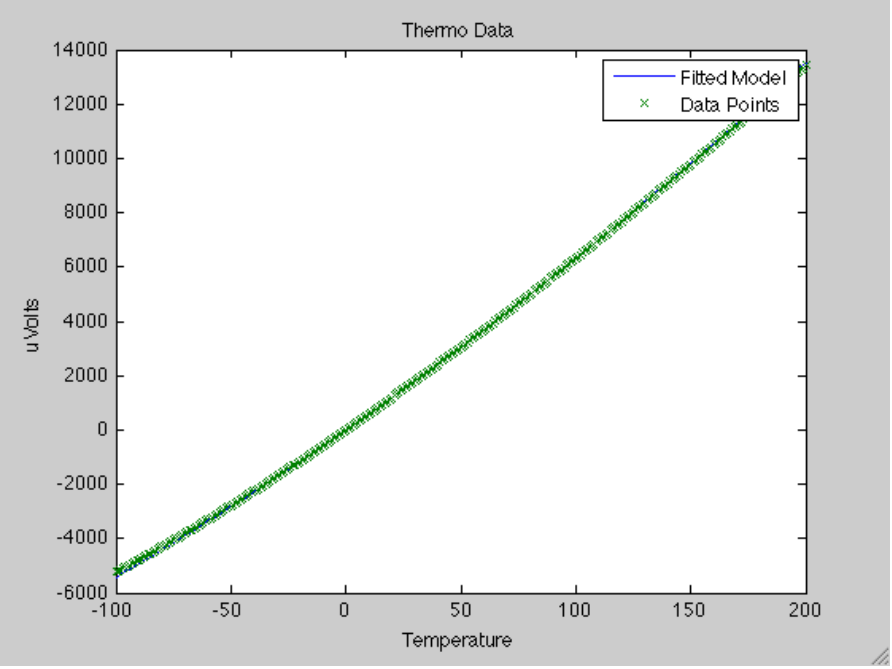
Part 1: Sensor Calibration

This project takes the data from a thermo sensor and finds a quadratic and cubic representation of the data from (0,100) degrees. The fit of the graph is then graphed with all the data points. In order to accomplish this the input for the function is Vand([ALL DATA], [selected Data], [number of terms in polynomial]). Only the data from the selected data input is used to find the best fit model. The best fit model is found using the method described in the reading.

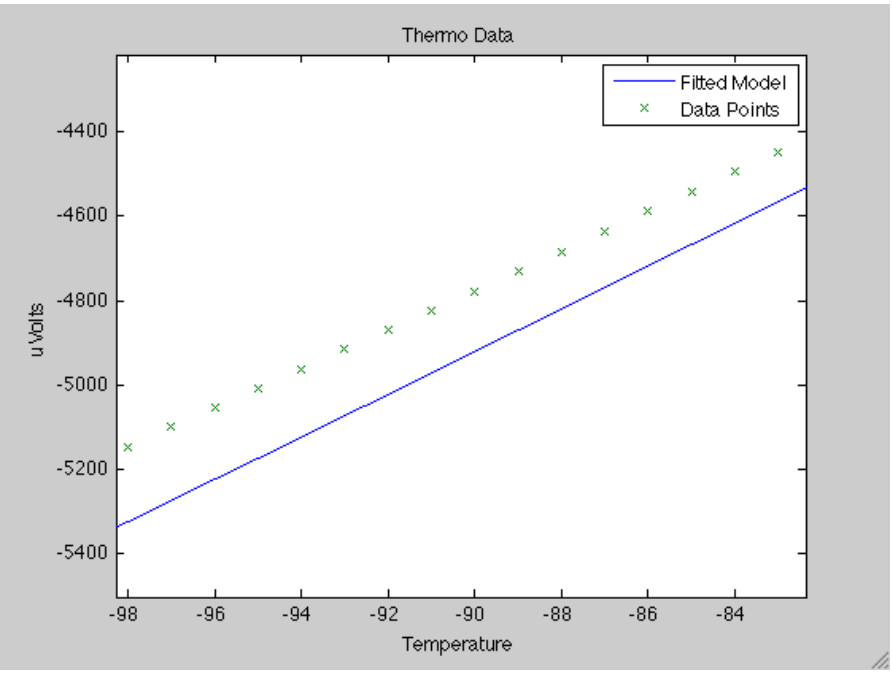
Next, the standard deviation of the modeled line to the given data points is found. This result is then displayed.

The following results from the thermo data for a quadratic:

```
>> Vand(data,data1_100,3)

Coefficents:

  -0.3904

   58.7079

0.0448


Standard Deviation:

  53.3033
```



The graph below shows a closer look at the data above. It is clear that the line is not a perfect fit for the data.



The following results from the thermo data for a cubic:

>> Vand(data,data1_100,4)

Coefficents:

0.3121

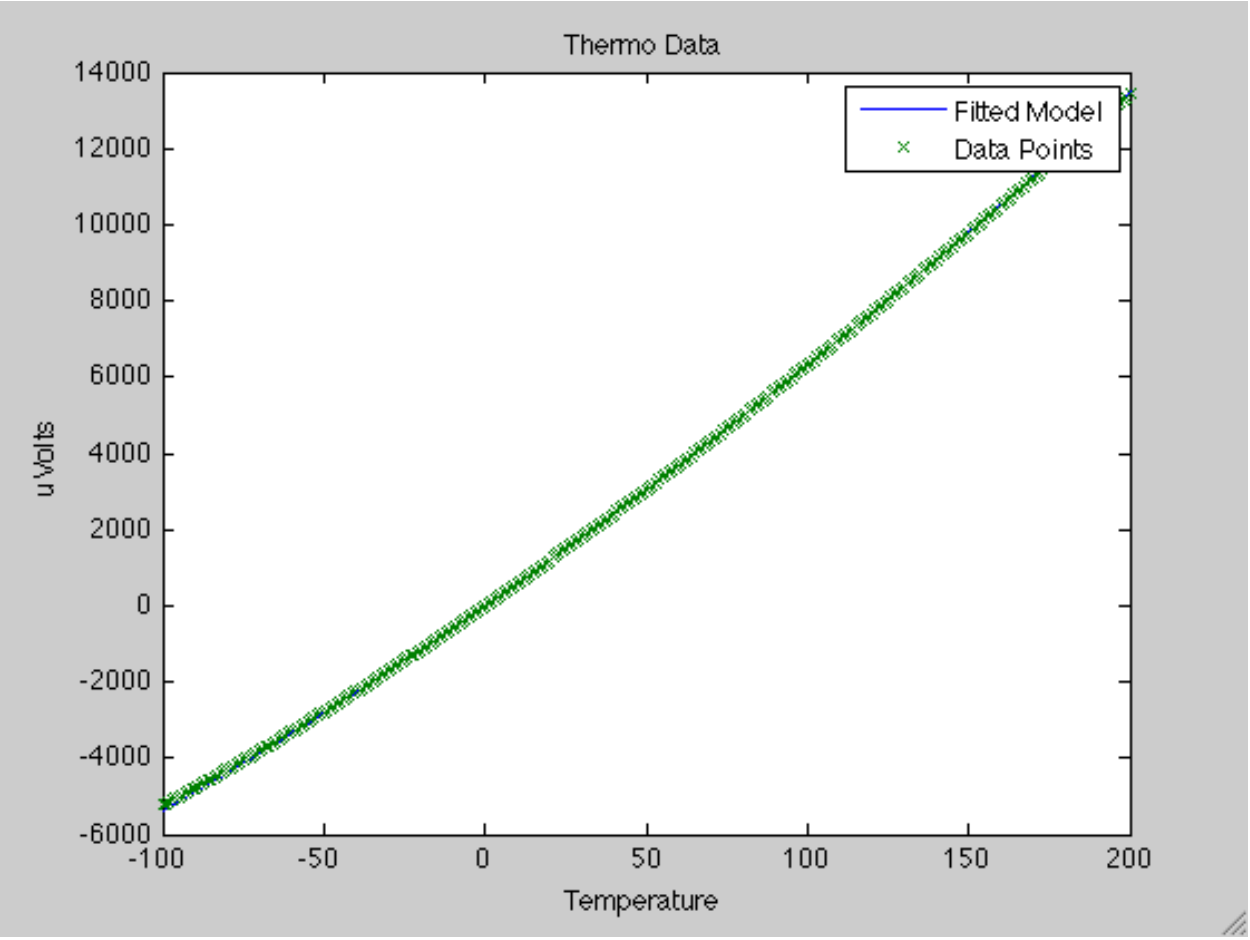
58.6214

0.0469

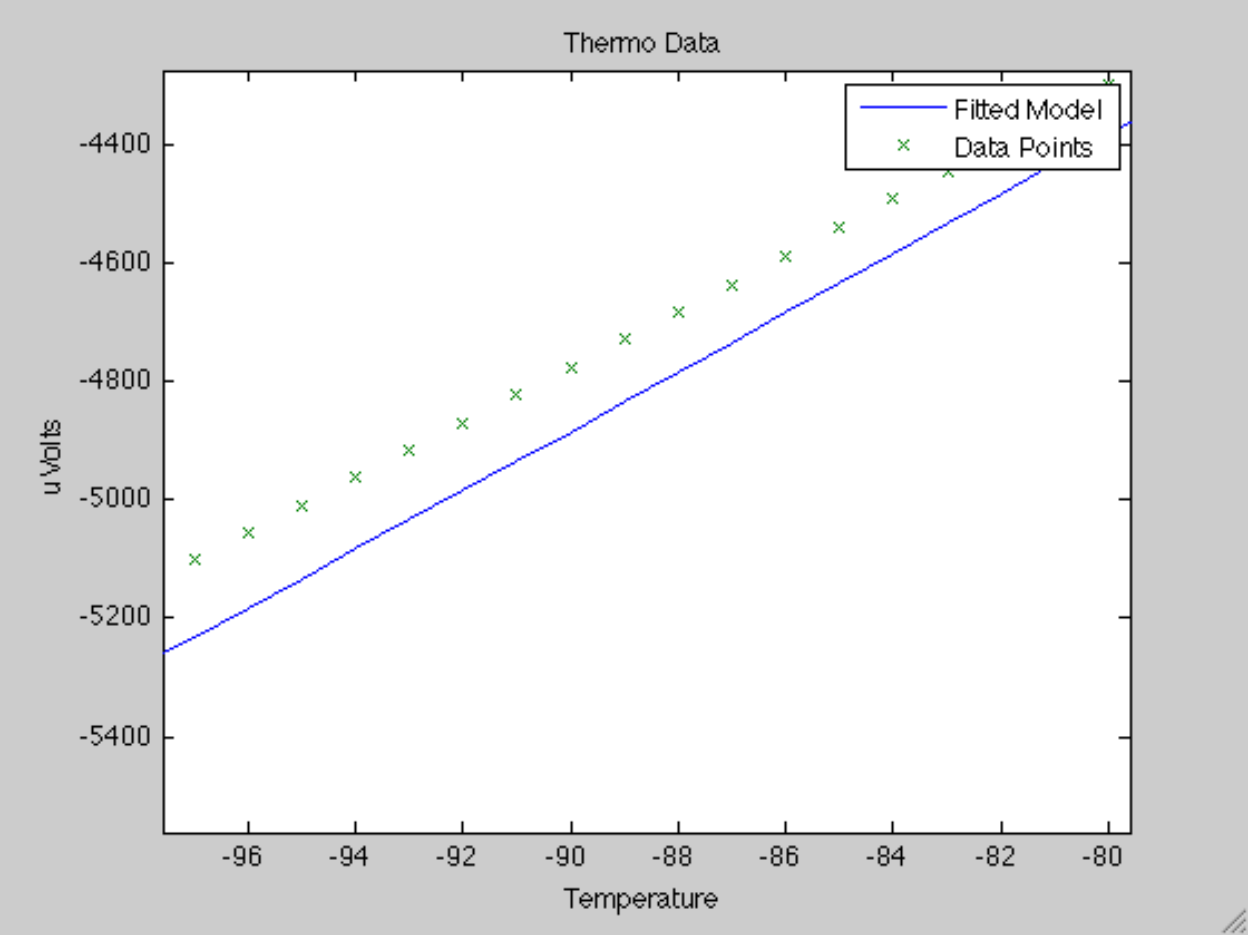
-0.0000

Standard Deviation:

36.9092



The graph below shows a closer look at the data above. It is clear that the line is not a perfect fit for the data.



The following results from the PRT data for a quadratic:

>> Vand(data,data1_100,3)

Coefficents:

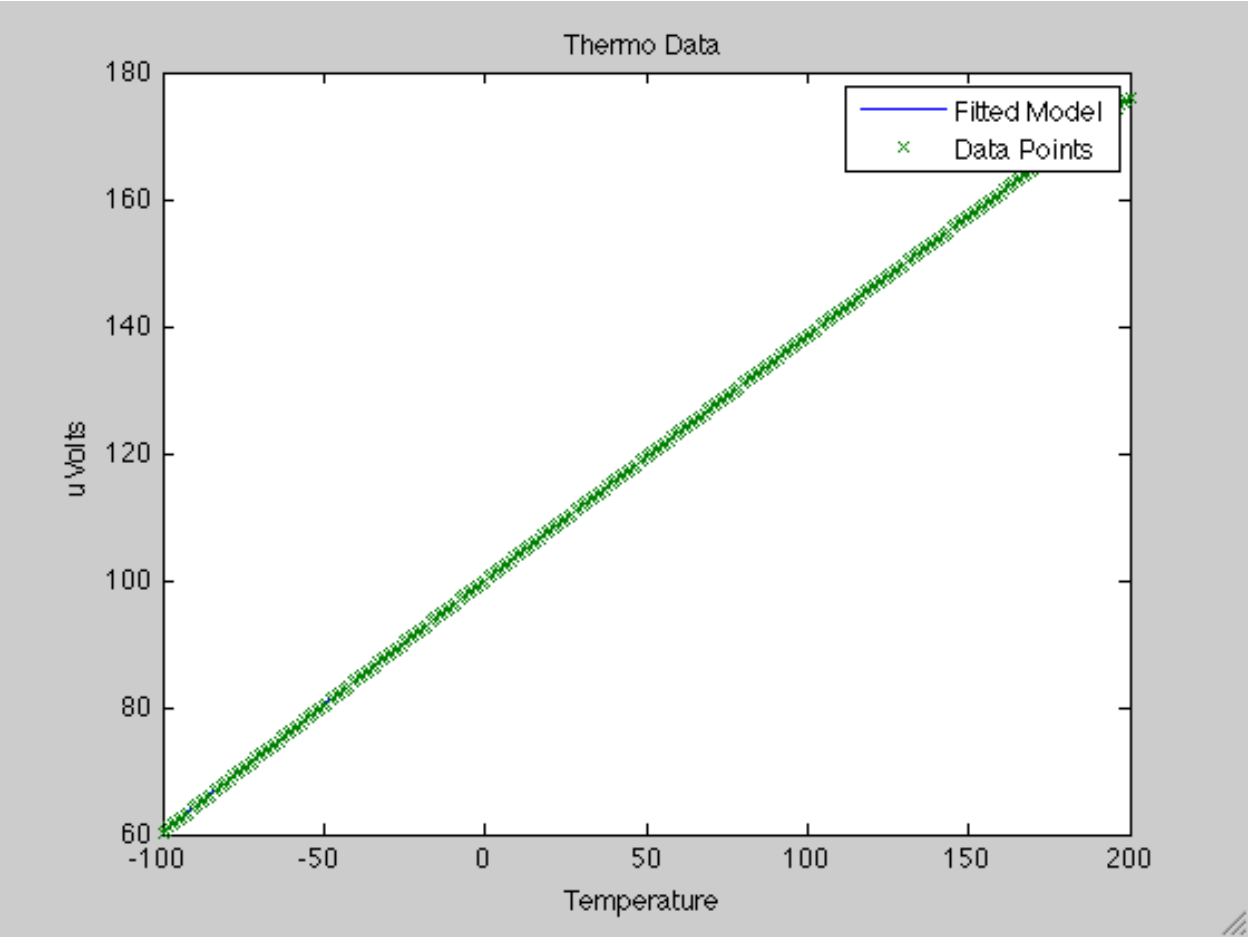
99.9983

0.3909

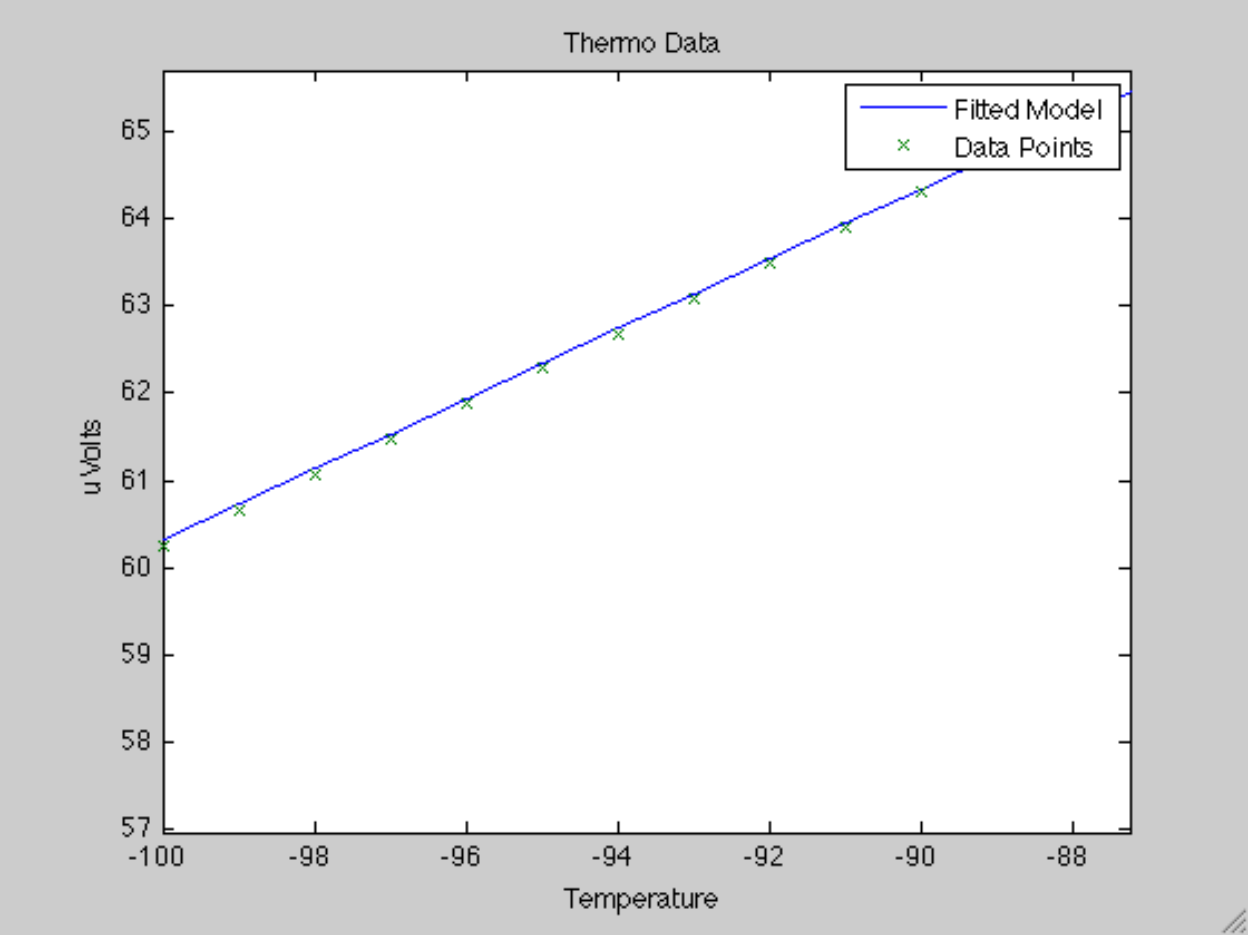
-0.0001

Standard Deviation:

0.0146



The graph below shows a closer look at the data above. It is clear that the line is not a perfect fit for the data. The model is much closer for this data set then for the first data set.



The following results from the PRT data for a cubic:

>> Vand(data,data1_100,4)

Coefficents:

99.9983

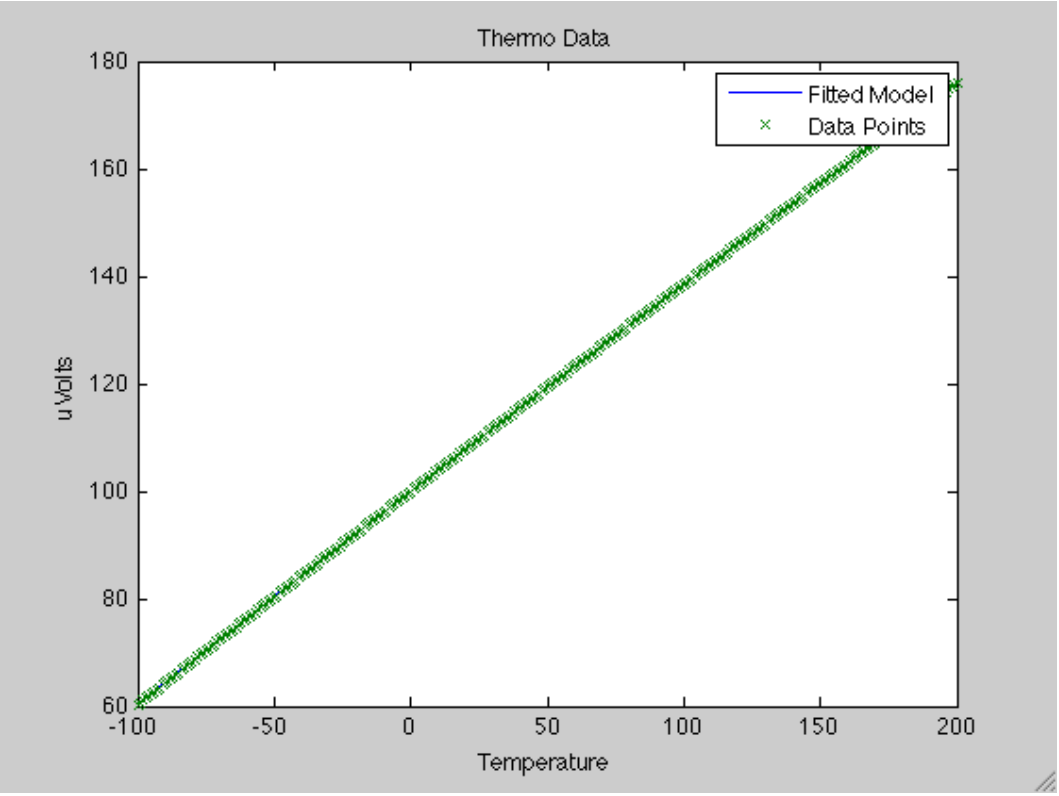
0.3909

-0.0001

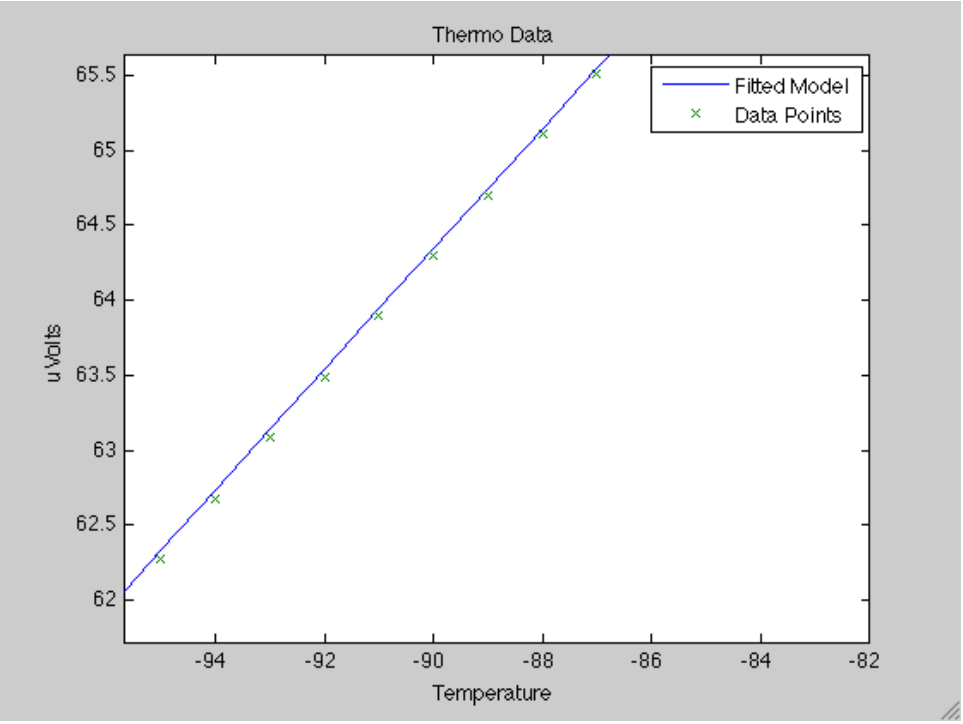
-0.0000

Standard Deviation:

0.0146



The graph below shows a closer look at the data above. It is clear that the line is not a perfect fit for the data. The model is much closer for this data set then for the first data set.



Part 2:

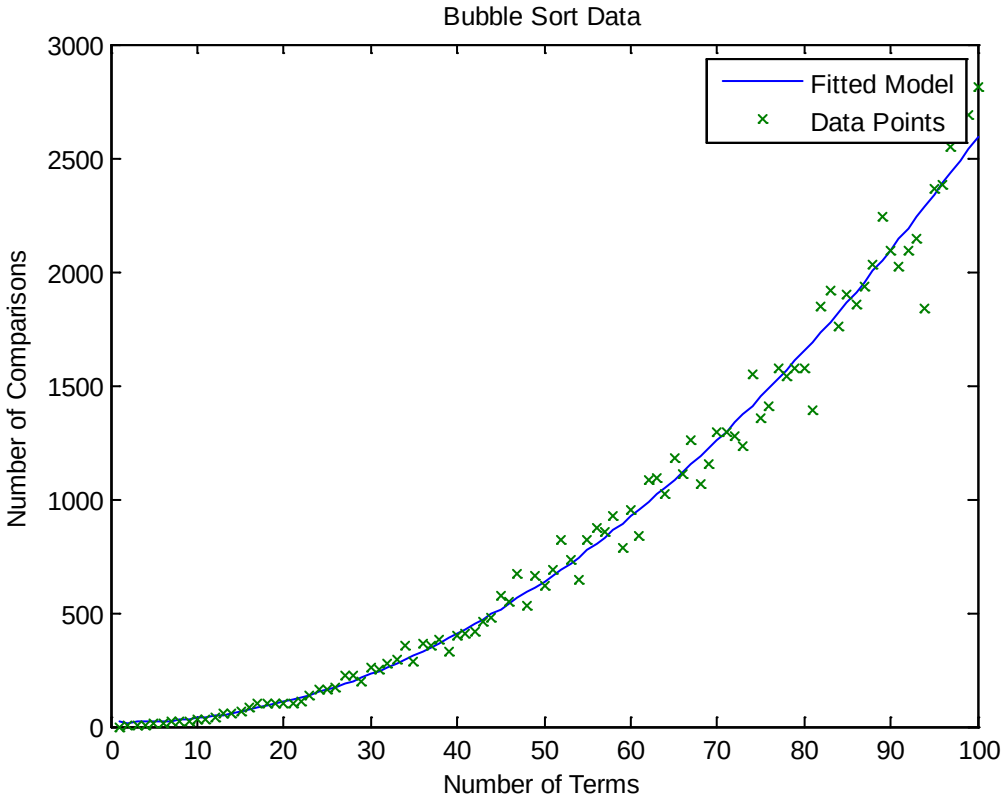
To create my sorting function I used for, while, and if-else statements and loops. I counted each time just before the if statement to get an accurate look at how many times the function compared two values.

If the random values happened to be in order already my function would perform 0 swaps. However, it would compare the terms N-1 times. This is because if the terms are already in the correct order the function moves on.

If the random values are in descending order prior to being sorted the number of comparisons would be (N^2-N) /2. As this formula suggests the data points gathered on the number of comparisons made from 1-100 terms in a vector can be modeled by a quadratic function.

The following script gathers the data mentioned above and then finds a quadratic model fit for the data.

```
>> bubble

Coefficents:

   18.3491

   -0.9371

    0.2667
```



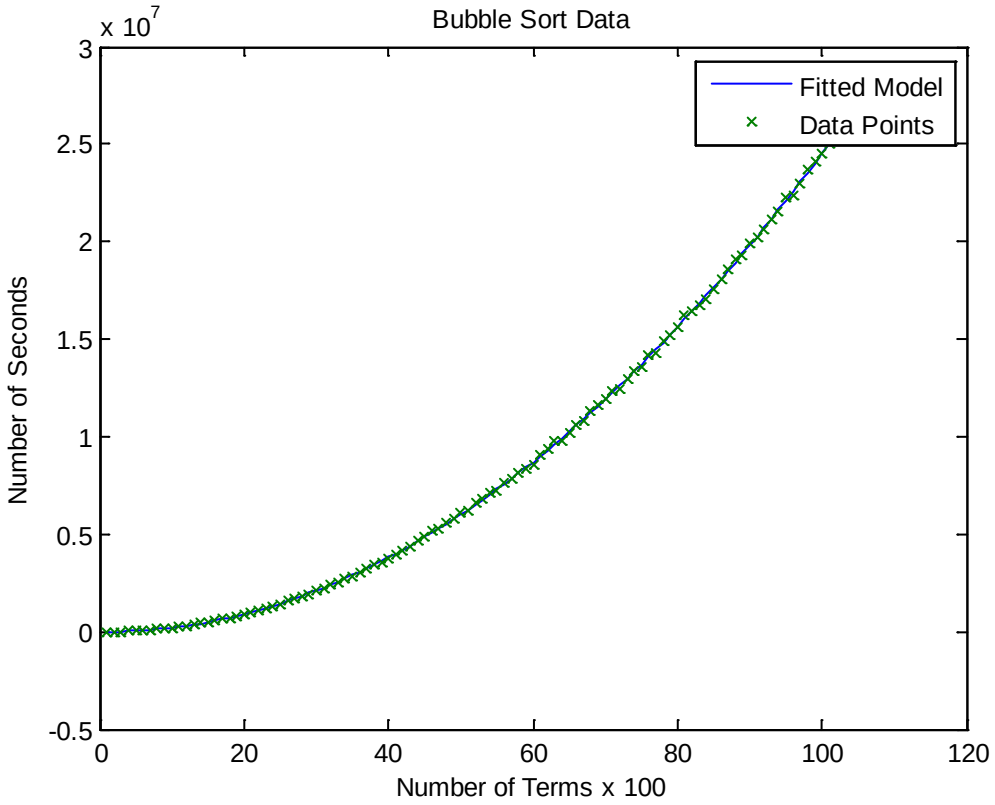
It is clear that the modeled fit is a good representation of the data gathered.

EXTRA CREDIT:

This next script runs the bubble_sort2 function and returns the time elapsed during the sorting process. This is done by placing a tic before the comparisons begin and a toc when the program is finished. Toc is then stored as the output of the function. Then a similar script to the one used to graph the data above is used on this new function. The data is gathered on much larger numbers of terms to get a more accurate look at the timing. The result is also a quadratic function as pictured below.

```
>> bubble2

Coefficents:

  1.0e+003 *


   -7.7282

   -4.1503
```

2.4916



The data is an even closer fit than that of the number of comparisons v. number of terms.