Optimization - Project 2

Minimization using derivatives

Nikolaos Konstas

Department of Electrical & Computer Engineering
Aristotle University of Thessaloniki

December 2022

Contents

1 Introduction

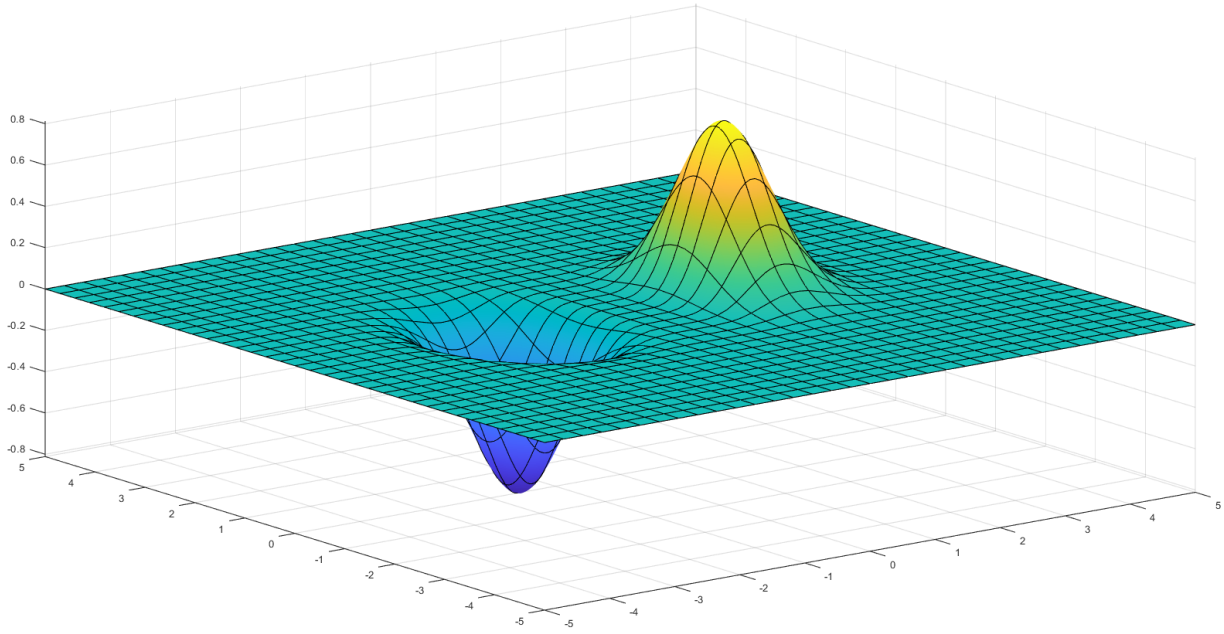
In this work it is requested to implement the algorithms:

- Steepest Descent Method,
- Newton Method,
- Levenberg-Marquardt Method.

To study the above algorithms, the objective function will be used:

$$f(x, y) = x^5 \, e^{-x^2 - y^2}$$

Below is the graph of $f(x, y)$:



From the three-dimensional representation of $f(x, y)$ we can see that the function has a total minimum approximately at the point (-1.6,0).

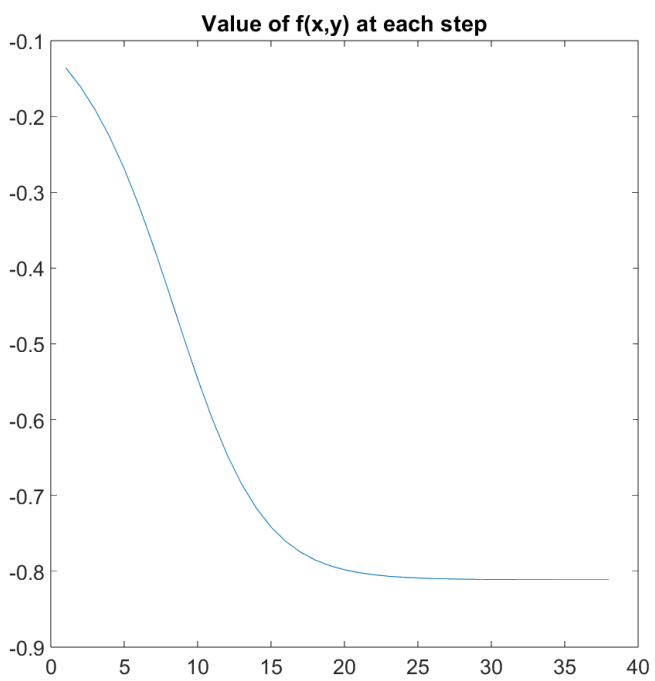
2 Maximum Descent Method

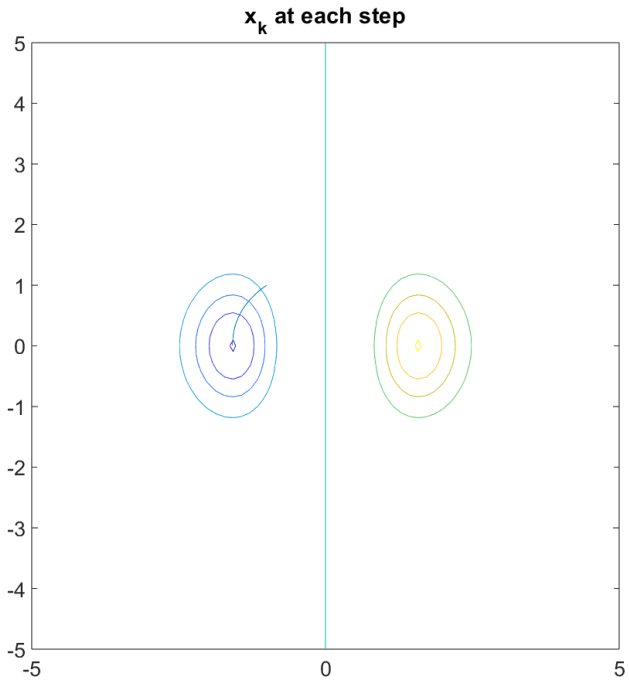
First, the Maximum Method method was implemented and tested, for which 3 methods were used to select the scaling factor γ_k at each step of the algorithm. First, a constant $\gamma_k = 0.1$ was used, then using the bisector method using a derivative (implemented in the first paper) γ_k was chosen at each step that minimizes the expression $f(x_k + \gamma_k \delta_k)$, while finally γ_k was chosen to satisfy the Armijo rule. In the last case, the values $\alpha = 10^{-3},\ \beta = 0.2,\ s = 0.1$ were used for the free parameters. Also, the algorithm was tested for 3 different starting points: $(0,0), (-1,1), (1,-1)$. In all cases the same termination criterion $|\nabla f(x_k)| was used < 0.01$.

Starting with the starting point $(x_0, y_0) = (0,0)$ we observe that regardless of the way of choosing γ_κ the algorithm terminates directly. This is logical as $\nabla f(0,0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and so the termination criterion is met.

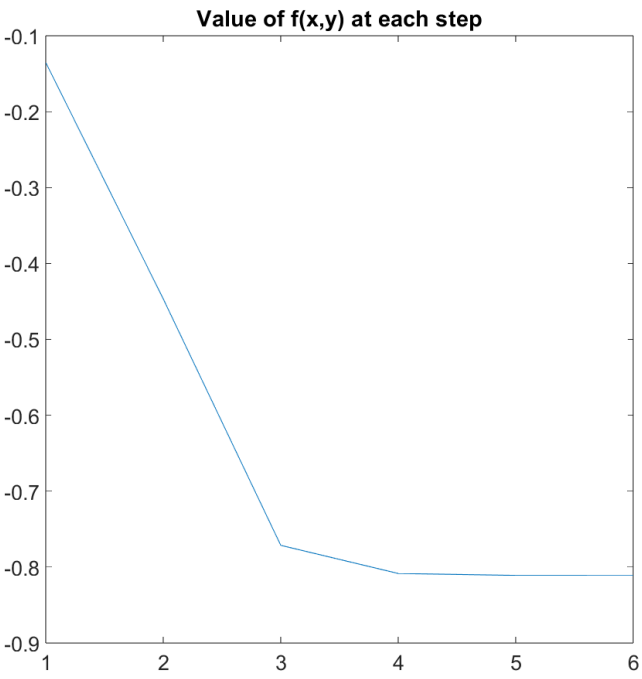
Now we use as starting point $(x_0, y_0) = (-1,1)$.

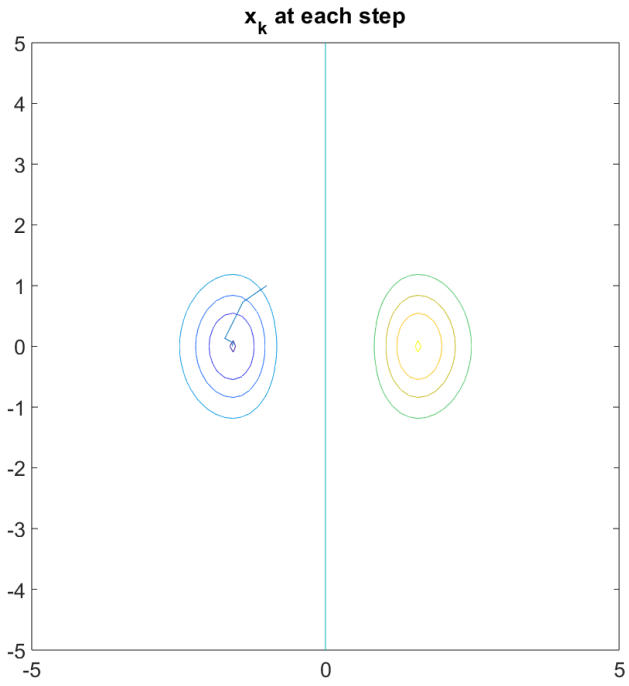
Constant γ_k:



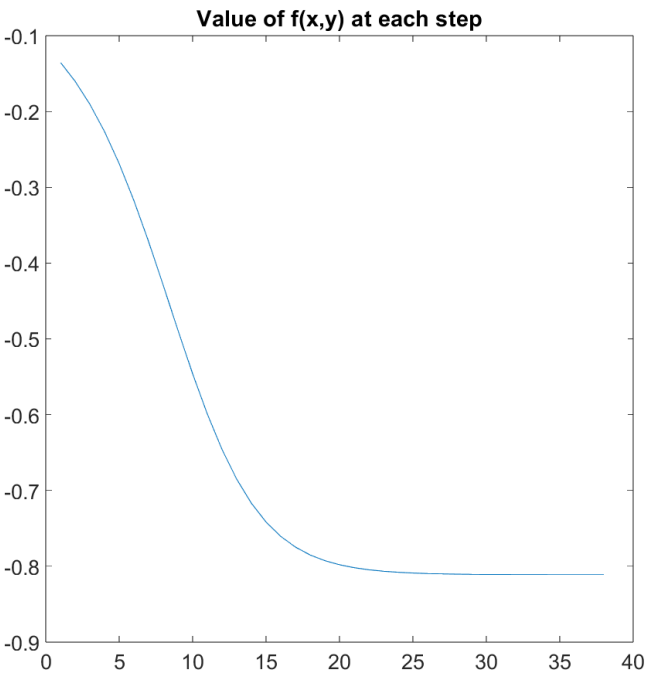


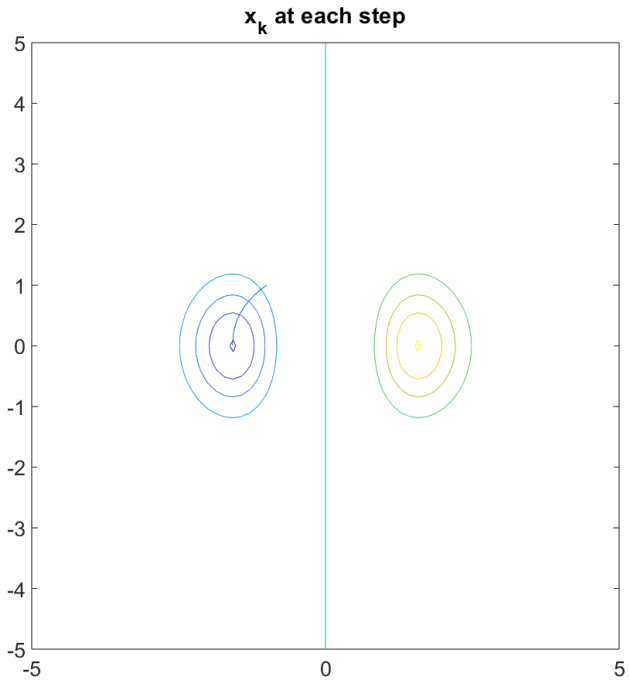
Choose γ_k to minimize $\mathbf{f(x_k + \gamma_k \delta_k)}$:





Choose γ_k with rule Armijo:



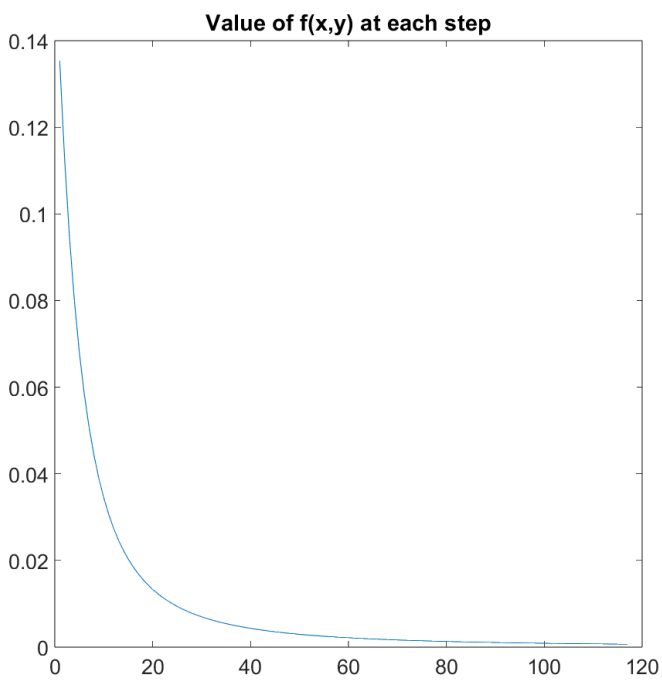


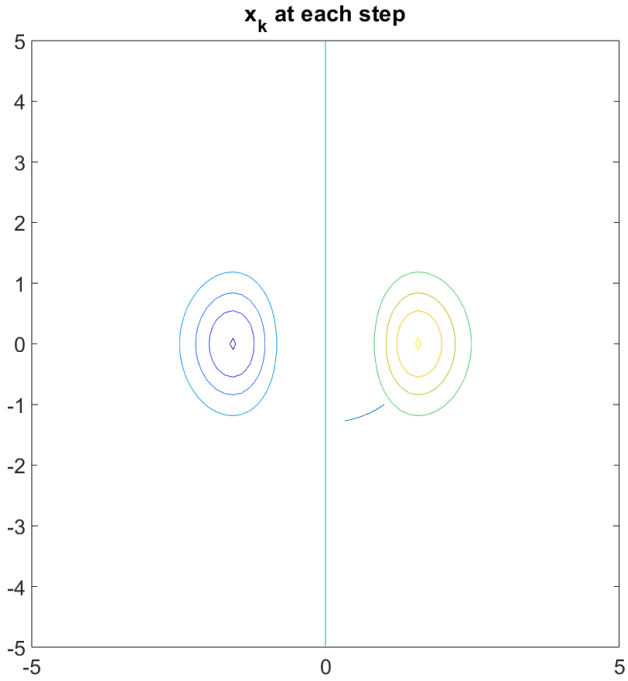
In all 3 cases we observe identical behavior with a negligible deviation at the final point of the order of 10^{-3} and the algorithm ends up at a point very close to the total minimum. The main difference we notice is the number of iterations needed, where the method with internal

minimization needs by far the fewest iterations. However, the number of iterations is not able to lead us to any conclusion as the 3 algorithms have different complexity. Thus we record the time it takes for the algorithms to complete and we observe that the algorithm for fixed γ_k terminates faster with the other 2 taking almost 10 times more time (the algorithm with internal minimization was slower). More detailed minimum points and execution time can be found by running the corresponding file (Task2.m).

Now we choose as starting point $(x_0, y_0) = (1, -1)$.

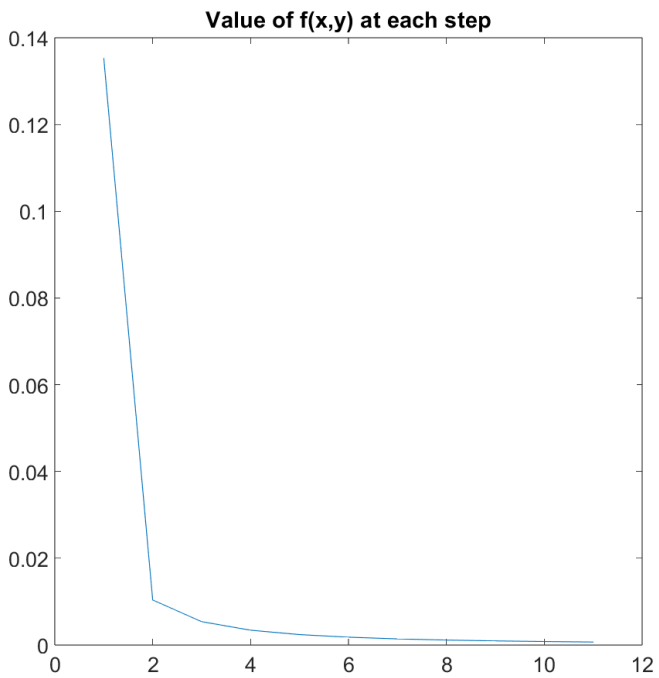
Constant γ_k:

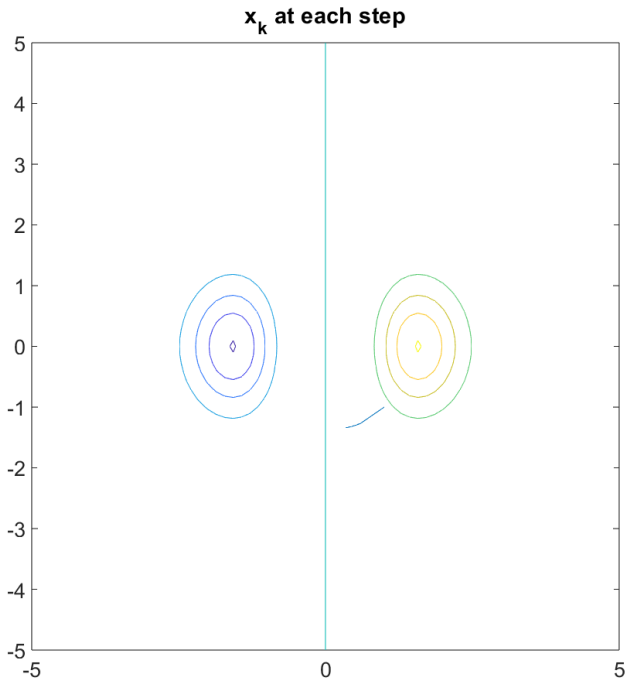




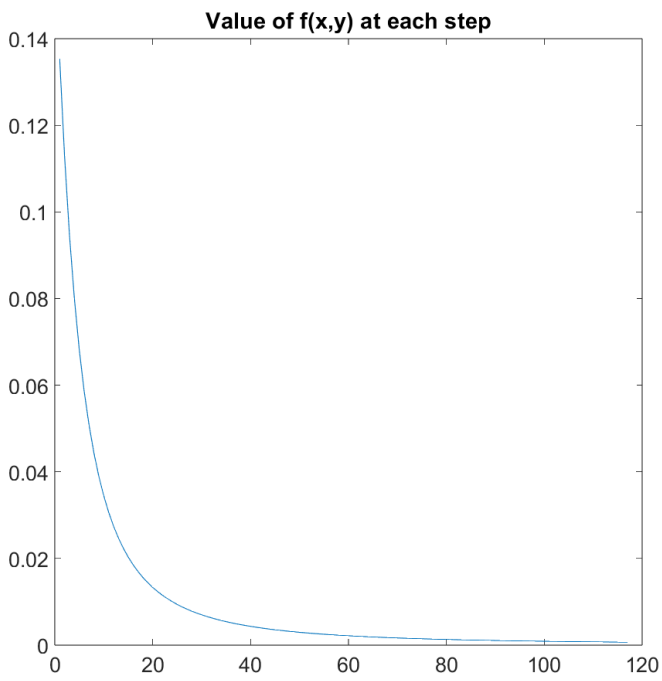
This time, the algorithm gets stuck in a local minimum and fails to reach the global minimum, as before. Nevertheless, the results of the 3 algorithms again agree, while the algorithm with constant γ_k is again faster and the one satisfying the Armijo rule is slower. The remaining figures are shown below.

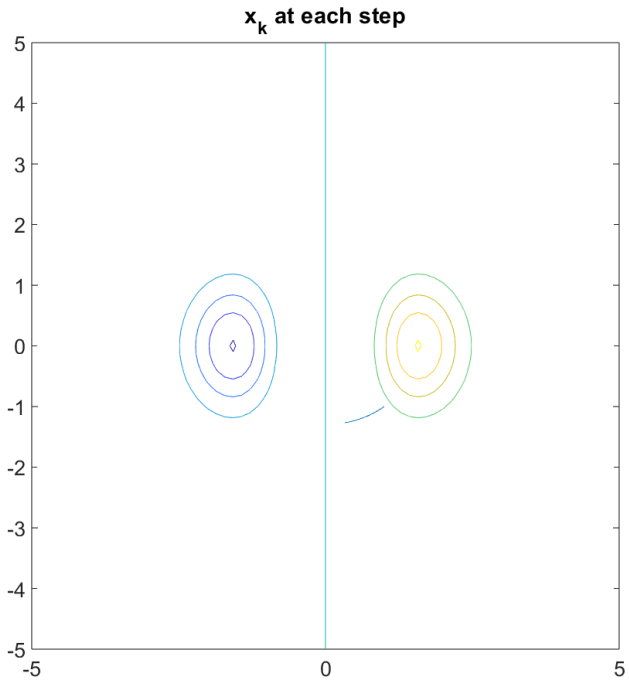
Choose γ_k to minimize $\mathbf{f(x_k + \gamma_k \delta_k)}$:





Choose γ_k with rule Armijo:





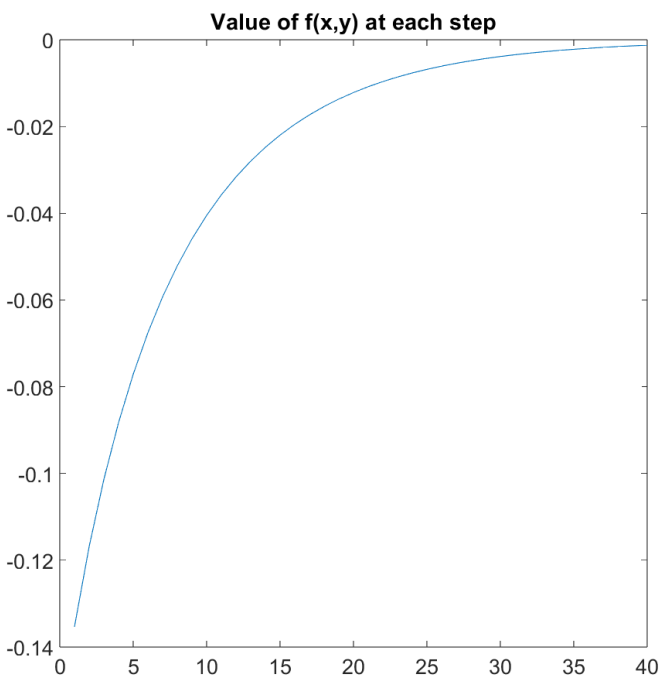
3 Newton Method

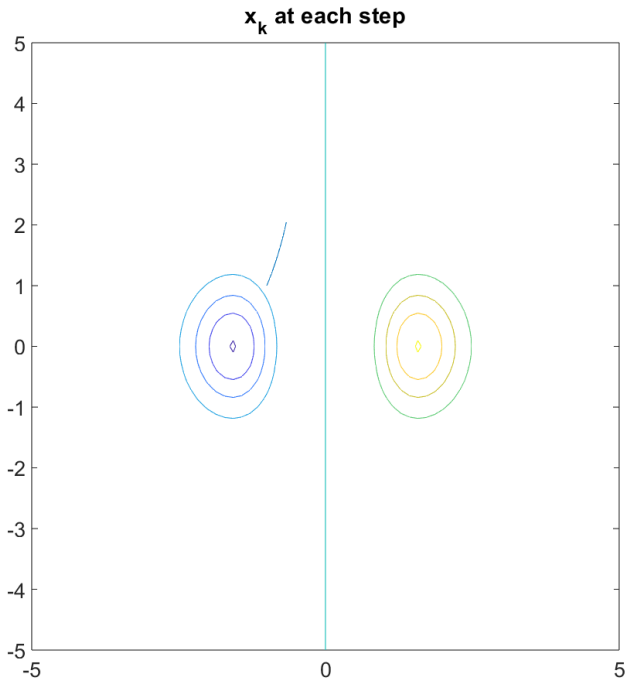
The Newton method was then implemented. Again 3 different ways of choosing γ_k were used, but also 3 different starting points.

Starting with the starting point $(x_0, y_0) = (0, 0)$, just like above the algorithm loops and terminates directly.

Of greater interest are the other starting points. Choosing $(x_0, y_0) = (-1, 1)$ shows the weaknesses of this method. As we know from the theory, this particular method is based on the condition that the Hessian matrix of f is positive definite. This does not happen and the problems are visible by applying the algorithm. For choices of γ_k with internal optimization and based on the Armijo rule, x_k does not converge to some value and the algorithm never terminates. By choosing a fixed γ_k , the value of f increases over time and essentially terminates when the algorithm locks into a local maximum. as shown below.

Constant γ_k:

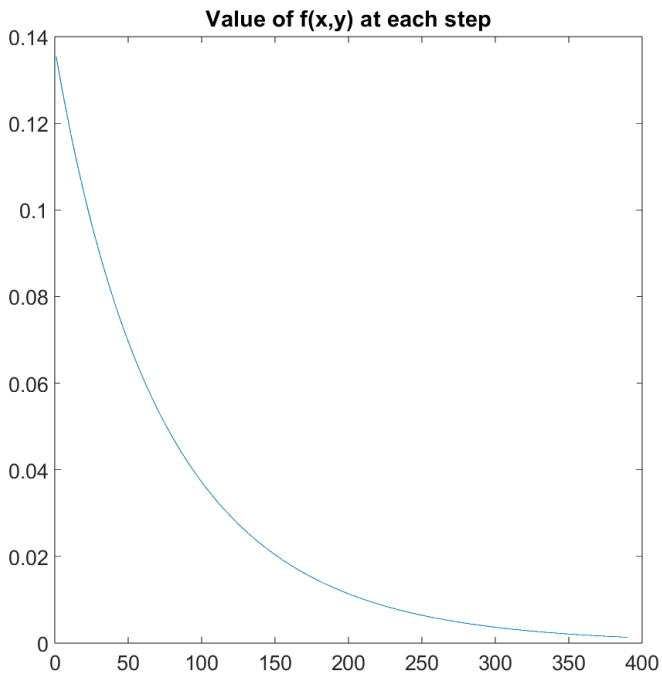


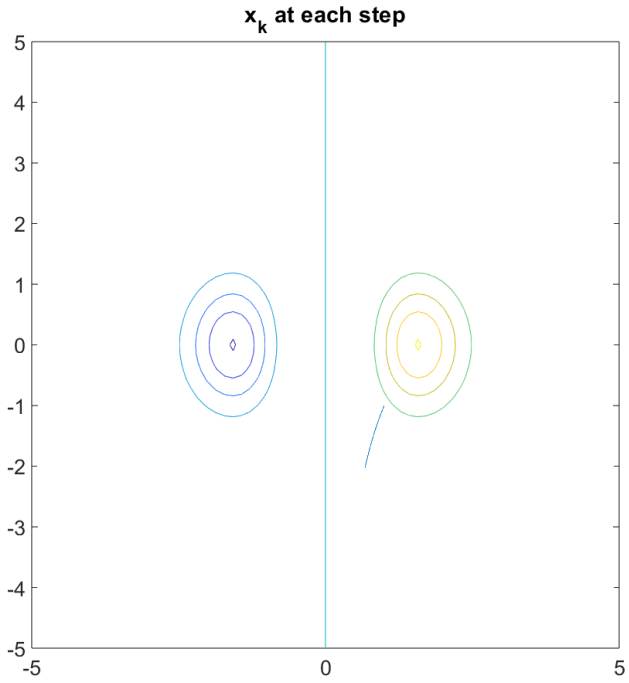


Now we choose as starting point $(x_0, y_0) = (1, -1)$.

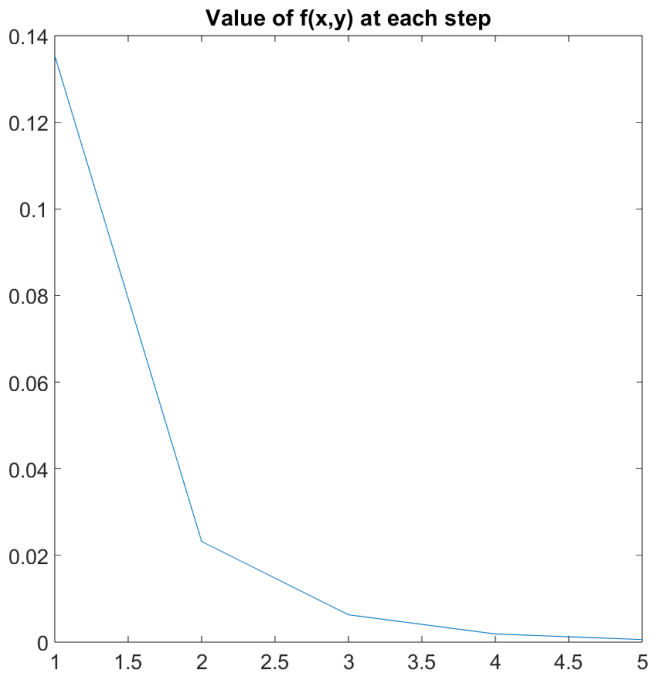
For this particular point, we have in all 3 cases a similar behavior to the Maximum Descent method, with x_k locking in some local minimum. The same is true of the execution time, with the constant γ_k being the fastest algorithm. Below are the corresponding figures.

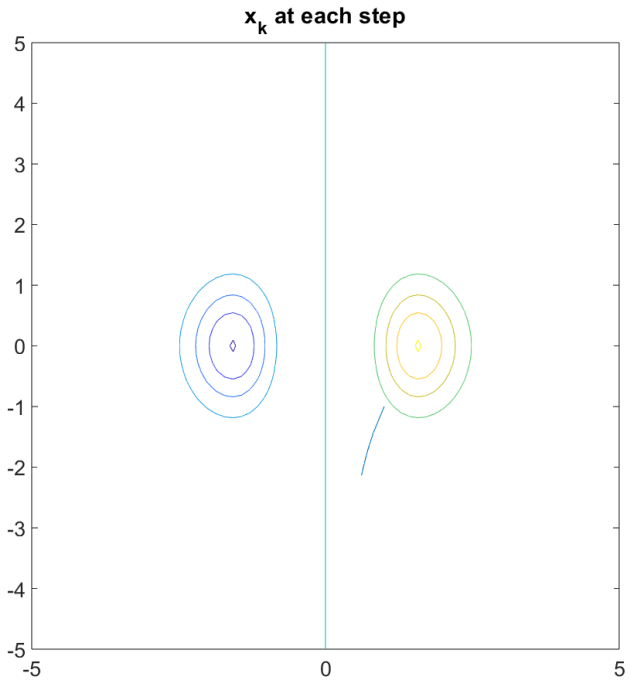
Constant γ_k:



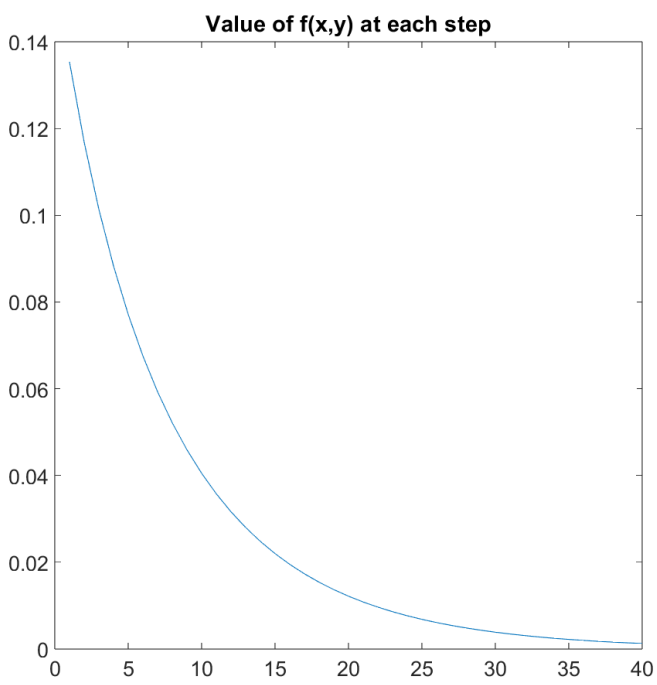


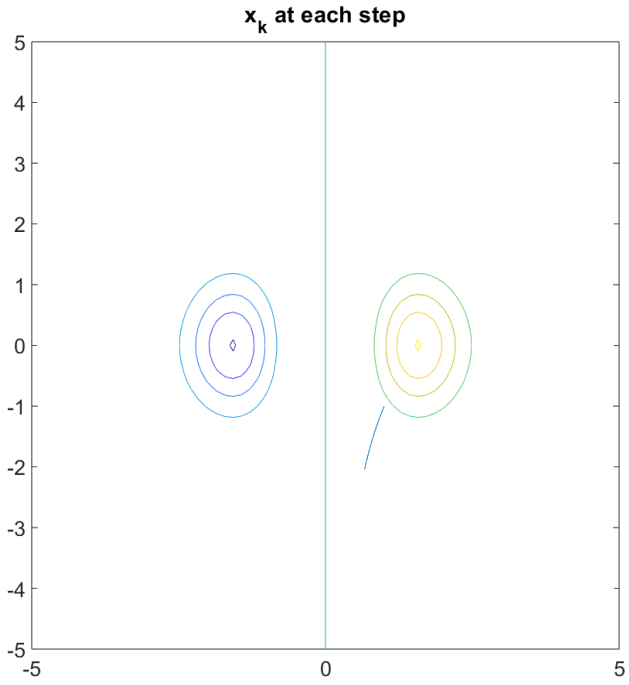
Choose γ_k to minimize $f(x_k + \gamma_k \delta_k)$:





Choose γ_k with rule Armijo:


Value of f(x,y) at each step
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
5
10
15
20
25
30
35
40


x_k at each step
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5

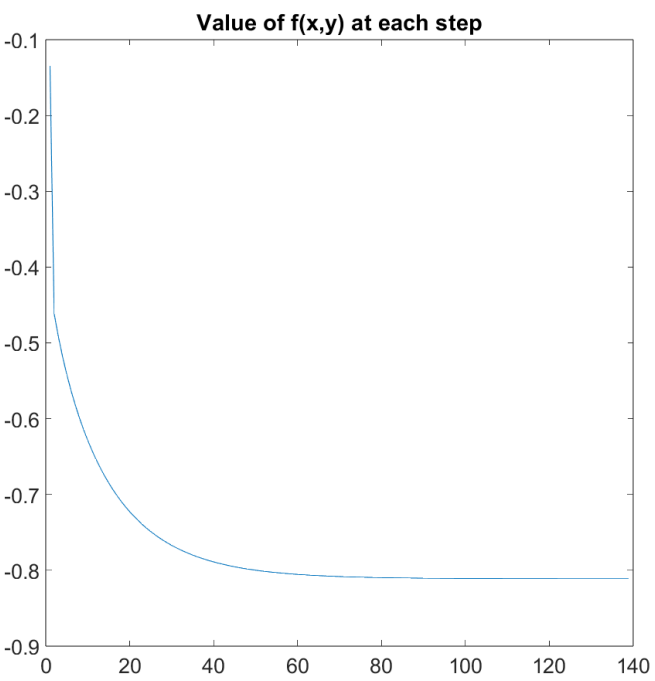
4 Levenberg-Marquardt Method

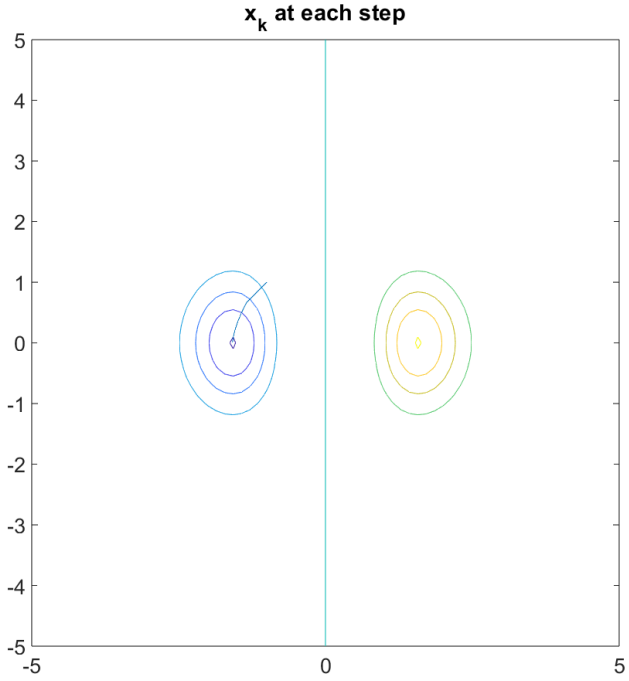
Lastly, the Levenberg-Marquardt method was implemented and applied. This particular method is a modification of the Newton method, based on the latter's well-functioning criteria. It solves the problem of the divinely determined matrix condition by adding to the Hessian matrix an appropriate quantity ($\nabla^2 f(x_k) + \mu_k I$, specifically in our implementation $\mu_k = \bar{\mu_k} + 0.1$).

Once again, starting with the starting point $(x_0, y_0) = (0, 0)$, just as above the algorithm loops and terminates directly.

Now we use as starting point $(x_0, y_0) = (-1, 1)$.

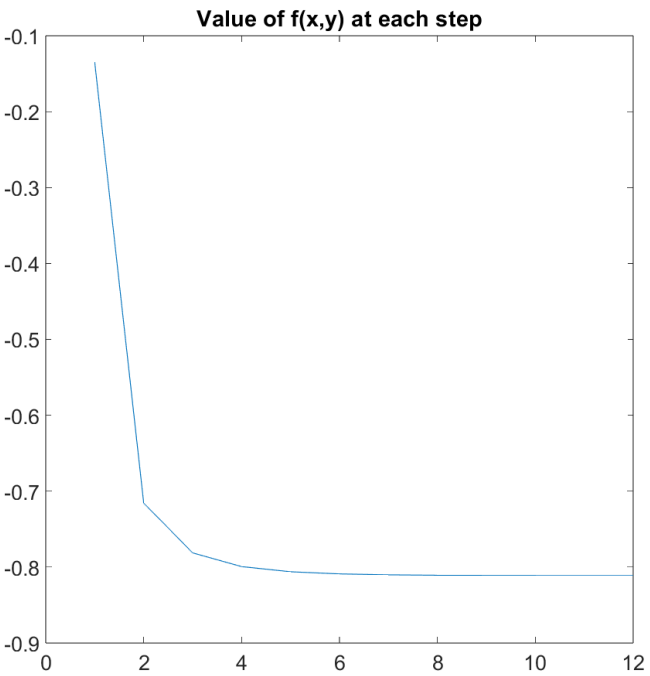
Constant γ_k:

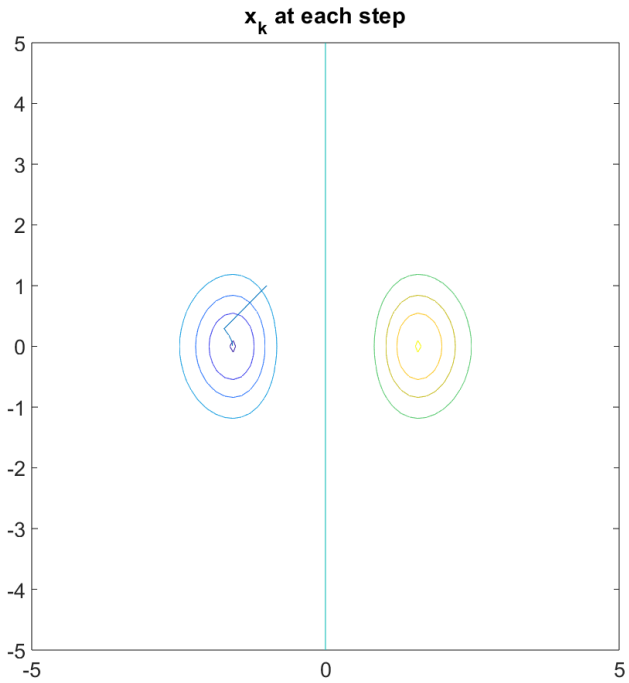




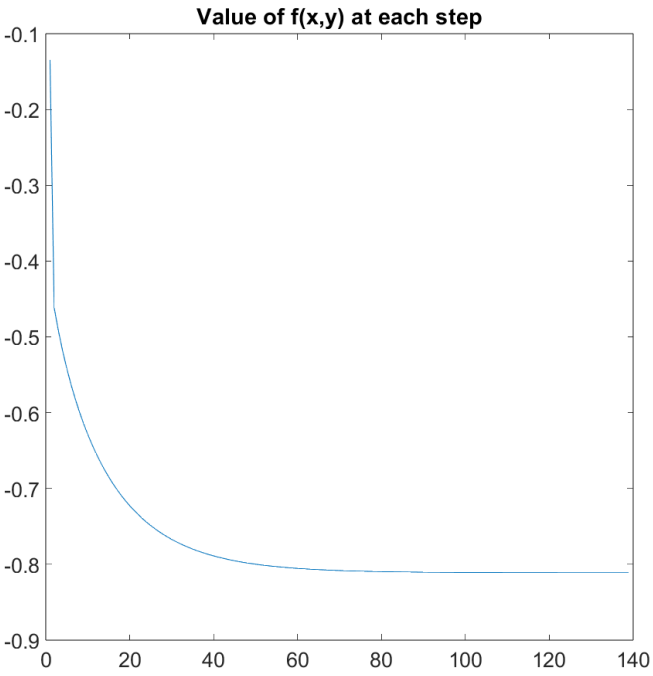
As we expected, the method works, without problems and produces results similar to the maximum descent method. Here are the rest of the figures.

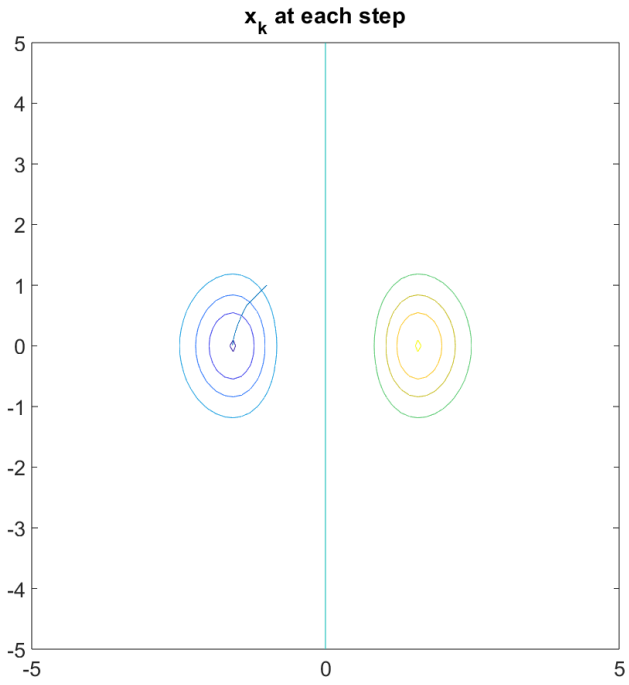
Choose γ_k to minimize $\mathbf{f(x_k + \gamma_k \delta_k)}$:





Choose γ_k with rule Armijo:



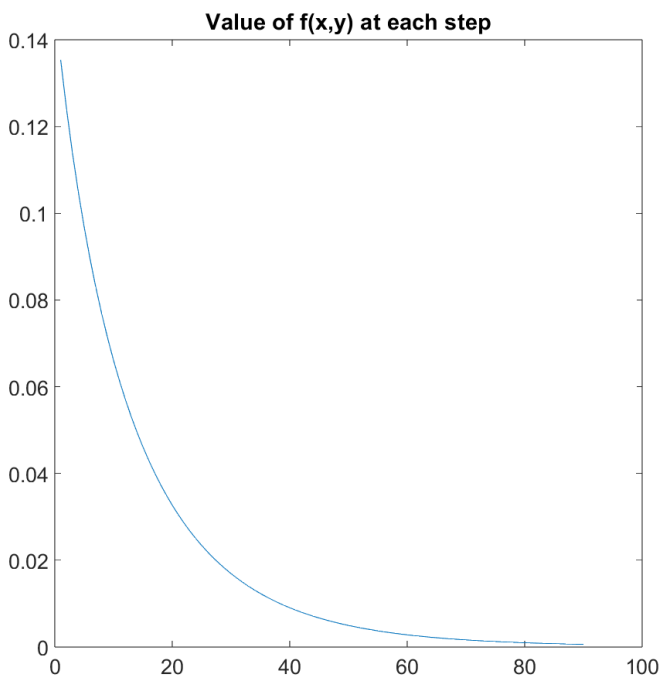


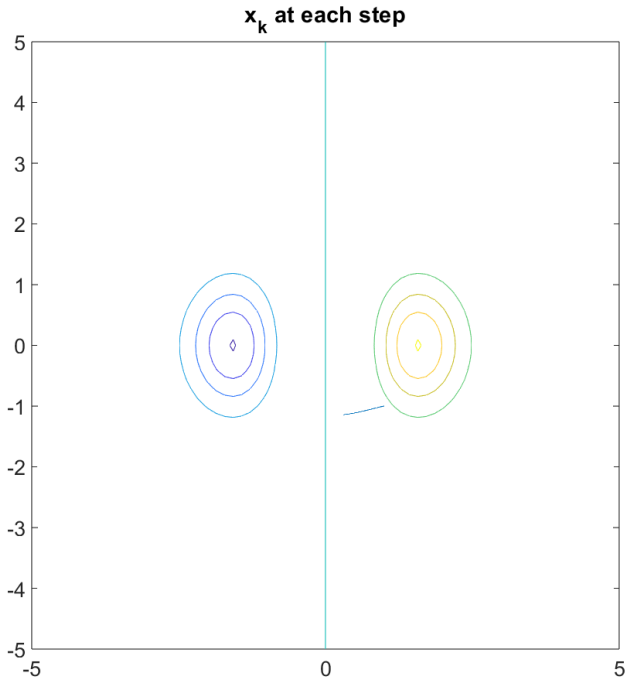
Finally, we choose $(x_0, y_0) = (1, -1)$ as the starting point.

Again we observe identical behavior to the Maximum Descent method, with x_k locking

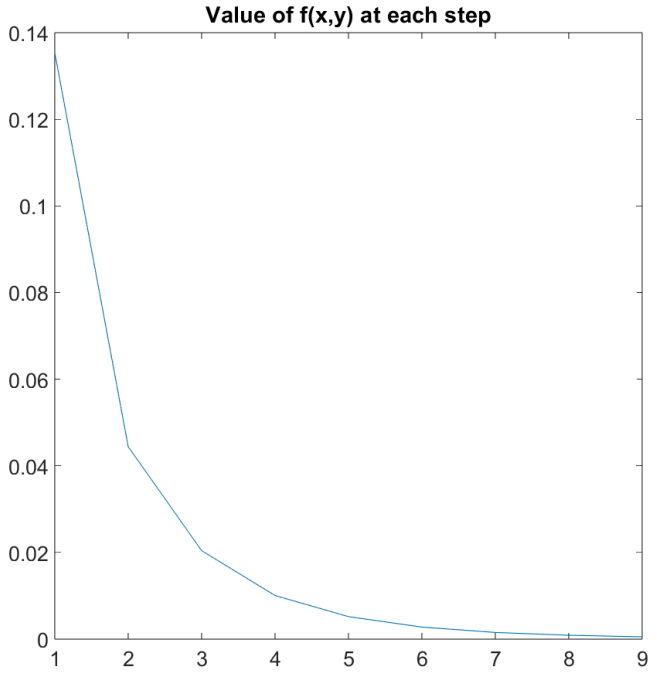
into some local minimum. The same is true of the execution time, with the constant γ_k being the fastest algorithm. Below are the corresponding figures.

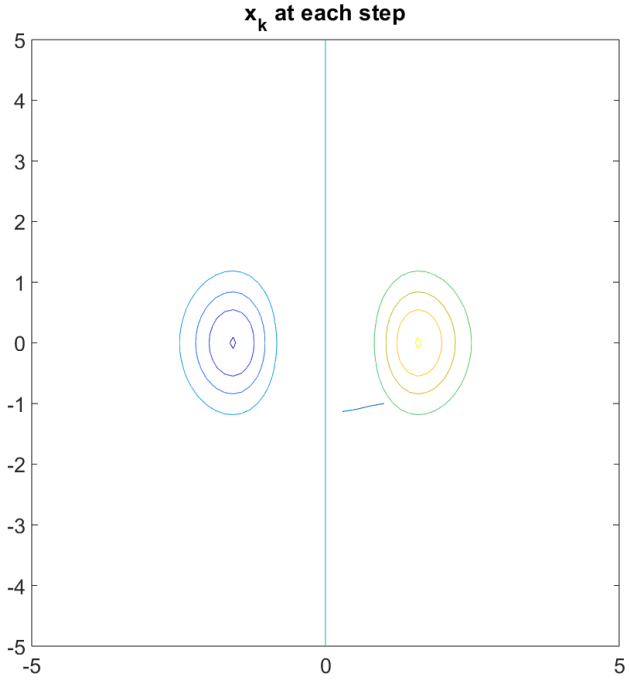
Constant $\gamma_\mathbf{k}$:



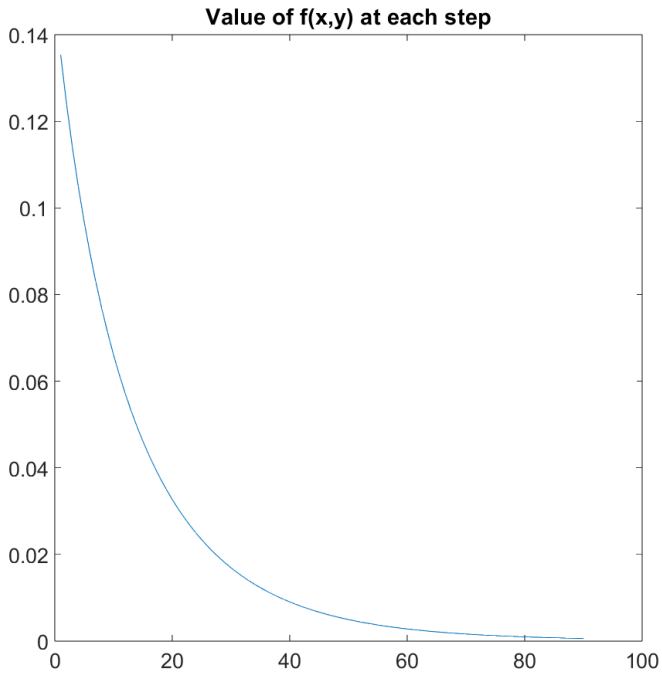


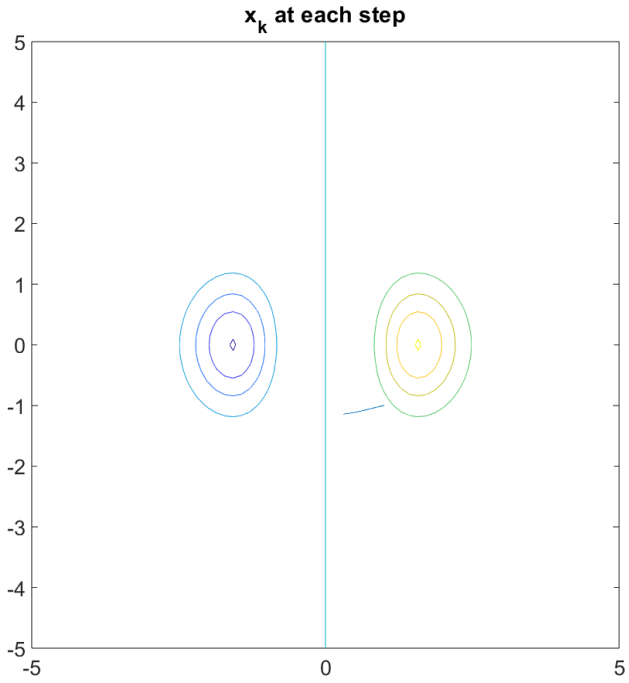
Choose γ_k to minimize $\mathbf{f(x_k + \gamma_k \delta_k)}$:





Choose γ_k with rule Armijo:





5 Conclusions

Initially comparing the methods with each other, it is very easy to rule out the Newton method, while we are free to choose any of the Maximum Descent and Levenberg-Marquardt methods, which work best producing identical results, with similar execution time. Another separation we can make is based on the choice of γ_k. As we observed above, although for fixed γ_k we need more iterations, the low complexity of the algorithm makes it faster than the other 2, so it is the optimal choice, in case it does not cause convergence problems. Finally, through our tests we understood that in methods, such as the above, the starting point of the algorithm is particularly important, since a bad choice of it can lead to getting trapped in some local minimum.