POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Matematica

Report NO4LSCP

Chiodo Martina - 343310
Vigè Sophie - 339268

Anno Accademico 2024-2025

INTRODUCTION

DIRE QUALCOSA SU COME SI CALCOLA ROC

The experimental rate of convergence can be approximated by

$$q \approx \frac{\log\left(\frac{\|\hat{e}^{(k+1)}\|}{\|\hat{e}^{(k)}\|}\right)}{\log\left(\frac{\|\hat{e}^{(k)}\|}{\|\hat{e}^{(k-1)}\|}\right)} \quad \text{for } k \text{ large enough} \tag{1}$$

where $\hat{e}^{(k)} = x^{(k)} - x^{(k-1)}$ approximates the error at the k-th iteration (if the exact solution is unknown).

Specificare gli STOPPING CRITERION per gli algoritmi

PROBLEM 25

Model

The function described in this problem is the following

$$
\begin{aligned}
F(\mathbf{x}) &= \frac{1}{2}\sum_{k=1}^{n} f_k^2(x) \\
f_k(\mathbf{x}) &= 10\left(x_k^2 - x_{k+1}\right), && \mod(k,2) = 1 \\
f_n(\mathbf{x}) &= x_{k_1} - 1, && \mod(k,2) = 0
\end{aligned}
$$

where n denotes the dimensionality of the input vector $\mathbf{x}$. As convention, we set $x_{n+1} = x_1$ when it is necessary, that is when the dimensionality n is an odd number.
The starting point for the minimization is the vector $\mathbf{x}_0 = [-1.2, 1, -1.2, 1, \ldots]$.

In order to compute the derivatives of this problem we have to consider separately the cases when n is even or odd. In the first case, the gradient of the function is given by

$$
\begin{aligned}
\frac{\partial F}{\partial x_k}(\mathbf{x}) &= \frac{\partial}{\partial x_k}\left[\frac{1}{2}f_{k-1}^2(\mathbf{x})\right] = -100(x_{k-1}^2 - x_k) && \mod(k,2) = 0 \\
\frac{\partial F}{\partial x_k}(\mathbf{x}) &= \frac{\partial}{\partial x_k}\left[\frac{1}{2}f_k^2(\mathbf{x}) + \frac{1}{2}f_{k+1}^2(\mathbf{x})\right] = 200x_k(x_k^2 - x_{k+1}) + (x_k - 1) && \mod(k,2) = 1
\end{aligned}
$$

If the dimensionality n is odd, the only changement is in the first component of the gradient, which becomes

$$
\frac{\partial F}{\partial x_1}(\mathbf{x}) = \frac{\partial}{\partial x_k}\left[\frac{1}{2}f_k^2(\mathbf{x}) + \frac{1}{2}f_{k+1}^2(\mathbf{x}) + \frac{1}{2}f_n^2(\mathbf{x})\right] = 200x_1(x_1^2 - x_2) + (x_1 - 1) - 100(x_n^2 - x_1)
$$

Looking at the structure of the problem we are considering, it is obvious that the Hessian matrix is a sparse matrix whose particular structure depends again on wheter n is even or odd. In the first case, the Hessian is a block tridiagonal matrix with the following non-zero terms

$$
\begin{aligned}
&\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) = 100, && \frac{\partial^2 F}{\partial x_k \partial x_{k+1}}(\mathbf{x}) = 0, && \frac{\partial^2 F}{\partial x_k \partial x_{k-1}}(\mathbf{x}) = -200x_{k-1} && \mod(k,2) = 0 \\
&\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) = 600x_k^2 - 200x_{k+1} + 1, && \frac{\partial^2 F}{\partial x_k \partial x_{k+1}}(\mathbf{x}) = -200x_k, && \frac{\partial^2 F}{\partial x_k \partial x_{k-1}}(\mathbf{x}) = 0 && \mod(k,2) = 1
\end{aligned}
$$

If n is odd, there are two changements in the Hessian matrix: the derivative $\frac{\partial^2 F}{\partial x_1^2}(\mathbf{x})$ is affected by the presence of x_1 in the term $f_n()$ and the extremal diagonals are not zero anymore. We report the terms of the Hessian matrix that differs from the previous case

$$
\begin{aligned}
\frac{\partial^2 F}{\partial x_1^2}(\mathbf{x}) &= 600x_k^2 - 200x_{k+1} + 101 \\
\frac{\partial^2 F}{\partial x_n \partial x_1}(\mathbf{x}) &= \frac{\partial^2 F}{\partial x_1 \partial x_n}(\mathbf{x}) = -200x_n
\end{aligned}
$$

By analyzing the derivatives of the problem, we can deduce that the gradient of the function is nullified by the vector composed of ones which also nullifies the value of $F(\mathbf{x})$.

Nealder Mead Method

	avg fbest	avg num of iters	avg time of exec (sec)	n failure	avg roc
10	3.202	85.273	2.8171	0	NaN
25	8.7449	206	4.764	0	9.8503
50	14.633	350.18	7.5195	0	NaN

Figura 1: Results obtained by running the symplex method on the problem 25.

We now report a table showing some general results obtained by running the Nealder Mead method on the problem.

Looking at the table, it is clear that even if the method satisfies the stopping criterion for all dimensionalities it does not reach the point we declared be a global minimum, actually the minimum value the algorithm finds increases with the dimensionality. We are not too surprised by the por performance of the symplex method because the algorithm solely relies on function evaluation and does not take advantage of the information contained in the derivatives of the objecive function.

Modified Newton Method - Exact Derivatives

We now report a table showing some general results obtained by running the Modified Newton method on the problem.

	avg fbest	avg gradf_norm	avg num of iters	avg time of exec (sec)	n failure	avg roc
1000	4.28e-11	2.0733e-05	25.727	2.3163	0	5.9605
10000	4.2443e-10	1.1244e-05	26.818	1.8316	0	6.1992
100000	2.8365e-14	2.2207e-06	27.636	3.8631	0	1.3941

Figura 2: Results obtained by running the Modified Newton method on the problem 25 using the exact derivatives.

As expected from the theoretical background we have about these methods, the Modified Newton method performs significantly better than the Nealder Mead method. The table shows that the method converges to a point in which the norm of the gradient is below the fixed tollerance for all dimensionalities tested, and the minimum value found is consistently close to zero. This is because the Modified Newton method leverages the gradient and Hessian information, allowing it to make more informed steps towards the minimum and thus to converge in fewer iterations.

However, we can notice that the ratio between the average time of execution and the average number of iterations is smaller for the symplex method. This means that each iteration performed by the Modified Newton method is more high-performance but also more costly in terms of computational effort.

Modified Newton Method - Approximated Derivatives

Approximating the derivatives of the function $F(\mathbf{x})$ using finite differences is more challenging than it appears due to potential numerical cancellation issues, which can occur when subtracting two nearly equal quantities. Additionally, we aim to derive a formula that minimizes computational cost.

As done previously we will first consider the case in which the dimensionality n is an even integer and then we will specify what changes if n is an odd number.

Let's begin by approximating the first order derivatives by using the centered finite difference formula with increment h_k. We keep track of the subscript k in order to derive formula which are valid both for the case with costant increments and the case in which the increments depend on the components respect to which we are differentiating. The general formula is

$$\frac{\partial F}{\partial x_k}(\mathbf{x}) \approx \frac{F(\mathbf{x}+h_k\vec{e}_k) - F(\mathbf{x}-h_k\vec{e}_k)}{2h_k} = \frac{\sum_{i=1}^{n} f_i(\mathbf{x}+h_k\vec{e}_k)^2 - \sum_{i=1}^{n} f_i(\mathbf{x}-h_k\vec{e}_k)^2}{4h_k}$$

but it would not be much wise to apply it directly to out problem because it would be unnecessary to evaluate all the terms $f_i^2(\mathbf{x})$ which are not affected by the varation of the k-th component of the vector $\mathbf{x}$. In particular, we can notice that if we are differentiating with respect to an even component the only term we need to compute is $f_{k-1}^2()$, while if we are differentiating with respect to an odd component we only need to expand the terms $f_k^2()$ and $f_{k+1}^2()$. Omitting the calculus, we obtain the following formula

$$\begin{aligned}
\frac{\partial F}{\partial x_k}(\mathbf{x}) &\approx \frac{f_{k-1}^2(\mathbf{x}+h_k\vec{e}_k) - f_{k-1}^2(\mathbf{x}-h_k\vec{e}_k)}{4h_k} = \frac{-40h_k(10x_{k-1}^2 - 10x_k)}{4h_k} && \text{mod } (k,2) = 0 \\
\frac{\partial F}{\partial x_k}(\mathbf{x}) &\approx \frac{f_k^2(\mathbf{x}+h_k\vec{e}_k) - f_k^2(\mathbf{x}-h_k\vec{e}_k) + f_{k+1}^2(\mathbf{x}+h_k\vec{e}_k) - f_{k+1}^2(\mathbf{x}-h_k\vec{e}_k)}{4h_k} \\
&= \frac{80x_kh_k(10x_k^2 + 10h_k^2 - 10x_{k+1}) - 4h_k(x_k-1)}{4h_k} && \text{mod } (k,2) = 1
\end{aligned}$$

If n is an odd number, the approximation of $\frac{\partial F}{\partial x_1}(\mathbf{x})$ will slightly change into

$$\begin{aligned}\frac{\partial F}{\partial x_1}(\mathbf{x}) &\approx \frac{f_1^2(\mathbf{x}+h_1\vec{e}_1) - f_1^2(\mathbf{x}-h_1\vec{e}_1) + f_2^2(\mathbf{x}+h_1\vec{e}_1) - f_2^2(\mathbf{x}-h_1\vec{e}_1) + f_n^2(\mathbf{x}+h_1\vec{e}_1) - f_n^2(\mathbf{x}-h_1\vec{e}_1)}{4h_1} \\ &= \frac{80x_1h_1(10x_1^2+10h_1^2-10x_2) - 4h_1(x_1-1) - 40h_1(10x_n^2-10x_1)}{4h_1}\end{aligned}$$

For what concerns the second order derivatives, we can apply a similar reasoning based on negletting the terms $f_i^2(\mathbf{x})$ which are not affected by the variation of the k-th component of $\mathbf{x}$ but starting from the general formula

$$\frac{\partial^2 F}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{F(\mathbf{x}+h_i\vec{e}_i+h_j\vec{e}_j) - F(\mathbf{x}+h_i\vec{e}_i) - F(\mathbf{x}-h_j\vec{e}_j) + F(\mathbf{x})}{h_ih_j}$$

Due to the particular structure of the problem we are considering, many second order derivatives are zero, thus we are going to approximate solely the ones we know are not null.

$$\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) \approx \frac{f_{k-1}^2(\mathbf{x}+2h_k\vec{e}_k) - 2f_{k-1}^2(\mathbf{x}+h_k\vec{e}_k) + f_{k-1}(\mathbf{x})}{2h_k^2} = \frac{200h_k^2}{2h_k^2}, \qquad \text{mod } (k,2) = 0$$

$$\begin{aligned}\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) &\approx \frac{f_k^2(\mathbf{x}+2h_k\vec{e}_k) - 2f_k^2(\mathbf{x}+h_k\vec{e}_k) + f_k(\mathbf{x}) + f_{k+1}^2(\mathbf{x}+2h_k\vec{e}_k) - 2f_{k+1}^2(\mathbf{x}+h_k\vec{e}_k) + f_{k+1}(\mathbf{x})}{2h_k^2} \\ &= \frac{40h_k^2(10x_k^2-10x_{k+1}) + 1400h_k^4 + 2400h_k^3x_k + 800x_kh_k^2 + 2h_k^2}{2h_k^2}, \qquad \text{mod } (k,2) = 1\end{aligned}$$

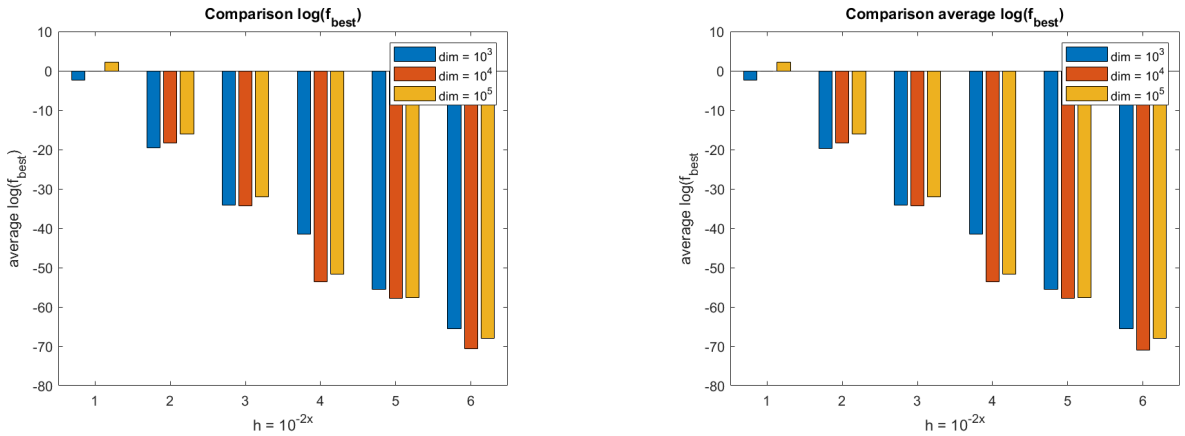
$$\begin{aligned}\frac{\partial^2 F}{\partial x_k \partial x_{k+1}}(\mathbf{x}) &\approx \frac{f_k^2(\mathbf{x}+h_k\vec{e}_k+h_{k+1}\vec{e}_{k+1}) - f_k^2(\mathbf{x}+h_k\vec{e}_k) - f_k^2(\mathbf{x}+h_{k+1}\vec{e}_{k+1}) + f_k(\mathbf{x})}{2h_kh_{k+1}} \\ &= \frac{20h_{k+1}(-10h_k^2-20h_kx_k)}{2h_kh_{k+1}}, \qquad \text{mod } (k,2) = 1\end{aligned}$$

We explicitely approximated just the superior diagonal, the inferior one is obtained by imposing the simmetry of the hessian matrix.

If the dimensionality n is an odd number, the changes only concern the term $\frac{\partial^2 F}{\partial x_1^2}(\mathbf{x})$ and the extremal diagonal which are not null anymore. In particular, these terms become

$$\begin{aligned}\frac{\partial^2 F}{\partial x_1^2}(\mathbf{x}) &\approx \frac{40h_1^2(10x_1^2-10x_2) + 1400h_1^4 + 2400h_1^3x_1 + 800x_1h_1^2 + 202h_1^2}{2h_k1^2} \\ \frac{\partial^2 F}{\partial x_n \partial x_1} &= \frac{\partial^2 F}{\partial x_1 \partial x_n} \approx \frac{20h_1(-20x_nh_n - 10h_n^2)}{2h_1h_n}\end{aligned}$$

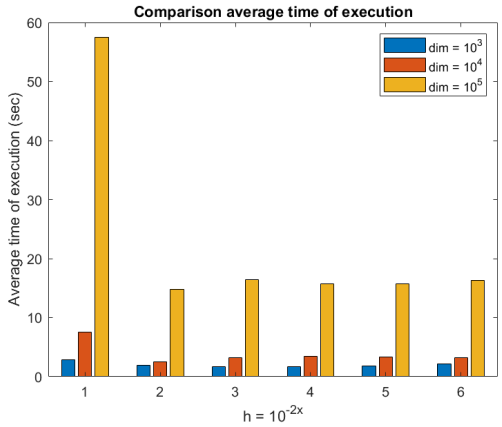
According to what we expect, seeking for the minimum using the approximations of the derivatives affects the performance of the Modified Newton method, expecially for larger values of the increment h. In fact, from the theory, we know that the finite difference formula approximates the analytical derivative with an error that depends on the value of the increment h. Specifically, the error diminishes as h approaches 0.



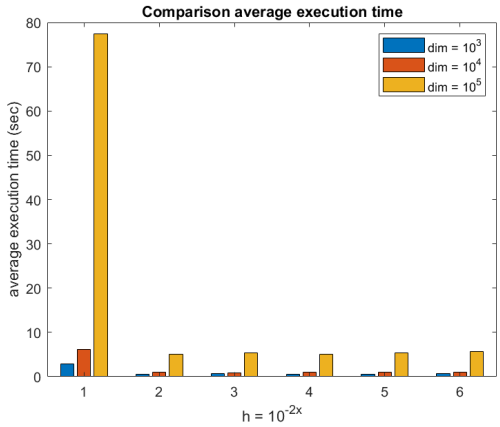
(a) Costant Increment h (b) Specific Increment

Figura 3: Values of the average $\log(f_{best})$ in function of the increment while running the Modified Newton Method with approximated derivatives on the problem 25.

Therefore, it is not surprising that for $h = 10^{-2}$, the algorithm often converges to a point whose value is not very close to 0 and requires significantly more iterations to meet the stopping criterion. This is clearly shown in the following bar plots (Figure 3), which display the average value of $\log(f_{best})$, where f_{best} is the minimum found by the Modified Newton method, as a function of the increment h used to approximate the derivatives. Notice that we applied a logarithmic transformation to the value of f_{best} for clarity, as the values were close to 0. As we can see from the barplot, as the value of h diminish the minimum found is smaller (i.e. $\log(f_{best})$ increases in absolute value while being a negative quantity) as a consequence of the more precise approximations of the derivatives the Modified Newton method uses.

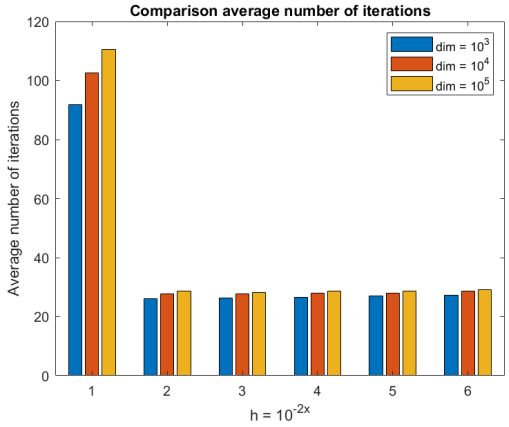


(a) Costant Increment h

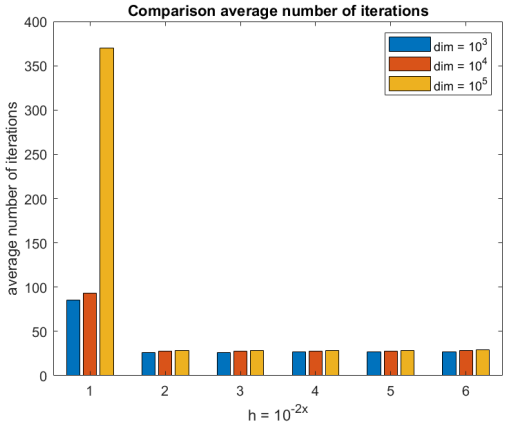


(b) Specific Increment

Figura 4: Average time of execution in function of the increment h while running the Modified Newton Method with approximated derivatives on the problem 25.



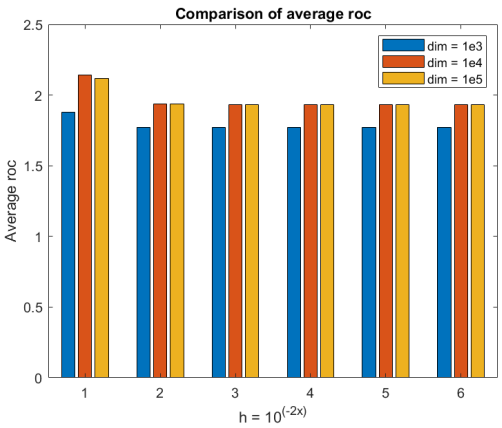
(a) Costant Increment h



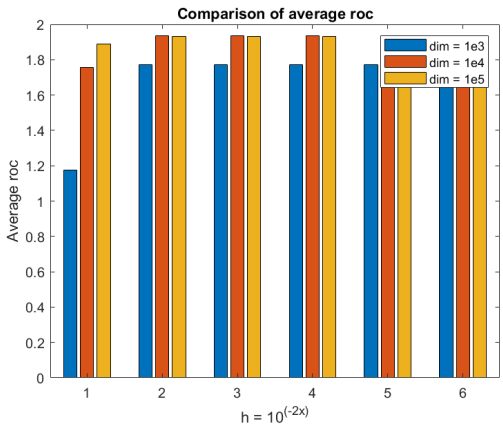
(b) Specific Increment

Figura 5: Average number of iterations in function of the increment h while running the Modified Newton Method with approximated derivatives on the problem 25.

It can be intresting notice from the barplots comparing the average number of iterations needed by the Modified Newton method (Figure (5)) that a specific increment based on the point in which we are approximating the derivative seems to make the method perform better than using the costant increment.



(a) Costant Increment h



(b) Specific Increment

Figura 6: Average values of the experimental rate of convergence in function of the increment h while running the Modified Newton Method with approximated derivatives on the problem 25.

PROBLEM 75

Model

The function described in this problem is the following

$$
\begin{aligned}
F(\boldsymbol{x}) &= \frac{1}{2}\sum_{k=1}^{n} f_k^2(\boldsymbol{x}) \\
f_k(\boldsymbol{x}) &= x_k - 1, && k = 1 \\
f_k(\boldsymbol{x}) &= 10(k-1)(x_k - x_{k-1})^2, && 1 < k \leq n
\end{aligned}
$$

where n is the length of the input vector $\boldsymbol{x}$. With the given starting point for minimization being

$$
\boldsymbol{x_0} = [-1.2, -1.2, ..., -1.2, -1]' \in \mathbb{R}^n.
$$

The gradient of $F(\boldsymbol{x})$ is the following (note that, besides the first and last components, all the others have the same structure).

$$
\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_k}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}\frac{1}{2}(f_1^2 + f_2^2)(\boldsymbol{x}) \\ \vdots \\ \frac{\partial F}{\partial x_k}\frac{1}{2}(f_k^2 + f_{k+1}^2)(\boldsymbol{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}\frac{1}{2}f_n^2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1 - 1 - 200 \cdot (x_2 - x_1)^3 \\ \vdots \\ 200 \cdot \Big((k-1)^2(x_k - x_{k-1})^3 - k^2(x_{k+1} - x_k)^3\Big) \\ \vdots \\ 200 \cdot (n-1)^2(x_n - x_{n-1})^3 \end{bmatrix}
$$

The Hessian matrix of $F(\boldsymbol{x})$ is sparse since only on three diagonals elements different from zeros are present. They are the following:

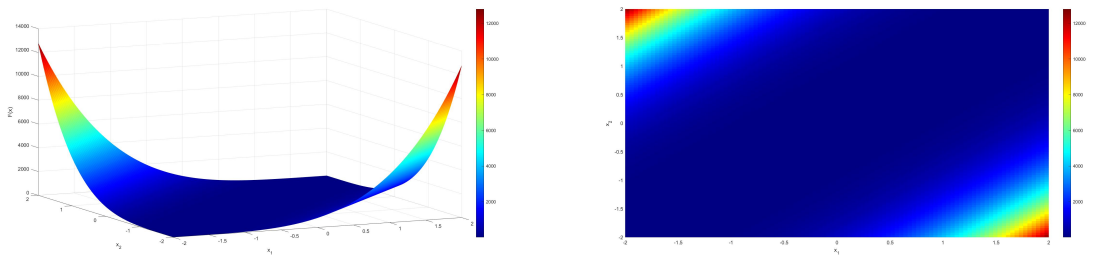
$$
\begin{aligned}
\frac{\partial^2 F}{\partial x_1^2}(\boldsymbol{x}) &= 1 + 600 \cdot (x_2 - x_1)^2 \\
\frac{\partial^2 F}{\partial x_k^2}(\boldsymbol{x}) &= 600 \cdot \Big((k-1)^2(x_k - x_{k-1})^2 + k^2(x_{k+1} - x_k)^2\Big), \quad 1 < k < n \\
\frac{\partial^2 F}{\partial x_n^2}(\boldsymbol{x}) &= 600 \cdot (n-1)^2(x_n - x_{n-1})^2 \\
\frac{\partial^2 F}{\partial x_k \partial x_{k-1}}(\boldsymbol{x}) &= -600 \cdot (k-1)^2(x_k - x_{k-1})^2, \quad 1 < k \leq n.
\end{aligned}
$$

It is easy to notice that F, being the sum of squared functions, is always non negative. Furthermore, $F(\boldsymbol{x}) = 0$ if and only if $\boldsymbol{x} = [1, 1, ..., 1]'$ since $f_1(\boldsymbol{x})^2 = 0$ if and only if $x_1 = 1$ and, for every $1 < k \leq n$, $f_k(\boldsymbol{x}) = 0$ if and only if $x_k = x_{k-1}$.

More formally, we can see that $\boldsymbol{x} = [1, 1, ..., 1]'$ solves the equation $\nabla F(\boldsymbol{x}) = \mathbf{0}$. Since F is convex (being the sum of convex functions) and differentiable, we know that any stationary point is a global minimum point for F.

Then $\boldsymbol{x} = [1, 1, ..., 1]'$ is the only global minimum point for F.

We plot the function for $n = 2$ in a neighborhood of the minimum point.



(a) (b)

Figura 7: function $F(\boldsymbol{x})$ for $n = 2$

We can easily see that the central area, where the minimum point is located, is quite flat. This means that the minimization methods used might have some troubles when reaching this area because they might get stuck before reaching the minimizer.

Nelder Mead Method

We runned the minimisation problem with Nelder Mead method using the following parameters:

$$\begin{aligned} \text{reflection } \rho &= 1.1 \\ \text{expansion } \chi &= 2.5 \\ \text{contraction } \gamma &= 0.6 \\ \text{shrinking } \sigma &= 0.5. \end{aligned}$$

The aim, with this choice, is to try to keep the simplex big enough so that the method will not get stuck too easily in the almost flat areas of the function's graph.

We now report a table summarizing the results obtained by running the Nelder-Mead method on the considered problem for dimensions n =10, 25, 50 and for a total of 11 starting points for each dimension, obtained as perturbations of the given one.

	average fbest	average number of iterations	average time of execution (sec)	number of failures	average rate of convergence
10	2.22	128.64	0.52	0.00	NaN
25	2.30	279.64	0.98	0.00	NaN
50	2.36	595.64	1.97	0.00	NaN

Figura 8: Results obtained by running the Nelder Mead method on problem 75.

We notice that the method reports zero failures, which means that it never stopped because of the maximum number of iterations allowed (in this case $200 \cdot n$) had been reached. However, the best value of the function F that has been found is not so close to the expected value (which as observed before should be 0). The problem is in the starting point. In fact, even with the random perturbations, it always falls in the flat area of the function. Tuning the parameters in a way that encourages the expansion of the simplex's area is not enough to prevent the method from getting stuck here. It can be seen that, if we use as a staring point for example $[0, 0, ..., 0]'$, the results are a bit closer to the exact one (around 0.35).

The Nelder Mead method does not guarantee convergence and it is sensitive to the starting point and this becomes evident in this optimisation problem.

Concerning the rate of convergence, the fact that `Nan` is always reported is due to the construction of the method itself. In fact not necessarily at every iteration the best point of the current simplex is updated; expecially when contraction and shrinking phases are reached, it means that new promising points were not found, so it is quite likely that the current best point does not change among consecutive iterations. This is a problem when it comes to apply the formula for the experimental rate of convergence (1), since it leads the denominator to be 0.

Modified Newton Method - Exact Derivatives

As previously shown, we can easily compute the exact derivatives for the gradient and the Hessian matrix of $F(x)$. The Hessian should be stored as a sparse matrix due to its large dimension. We can then apply the Modified Newton Method to the considered problem, obtaining the following results.

	avg fbest	avg gradf_norm	avg num of iters	avg time of exec (sec)	n failure	avg roc
1000	1.6195e-11	5.5308e-07	1.0655e+02	5.1004e-01	0.0000e+00	1.0000e+00
10000	2.4732e-10	7.4256e-07	8.9209e+02	3.6626e+00	0.0000e+00	1.0000e+00
100000	1.2570e-12	5.4617e-07	8.8986e+03	5.7716e+02	0.0000e+00	9.9663e-01

Figura 9: Results obtained by running the Modified Newton method on problem 75 using exact derivatives.

As expected, this method is performing much better due to the exploitation of the information contained in the gradient and the Hessian. In every tested dimension n it reaches the exact solution within the maximum number of iterations fixed for the corresponding dimension (in this case n has been used). For the others parameters the following values have been used:

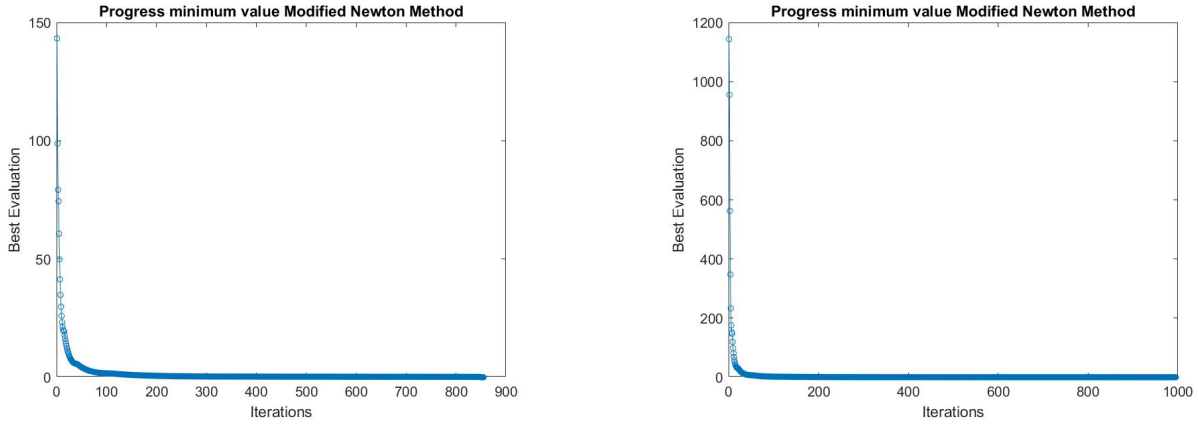
- tolerance for the norm of the gradient: 10^{-6} for every dimension

- parameter $\rho \in (0,1)$ for the reduction of the steplength in backtracking: $\rho = \begin{cases} 0.4 & n = 10^3 \\ 0.3 & n = 10^4 \\ 0.4 & n = 10^5 \end{cases}$
- parameter $c_1 \in (0,1)$ for the Armijo condition: $c_1 = \begin{cases} 10^{-4} & n = 10^3 \\ 10^{-4} & n = 10^4 \\ 10^{-3} & n = 10^5 \end{cases}$
- maximum number of backtracking steps allowed: $\texttt{btmax} = \begin{cases} 36 & n = 10^3 \\ 28 & n = 10^4 \\ 36 & n = 10^5 \end{cases}$

The values of ρ and `btmax` for every dimension has been chosen in such a way that stagnation is not allowed. In fact $\rho^{\texttt{btmax}} > \epsilon_m$, where the machine precision is $\epsilon_m \approx 10^{-16}$.

That experimental rate of convergence is approximately 1, so we are losing some of the strength of pure Newton method.

In the following figure we can see two examples of the progress of the minimum value of $F(x)$. Notice how



(a) $n = 10^3$ (b) $n = 10^4$

Figura 10: Example of convergence to zero of the value of $F(x)$.

fast it decreases in the first iterations, while the convergence becomes smaller when entering the almost flat area of the function.

Modified Newton Method - Approximated Derivatives

Let us now analyze what happens if we suppose not to be able to compute the exact derivatives of function F.

Using forward difference with step h_k (where h_k can either be constant or $h_k = h|\hat{x}_k|$ where $k = 1, ..., n$ and $\hat{x}$ is the point where the approximation is calculated), we can obtain an approximation of the gradient of F. Note that we can exploit the structure of the function in order to avoid the evaluation of the whole F. This makes the evaluation much faster.

$$\begin{aligned} \frac{\partial F}{\partial x_k}(x) &\approx \frac{F(x + h_k \vec{e_k}) - F(x)}{h_k} = \frac{f_k^2(x + h_k \vec{e_k}) + f_{k+1}^2(x + h_k \vec{e_k}) - f_k^2(x) - f_{k+1}^2(x)}{2h_k} \quad 1 \le k < n \\ \frac{\partial F}{\partial x_n}(x) &\approx \frac{f_n^2(x + h_n \vec{e_n}) - f_n^2(x)}{2h_n}. \end{aligned}$$

The same reasoning can be applied to approximate the Hessian using the general formula

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \approx \frac{f(x + h_i \vec{e_i} + h_j \vec{e_j}) - f(x + h_i \vec{e_i}) - f(x + h_j \vec{e_j}) + f(x)}{h_i h_j}$$

obtaining the following

$$\begin{aligned}
\frac{\partial^2 F}{\partial x_k^2} &\approx \frac{f_k^2(x+2h_k\vec{e_k}) + f_{k+1}^2(x+2h_k\vec{e_k}) - 2f_k^2(x+h_k\vec{e_k}) - 2f_{k+1}^2(x+h_k\vec{e_k}) + f_k^2(x) + f_{k+1}^2(x)}{2h_k^2} \quad 1 \leq k < n \\
\frac{\partial^2 F}{\partial x_n^2} &\approx \frac{f_n^2(x+2h_n\vec{e_n}) - 2f_n^2(x+h_n\vec{e_n} + f_n^2(x))}{2h_n^2} \\
\frac{\partial^2 F}{\partial x_k \partial x_{k-1}} &\approx
\end{aligned}$$

PROBLEM 76

Model

The function described in this problem is the following

$$F(\mathbf{x}) = \frac{1}{2}\sum_{k=1}^{n} f_k^2(x)$$
$$f_k(\mathbf{x}) = x_k - \frac{x_{k+1}^2}{10}, \quad 1 \leq k < n$$
$$f_n(\mathbf{x}) = x_n - \frac{x_1^2}{10}$$

where n denotes the dimensionality of the input vector $\mathbf{x}$.
The starting point for the minimization is the vector $\mathbf{x}_0 = [2, 2, \ldots, 2]$.

To be able to say something more about the behaviour of the problem is useful to look at the gradient of the function $F(\mathbf{x})$ and at its Hessian matrix.

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_k}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}\frac{1}{2}\left[f_n^2 + f_1^2\right](\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_k}\frac{1}{2}\left[f_{k-1}^2 + f_k^2\right](\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n}\frac{1}{2}\left[f_{n-1}^2 + f_n^2\right](\mathbf{x}) \end{bmatrix} = \begin{bmatrix} -\frac{x_1}{5}\left(x_n - \frac{x_1^2}{10}\right) + \left(x_1 - \frac{x_1^2}{10}\right) \\ \vdots \\ -\frac{x_k}{5}\left(x_{k-1} - \frac{x_k^2}{10}\right) + \left(x_k - \frac{x_{k+1}^2}{10}\right) \\ \vdots \\ -\frac{x_n}{5}\left(x_{n-1} - \frac{x_n^2}{10}\right) + \left(x_n - \frac{x_1^2}{10}\right) \end{bmatrix}$$

Due to the particular structure of the function, the Hessian matrix as a sparse structure, with only 3 diagonals different from zero. The non-zero elements are the following:

$$\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) = -\frac{1}{5}x_{k-1} - \frac{3}{50}x_k^2 + 1, \quad 1 < k \leq n \qquad \frac{\partial^2 F}{\partial x_1^2}(\mathbf{x}) = -\frac{1}{5}x_n - \frac{3}{50}x_1^2 + 1,$$
$$\frac{\partial^2 F}{\partial x_k \partial x_{k+1}}(\mathbf{x}) = -\frac{1}{5}x_{k+1}, \quad 1 \leq k < n \qquad \frac{\partial^2 F}{\partial x_n \partial x_1}(\mathbf{x}) = -\frac{1}{5}x_1$$
$$\frac{\partial^2 F}{\partial x_k \partial x_{k-1}}(\mathbf{x}) = -\frac{1}{5}x_k, \quad 1 < k \leq n \qquad \frac{\partial^2 F}{\partial x_1 \partial x_n}(\mathbf{x}) = -\frac{1}{5}x_n$$

We can now easily notice that the gradient of the function is null when all the components of the vector $\mathbf{x}$ are equal to 0, in this case the Hessian matrix is positive definite, so the point $\mathbf{x} = \mathbf{0}$ is a minimum of the function $F(\mathbf{x})$. Because of the definition of the function, 0 is the lowest value the function can assume, so the minimum found is global.

Nealder Mead Method

We now report a table containing some general results obtained by running the Nealder Mead method on the function $F(\mathbf{x})$.

	avg fbest	avg num of iters	avg time of exec (sec)	n failure	avg roc
10	5.555e-05	218.64	4.5116	0	NaN
25	4.1038e-05	1680.4	31.524	0	NaN
50	29.039	14007	269.17	10	NaN

Figura 11: Resultats obtained by running the symplex method on the problem 76.

First thing we can notice is that for smaller dimensionalities the symplex method is able to find the minimum in a reasonable amount of time, but when the dimensionality becomes higher the method starts failing. From the plot in figure (12), we can see that for most points belonging to $\mathbb{R}^{50}$, the method keeps iterating until the maximum number of iterations is reached without satisfying the stopping criterion. This behaviour can probably be explained by the fact that when the dimensionality increases the starting point is more far from the minimum due to its definition, so the method needs to perform more iterations to reach the minimum.

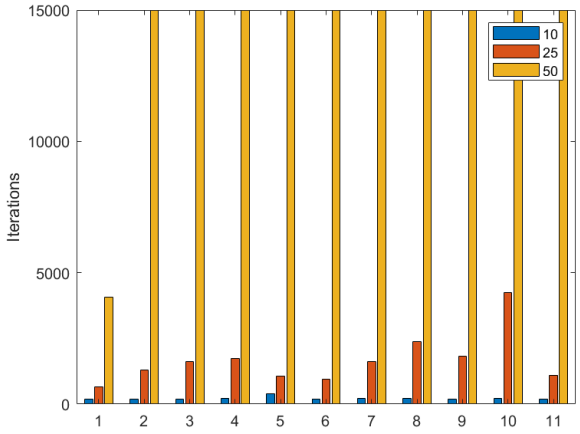
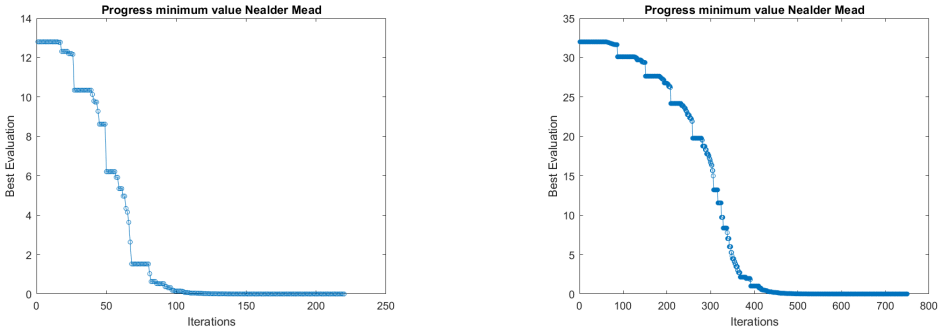


Figura 12: Number of iterations needed by the Nealder Mead method to find the minimum of the problem 76 for each starting point.

From the previous table, we can notice that the experimental rate of convergnce is always `Nan`: this is due to the fact that in the last iterations the value of $\mathbf{x}^{(k)}$ does not change much and thus it yields a division by zero in the formula (1) which defines the experimental rate of convergence. This can be seen in the following plots, showing that, in the last iterations, the approximated value of the minimum seems to be stationary.



(a) dimension 10

(b) dimension 25

Figura 13: Plots of the progresses of the Nealder Mead method for different dimensionalities for the problem 76.

Modified Newton Method - Exact Derivatives

We now report a table containing some general results obtained by running the Modified Newton method on the function $F(\mathbf{x})$. We obviously expect the method to perform better than the symplex method because of the exact derivatives used in the computation of the descent direction.

	avg fbest	avg gradf_norm	avg num of iters	avg time of exec (sec)	n failure	avg roc
1000	2.9818e-10	1.1915e-05	5.4545	0.028048	0	1.7721
10000	2.9521e-16	2.369e-08	4.9091	0.025717	0	1.9344
100000	3.2292e-15	7.6604e-08	5	0.24656	0	1.9326

Figura 14: Resultats obtained by running the Modified Newton Method on the problem 76 using the exact derivatives.

This time, the method always converges to the minimum point in very few iterations, even for higher dimensionalities. We can also appreciate the fact that the approximated rate of convergence is close to 2, as expected for a Newton method. Comparing this table with the previous one (showing the results obtained by running the symplex method), we can see that the Modified Newton method identifies as minimum a point in which the evaluation of the function is much smaller. This behavior aligns with theoretical expectations,

as the Modified Newton method leverages the exact derivatives of the function $F(\mathbf{x})$ to determine the descent direction, while the symplex method depends only on function evaluations.

Modified Newton Method - Approximated Derivatives

Approximating the derivatives of the function $F(\mathbf{x})$ using finite differences is more challenging than it appears due to potential numerical cancellation issues, which can occur when subtracting two nearly equal quantities. Additionally, we aim to derive a formula that minimizes computational cost.

Let's begin by approximating the first-order derivatives of the function $F(\mathbf{x})$ using the centered finite difference formula with step h_k. The subscript k is specified because the following formula are valid both with a constant increment, $h_k = k$ for all $h = 1, \dots, n$, and with a specific increment $h_k = h|\hat{x}_k|$ $k = 1, \dots, n$, where $\hat{\mathbf{x}}$ is the point at which we approximate the derivatives.

$$\frac{\partial F}{\partial x_k}(\mathbf{x}) \approx \frac{F(\mathbf{x} + h_k\vec{e}_k) - F(\mathbf{x} - h_k\vec{e}_k)}{2h_k} = \frac{\sum_{i=1}^{n} f_i(\mathbf{x} + h_k\vec{e}_k)^2 - \sum_{i=1}^{n} f_i(\mathbf{x} - h_k\vec{e}_k)^2}{4h_k}$$

We can observe that each term f_i^2 only depends on x_i and x_{i+1}, so $f_i(\mathbf{x} + h_k\vec{e}_k)^2 - f_i(\mathbf{x} - h_k\vec{e}_k)^2 = 0$ for all $i \neq k-1, k$ (or $i \neq 1, n$ if we are considering $k = 1$). This allows to simplify the formula, even in order to decrease the computational cost, as follows

$$\frac{\partial F}{\partial x_k}(\mathbf{x}) \approx \frac{f_{k-1}(\mathbf{x} + h_k\vec{e}_k)^2 - f_{k-1}(\mathbf{x} - h_k\vec{e}_k)^2 + f_k(\mathbf{x} + h_k\vec{e}_k)^2 - f_k(\mathbf{x} - h_k\vec{e}_k)^2}{4h_k} \quad 1 < k \leq n$$

$$\frac{\partial F}{\partial x_k}(\mathbf{x}) \approx \frac{f_n(\mathbf{x} + h_k\vec{e}_k)^2 - f_n(\mathbf{x} - h_k\vec{e}_k)^2 + f_k(\mathbf{x} + h_k\vec{e}_k)^2 - f_k(\mathbf{x} - h_k\vec{e}_k)^2}{4h_k} \quad k = 1$$

In order to avoid numerical cancellation, the numerator has been expanded obtaining the following formula

$$\frac{\partial F}{\partial x_1}(\mathbf{x}) \approx \frac{4h_k x_1 - 2/5 h_k x_2^2 - 4/5 h_k x_n x_1 + 8/100 h_k x_1 (x_1^2 + h_k^2)}{4h_k}$$

$$\frac{\partial F}{\partial x_k}(\mathbf{x}) \approx \frac{4h_k x_k - 2/5 h_k x_{k+1}^2 - 4/5 h_k x_{k-1} x_k + 8/100 h_k x_k (x_k^2 + h_k^2)}{4h_k}$$

$$\frac{\partial F}{\partial x_n}(\mathbf{x}) \approx \frac{4h_k x_n - 2/5 h_k x_1^2 - 4/5 h_k x_{n-1} x_n + 8/100 h_k x_n (x_n^2 + h_k^2)}{4h_k}$$

We can now proceed to approximate the second order derivatives of the function $F(\mathbf{x})$ using the centered finite difference formula; this time we need to use two different increments h_i and h_j based on the two components with respect to which we are differentiating. The general formula is the following

$$\frac{\partial^2 F}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{F(\mathbf{x} + h_i\vec{e}_i + h_j\vec{e}_j) - F(\mathbf{x} + h_i\vec{e}_i) - F(\mathbf{x} - h_j\vec{e}_j) + F(\mathbf{x})}{h_i h_j}$$

The approximation of the Hessian matrix has to be approached taking into account its sparsity in order to reduce the computational cost, indeed in the Matlab script we have implemented a function that approximates the Hessian matrix just by computing the non-null terms which are the following

$$\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) \approx 2h_k - \frac{2}{5}x_{k-1}h_k + \frac{12}{100}x_k^2 h_k^2 + \frac{24}{100}x_k h_k^3 + \frac{14}{100}h_k^2 \qquad 1 < k \leq n$$

$$\frac{\partial^2 F}{\partial x_k^2}(\mathbf{x}) \approx 2h_k - \frac{2}{5}x_n h_k + \frac{12}{100}x_k^2 h_k^2 + \frac{24}{100}x_k h_k^3 + \frac{14}{100}h_k^2 \qquad k = 1$$

$$\frac{\partial^2 F}{\partial x_k \partial x_{k+1}}(\mathbf{x}) \approx -\frac{2}{5}h_k h_{k+1} x_{k+1} - \frac{1}{5}h_k^2 h_{k+1} \qquad 1 \leq k < n$$

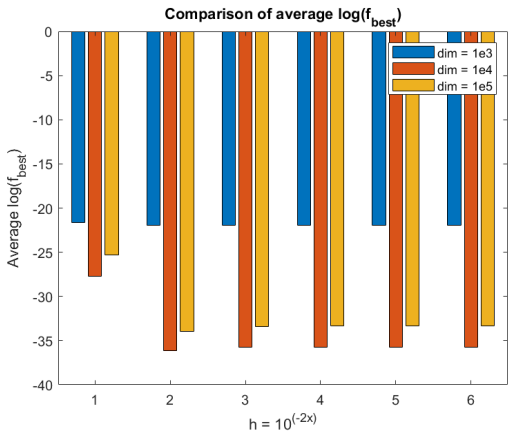
The values of the inferior diagonal are obtained by exploiting the symmetry of the Hessian matrix.

The terms have been computed following the same approach described above: the numerator has been expanded negletting the $f_i^2()$ that are not affected by the varation of the components with respect to which we are differentiating.

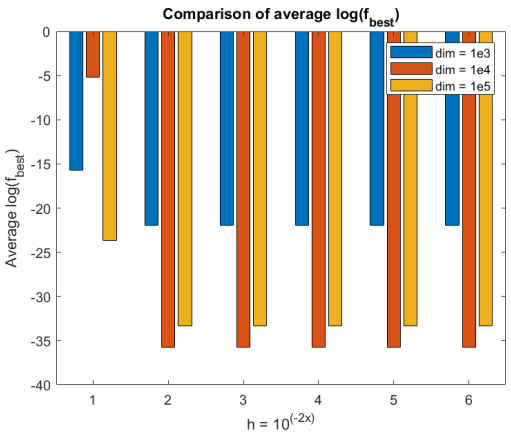
We now report some barplots showing the results obtained by running the Modified Newton method on the function $F(\mathbf{x})$ using the approximated derivatives.

As we can see from the plots (15), expecially for larger values of the increments, the algorithm converges to a point such that the value of the function is higher accordingly to the fact that the approximated derivatives are less accurate. Nonetheless, the method succeeds to find an acceptable approximation of the minimum value even when computing the descent direction with just an approximation of the derivatives.

The others plots show that the average time of execution and the average rate of convergence are not significantly affected by the approximation of the derivatives for none of the values oh the increment h.

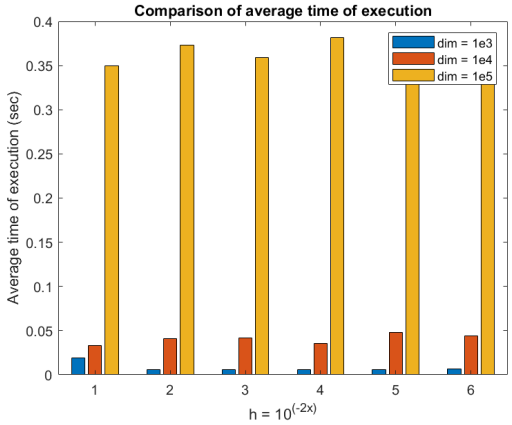


(a) Costant Increment h

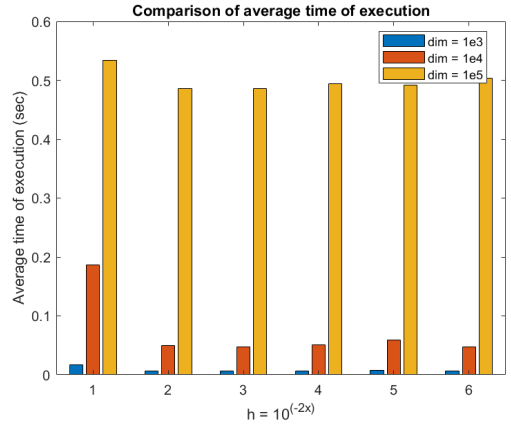


(b) Specific Increment

Figura 15: Values of the average $\log(f_{best})$ in function of the increment while running the Modified Newton Method with approximated derivatives on the problem 76.

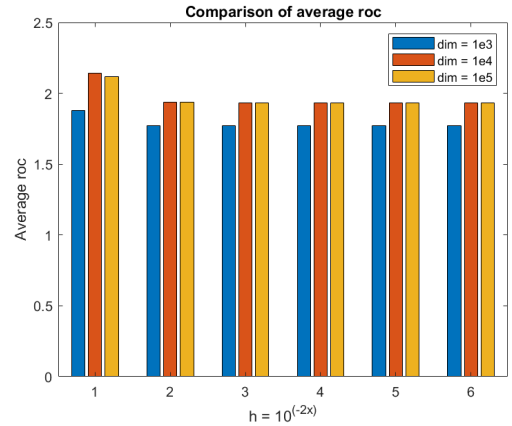


(a) Costant Increment h

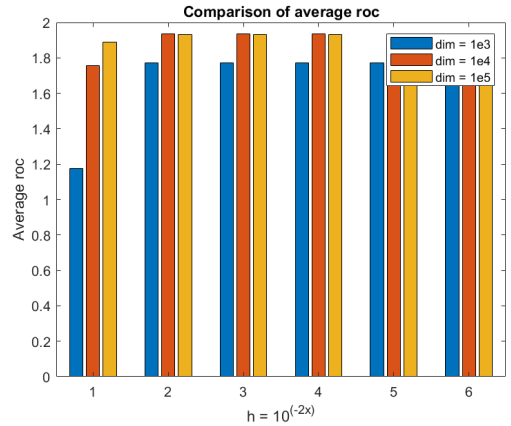


(b) Specific Increment

Figura 16: Average time of execution in function of the increment h while running the Modified Newton Method with approximated derivatives on the problem 76.



(a) Costant Increment h



(b) Specific Increment

Figura 17: Average values of the experimental rate of convergence in function of the increment h while running the Modified Newton Method with approximated derivatives on the problem 76.

CONCLUSION