# One-Dimensional Line Search Methods Assignment

## Advanced Design 814

Andreas Joss

16450612

March 12, 2015

# 1 Introduction

This report briefly explains the three different one-dimensional line search methods which are used for this assignment. Furthermore, the algorithms are implemented in code and the results thereof are shown and discussed.

# 2 Theory

## 2.1 Newton-Raphson Method

The Newton-Raphson method can be derived from the Taylor Series expansion which is used to approximate functions. The Newton-Raphson method is described as:

$$x^* \approx x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)} \tag{1}$$

where $x^{k+1}$ describes the next solution and $x^k$ describes the present solution.

## 2.2 Newton-Raphson Method Modified

The purpose of this modification is to ensure that a solution is found for the given problem function, for cases where the normal Newton-Raphson method would diverge. This modified method might require more iterations than the unmodified method, nevertheless convergence is more likely.

The modified Newton-Raphson method is described as:

$$x^* \approx x^{k+1} = x^k - \frac{f'(x^k)}{\max(f''(x^k), \epsilon)} \tag{2}$$

For the denominator of Equation 5, the maximum value is selected being either the second derivative of the function at the current solution, or a value $\epsilon$ which is normally selected in the order of $\epsilon = 10^{-6}$. One problem arises in the unmodified Newton-Raphson method (Equation 1) when the denominator is equal to zero or very close to zero. This causes the newly calculated solution $x^{k+1}$ to assume a value placed very far, or even infinitely far, from the previous solution $^k$, which as a result makes it very hard for the unmodified Newton-Raphson method to converge. The modification of the Newton-Raphson method ensures that the denominator is at least as large as $\epsilon$. The second problem of the unmodified method, is when the second derivative is found to be $f^{''}(x^k) < 0$. This means that the function is non-convex (concave) in that region and that the next solution will move closer to a non-converging region. The modified method, would then prevent the next solution to move into the aforementioned direction, by replacing $f^{''}(x^k)$ with a small positive value $\epsilon$.

## 2.3 Golden Section Method

For this method to be useful, it is assumed that the objective function $f(x)$ has a minimum within an interval $[a, b]$ (which is selected from user experience), and that $f(x)$ is descending for $x < x^*$ and $f(x)$ is ascending for $x > x^*$. New points $x_1$ and $x_2$ are selected when certain criteria are met.

If $f(x_2 > f(x_1$ then the new interval becomes $[a, x_2]$ with $b$ obtaining the new boundary value, otherwise the new interval will be $[x_1, b]$ with $a$ obtaining the new boundary value. Thus, the interval is reduced until $x^*$ is reached within a prescribed accuracy. The ratio by which the interval is reduced, is determined by the golden ratio $r = 0.618034$. The algorithm steps are explained in full by [1:37].

# 3 Selecting the Appropriate Method

## 3.1 Newton-Raphson Methods

From Equation 1 and 5 it is clear that 1st and 2nd order derivative information of the objective test function is required to use the method. Depending on the problem at hand, it is up to the user to decide how practical this method is regarding the expensiveness to obtain this information. Even if the 1st and 2nd order derivatives are obtained, the method could still struggle to deliver a global minimum due to its susceptibility to noise. For the modified Newton-Raphson method, values for $\epsilon$ can be chosen experimentally for specific problems. If the method still diverges, then a larger $\epsilon$ value could be selected.

## 3.2 Golden Section Method

From Equation 3 it is seen that only zero order information is required from the objective test function. This is useful when the 1st and 2nd order derivative information is hard and expensive to obtain, but zero order information is more accessible. This method is consequently much less susceptible to noise.

# 4 Example Functions

Example functions are used to evaluate the effectiveness of each algorithm and briefly compare the algorithm results with each other. Each graph displays several parameters for the particular optimization at hand as well as the solution that is found. Each test function has its own specified boundaries and also the tolerance of the accuracy of the desired solution. The following test functions are used:

(i) minimize $F(\lambda) = \lambda^2 + 2e^{-\lambda}$ over $[0, 2]$ with $\epsilon = 0.01$

(ii) maximize $F(\lambda) = \lambda \cos\lambda$ over $[0, \frac{\pi}{2}]$ with $\epsilon = 0.001$

(iii) minimize $F(\lambda) = 4(\lambda - 7)/(\lambda^2 + \lambda - 2)$ over $[-1.9, 0.9]$ with  by performing no more than 10 function iterations.

(i) minimize $F(\lambda) = \lambda^4 + -20\lambda^3 + 0.1\lambda$ over $[0, 20]$ with $\epsilon = 10^{-5}$

# 5 Results

The algorithm are programmed in the Python language, due to future work which will also require extensive Python scripting. The Matplotlib library is used to display the results in a neat and informative manner. In each case the green dot indicates the solution that is found by the algorithm.

## 5.1 1-D line search based on the Newton-Raphson Method

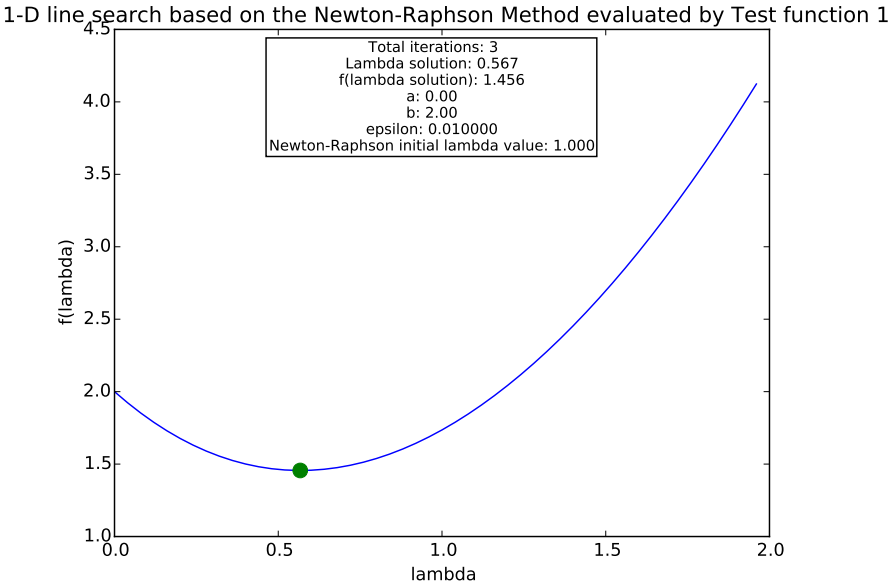


Figure 1: 1-D line search based on the Newton-Raphson Method evaluated by first example function

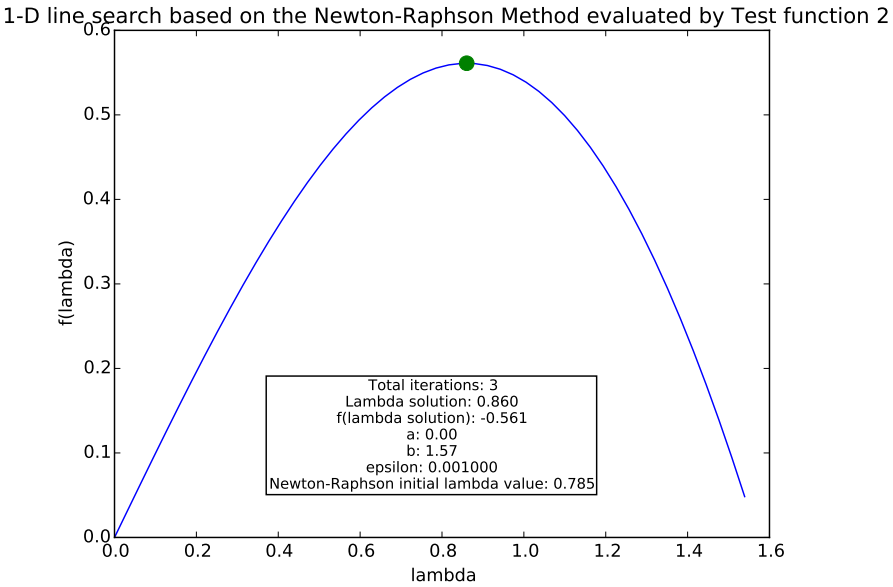


Figure 2: 1-D line search based on the Newton-Raphson Method evaluated by second example function

Figure 1 and Figure 2 show that this algorithm manages to successfully converge near or on the optimal point. Note for both test functions, only three iterations are required.

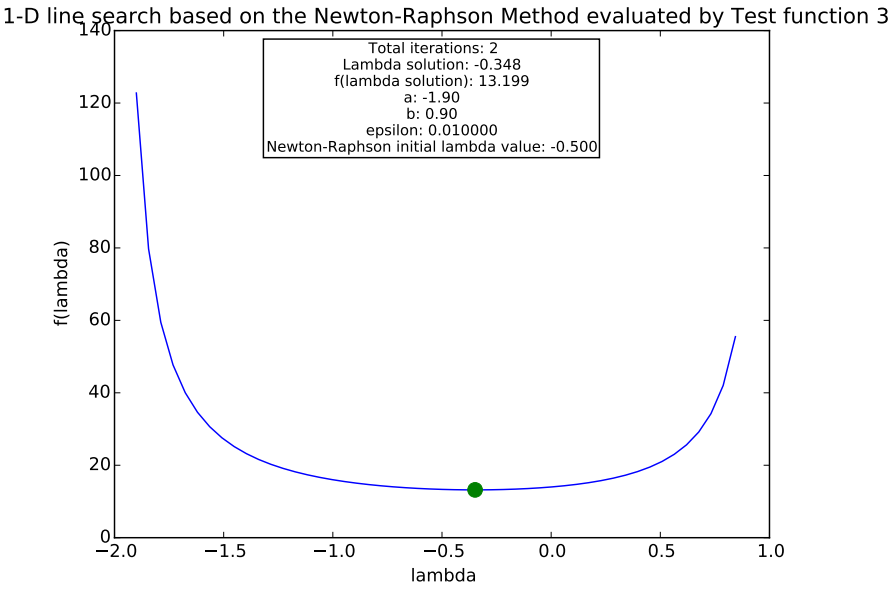


Figure 3: 1-D line search based on the Newton-Raphson Method evaluated by third example function

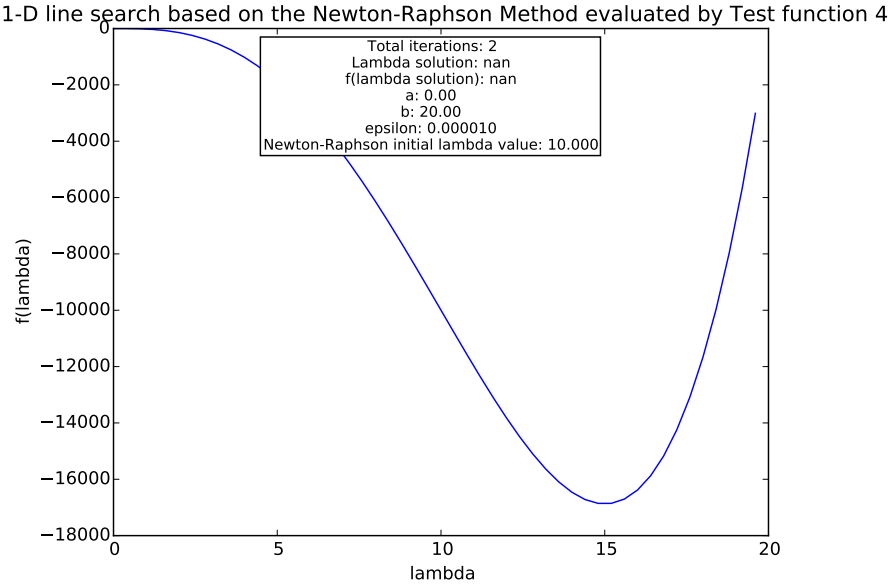


Figure 4: 1-D line search based on the Newton-Raphson Method evaluated by fourth example function

Figure 3 indicates that this algorithm manages to successfully converge near or on the optimal point. Note that only two iterations are required to adhere to the tolerance value. From Figure 4, it is clear that the algorithm did not converge to a solution. This is due to the fact that the initial starting solution is not chosen adequately for this particular problem.

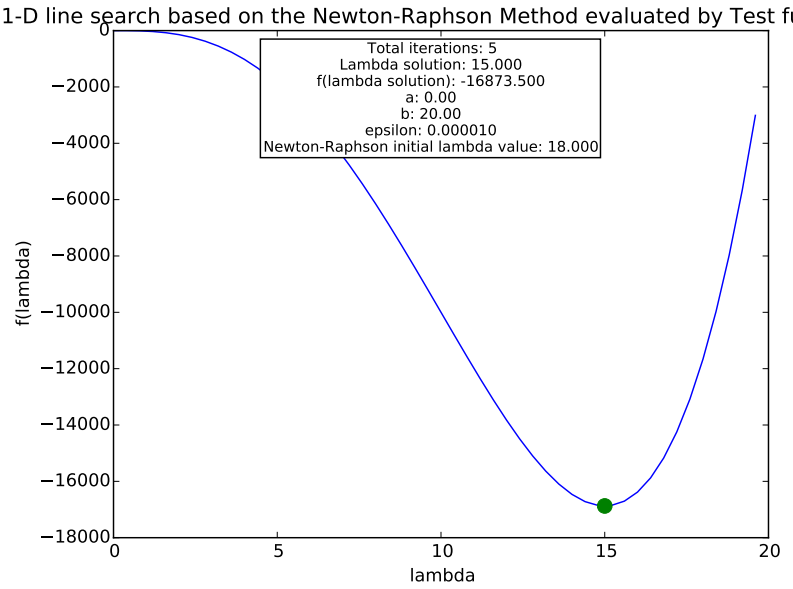


Figure 5: 1-D line search based on the Newton-Raphson Method evaluated by fourth example function, with a new starting solution selected

This time, from Figure 5, it is visible that the algorithm successfully converged to a solution. This is because a different starting estimated solution is assumed compared to the estimated starting solution of Figure 4.

## 5.2 Modification of the Newton-Raphson was discussed in class

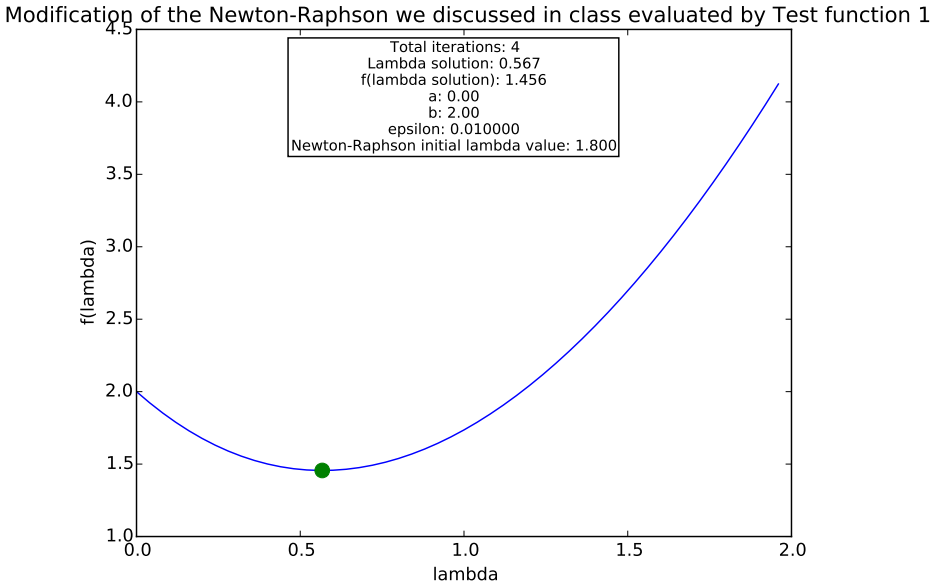


Figure 6: Modification of the Newton-Raphson evaluated by first example function

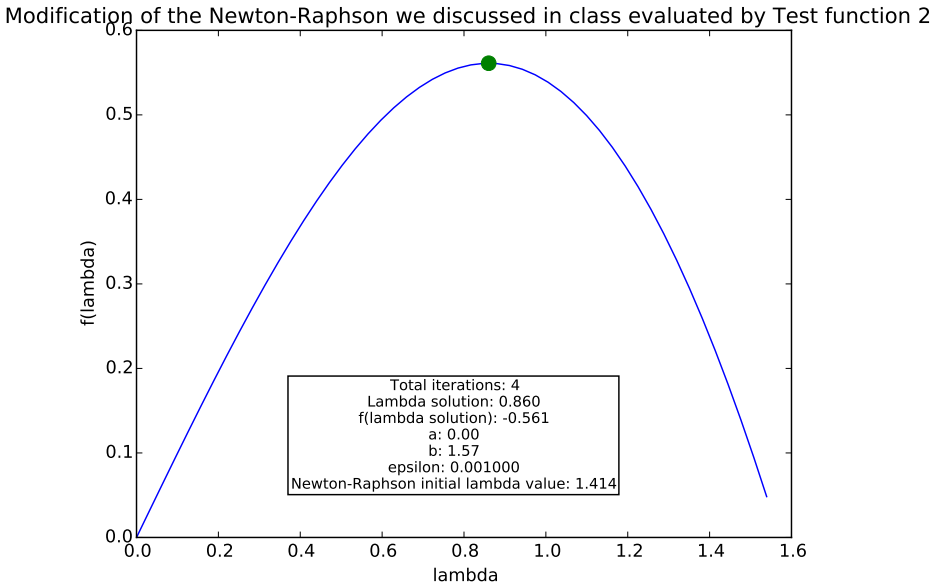


Figure 7: Modification of the Newton-Raphson evaluated by second example function

Figure 6 and Figure 7 show that this algorithm manages to successfully converge near or on the optimal point. Note for both test functions, only three iterations are required.

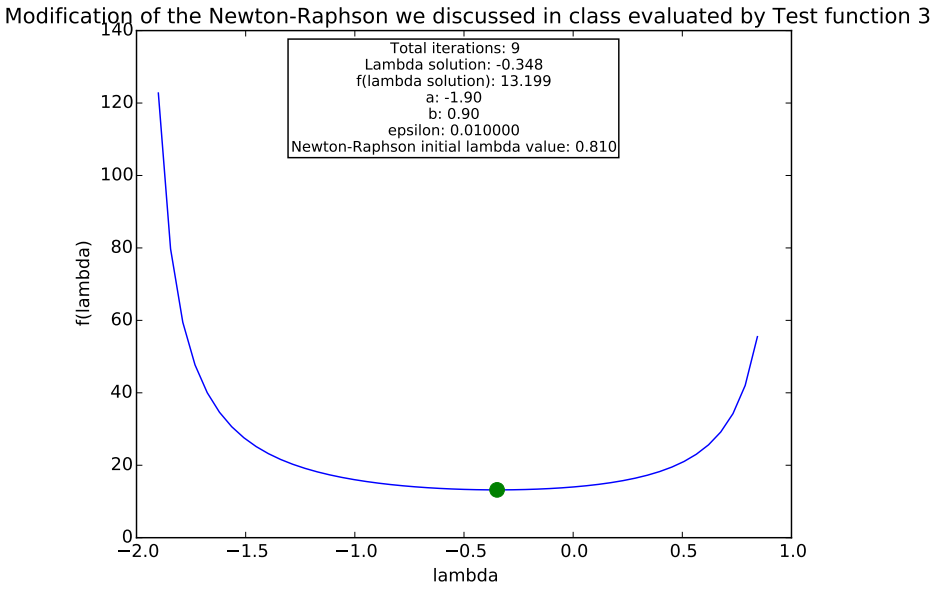


Figure 8: Modification of the Newton-Raphson evaluated by third example function

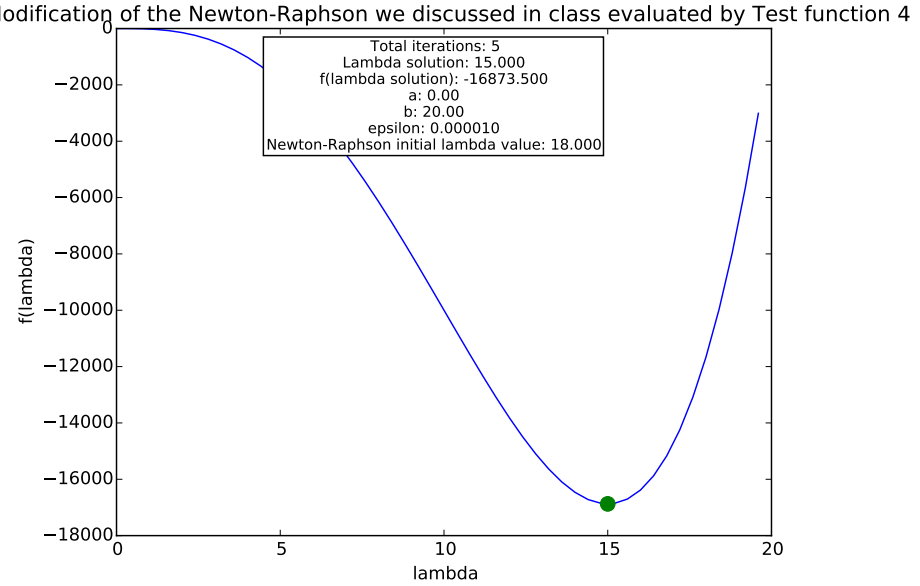


Figure 9: Modification of the Newton-Raphson evaluated by fourth example function

Figure 8 indicates that this algorithm manages to successfully converge near or on the optimal point. Note that only two iterations are required to adhere to the tolerance value. From Figure 9, it is clear that the algorithm did not converge to a solution. This is due to the fact that the initial starting solution is not chosen adequately for this particular problem.

## 5.3 Golden Section Method

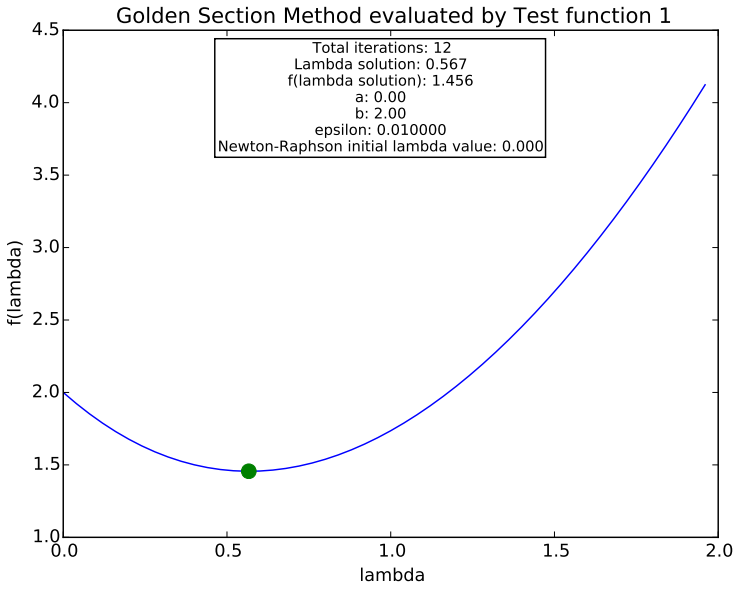


Figure 10: Golden Section Method evaluated by first example function

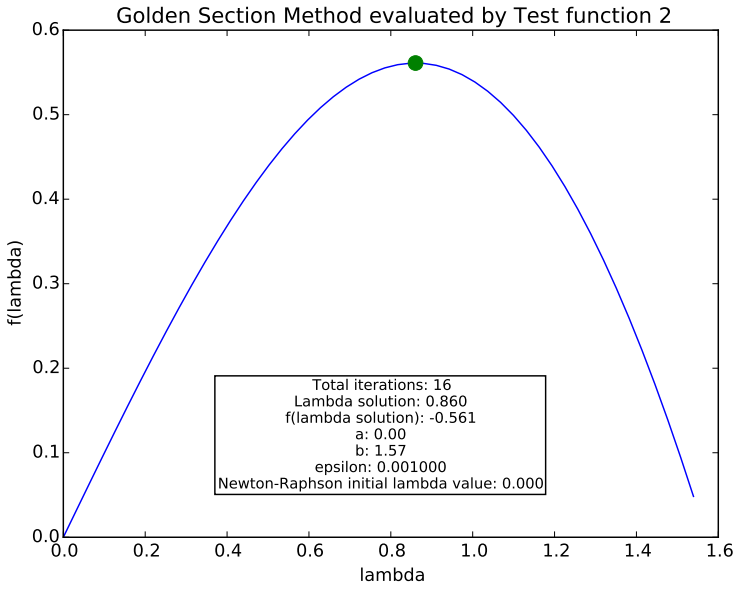


Figure 11: Golden Section Method evaluated by second example function

Figure 10 and Figure 11 show that this algorithm manages to successfully converge near or on the optimal point. Note that much more iterations are required to achieve the equal accuracy of solutions as indicated by Figure 1 and Figure 2.

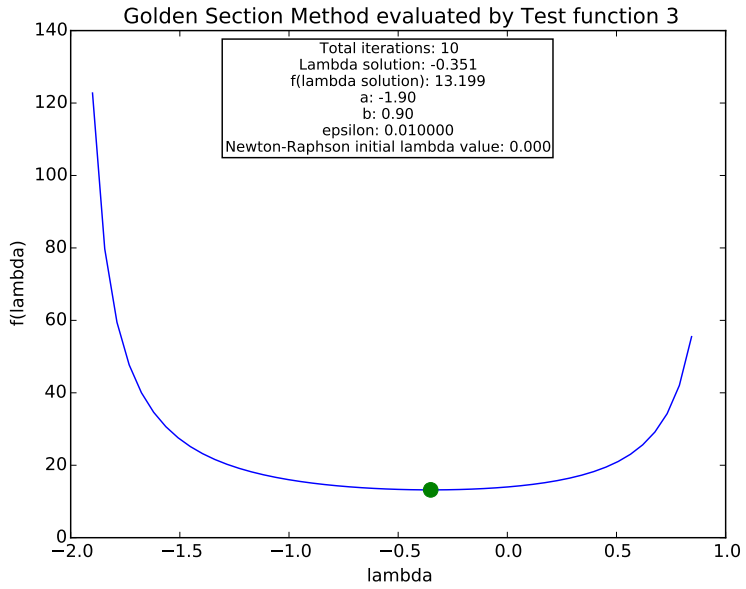


Figure 12: Golden Section Method evaluated by third example function

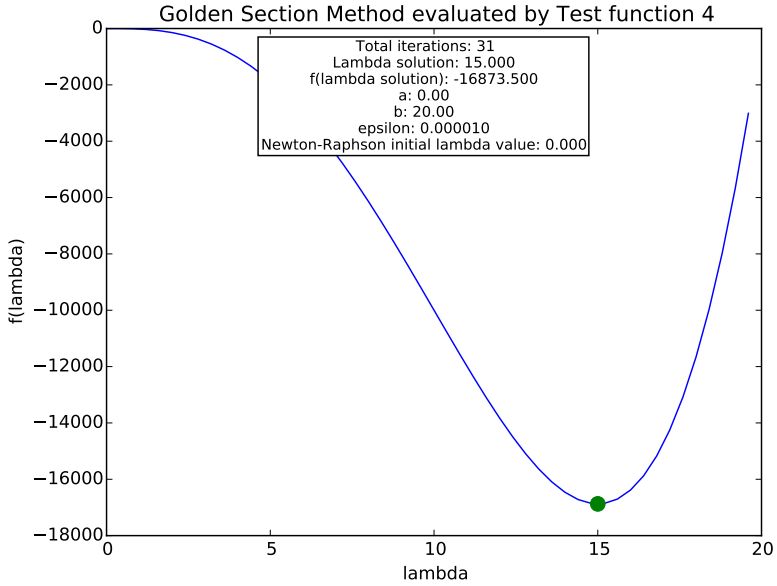


Figure 13: Golden Section Method evaluated by fourth example function

Figure 12 indicates that this algorithm manages to successfully converge near or on the optimal point. It is also shown that the number of iterations are limited to 10 for the third test function. From Figure 13, it is clear that the algorithm did converge to a solution. Notice that 31 iterations are required to obtain the solution, as compared to Figure 5 where only 5 iterations are required (given the starting point chosen for that example).

# 6 Conclusion

From the results it seems that the PSO algorithm is a handy tool to use when a cost function of 10-20 variables need to be optimized. Some of the example functions in this report, such as the Quadric function are evidently more difficult for the PSO algorithm to solve than for instance the Rosenbrock function. Reasonable solutions could probably still be obtained for a 10-20 dimensional problem but it does not seem that this algorithm would be sufficient for a 30 dimensional problem. It is also noted that an increase of the population size seems to result in better solutions or at least faster convergence to some local minimums. Furthermore, it is interesting to note how sensitive the example problems and the PSO algorithm is toward the inertia factor. A slight increase in the inertia factor often resulted in a diverging solution.

All considered the PSO algorithm is relatively easy to implement and is worth a try to solve a multi-dimensional optimization problem.

# 7 Bibliography

1. Wilke DN, Kok S, Groenwold AA. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. International Journal for Numerical Methods in Engineering 2006; 70:962-984.