Report NO4LSP

Chiodo Martina - 343310
Vigè Sophie - 339268

INTRODUZIONE

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.

Specificare gli STOPPING CRITERION per gli algoritmi

PROBLEMA 25

Model

The function described in this problem is the following

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

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

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

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

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

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

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

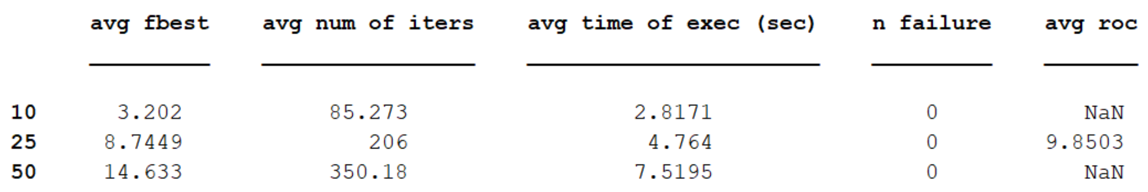
By analyzing the derivatives of the problem, we can deduce that the gradient of the function is nullified by the vector composed of ones which also nullifies the value of $F(\mathbf{x})$.

Nealder Mead Method

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

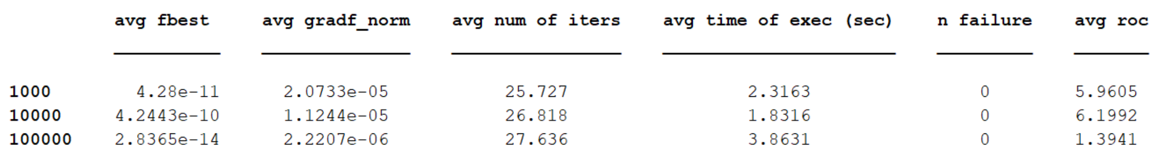
Modified Newton Method - Exact Derivatives

Modified Newton Method - Approximated Derivatives



	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: Resultats obtained by running the symplex method on the problem 25.



	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: Resultats obtained by running the Modified Newton method on the problem 25 using the exact derivatives.

PROBLEMA 75

PROBLEMA 76

Model

The function described in this problem is the following

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

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

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:

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

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 3: 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 (4), 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.

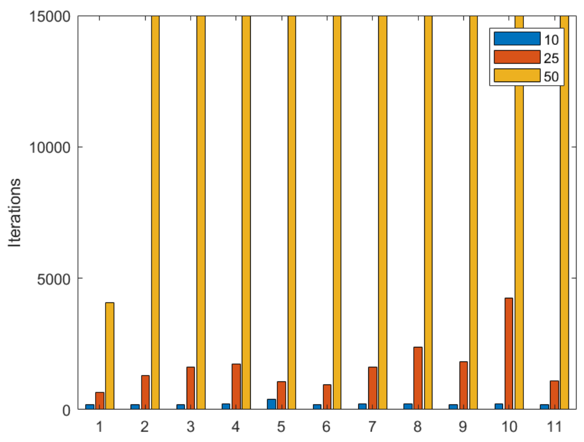
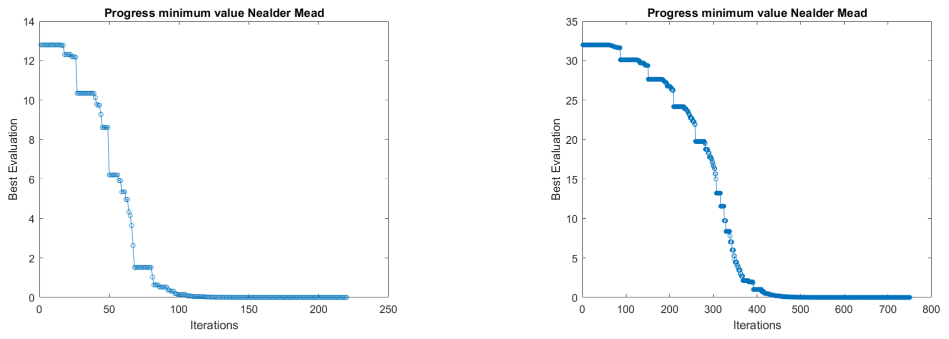


Figura 4: Number of iterations needed by the Nealder Mead method to find the minimum of the problem 76 for each starting point.

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 5: 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 6: 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

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

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

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

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

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

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

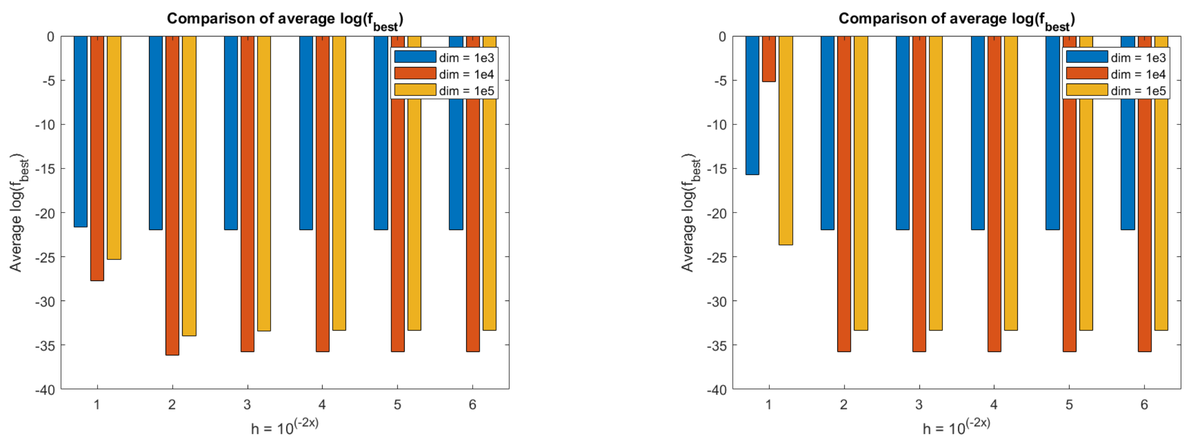
The values of the inferior diagonal are obtained by exploiting the symmetry of the Hessian matrix.

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.

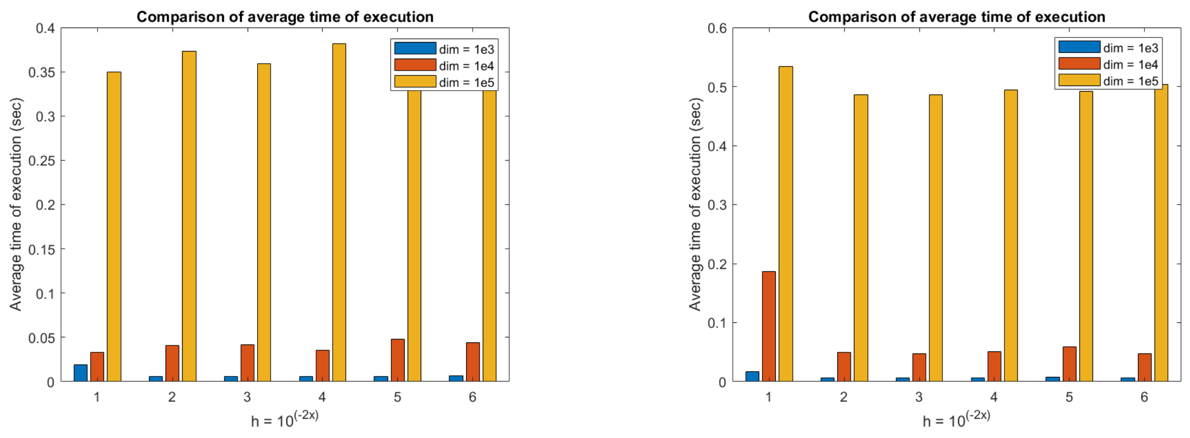
As we can see from the plots (7), expecially for larger values of the increments, the algorithm converges to a point such that the value of the function is higher accordingly to the fact that the approximated derivatives are less accurate. Nonetheless, the method succeeds to find an acceptable approximation 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.



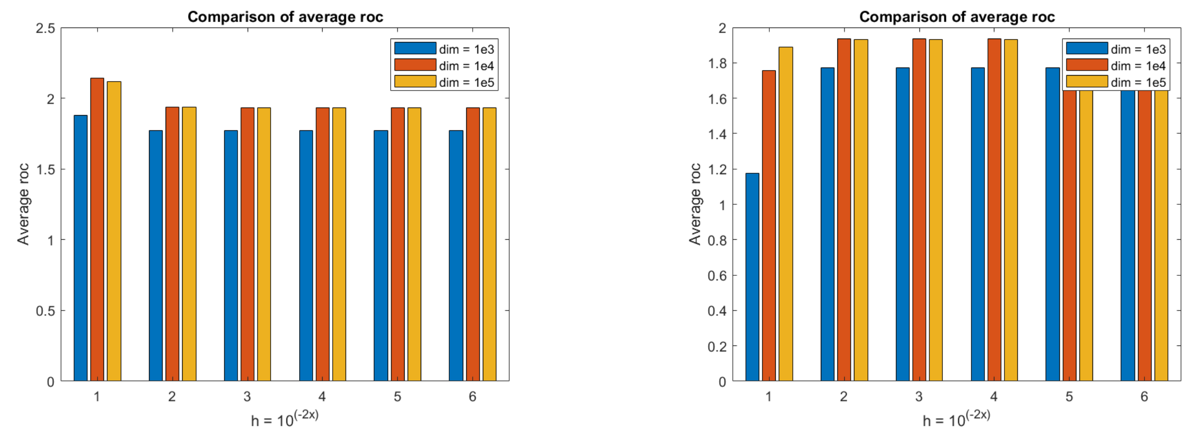
(a) Costant Increment h (b) Specific Increment

Figura 7: 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 8: 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 9: 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.

CONCLUSIONI