Christopher Schretzmann

Plot Salt Smooth

For this portion of the project, I decided to use the function x^2 + 13.

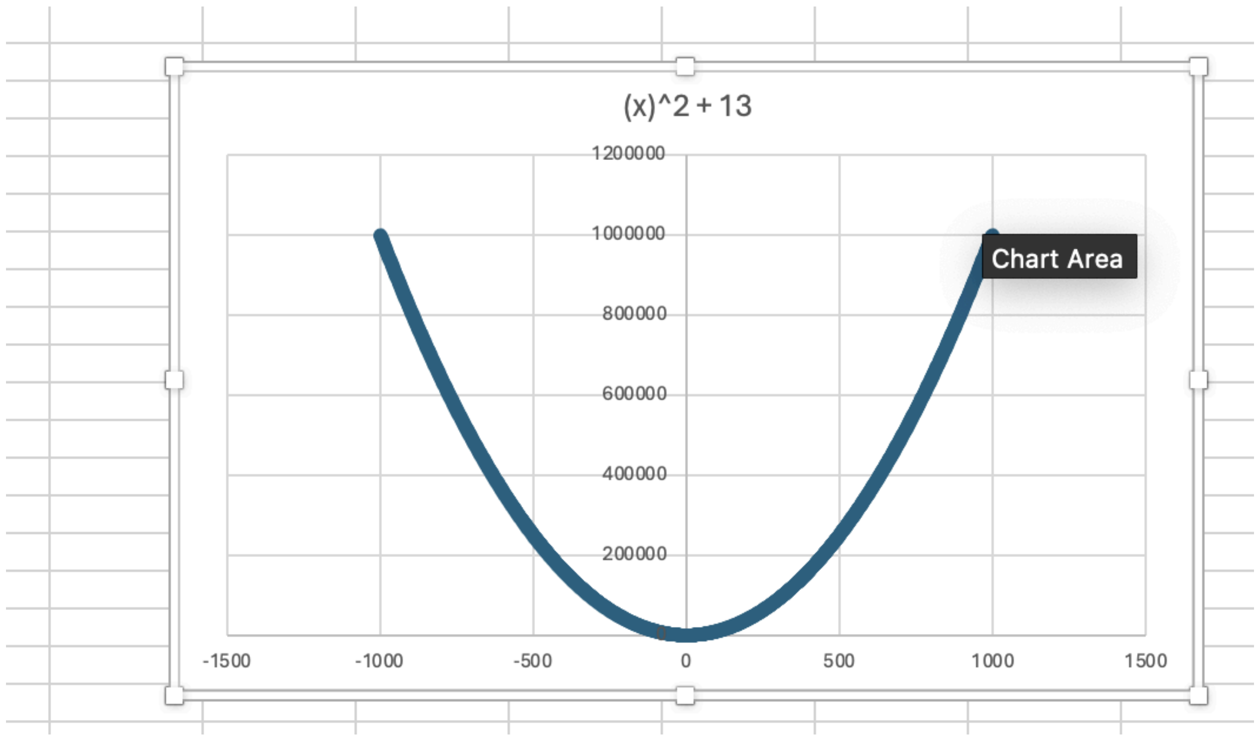
It must be noted that the zoom level, resolution of the monitor, and amount of points change the outcomes of all of these tests greatly.

Plotting Initial Function

For my initial part, I have a program that plots a function and assigns it to a hash map for a given range. For the purpose of this, my bounds are -1000, 1000 so you can see the parabola from both sides of the y-axis. This hash map is then output into a CSV file. Later in the project, I started using array lists but never changed the hashmap because it truly makes no difference for this implementation.

My graph below is the CSV file read in Excel and plotted.

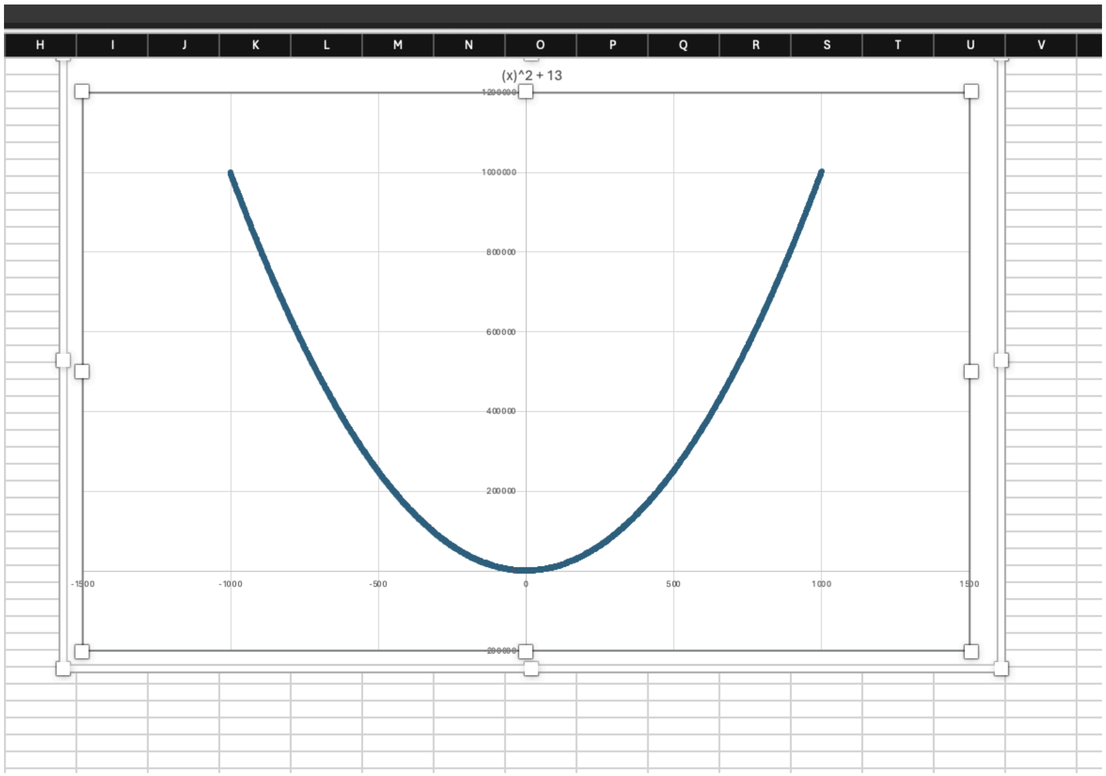
Initially -1000 - 1000



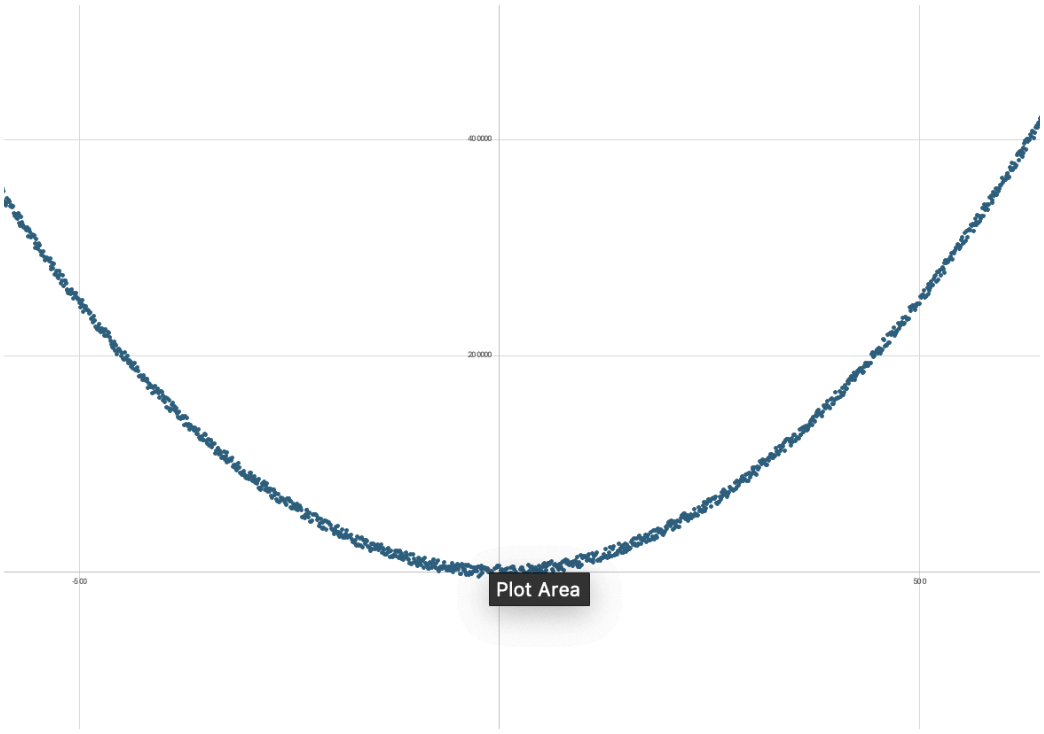
Salting is the act of randomizing data on a graph to make it more scattered and accurate. This program takes the CSV created in the last file and salts the data. The program has an assign value method. This stores the information from the CSV and puts the x and y axes into separate ArrayLists. Now that I have the initial data stored from the CSV, I have a salt function. This takes a y value for the data and min/max parameters. Using the Math. random function, I generate a random number within the given range. Since the salting has PLUS OR MINUS, I created another random variable that randomizes two numbers and if it is 1, add the value and if it is zero, subtract the value. This program does not store the salted data in a data structure, it just outputs it into the CSV file. The print function was just to test the arrays.

For the first test, I made the sat value range the same as the points. From this zoomed out, it is hard to see, but there is not that much variance.

Salt Value -1000, 1000

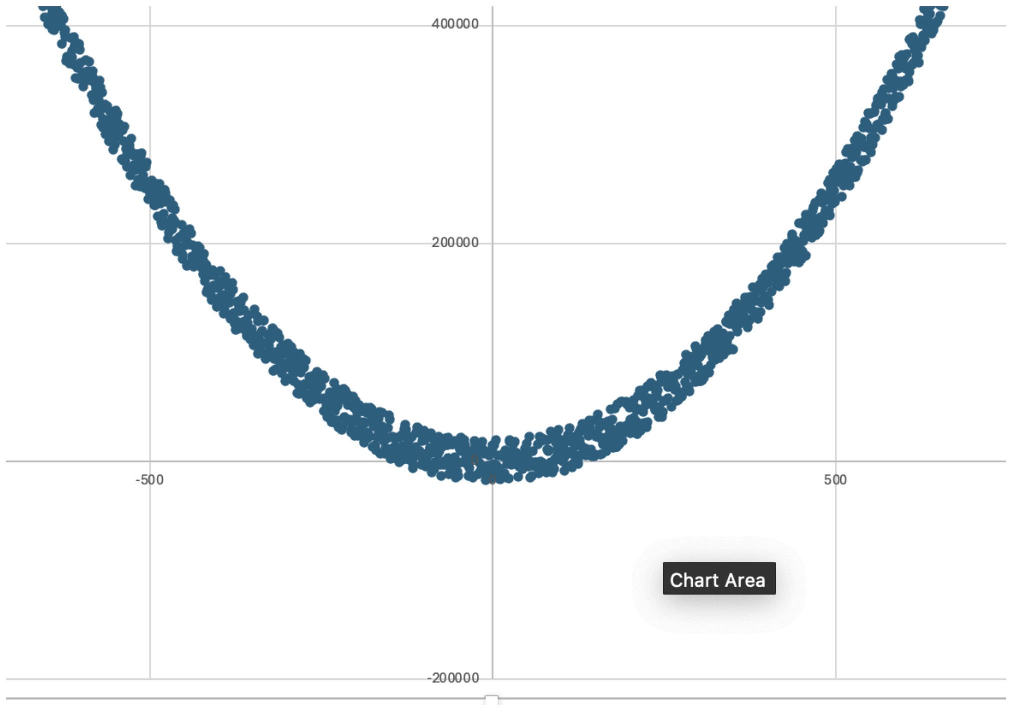


For the range of -5000 - 5000, It still looks relatively smooth until you zoom in.
Salt value -5000, 5000

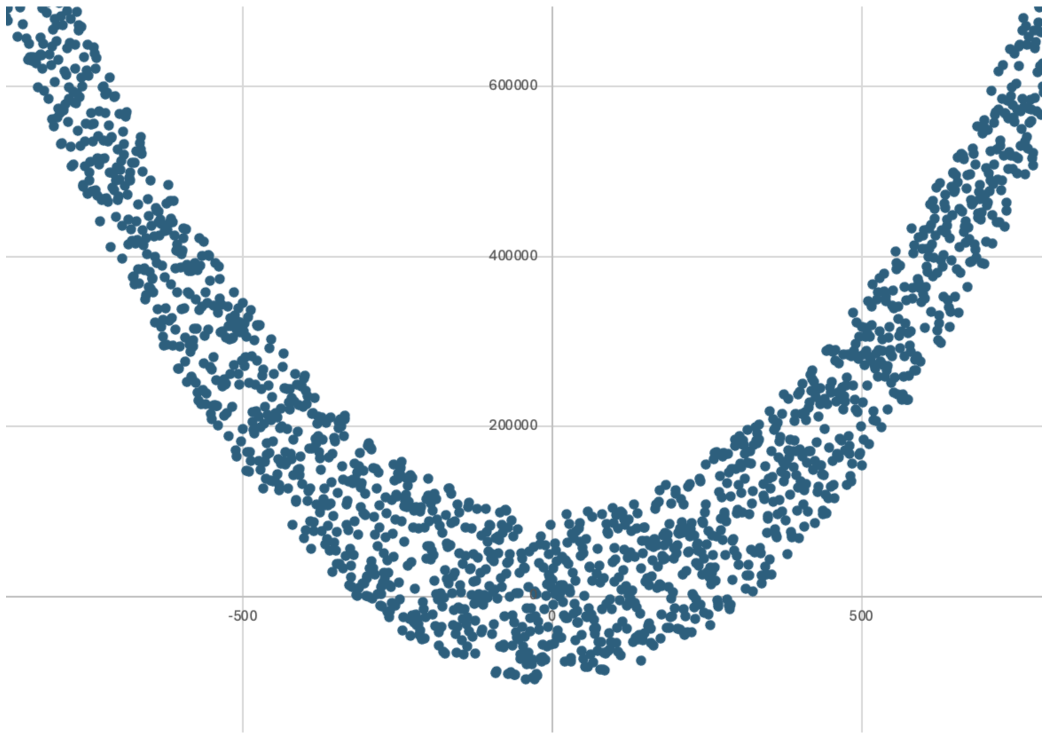


Since there is not that much difference in the salt for this many points and this function, I chose to put the range to -20000-20000. The line started to get thicker and if you zoom in, you can see the scattering of the points.

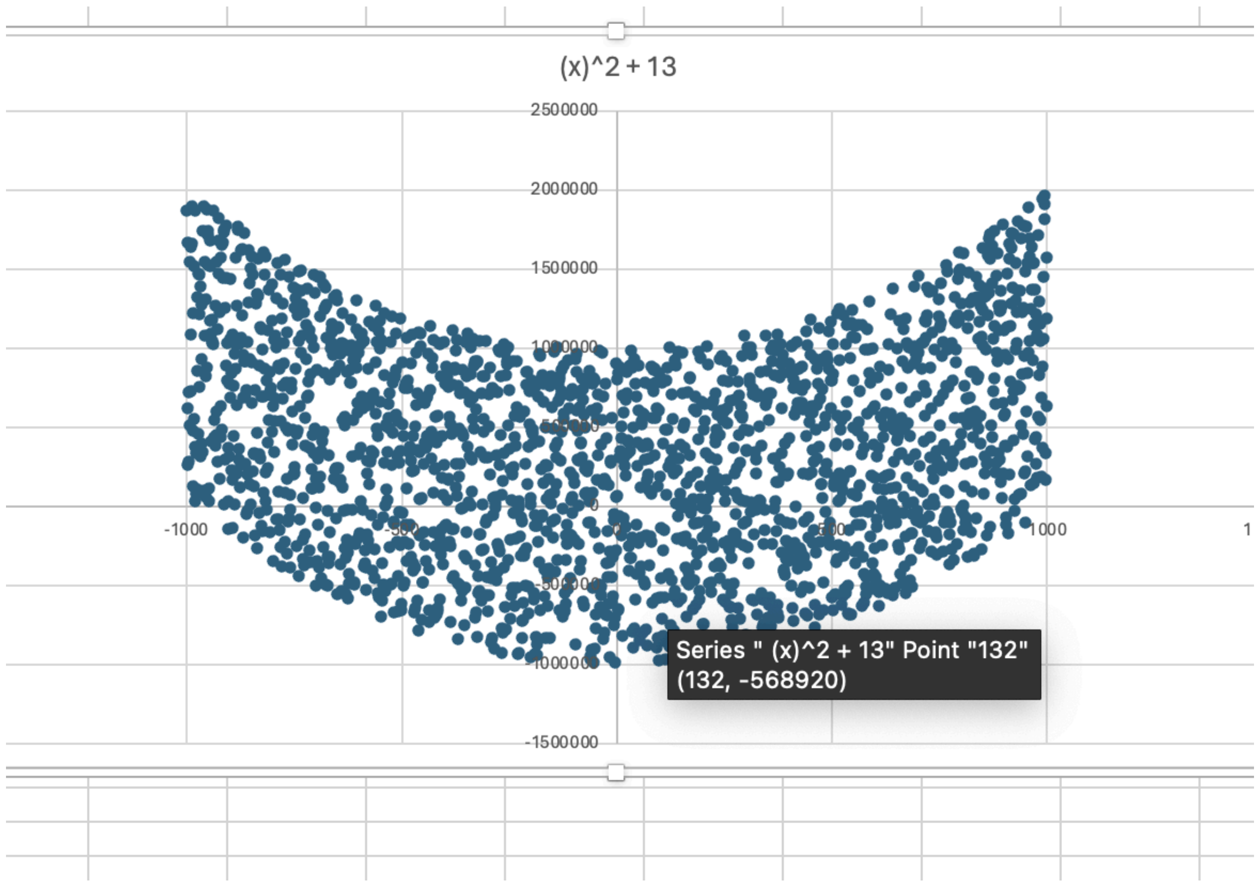
Salt Value -20000, 20000



I tested so many numbers and for the amount of points I have and the current resolution, the changes are minimal so I started changing the range in larger increments. With this range, it is more obvious that the points are scattered.
Salt Value -100000, 100000



To go very extreme I used the range -1000000-1000000. As you can see, the amount of random values makes the plot very scattered. For this amount of points and current zoom level, there is a visible behavior of every salt value.
Salt value -1000000, 1000000



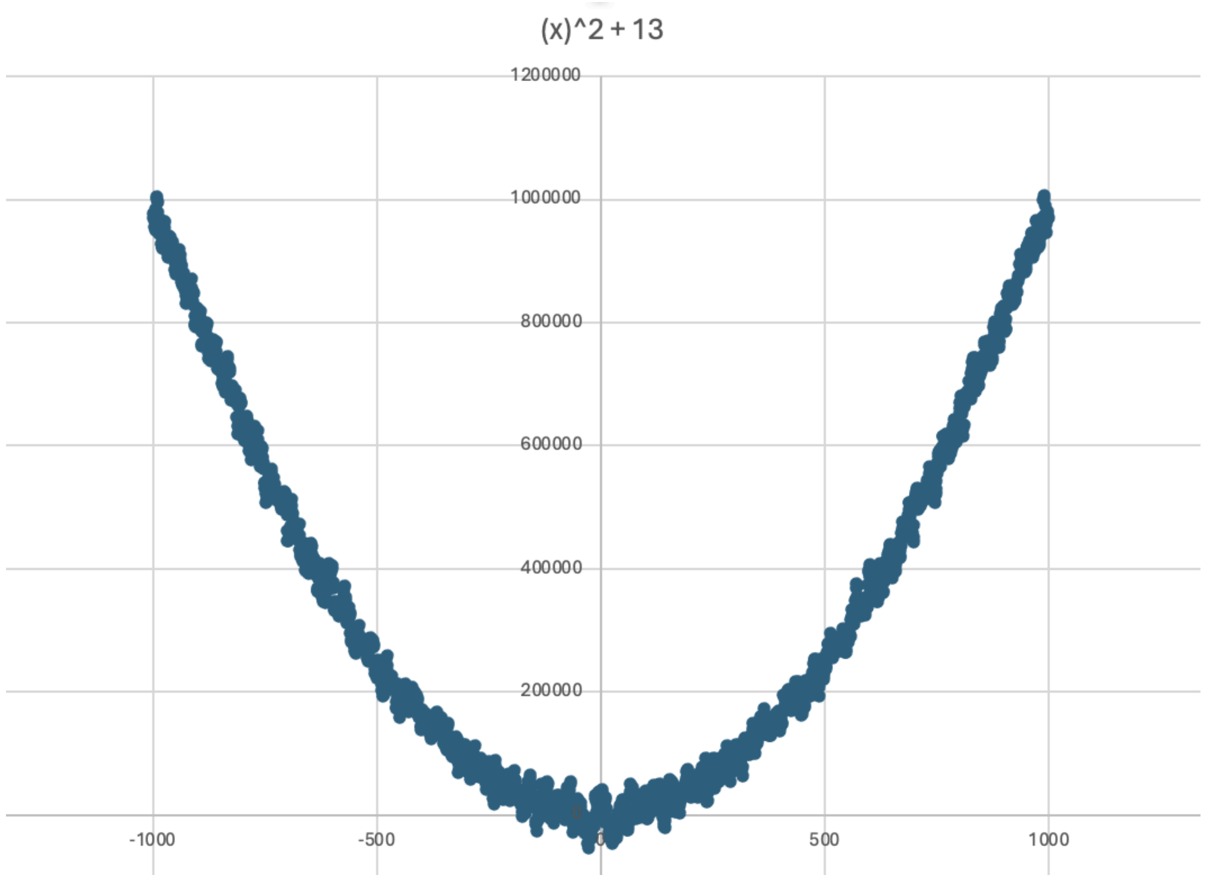
Smoothing

For smoothing, I am using the data from the salting range of -100000-100000

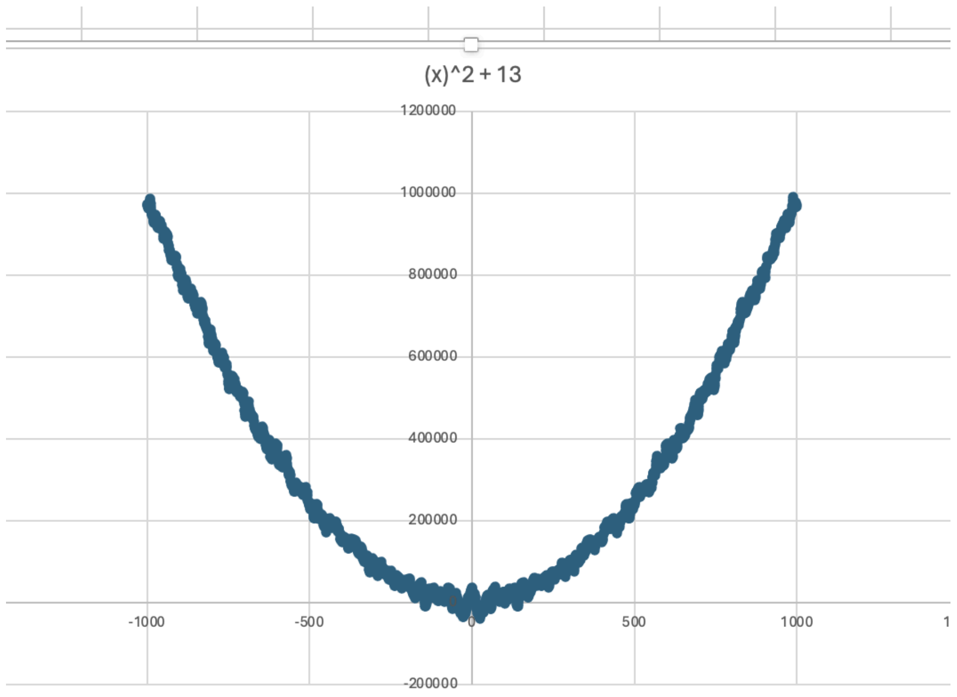
What if we want to turn the salted function back to normal? Is that possible? The smoothing program takes the salted data as a CSV and attempts to smooth it back to the original function. Just like the other programs, there is an assignValue method that assigns the data to an array list in this case. The smoothing function takes the average of the specified window and assigns that to the point of the new data. Initially, I was taking the average for the window slightly wrong and not initializing my counter to zero before every pass because it was in the wrong loop. After fixing the bug that gave my data weird behavior, I fixed it. After the salting of the original data, it exports it to a CSV just like the other programs above.

For the initial smoothing value, I input 5. This means for each point, it takes an average of 5 above and 5 below. With respect to the amount of salting, the bigger the salt value, the larger the smoothing value will have to be to smooth out that large variance. As you can see, the graph points are much less scattered or salted.

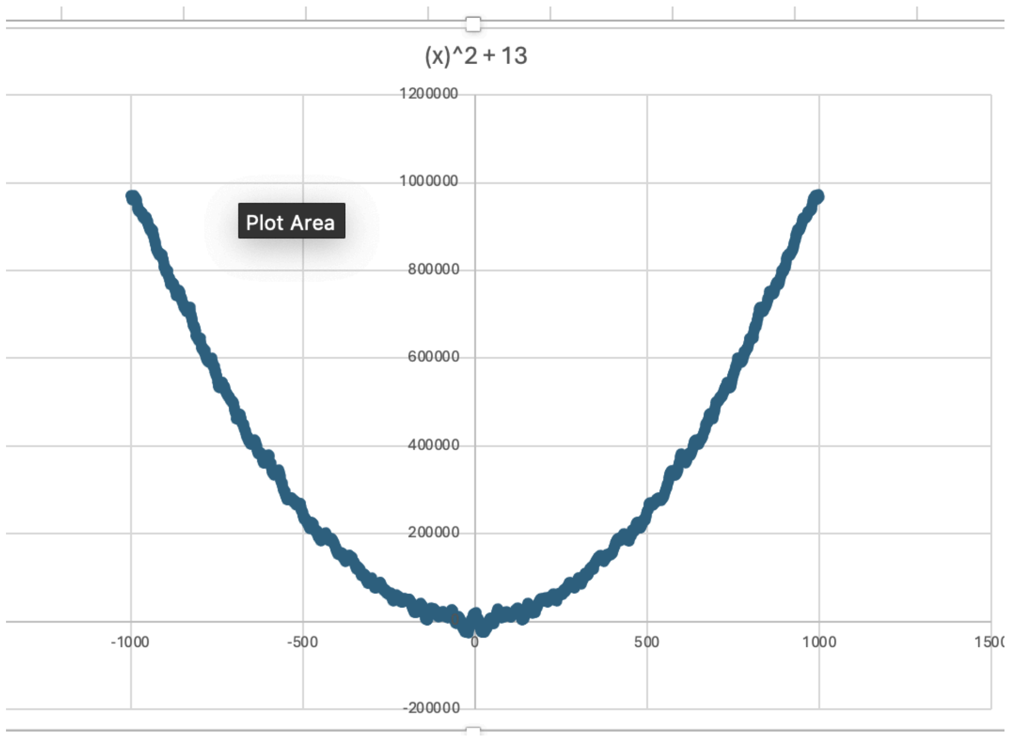
Smoothing with -100000, 100000
Smooth value 5



With the value of 10, you can see the graph doesn't get much smoother, but we see some improvement.
Smooth value 10

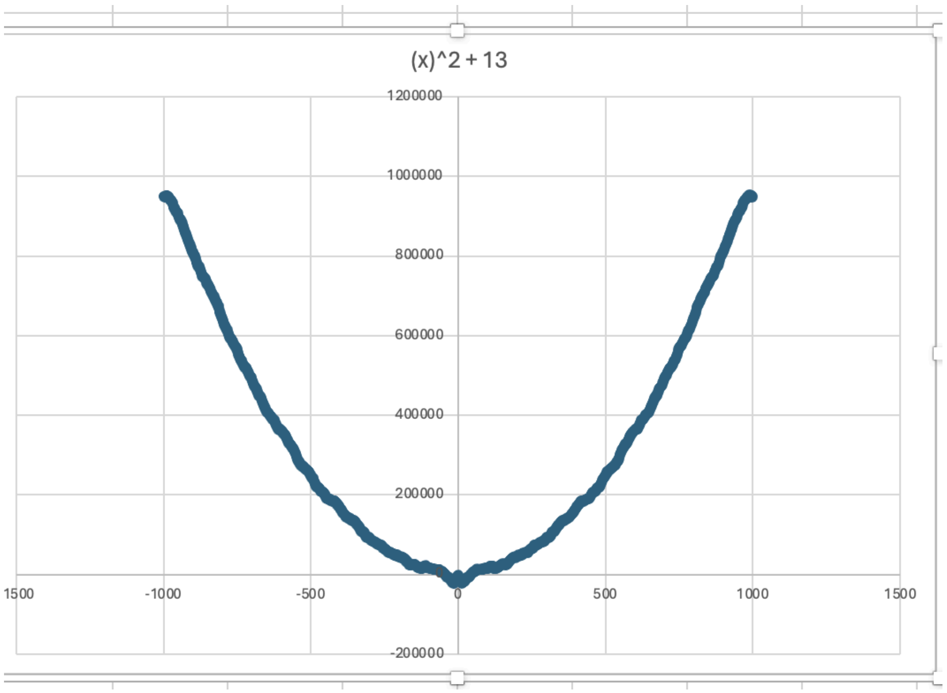


At the value of 20, the graph is even more smooth.
Smooth value 20

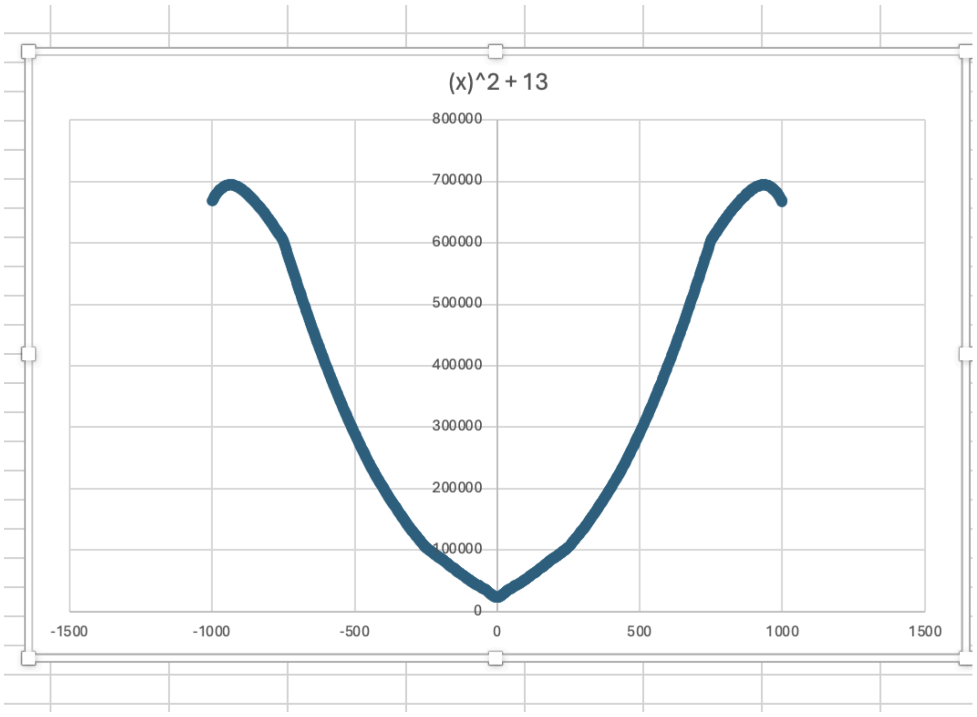


When the window is 50, the line is very smooth, but the end starts to have odd behavior.

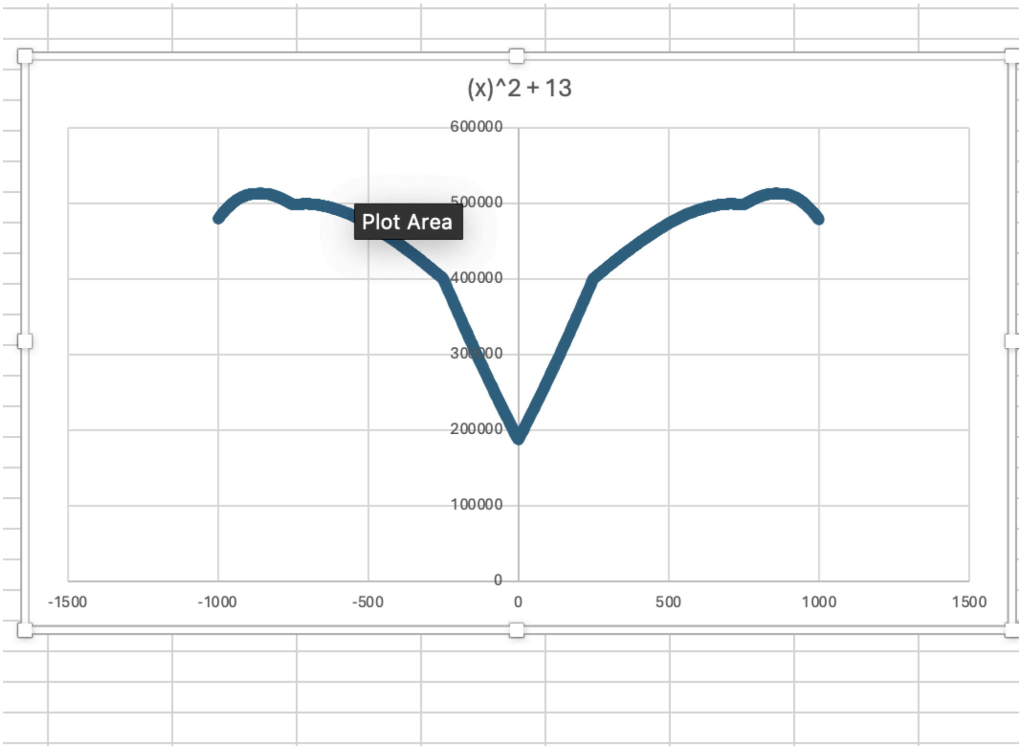
Smooth value 50



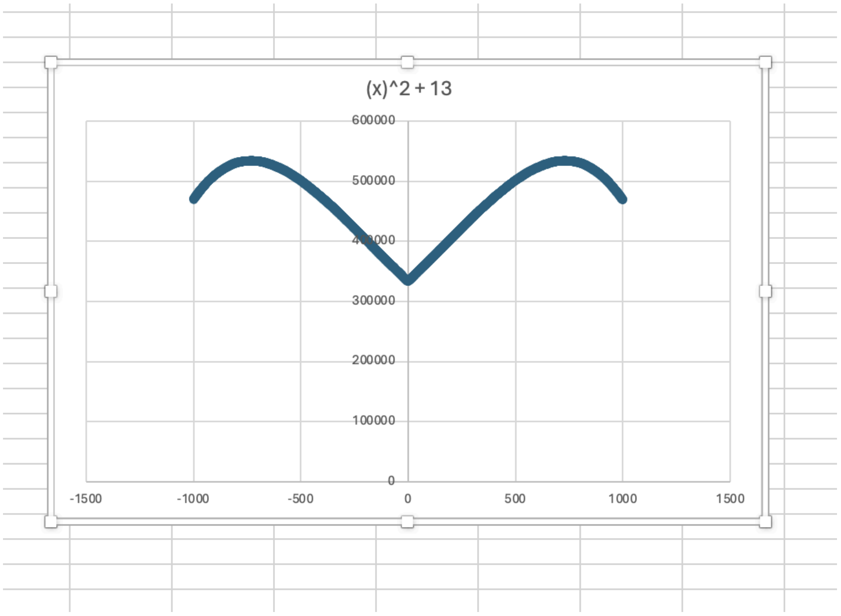
Just for fun, I kept increasing the smooth value. Based on the implementation of the algorithm I created, the endpoints definitely get weird.
Smooth value 500



Smooth value 1500


(x)^2 + 13
600000
500000
400000
300000
200000
100000
0
-1500
-1000
-500
0
500
1000
1500
Plot Area

Smooth 2000


(x)^2 + 13
600000
500000
400000
300000
200000
100000
0
-1500
-1000
-500
0
500
1000
1500