Task: get the b-a curve of values such that there exists the optimal period 4 sink somewhere near the location of the stable-unstable manifold tangency.

Given two starting b, and corresponding a for period 4 sink alog with the point p_xy, I am able to calculate all the remaining points on the curve b-a for the optimal period 4 sink, ie trace of DF^4(p_xy) is numerically zero (my average is around the order of 10^-9, however if I run the program slower to a higher accuracy, it can get to 10^-14).

My method works similar to the previous one I used for calculating the manifold tangencies, where I used a linear approximation of the previous two parameters to calculate the next initial guess. In the code, it looked like this:

```
slope_a = np.double((a_4sink[i-1] - a_4sink[i-2]) / (b_i[i-1] - b_i[i-2]))
slope_px = np.double((px_4sink[i-1] - px_4sink[i-2]) / (b_i[i-1] - b_i[i-2]))
slope_py = np.double((py_4sink[i-1] - py_4sink[i-2]) / (b_i[i-1] - b_i[i-2]))



a_guess = a_4sink[i-1] + slope_a*(b_i[i] - b_i[i-1])
px_guess = px_4sink[i-1] + slope_px*(b_i[i] - b_i[i-1])
py_guess = py_4sink[i-1] + slope_py*(b_i[i] - b_i[i-1])
```

The main difference in the program is that afterwards, I also take advantage of the fact we are working with a sink, and to improve accuracy, I use Newton's Method again with points px_better, py_better, which take the given p_x,p_y I obtained with the first root-solver, and then iterate with the Henon map 10,000 times by period 4, or 40,000 times total. This looks like this:

```
sink4_i = optimize.fsolve(F_sink,[a_guess,px_guess,py_guess,dx_4sink[i-1],dy_4sink[i-1]],xtol = 0.1)

better_p = NhMap_n(sink4_i[0],b_val,sink4_i[1],sink4_i[2],4,10000)

sink4_i = optimize.fsolve(F_sink,[sink4_i[0],better_p[0],better_p[1],sink4_i[3],sink4_i[4]],xtol = 0.1)
```

This method of correction increases the precision of the root-solver, which I verify by observing the relative error of the initial guess inputted into the Newton Method, and the outputted root. For the second newton method with the better p-values, the relative error is 1/10 or 1/100 of the first.

Furthermore, I verify the quality of my method by taking the final p-value, iterating it 10,000 times, and calculating the distance of the two values. It is never more than 10^-10. What's more, the trace of DF^4 of both values is not more than about 10^-9. Here is a sample result.

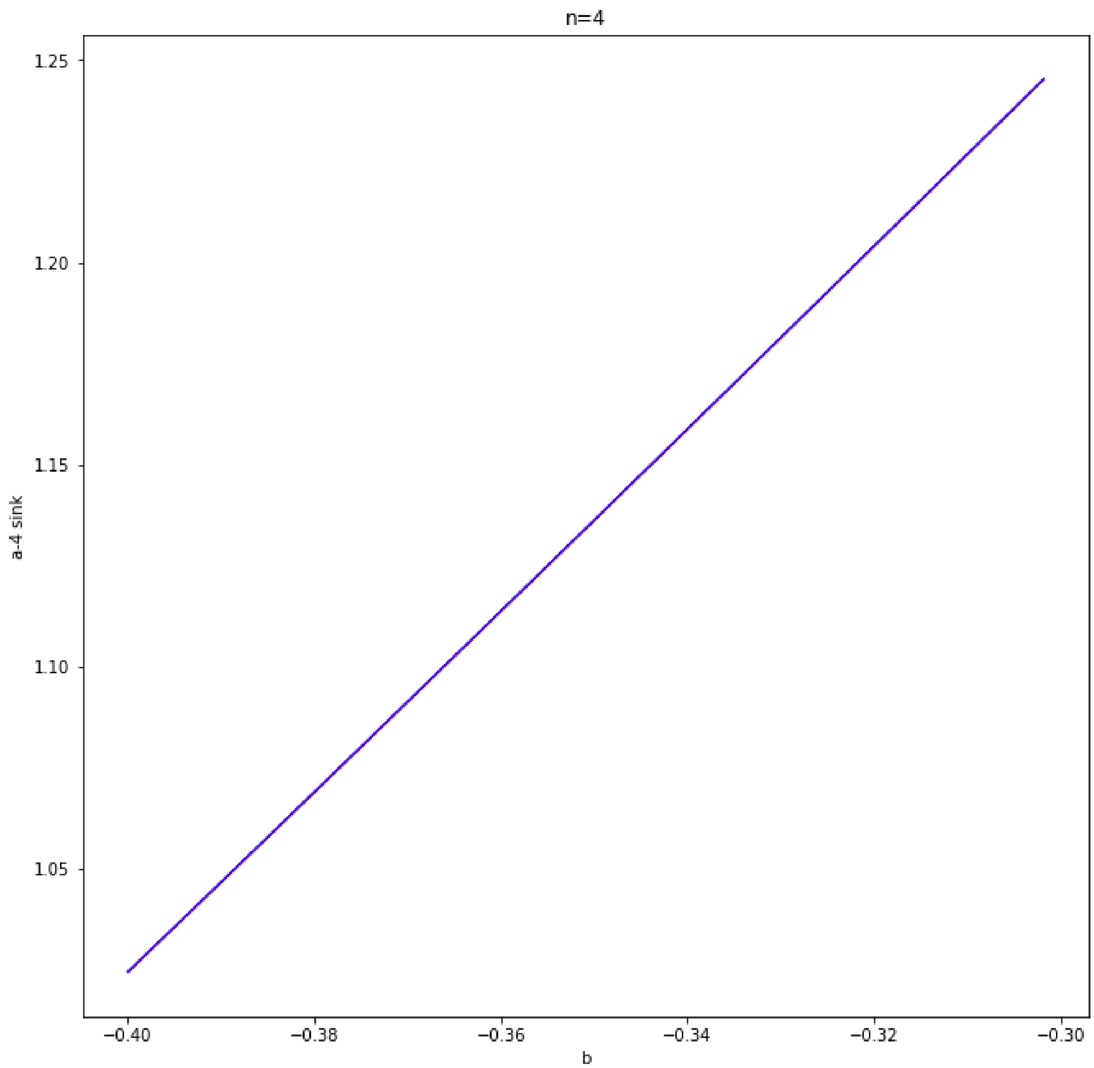
```
...
b:  -0.34979999999999994
a:  1.1366654951793924
dx:  -8744601338.01051
dy:  1523375629913.158
p:  1.3925187422889855 ,  -0.01305465693820011
4p:  (1.3925187422856329, -0.01305465690311669)
4*10,000p:  [1.3925187422868166, -0.013054656913321083]
^Trace DF4:  1.1212298346718885e-09
p-F Error:  [array([-3.34399175e-12]), array([3.50184708e-11]), 3.3526514897630477e-12, -3.5083418808978806e-11, -4.5
97945253703717e-10]
10,000 4p-Error:  [array([4.21884749e-15]), array([-2.15383267e-14]), 2.220446049250313e-16, -1.6653345369377348e-15,
1.1212298346718885e-09]
p sink diff error:  2.4973389749644544e-11
-------------------------------------------
```

The Error arrays above are from putting in the obtained root into F_sink, my error function I wanted to minimize in R^5.

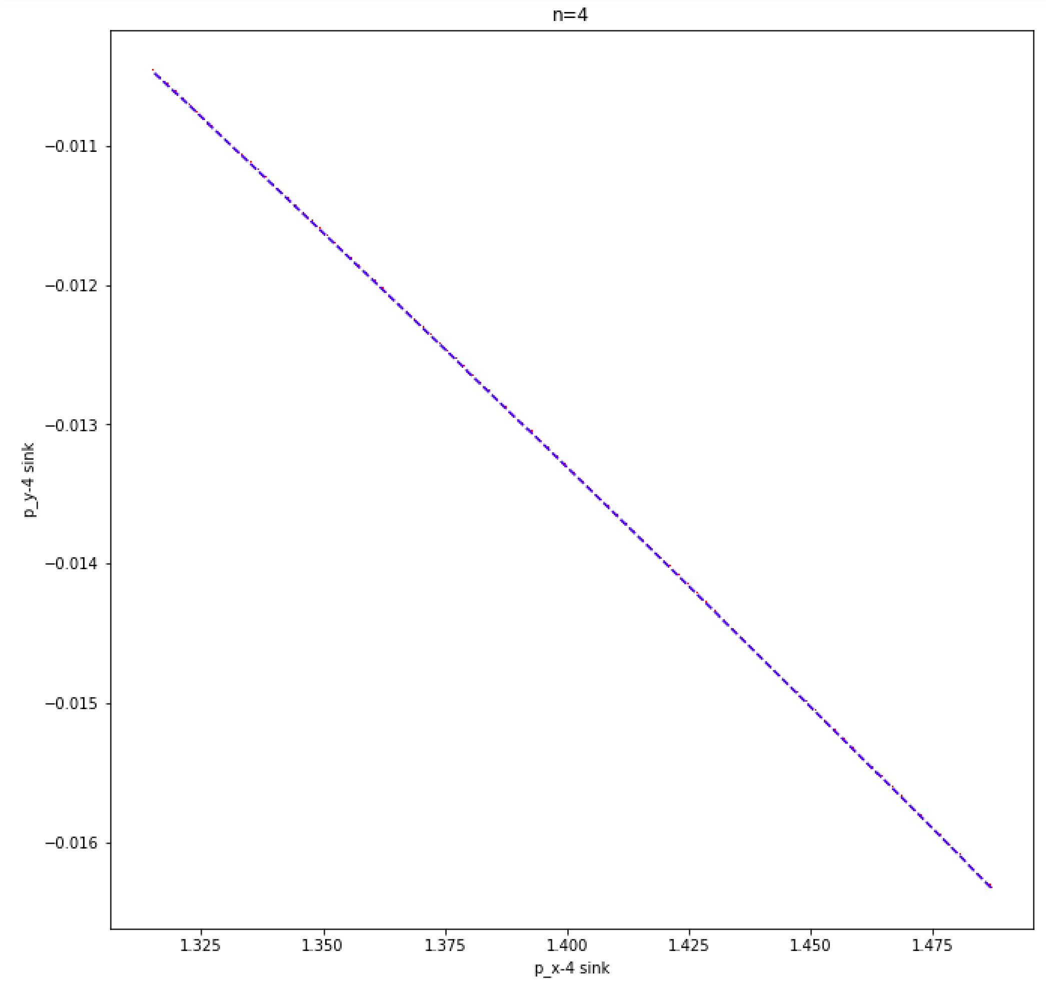
I also verified the points I was getting are in fact prime period 4. For some arbitrary point 1.3959993191801434 , -0.013172686175069824 with corresponding sink parameters (b,a) as (-0.35179999999999995, 1.13216811838981), the orbit of the point goes as follows:

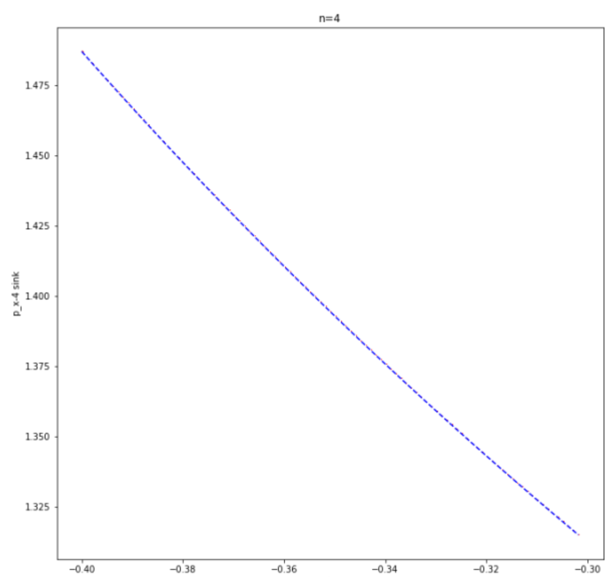
```
0 :  (1.3959993191801434, -0.013172686175069824)
1 :  (-1.2017510407314107, 1.3959993191801434)
2 :  (-1.1261960563352322, -1.2017510407314107)
3 :  (-0.013172686245485232, -1.1261960563352322)
4 :  (1.3959993191884605, -0.013172686245485232)
```

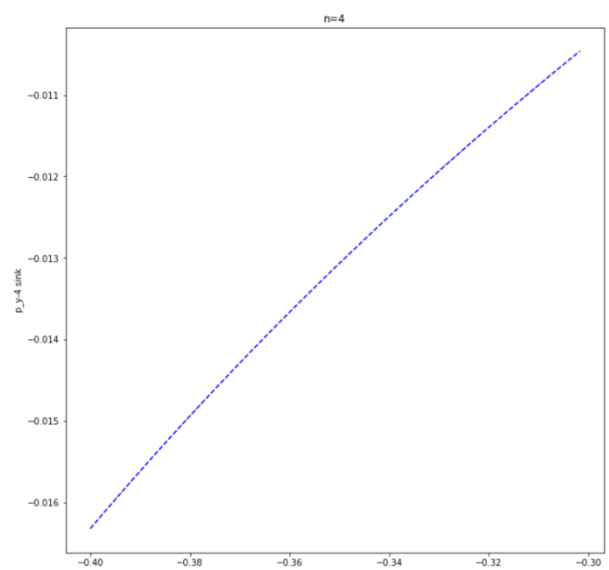
Now I will show graphs of my results. This is the b-a curve for the n=4 sink:



Here is how the period 4 point p_xy changes while varying b, along with the individual coordinates with b as an axis next.

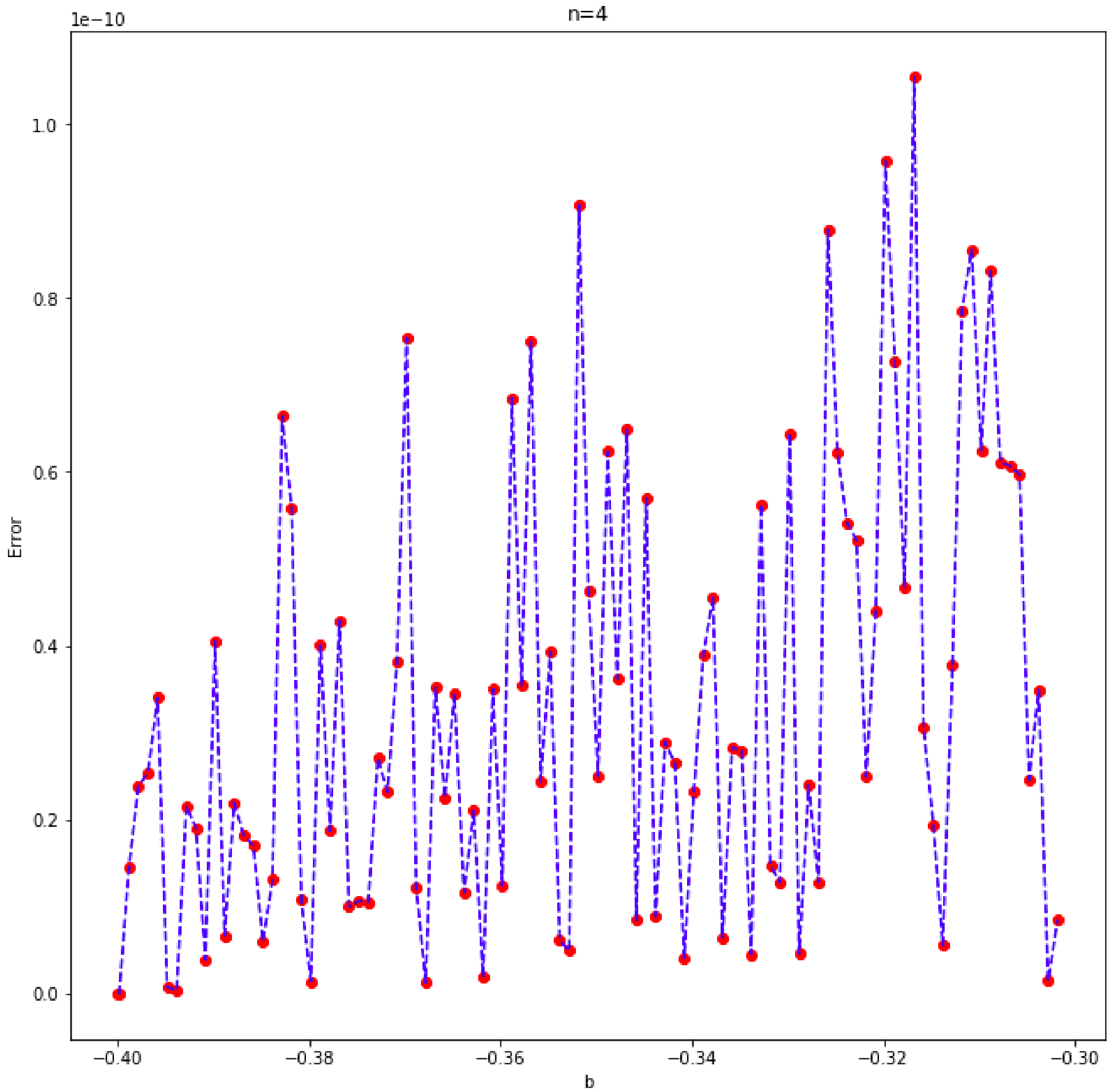




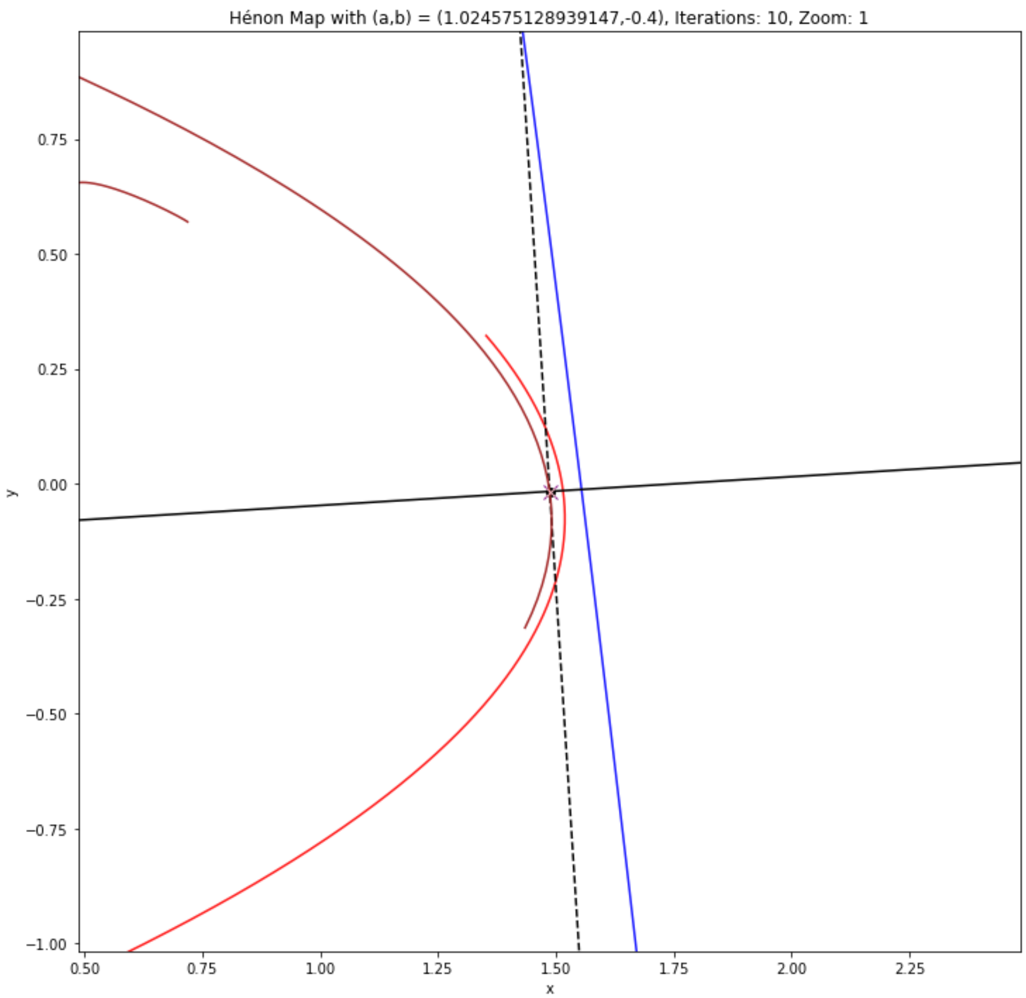


And here is the error I described above by calculating the difference between the final point and the 40,000th iterate of it. The average is on top.

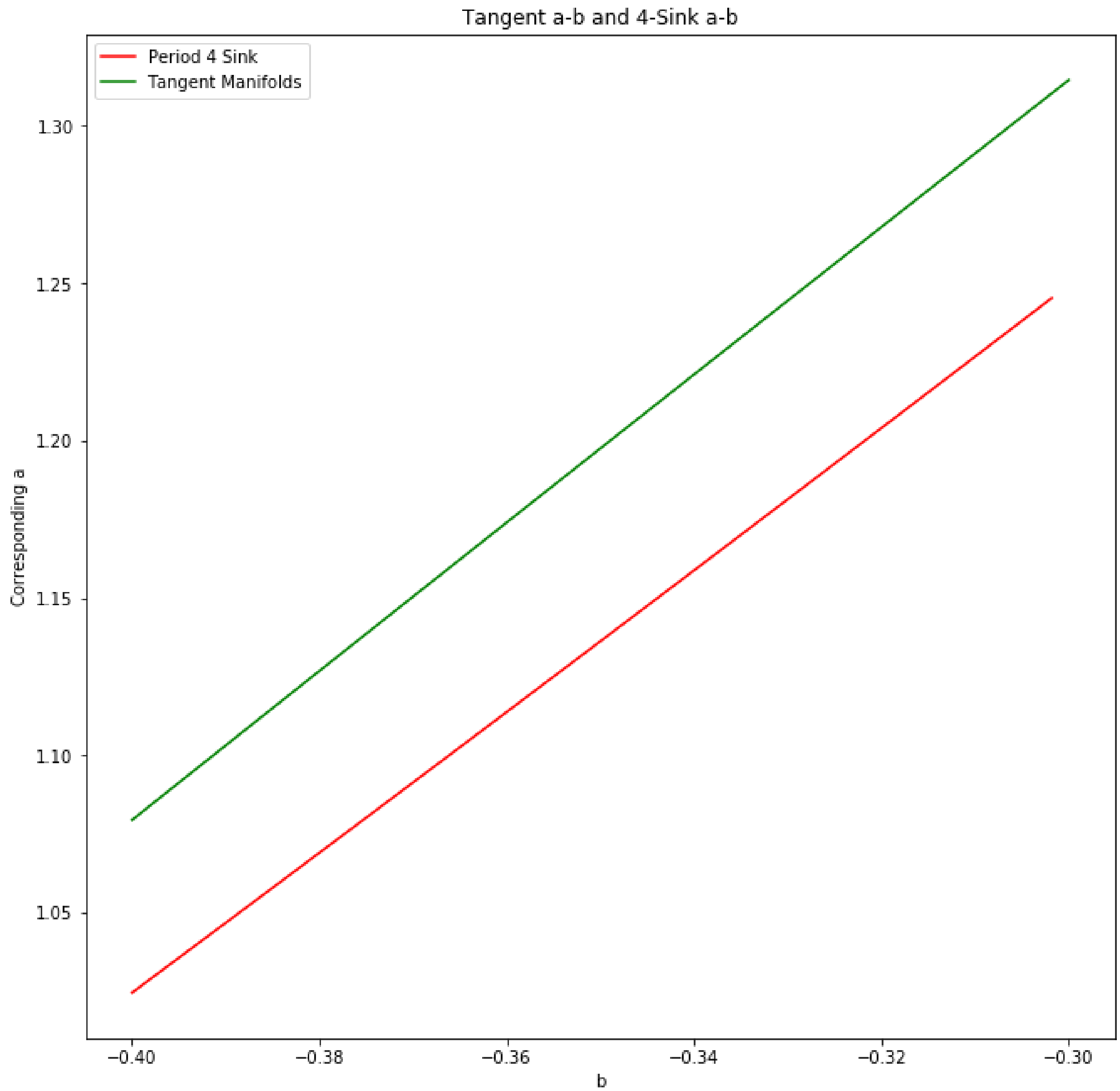
3.26081833322675e-11



Here is an image of what one of these points looks like geometrically, for b = -0.4, along with the stable manifold, the line with directions dx,dy obtained through the algorithm, the corresponding perpendicular line's image under the Henon 4-Map, and periodic point p itself.



Finally, I graph the tangent curve along with the period 4 sink.


Tangent a-b and 4-Sink a-b
Period 4 Sink
Tangent Manifolds
Corresponding a
1.30
1.25
1.20
1.15
1.10
1.05
−0.40
−0.38
−0.36
−0.34
−0.32
−0.30
b

I ran into a few problems while trying to do this, but was able to resolve them. First, there was a big issue with the point I obtained, p_xy, let's say 1.396 , -0.01317 converging to some fixed point around approximately (0.5, 0.5) after I iterated 40,000 times. This tended to happen if my a-value was too far off, which had to be resolved by taking the linear guess instead of the previous result, when my increase in b was too large. Furthermore, I found that using the p_better values and re-doing the Newton method helped too. Therefore for each iteration of b, I use root finding twice. However, with all the precautions I took to ensure accurate starting parameters, I was able to run the program to get the a values for b=[-0.4,-0.3] in about 5 minutes.

Next, I am planning to do further work in the other tasks you've asked me to do, such as make an algorithm to find the most optimal starting guess a for an arbitrary period n. I will try this in the method we've discussed, first by trying the smarter bisection search. Then will try to make my algorithm itself work for arbitrary n, for now I've only got it coded for n=4, but that shouldn't be a difficult thing to change.

Thank you for your time.