Asset Pricing Theory — Problem Set 4 *

Ali Bahramisani†

Version: March 2, 2025

*The latest version of this document and related materials can be accessed at: Github Repository.

†Stockholm School of Economics, Department of Finance, Email: ali.bahramisani@hhs.se, Website: alibahramisani.github.io

1 Optimization

In this section, I use the first order taylor approximation for the Newton-Raphson method to solve the problem.

1.1 A Polynomial

I tried to solve the problem with Newton-Raphson iteration method, as well as using Python built in solvers. The results are presented in Figure 1.

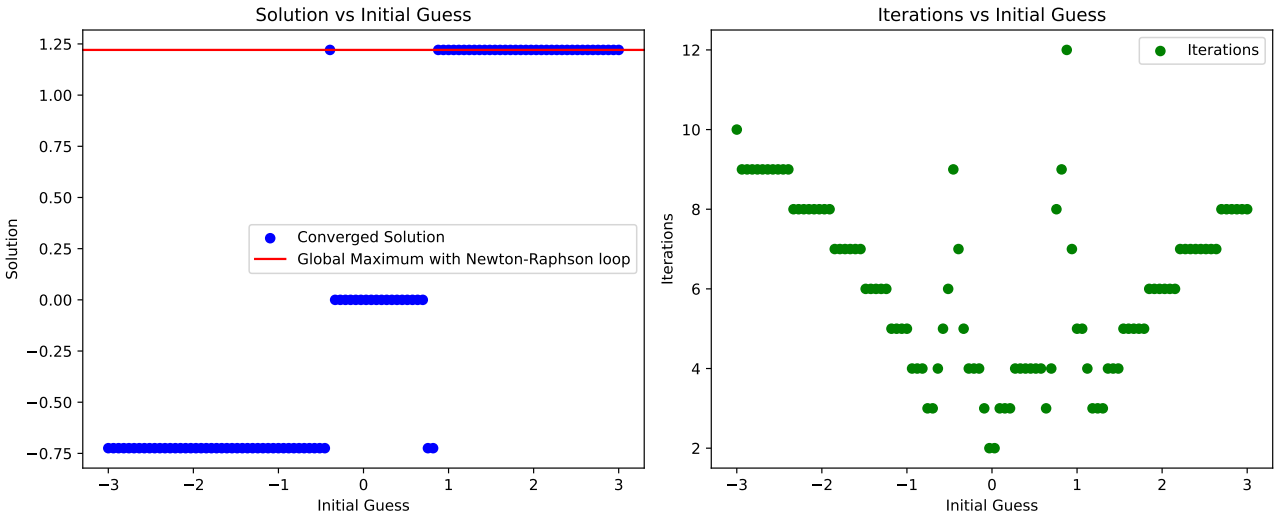

Figure 1: Obtained solution and the number of iterations against the initial guess

As it is shown, the Newton-Raphson method with the first order taylor approximation, conditional on the initial guess, could not precisely solve the problem. Also, since this method is heavily contingent on the local derivations, the initial guess is crucial for the convergence and the solution. We can see that as the initial guess is closer to the local optimas, the method converges faster to those optimas. However, some of them are not global maximas, and the method converges to those local maximas. Considering the global maximum that Newton-Raphson method converges to, only 37% of intitial guesses converge to the global maximum.

1.2 The sigmoid function

The true solution is obviously zero. I, again, wolve the problem with the first order taylor approximation of Newton-Raphson method. The results are presented in Figure 2. As we can see, initial guesses far from the solution are diverging from the true solution. However, the method converges to the true solution for the initial guesses close to the true solution. Initial guesses far from the true solution are diverging due to the fact that for small and high initial guesses, the first derivation of the sigmoind function is extremly large and close to zero, respectively. This makes the Newton-Raphson method to diverge from the true solution.

Put differently, by calculating the average update steps, we can see that the method converges to the true solution for the initial guesses close to the true solution. However, for the initial guesses far from the true solution, the method diverges from the true solution. The average update steps are defined as:

$$\text{Average update steps} = \frac{1}{N}\sum_{i=1}^{N} |x_{n+1} - x_n| \tag{1}$$

where N is the number of iterations, and x_n is the solution at the nth iteration. As we can see the average update steps in Figure 2, they are decreasing as the initial guess is closer to the true solution.

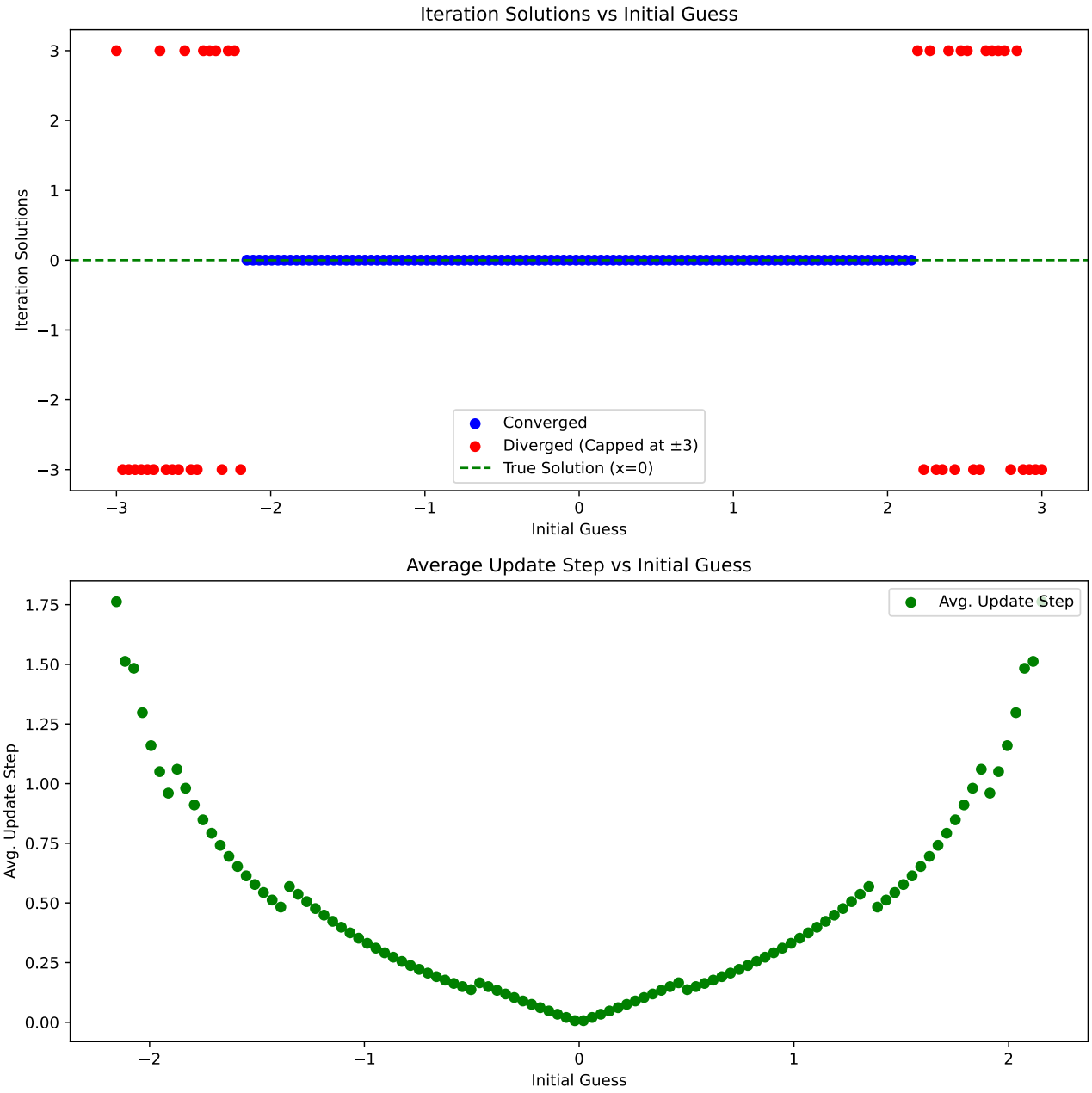

Figure 2: Obtained solution and the average update steps against the initial guess

All in all, combining the results in Figure 1 and Figure 2, we can see that the Newton-Raphson method is pretty fast when the initial guess is closer to the true solution. However, for the initial guesses far from the true solution, the method diverges from the true solution. If the function is differentiable this is an easy method to impelement. However, for non-differentiable functions, we cannot use it. In regions that the derivations are close to zero, the method diverges from the true solution. Also, this method maybe useful for local optimization. Finally, the method is contingent on the order of derivations, which as it increases it is more precise, but also more computationally expensive.

1.3 The bisection algorithm

It takes 19 iterations to converge to the solution with the bisection method depicted in Figure 3. Compare to Newton-Raphson method, bisection method is slower. The bisection method is not contingent on the initial guess.

It is contingent on the first derivation of the function and how we define the intervals. However, it is slower than the Newton-Raphson method. Bisection alwasy find a root, but it is not necessarily the global maximum. As it is depicted in the Figure 3, there are three roots for the function, only one of them is the global maximum which is specified with the horizontal green line. If we split the intervals into small intervals, it is true that we may find all the roots and then we can check which one is the global maximum. However, it is computationally expensive.

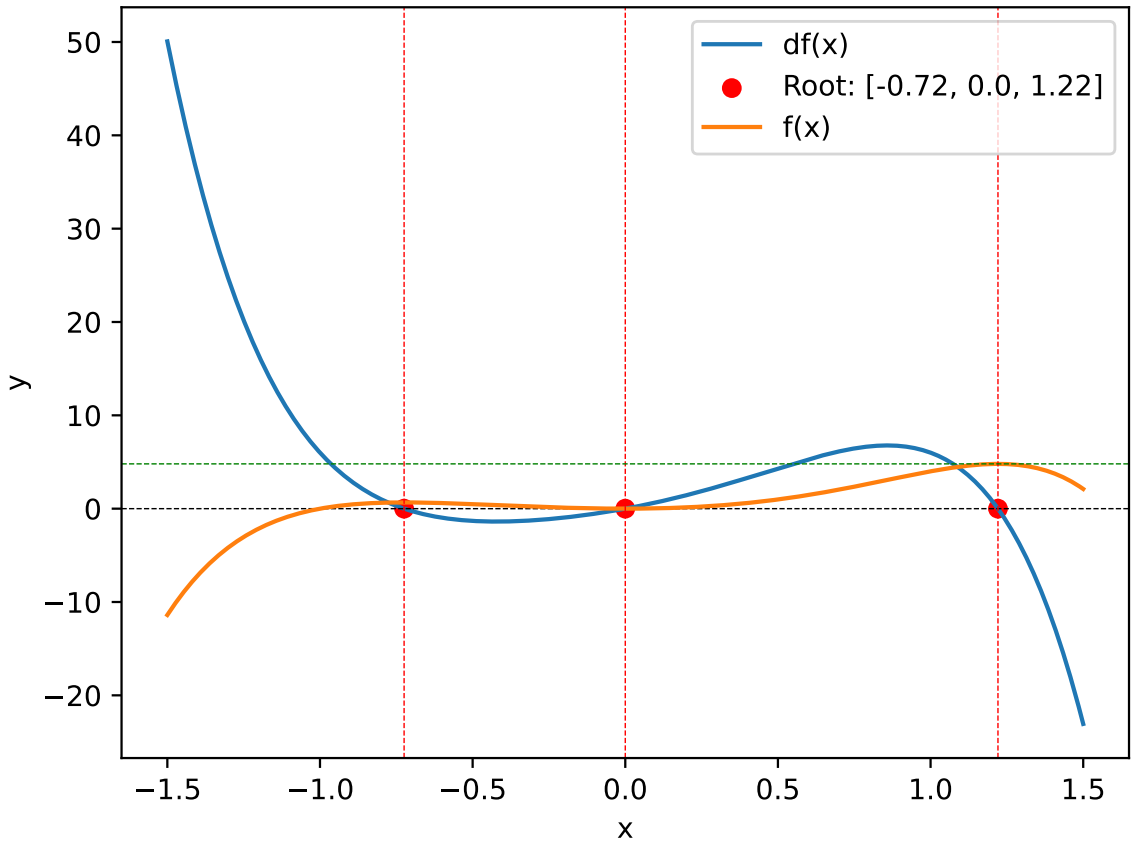

Figure 3: Bisection Method: Roots of $f(x)$

2 Numerical Integral

1. We have the following for the number of intervals:

$$|10^{-4}| \leq \frac{2.57474^3}{12n^2} \cdot 160 \rightarrow n_{Trapezoidal} \geq 1507 \tag{2}$$

$$|10^{-4}| \leq \frac{2.57474^5}{180n^4} \cdot 360 \cdot 1.57474^2 \rightarrow n_{Simpson} \geq 50 \text{ It must be even!} \tag{3}$$

2, 3. The results are presented in Figure 4. As we can see, the Simpson's method is more precise in lowe iterations, then Trapezoidal method is more precise as N increases. However, for Monte Carlo, it is more dependent on the random process. For instance, we can see with seed **137**, and 1000 iterations, its error is less than the error of the Simpson's method.

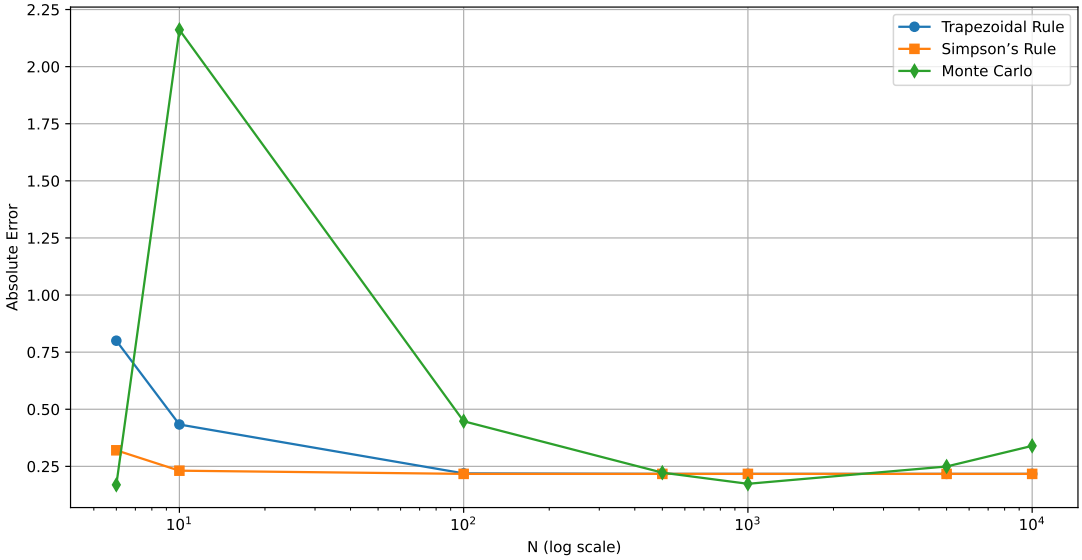

Figure 4: Absolute Errors vs N

4. Taking log from both sides of the error terms, we have:

$$\log E_{\text{Trap}} \sim -\log(n^2) \tag{4}$$

$$\log E_{\text{Simp}} \sim -\log(n^4) \tag{5}$$

$$\log E_{\text{MC}} \sim -\log(\sqrt{n}) \tag{6}$$

It is consistent with the results in the Figure 4. As we can see, the error of the Simpson's decreasing faster than the other method as n increases.

5. Trapezoidal method is simpler ti implement, but it is slower in covergance rather than Simpson method. Simpson method is more precise in lower iterations, bu it requires an even number of intervals. Both of them also require the function to be twice differentiable. For Simpson, it is more restrictive, as it requires the function to be four times differentiable. Monte Carlo method is more flexible, as it does not require the function to be differentiable. However, it is more dependent on the random process, and it is slower than the other methods. It is also more computationally expensive as requires a large number of samples for high accuracy.

3 Portfolio selection and Value at Risk

In this question, I implictitly assume that returns are normaly distributed and will derive the confidence intervals based on this assumption. Bob can reject that Alice's portfolio were optimaly chosen only if there is no interaction between Alice's MVF and Bob's CI. The results are presented in Figure 5. I used seed number **137** for the Monte Carlo simulation.

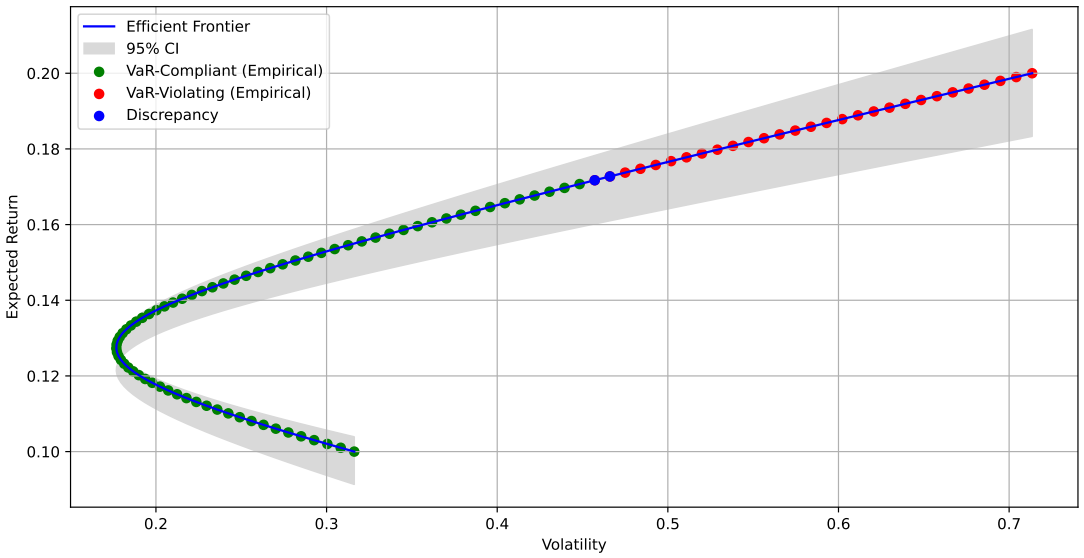

Figure 5: Efficient Frontier with Confidence Intervals and Empirical VaR Compliance

To find the upper bound on the volatility we can use the following formula:

$$\sigma_{\max} = \frac{r_{\text{target}} + \bar{V}}{\Phi^{-1}(\lambda)} \tag{7}$$

where I use the target returns as $r_{\text{target}} \in [0.1, 0.2]$. Using this formula we can find the upper bound on the volatility:

$$\sigma_{\max}(r_{\text{target}}) = 0.143 + 1.91 r_{\text{target}} \tag{8}$$

Comparing the theoretical and empirical VaR, we need to consider that the theoretical one is not necessarily like the empirical one and that is because both returns are not necessarily normally distributed. The normal distribution is symmetric, but the returns are not necessarily symmetric. The only difference between them is mostly because of the randomness of the Monte Carlo data generation algorithm (blue dots in Figure 5).