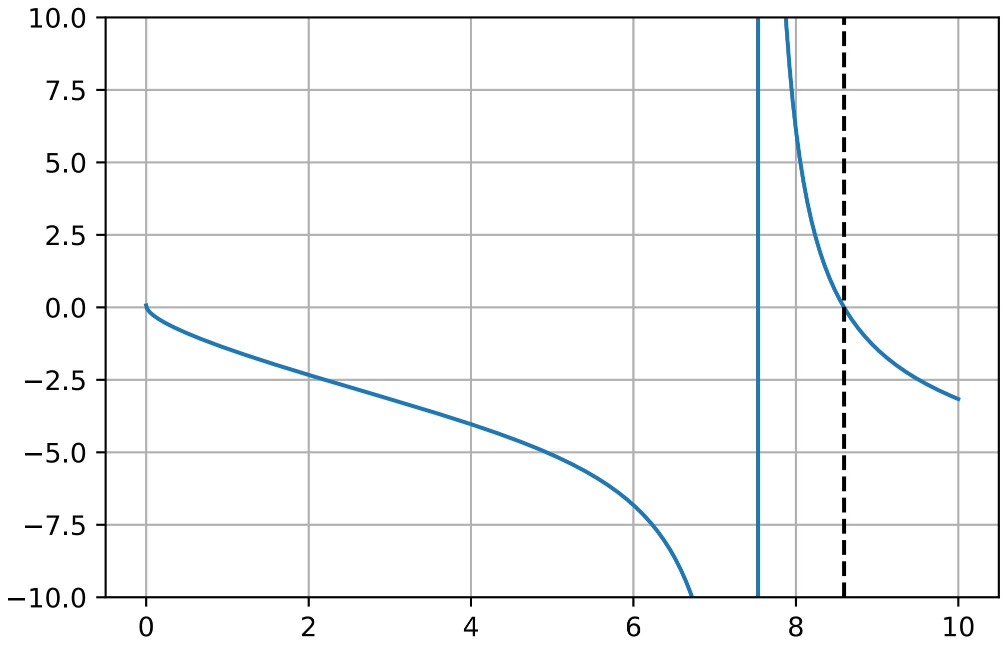


I find roots for the even function at:

E_B = 8.592785275230653 (bisection method), precision = -3.882227872509247e-12

And

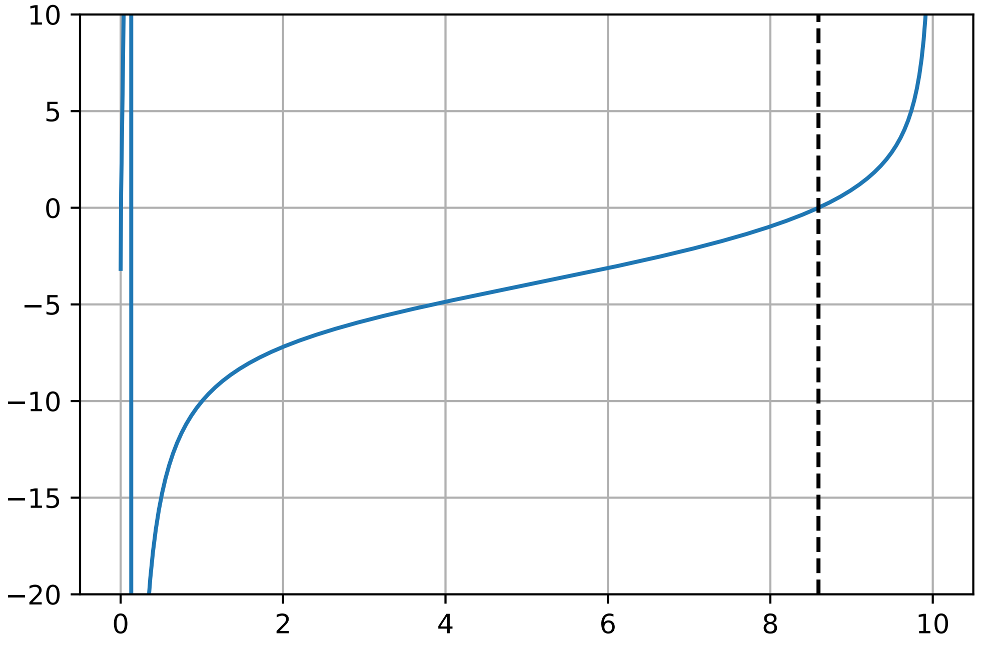
E_B = 8.592785275229831 (Newton-Raphson method), precision = 3.68594044175552e-14

Evaluating the function at the determined roots, I find the persicion as:

P = -3.882227872509247e-12 (bisection method)

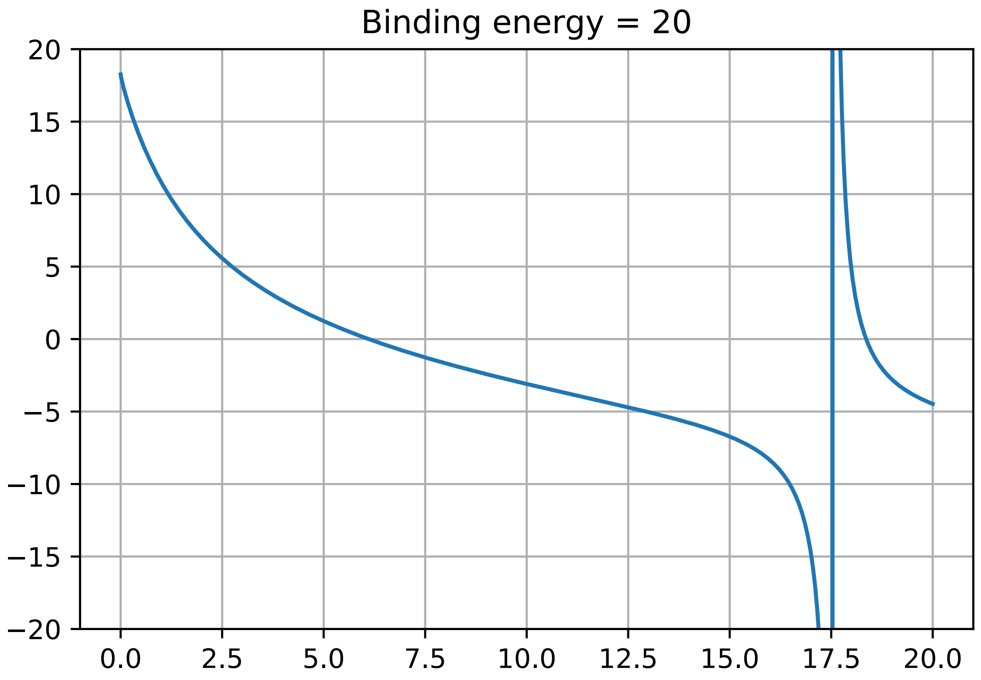
And

P = 3.68594044175552e-14



The odd version of the function doesn’t have a discontinuity as near to the root as the even function does. This means when you use the root-finding routines, your initial guess can be further away from the actual root, and the program can still find the root. Using both methods I find:

root bisection method, E_B = 8.592785275230653
root Newton-Raphson method, E_B = 8.592785275229838



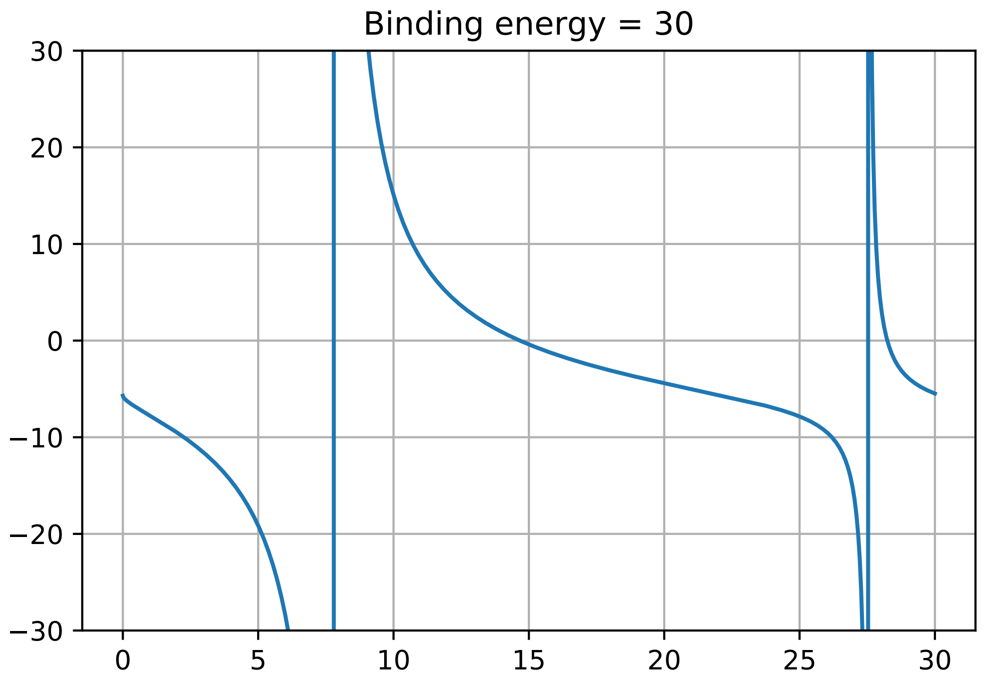
When you increase the binding potential from 10 - 20, instead of one root, there are now two. The values for these roots are:

E_B = 6.108467017546673, and E_B = 18.360519852466958 (bisection method)

With precessions of:

P = 9.676703882632864e-13, and P = -1.673328142715036e-12

The function for a binding energy of 20 also differs from a binding energy of 10 because it starts at a positive f(E_B) as opposed to near zero.



Increasing the binding energy to 30 you again find two roots, but the shape of this function more closely resembles the function for binding energy equal to 10. That is the function begins at a negative value closer to zero than the function for binding energy equal to 20.

The roots and precision for this function are:

E_B = 14.666053424209167, and E_B = 28.24111348793076 (bisection method)
P = -4.511946372076636e-13 and P = 5.071498776487715e-12

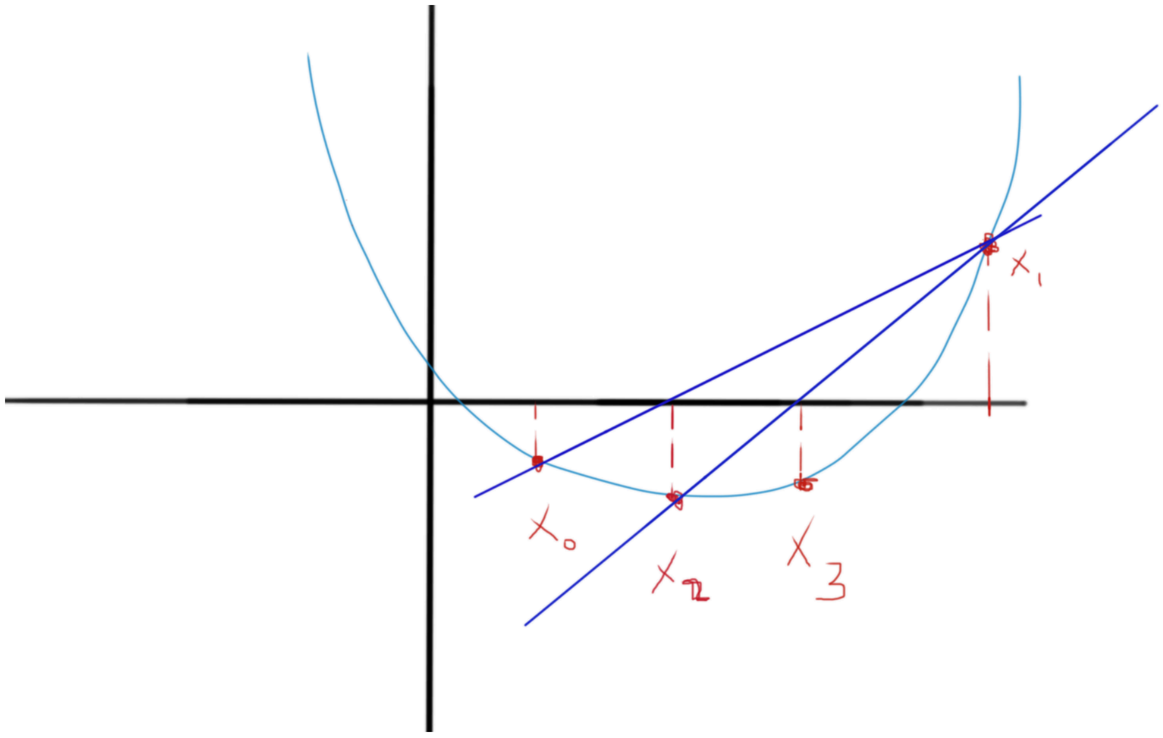
Looking at the spicy.optimize documentation, Brent's method is cited as, "Generally considered the best of the rootfinding routines here". Testing it against the initial function where binding energy is related to 10, I find the root as:

E_B = 8.592785275229838

With the precision being an order of magnitude better than the Newton-Raphson method:

P = 3.552713678800501e-15

This method appears to combine multiple numerical techniques to attain the best speed and accuracy. It leverages aspects of the bisection method in that it requires the function to have opposite signs at two initial guesses. With each step, the interval between the interval surrounding the root, initially the guess, will decrease. However, the method does exclusively use the bisection method to narrow the interval, rather it uses the bisection method as a sort of last resort since the bisection method always converges to a solution, but is does so slower than other techniques. To begin, Brent's method tries to narrow the interval using the secant method, or another interpolation method. Using the picture below to illustrate, if it uses the secant method, it will run a line through the two endpoints of the interval, labelled x0, and x1. This secant line will have a intersect the x-axis, provided that f evaluated at x0, and x1 have opposite signs. The point where the secant line crosses the x-axis will be labelled x2, and will create the next iteration or interval in the secant method. The method will repeat over the interval [x1,x2], and then find a new point x3, closer to the root. This repeats until the method reached some tolerance. Depending on the shape of the function, I.e. if the function has a low



rate of change, or horizontal asymptotes, the secant method will not converge to a solution.If the secant method cannot converge, or isn't efficient to some tolerance, Brent's method will revert to the bisection method at that step.