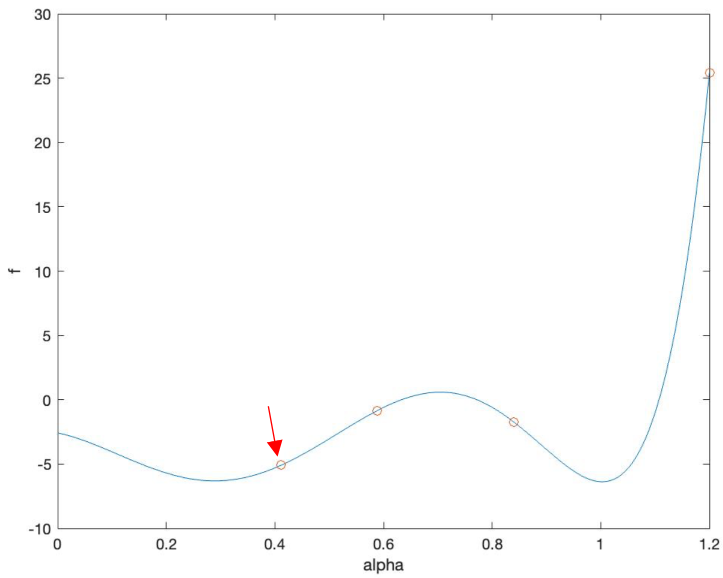
Jacob Snyder

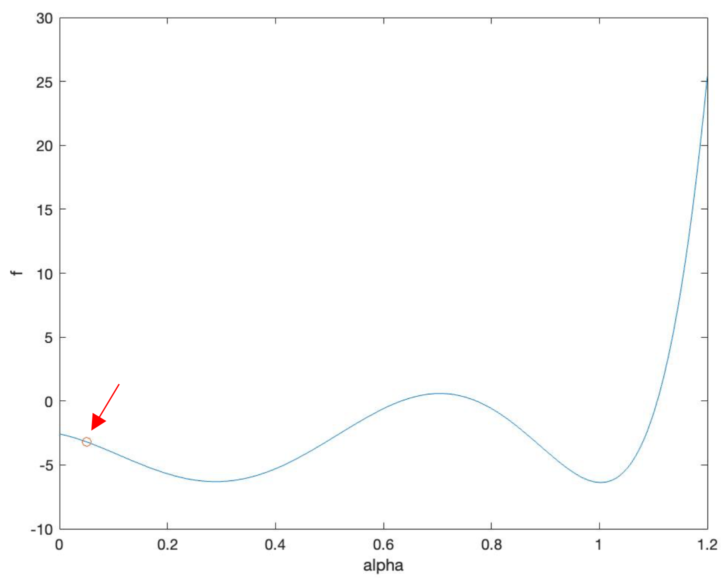
ACM213 Problem Set 1

red arrows are used throughout to indicate the final iteration point

1.a) Classical backtracking line searches

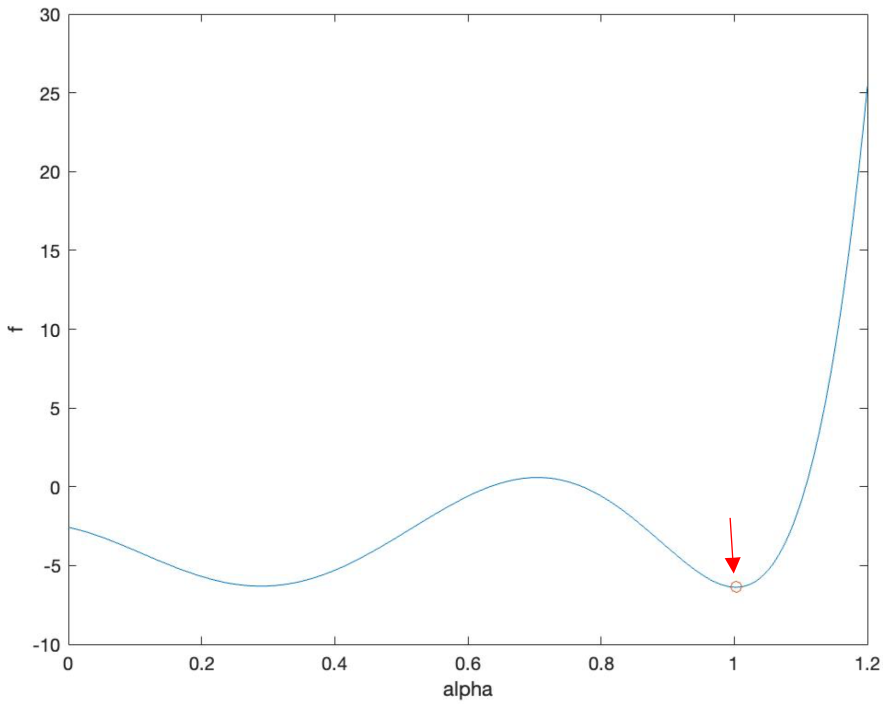


The starting point is on the far right, and the ending point is on the far left. Alpha_0 = 1.2

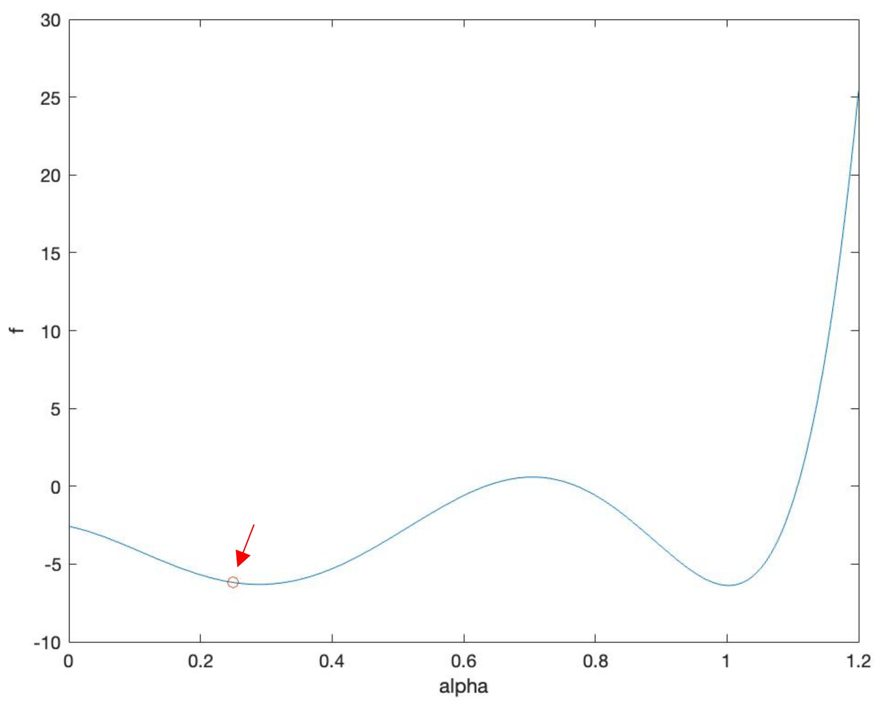


The only point is both the starting and ending point. Alpha_0 = 0.05

1.b) Bracketing and Pinpointing



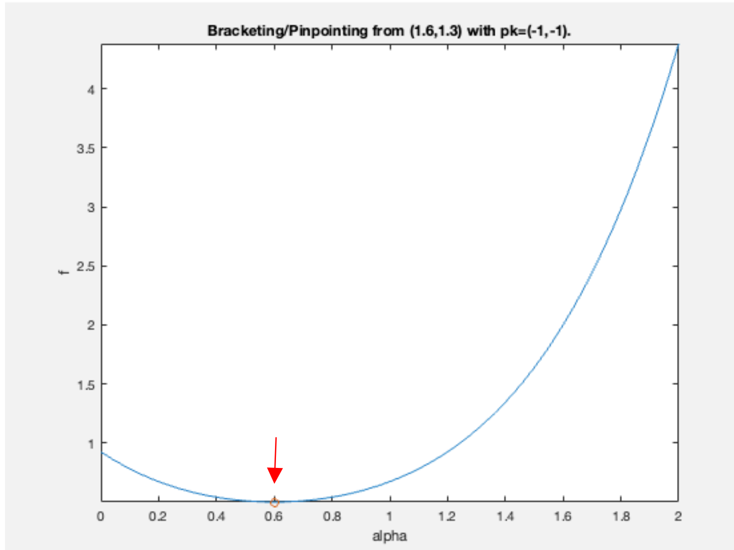
Mu1 = 10^-4, mu2 = 10^-3, sigma = 1.5



Mu1 = 0.5, mu2 = 0.8, sigma = 2

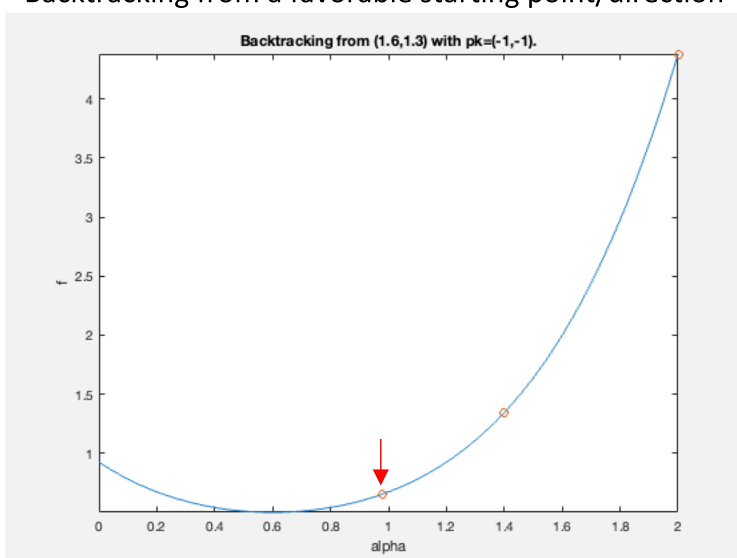
1.c)

Bracketing/Pinpointing from a favorable starting point/direction



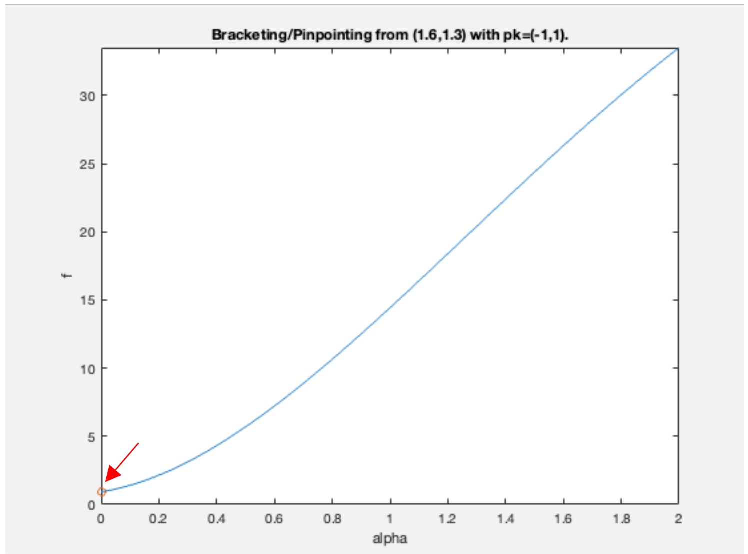
Considered 9 brackets, 5 alphas considered within the bracket

Backtracking from a favorable starting point/direction



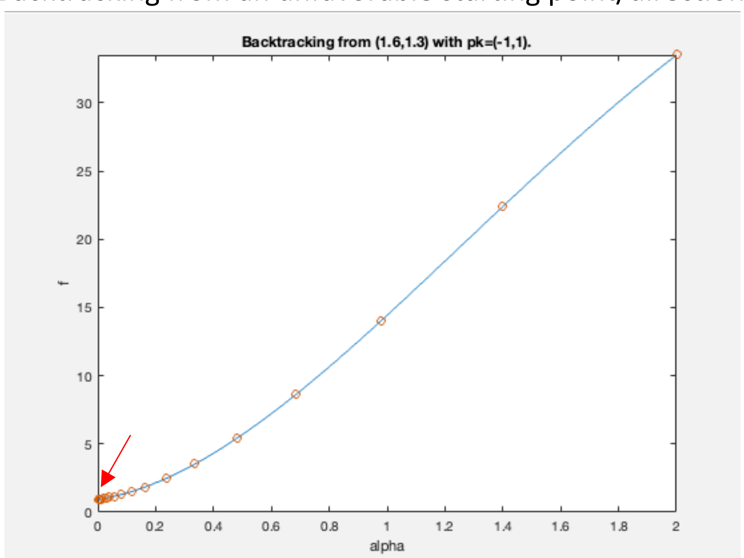
3 iterations

Bracketing/Pinpointing from unfavorable starting point/direction



2 brackets considered, 8 alphas considered within the bracket

Backtracking from an unfavorable starting point/direction



100 iterations (cutoff)

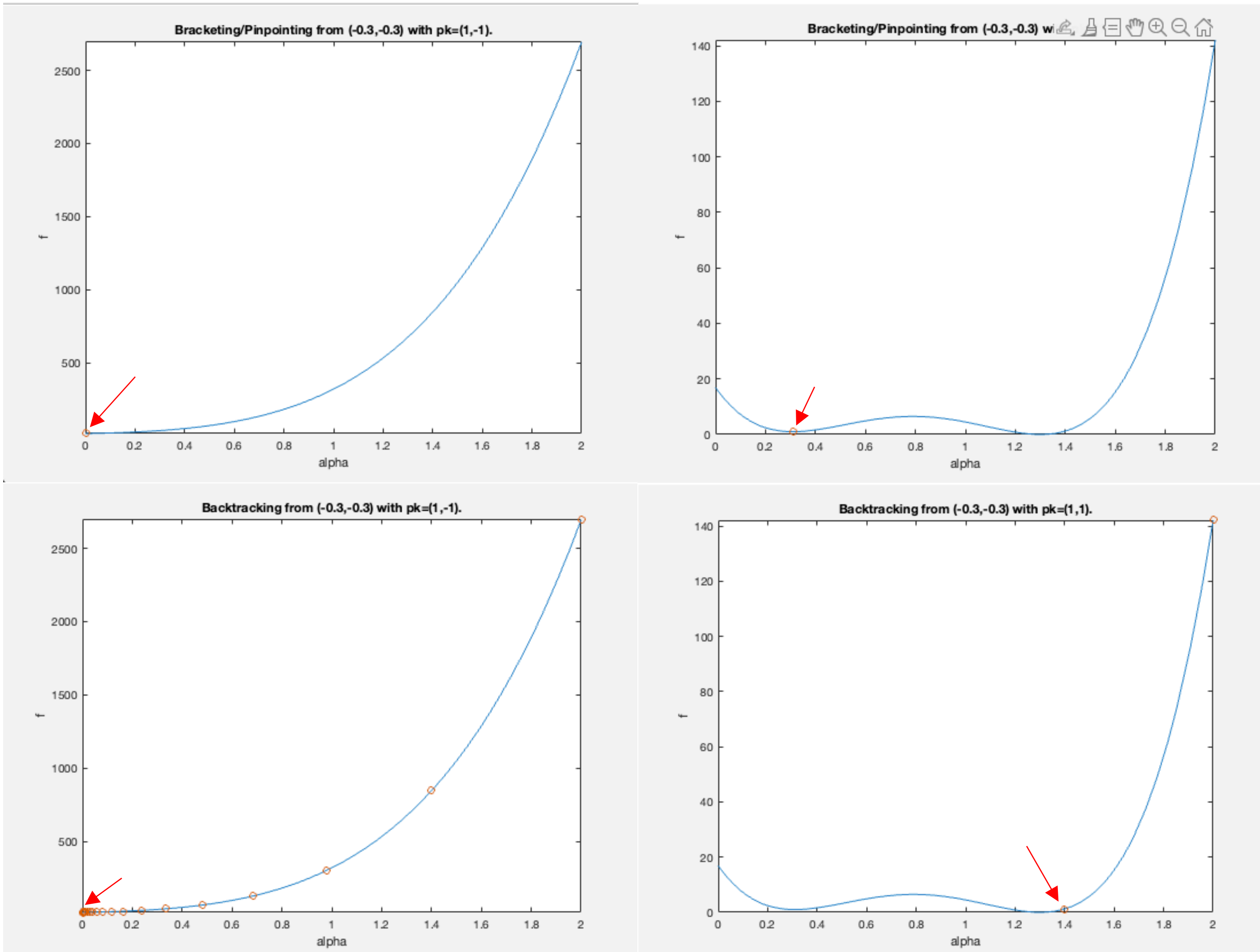
We can see that the backtracking approach works well when the starting position and direction are well-calibrated, but it performs very poorly when the starting position and direction are not good.

In contrast, the bracketing/pinpointing approach handles both scenarios with a medium amount of computations (worse than good-backtracking but much better than bad-backtracking).

One might decide to use backtracking if they have very high confidence that their starting position/direction is accurate. They could instead use bracketing/pinpointing if they had low-medium confidence in the accuracy of the starting position/direction.

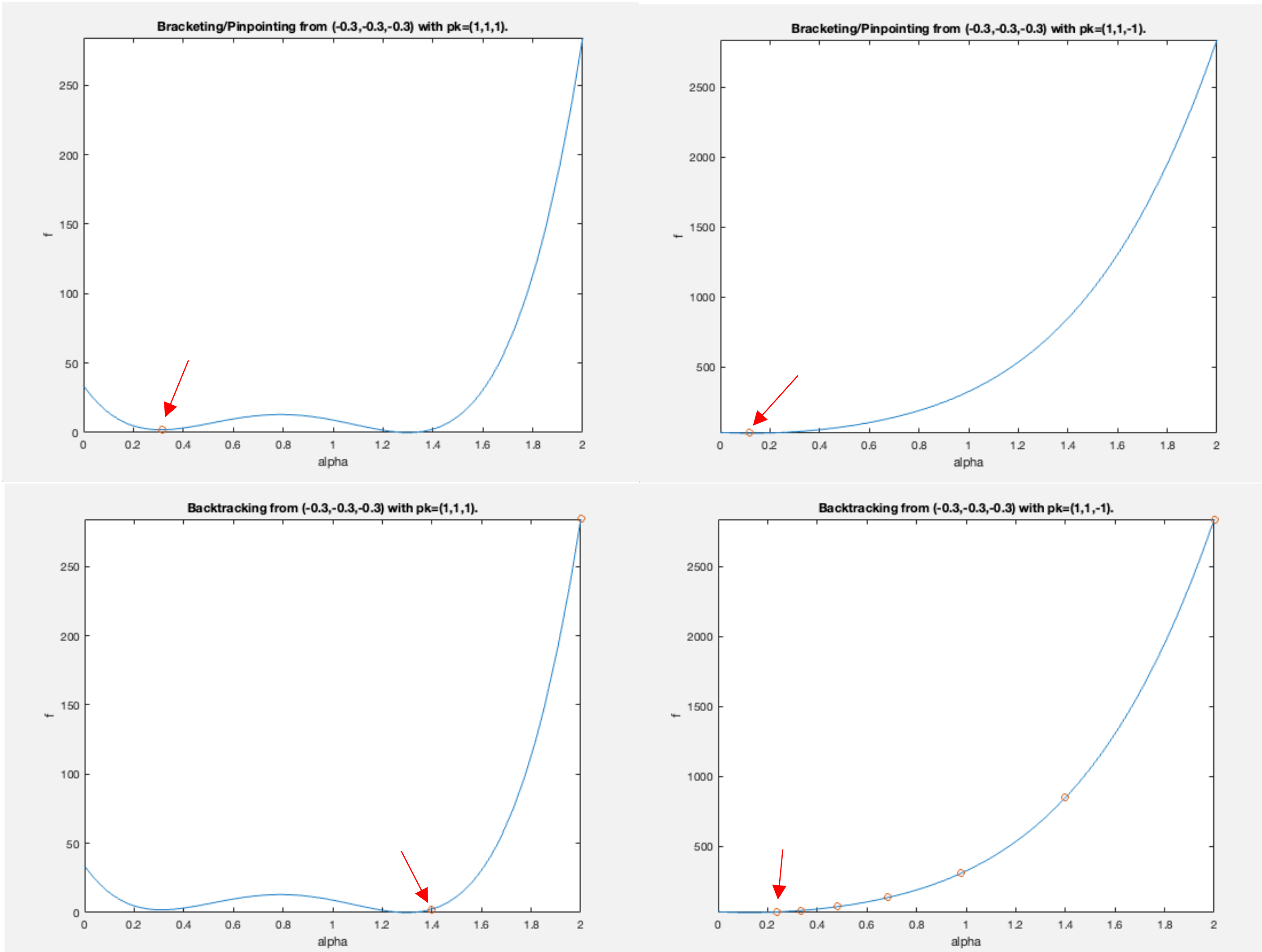
1.d)

n=2 Rosenbrock function



We observe the same results as in (1.c) when n=2 for the Rosenbrock function.

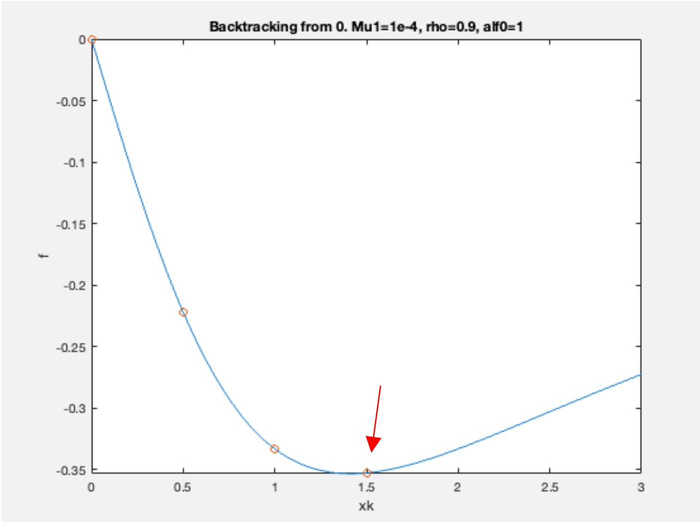
n=3 Rosenbrock function

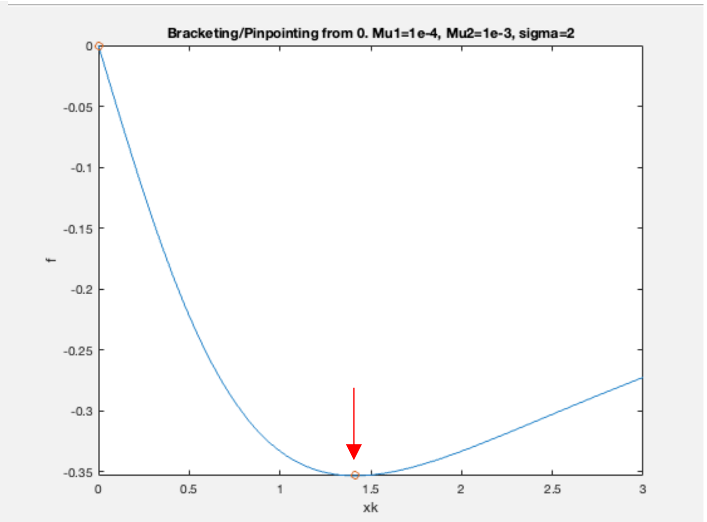


For n=3, we see the same results as we did as in n=2, where going in a bad direction seriously hinders the backtracking approach. The algorithms are robust for n>2.

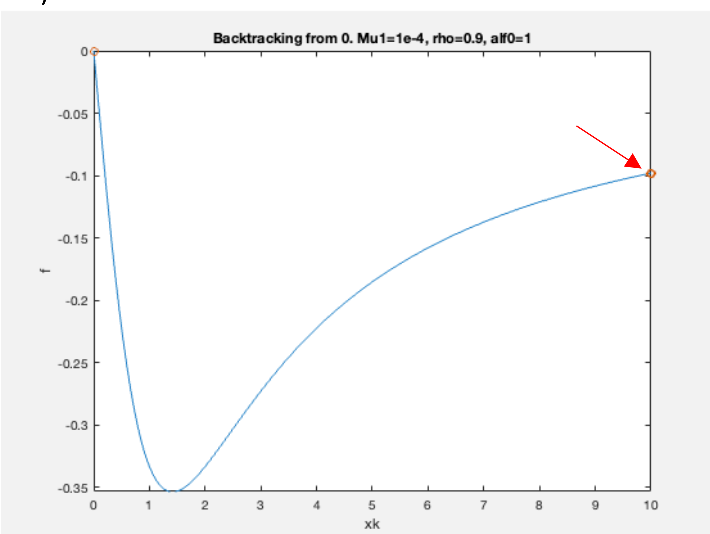
1.e) Time spent: roughly 5 hours

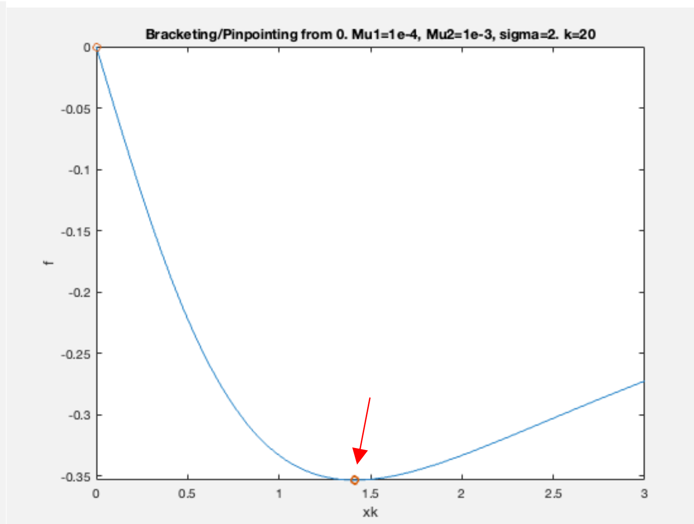
2.a)





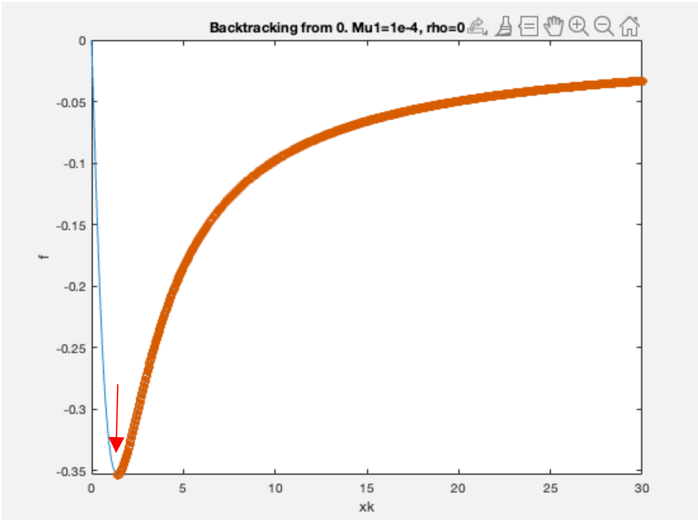
2.b) k=20

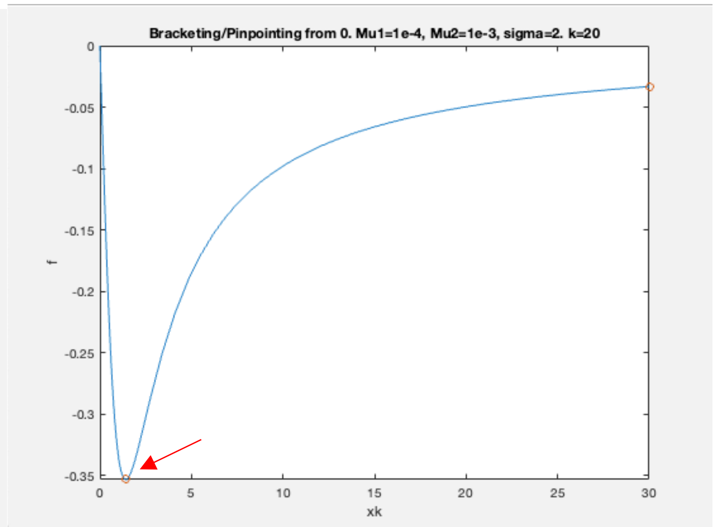




The bracketing/pinpointing algorithm still works well, but the backtracking algorithm has issues when the direction vector is increased dramatically in magnitude (the leftmost point is the starting point and the rightmost point is the ending point). Specifically, it has no reason to backtrack further than it needs to, so it will tend to "skip over" minima if the direction vector is very large in magnitude.

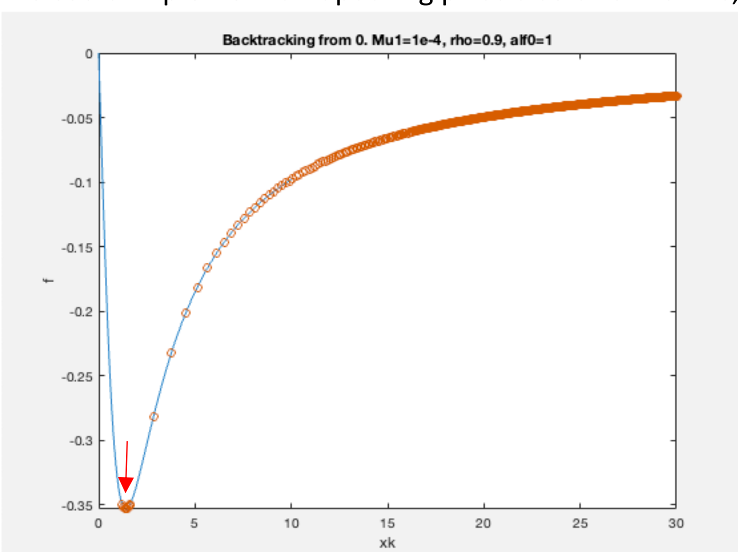
2.c)





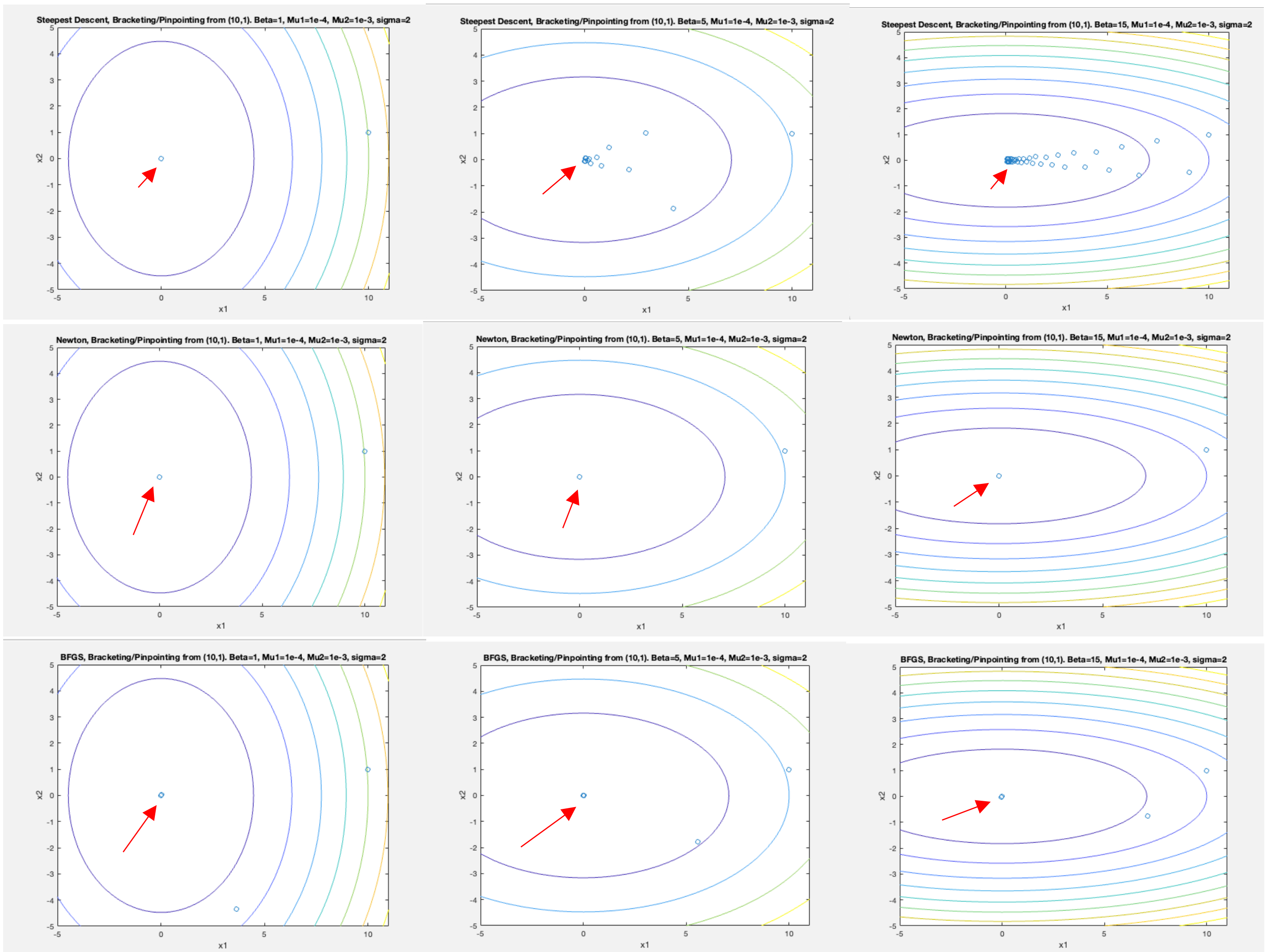
The bracketing/pinpointing algorithm performs well again here. The backtracking approach eventually reaches the minimum, but takes many many iterations to do so (the starting point is the point furthest to the right, the ending point is the minimum). The issue here is the definition of pk, being dependent on the slope of f. Since we start at a very flat section of f (at x=30), the backtracking approach can only take very small steps.

We could implement an updating pk as a solution to this, but that causes oscillations about the minimum:



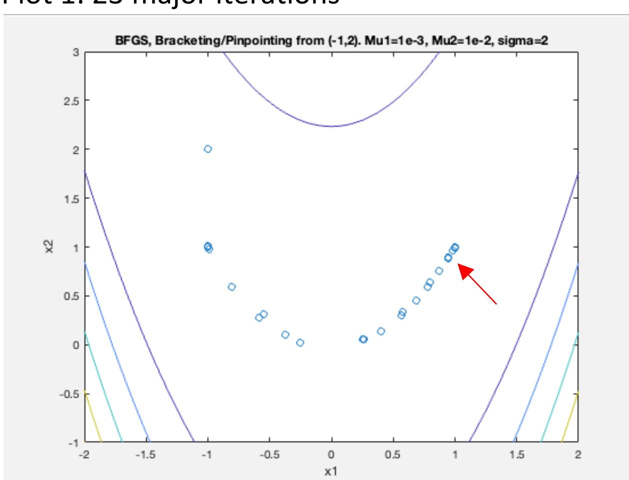
2.d) 1 hour

3.a)

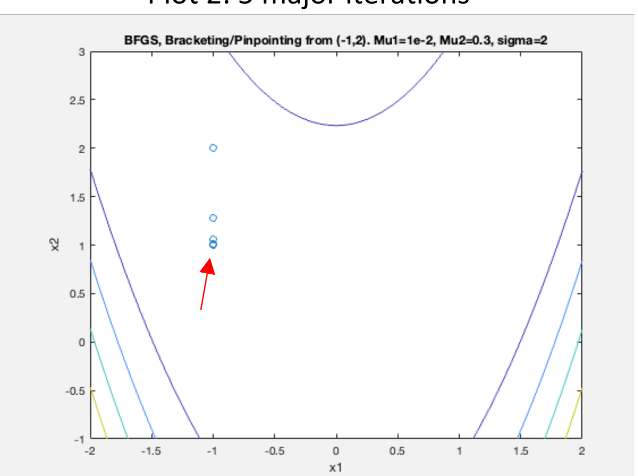


3.b)

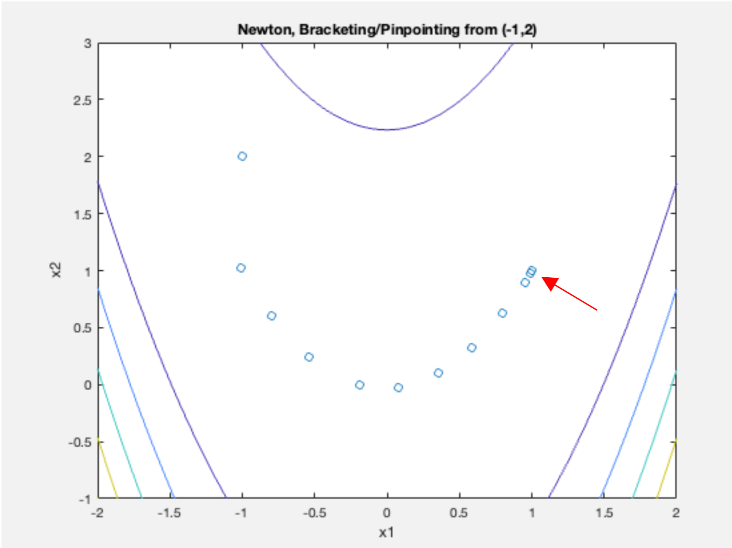
Plot 1: 23 major iterations



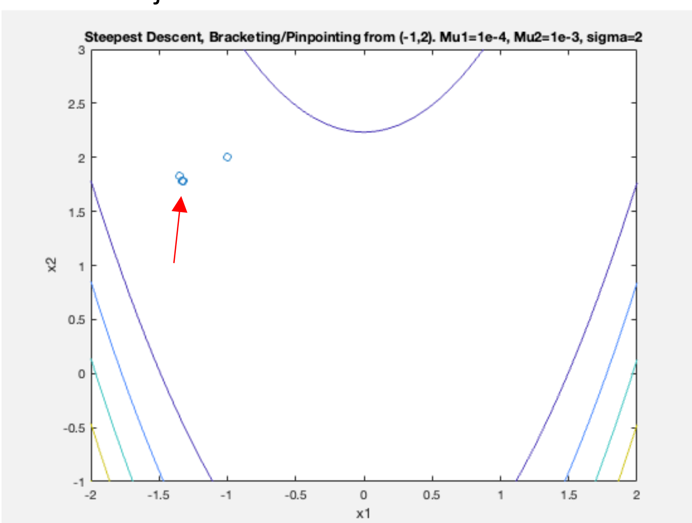
Plot 2: 5 major iterations



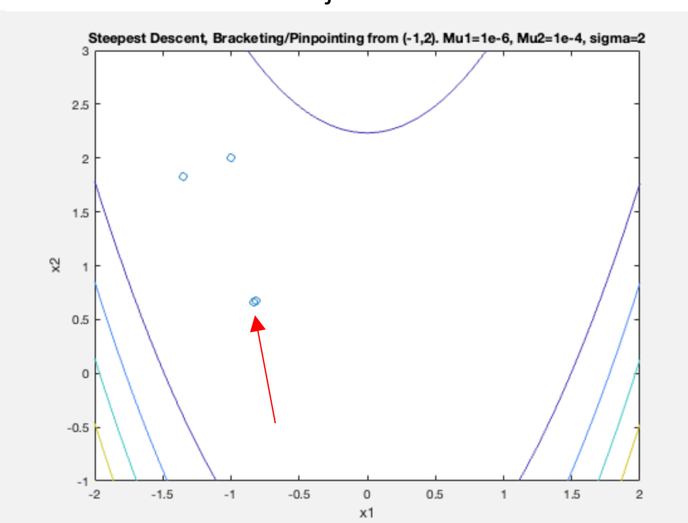
Plot 3: 12 major iterations



Plot 4: 4 major iterations

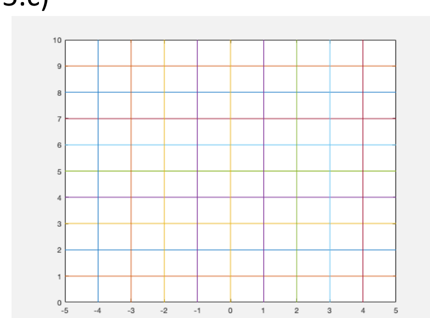


Plot 5: 4 major iterations



The function has a shallowly descending "minimum valley" rather than a single lowest point, so the BGFS and Newton methods struggle with this valley and perform many iterations iterating across it. Steepest descent is not able to reach the end of the valley (although this might be solvable by tuning parameters).

5.c)



the lines appear close to each other

5.d) 3 hours