Numerical Optimization for Large Scale Problems Assignment

Sofia Bernardoni 347900
Gaia Giulio 345989

February 2025

Index

1 Introduction: Assignment on Unconstrained Optimization

This report contains the description and the implementation of two different numerical methods for unconstrained optimization and the analysis of the results obtained applying both of them on some functions. We have chosen to implement the Nelder-Mead method and the truncated Newton method. The objective of the assignment is to implement the methods, using a back-tracking strategy for the line search in the truncated Newton method, and then test them on the Rosenbrock function:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

with two different initial condition $x^{(0)} = (1.2, 1.2), x^{(1)} = (-1.2, 1)$.
While testing the methods we were also asked to perform parameters tuning in case the standard ones did not work well.
Then we had to select three problems from [1].
In order to test the algorithms on these with replicable results, we set the seed equal to 345989=min{345989, 347900}, as asked.
We tested the Nelder-Mead method on the three functions in three different dimensions n=10,25,50, with eleven distinct starting points each. These were randomly generated with uniform distribution in the hyper-cube $x(0) \in [\bar{x}_1 - 1, \bar{x}_1 + 1] \times \cdots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$
With respect to the truncated Newton method we had to test it on the three functions in three different dimensions with eleven different starting points each, as done for the Nelder-Mead method, but with this method the considered dimensions were $n = 10^3, 10^4, 10^5$. In this case, since the method is a second-order one (i.e. it exploits the function's second order derivatives), we also had to apply the codes both with exact derivatives and with approximated ones. The latter were computed using finite differences with respect to six different values of the increment $h = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}$. Moreover, we had to test the method with finite differences using a specific increment h_i when differentiating with respect to the variable x_i

$$h_i = 10^{-k}|x_i|, \quad k = 2, 4, 6, 8, 10, 12, \quad i = 1, \ldots, n,$$

where $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is the point at which the derivatives have to be approximated. We have always exploited the structure of the selected functions to implement the finite differences.
Furthermore, we have tested the truncated Newton method with preconditioning too, as it involves the solution of a linear system with the conjugate gradient method which performance depends on the coefficient matrix conditioning.

2 Methods

2.1 Nelder Mead Method

The Nelder-Mead method is an iterative optimization method, classified as a "0-order" one since it uses no information from derivatives of the function we are optimizing. It is a Simplex-type method because at every step it starts with a given simplex and ends with a different one that improves the approximation of the solution. The Nelder-Mead algorithm is based on four operations: reflection, expansion, contraction and shrinking, depending on some parameters. At every iteration k, as mentioned, we have a non singular simplex S_k defined by n+1 points which we assume to be ordered so that

$$f(x_1^k) \leq f(x_2^k) \leq \ldots \leq f(x_{n+1}^k)$$

The algorithm is based on four phases:

1. Reflection phase:
   Calculate the barycenter of the simplex of the first n points $\bar{x}^k = \frac{1}{n}\sum_{i=1}^{n} x_i^k$ and then compute the reflection point with parameter $\rho > 0$ (whose standard choice is 1):

   $$x_R^k = \bar{x}^k + \rho(\bar{x}^k - x_{n+1}^k)$$

   If $f(x_1^k) \leq f(x_R^k) < f(x_{n+1}^k)$ then accept x_R^k as a new point for the simplex S_{k+1} replacing x_{n+1}^k

[1] ttps://www.researchgate.net/publication/45932888_Test_Problems_in_Optimization

2. Expansion phase:
   If $f(x_R^k) < f(x_1^k)$ then compute the expansion point

$$x_E^k = \bar{x}^k + \chi(x_R^k - \bar{x}^k)$$

   with $\chi > 1$, typically 2, and if $f(x_E^k) < f(x_R^k)$ accept x_E^k as the new point for S_{k+1} and stop, else accept x_R^k as the new point for S_{k+1} and stop .

3. Contraction phase:
   If $f(x_R^k) \geq f(x_n^k)$ then compute the contraction point between $\bar{x}^k$ and the best point among x_R^k and x_{n+1}^k:

$$x_C^k = \bar{x}^k - \gamma(\bar{x}^k - x_{n+1}^k) \text{ if } f(x_{n+1}^k) < f(x_R^k)$$
$$x_C^k = \bar{x}^k - \gamma(\bar{x}^k - x_R^k) \text{ if } f(x_R^k) < f(x_{n+1}^k)$$

   The parameter $\gamma \in (0,1)$ is typically $\gamma = 1/2$. if $f(x_C^k) < f(x_{n+1}^k)$ then accept x_C^k as the new point for S_{k+1} and stop, otherwise continue with the fourth phase.

4. Shrinking phase:
   Shrink the simplex around the the best point:

$$\begin{cases} \hat{x}_i^{k+1} = x_1^k + \sigma(x_i^k - x_1^k) & \forall i = 2...n+1 \\ \hat{x}_1^{k+1} = x_1^k \end{cases}$$

   and the new simplex S_{k+1} is defined by $\hat{x}_i^k$. The shrinking parameter is $\sigma \in (0,1)$, whose standard choice is $1/2$.

We have chosen to initialize the simplex by taking the starting point and generating n points, each with one component increased by the variable δ:

$$S_0 = (x_0, x_0 + \delta I)$$

For the chosen functions we have set a tolerance of 10^{-13}. This tolerance controls the distance between the function value at the best and worst points of the simplex.

For this algorithm, it was not possible to calculate the experimental convergence rate since, at each iteration, the best function value is saved, which, due to the way the algorithm is designed and the way space exploration is performed through the simplex, may not change for several iterations. Therefore, it is not possible to use the formula for the experimental rate of convergence

$$q \approx \frac{\log\left(\frac{\|\hat{e}_{k+1}\|}{\|\hat{e}_k\|}\right)}{\log\left(\frac{\|\hat{e}_k\|}{\|\hat{e}_{k-1}\|}\right)} \qquad \hat{e}_k := x_k - x_{k-1}$$

After some tuning of the parameters, we decided to choose the ones reported in the table 1 for n=10 and those in the table 2 for both n=25 and n=50.

ρ	χ	γ	σ	δ	tolerance	maxiter
1.1	1.8	0.8	0.9	0.1	1e-13	1e06

Table 1: Parameters and Hyper-parameters used in Nelder-Mead method for $n = 10$

ρ	χ	γ	σ	δ	tolerance	maxiter
1.1	2.5	0.8	0.9	1	1e-13	1e06

Table 2: Parameters and Hyper-parameters used in Nelder-Mead method for $n = 25$ and $n = 50$

2.2 Truncated Newton Method

The Truncated Newton Method is an optimization method used to find the minimum of a function. It is based on the well-known Newton method, which is iterative and uses the gradient (first derivative) and the Hessian matrix (second derivative) of the function to find at every iteration the best direction to choose to move towards the wanted minimum. For this reason, it is classified as a second order method. At every iteration the Newton method considers a quadratic approximation of the function f around x^k

$$m_k(x) = f(x^k) + \nabla f(x^k)^T(x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k)$$

and then finds the descent direction p, defined as $p = x - x^k$, computing the minimum of $m_k(p)$. If the $\nabla^2 f(x^k)$ positive definiteness is assumed, p can found as a stationary point for $m_k(p)$; then it satisfies

$$\nabla m_k(p) = 0 \implies \underbrace{\nabla^2 f(x^k)}_{A}\,\underbrace{p^k}_{z} = \underbrace{-\nabla f(x^k)}_{b} \tag{1}$$

So, the idea behind the truncated Newton method is to use the Conjugate Gradient (CG) method, that is an iterative method for linear systems' solution, to solve (1). Since we don't know if the Hessian matrix A is positive definite, at every inner iteration we check if the direction computed satisfies the negative curvature condition

$$p_{INN}^{(i)T} A p_{INN}^{(i)} \leq 0 \tag{2}$$

where $p_{INN}^{(i)}$ is the direction computed at the i-th inner iteration of CG method applied to $Az = b$.
Now, if the curvature condition (2) is satisfied, it means that certainly A is not positive definite, so:

- if $i = 0$ we stop with $p^k = z^0 + p_{INN}^0$ which is a decent direction (p_{INN}^0 is $-\nabla J(x^k)$)
- if $i > 0$ we stop with p_{INN}^{i-1}, i.e. the last approximated solution of $Az = b$ which does not satisfy the negative curvature condition

This way p^k is guaranteed to be a descent direction.
The steplength α^k along p^k is then computed with an inexact line search strategy using backtracking and the Armijo conditions. Backtracking consists of starting with a given steplength α_0^k (we used the classical choice $\alpha_0^k = 1$) and iteratively contracting that value ($\alpha_{j+1}^k = \rho\alpha_j^k$) until a feasible steplength value is reached, where α is said to be feasible if it satisfies the following Armijo condition:

$$f(x^k + \alpha p^k) \leq f(x^k) + c_1 \alpha \nabla f(x^k)^T p^k$$

Once p^k and α^k are found, the point x^{k+1} of the next iteration can be easily computed:

$$x^{k+1} = x^k + \alpha^k p^k$$

To mitigate the cost related to the solution of the linear system (1), it is exploited the idea of the inexact Newton method where the direction p^k is computed with a computational cost and accuracy related to how far we are from a possible solution, i.e. using an adaptive tolerance depending on $\|\nabla f(x^k)\|$ for the solution of (1):

$$\|\nabla^2 f(x^k)p^k + \nabla f(x^k)\| \leq \eta_k \|\nabla f(x^k)\|$$

where η_k is called forcing term.
We have decided to test our algorithm using two different forcing terms: $\eta_k^1 = min(0.5, \sqrt{\|\nabla f(x^k)\|})$ and $\eta_k^2 = min(0.5, \|\nabla f(x^k)\|)$, that will be referred respectively as superlinear and quadratic, since there exists a theorem guaranteeing these two different rates of convergence with these choices of η_k while using the inexact Newton method. In our case this convergence was not guaranteed, as we were not applying the inexact Newton method but the truncated version that works even without the Hessian matrix positive definiteness, as explained before.
The parameters that we had to set are the following:

- $kmax$ = maximum number of iterations.
- $tolgrad$ = tolerance on $\|\nabla f(x^k)\|$, which is the stopping criterion.
- cg_maxit = maximum number of CG iterations.

- z_0 = initial condition for the CG method.
- $c_1 \in (0,1)$ = coefficient for the Armijo condition, which standard choice is 10^{-4}
- $\rho \in (0,1)$ = coefficient for the α_k update, which standard choice is 0.5
- $btmax$ = maximum number of backtrackings

We have usually used the standard choices for the backtracking parameters and the following for the other ones: $kmax = 1500$, $tolgrad = 5e-7$, $cg\_maxit = 50$ for smaller problems (until $n = 10^3$) and $cg\_maxit = 100$ for bigger ones, z_0 null vector and $btmax = 50$, chosen accordingly to ρ in order to always get a non null steplength (with $\rho = 0.5$ and $btmax = 50$ the smallest reachable value is about $8.88e-16$). All the cases where a parameter has been changed will be pointed out, otherwise these are the values that have to be considered.

kmax	**tolgrad**	**cg_maxit small**	cg_maxit	z_0	**btmax**	ρ
1500	$5 * 10^-7$	50	100	$\underline{0}$	50	0.5

Table 3: Parameters and Hyper-parameters used in Truncated Newton method

3 Test on Rosenbrock function

Here follow the Rosenbrock function and the two different starting points considered:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2)$$
$$x^{(0)} = (1.2, 1.2)$$
$$x^{(1)} = (-1.2, 1)$$

3.1 Nelder Mead method

We have tested the Nelder Mead algorithm with different combinations of parameters on the Rosenbrok function for both the initial conditions. The following tables show the results.

Rho	**Chi**	**Gamma**	**Sigma**	**Delta**	**Time**	**Final Value**	**Iterations**
1	2	0,5	0,5	0,1	0,0064363	5,36312E-15	84
1,2	4	0,7	0,3	0,1	0,0017902	1,15708E-15	77
1,2	4	0,7	0,3	0,5	0,0044761	1,27148E-15	106
1,4	5	0,8	0,2	0,1	0,0028513	8,09746E-16	90
1,65	4,55	0,95	0,15	0,1	0,0074774	4,33493E-15	109
2	4	0,7	0,5	0,5	0,002954	1,30366E-15	94

Table 4: Results for x_0 values

Rho	**Chi**	**Gamma**	**Sigma**	**Delta**	**Time**	**Final Value**	**Iterations**
1	2	0,5	0,5	0,1	0,0031813	7,25102E-16	151
1,2	4	0,7	0,3	0,1	0,0030887	4,03414E-16	143
1,2	4	0,7	0,3	0,5	0,0031312	7,84565E-15	144
1,4	5	0,8	0,2	0,1	0,004755	3,2104E-14	128
1,65	4,55	0,95	0,15	0,1	0,0046375	6,57864E-12	150
2	4	0,7	0,5	0,5	0,0033451	3,31228E-14	107

Table 5: Results for x_1 values

The parameter reported in tables 4 and 5 are:

- $\rho > 0$ Reflection parameter
- $\chi > 1$ Expansion parameter

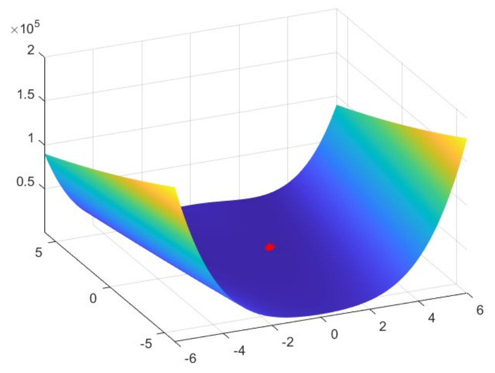
- $0 < \gamma < 1$ Contraction parameter
- $0 < \sigma < 1$ Shrinking parameter
- $\delta > 0$ the parameter for the initial simplex generated by the Nelder-Mead method

As can be seen from the tables 4 and 5, each test reaches the convergence and the number of iterations required to reach convergence is lower when starting from x_0 than from x_1. The computation time is comparable and changes depending on the parameters, but in both cases, it is on the order of 10^{-3} seconds.

3.2 Truncated Newton method

Starting point	**f terms**	f_x	**N Iter**	**Time**	**Violations**	**cg iterations**	**bt iterations**
x_0=[1.2;1.2]	Superlineare	5.55313×10^{-18}	9	0.0128747	0	1.444444444	0.111111111
x_0=[1.2;1.2]	Quadratica	5.55313×10^{-18}	9	0.0028837	0	1.444444444	0.111111111
x_1=[-1.2;1]	Superlineare	7.4715×10^{-28}	64	0.0008508	37	0.625	0.421875
x_1=[-1.2;1]	Quadratica	7.4715×10^{-28}	64	0.0006823	37	0.625	0.421875

Table 6: Results for x_0 and x_1 with truncated Newton method


(a) 3D plot of the Rosenbrock function with the sequence found by the algorithm strarting from $x_0 = [1.2, 1.2]$

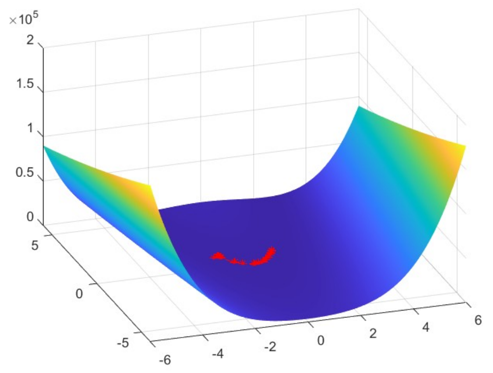
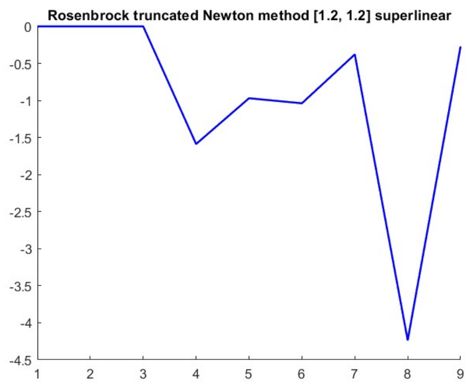
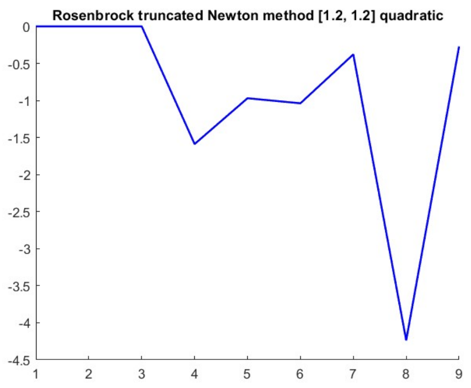

(b) 3D plot of the Rosenbrock function with the sequence found by the algorithm starting from $x_1 = [-1.2, 1]$

Figure 1: 3D plot of Rosenbrock function

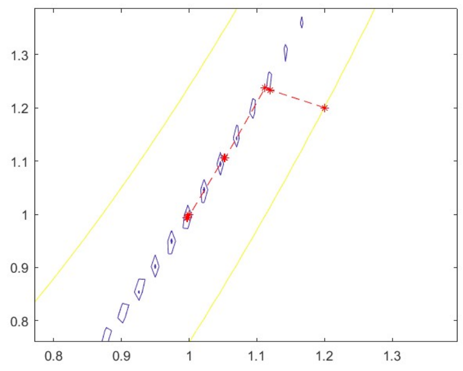


(a) Experimental rate of convergence starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

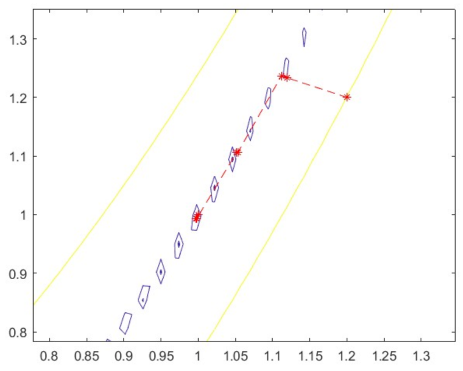


(b) Experimental rate of convergence starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 2: Experimental rate of convergence starting from x_0

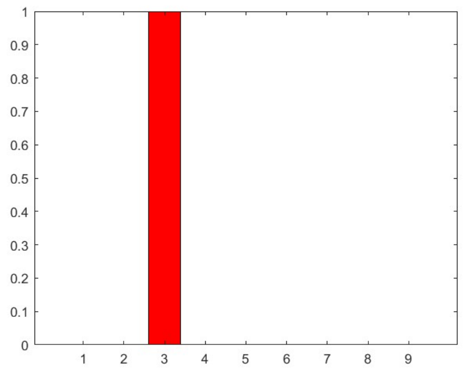


(a) Contour line of the function value and the sequence generated by the algorithm starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

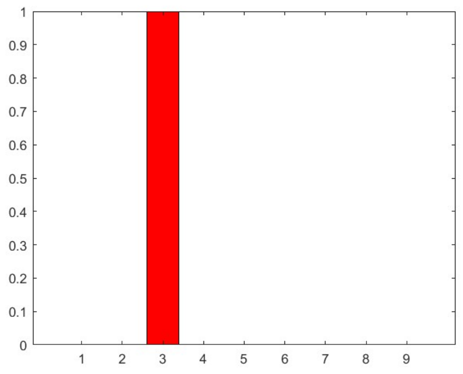


(b) Contour line of the function value and the sequence generated by the algorithm starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 3: Contour line starting from $x0$

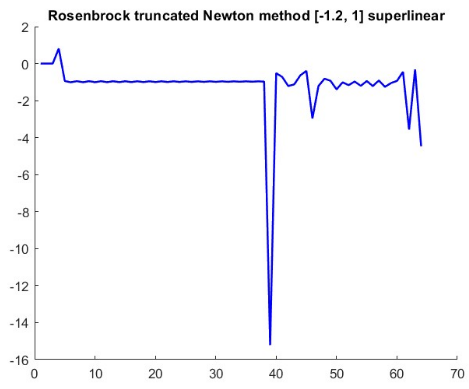


(a) Number of backtracking iterations required ad every outer iteration starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

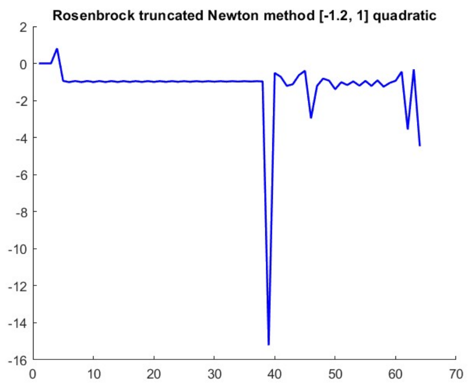


(b) Number of backtracking iterations required ad every outer iteration starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 4: Backtracking starting from x_0

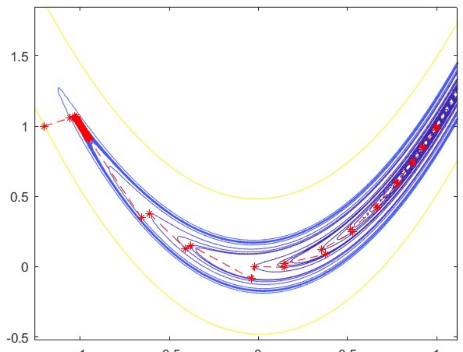


(a) Experimental rate of convergence starting from $x_1 = [-1.2, 1]$ with a superlinear forcing term of tolerance

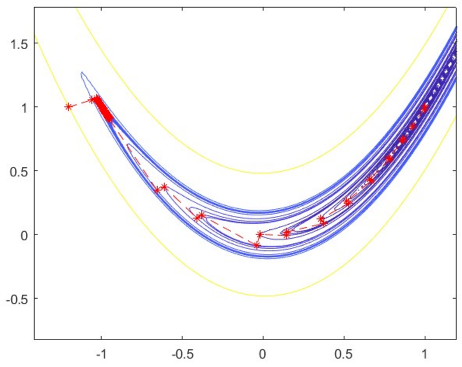


(b) Experimental rate of convergence starting from $x_1 = [-1.2, 1]$ with a quadratic forcing term of tolerance

Figure 5: Experimental rate of convergence starting from x_1

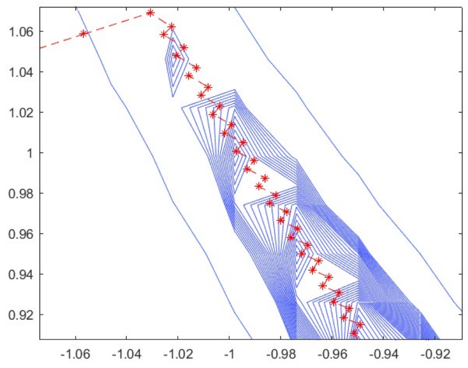


(a) Contour line of the function value and the sequence generated by the algorithm starting from $x_1 = [-1.2, 1]$ with a superlinear forcing term of tolerance

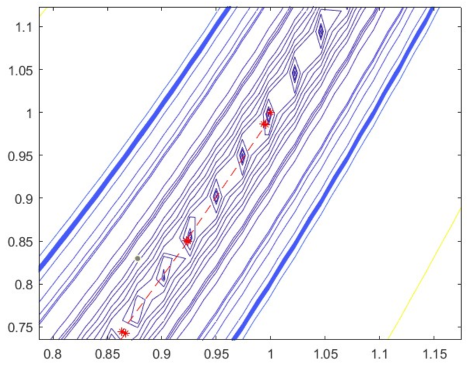


(b) Contour line of the function value and the sequence generated by the algorithm starting from $x_1 = [-1.2, 1]$ with a quadratic forcing term of tolerance

Figure 6: Contour line starting from x_1

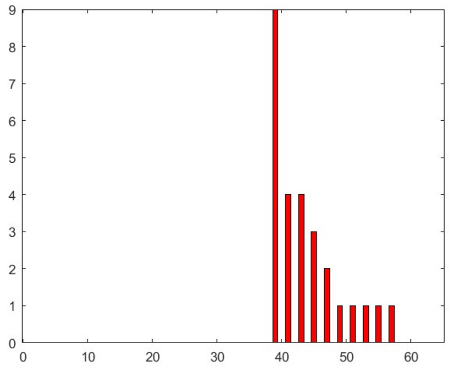


(a) The first iterations of the algorithm starting from $x_1 = [-1.2, 1]$

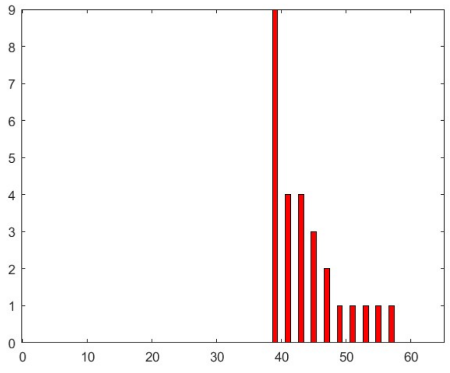


(b) The last iterations of the algorithm starting from $x_1 = [-1.2, 1]$

Figure 7: Zoom of contour line starting from x_1



(a) Number of backtracking iterations required ad every outer iteration starting from $x_1 = [-1.2, 1]$ with a superlinear forcing term of tolerance



(b) Number of backtracking iterations required ad every outer iteration starting from $x_1 = [-1.2, 1]$ with a quadratic forcing term of tolerance

Figure 8: Backtracking iterations starting from x_1

As can be seen from the table 6, starting from x_0, only nine iterations are required to reach convergence. It can also be noted that the algorithm never violates the negative curvature condition; therefore, the CG method stops when it has found the solution. Since we are in $\mathbb{R}^2$, the CG method reaches convergence in at most 2 iterations because the directions found by the method are A-conjugate, and in $\mathbb{R}^2$, there are only two possible relatively A-conjugate directions.

Looking at the average number of iterations performed by CG, which is approximately 1.4, it can be affirmed that some cases required both iterations.

The average number of CG iterations could therefore justify the higher ratio between execution times and the numbers of iterations of the tests starting from x_0, compared to the correspondent value computed for the tests starting from x_1. The latter need, in fact, more iterations and take significantly less time on average. This happens because, especially in the first iterations, the negative curvature condition is violated at the first step and, consequently, the steepest descent direction is chosen without solving the linear system with CG.

As can be seen from figure 7, when starting in x_1 the directions chosen in the first iterations, given by $-\nabla f(x^k)$, are perpendicular to each other, resulting in a zigzag behavior with very small steps, even though for each of them $\alpha = 1$ is accepted as steplength, with no need of backtracking (as can be noticed in figure 8). This happens because the distance between one point and the following one not only depends on α, but also on the gradient's modulus. Backtracking is used from iteration 39, after exiting the initial region where negative curvature was always violated.

Looking at table 6 and at figures from 2 to 8, it can be observed that the algorithm has the same behavior with both forcing terms.

4 Chosen problems

We have chosen the three following problems:

1. F79:

$$\begin{aligned} F(x) &= \frac{1}{2}\sum_{k=1}^{n} f_k^2(x) \\ f_k(x) &= \left(3 - \frac{x_k}{10}\right) x_k + 1 - x_{k-1} - 2x_{k+1}, \quad 1 \le k \le n \\ x_0 &= x_{n+1} = 0, \quad x_l = -1, \quad l \ge 1 \end{aligned}$$

2. F27:

$$\begin{aligned} F(x) &= \frac{1}{2}\sum_{k=1}^{n+1} f_k^2(x) \\ f_k(x) &= \frac{1}{\sqrt{100000}}(x_k - 1), \quad 1 \le k \le n \\ f_{n+1}(x) &= \sum_{i=1}^{n} x_i^2 - \frac{1}{4} \\ x_l &= l, \quad l \ge 1 \end{aligned}$$

3. F16:

$$\begin{aligned} F(x) &= \sum_{i=1}^{n} i\left[(1 - \cos(x_i)) + \sin(x_i) - 1 - \sin(x_{i+1})\right] \\ x_0 &= x_{n+1} = 0, \quad x_i = 1, \quad i \ge 1 \end{aligned}$$

5 Tables with results

In this section we present the results obtained applying the two optimization methods on the three selected problems in the three different dimensions.
Here follows a brief legend:

- n = dimension
- $Exact$ = results obtained with exact derivatives
- $FD1$ = results obtained with classical finite differences (increment h)

$$\frac{\partial F(x)}{\partial x_k} = \frac{F(x + he_k) - F(x - he_k)}{2h}$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} = \frac{F(x + he_k + he_j) - F(x + he_k) - F(x + he_j) + F(x)}{h^2}$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} = \frac{F(x + he_k) - 2F(x) + F(x - he_k)}{h^2}$$

- $FD2$ = results obtained with finite differences with increment depending on the considered point (increment $h_i = h|x_i|$)

$$\frac{\partial F(x)}{\partial x_k} = \frac{F(x + h|x_k|e_k) - F(x - h|x_k|e_k)}{2h|x_k|}$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} = \frac{F(x + h|x_k|e_k + h|x_j|e_j) - F(x + h|x_k|e_k) - F(x + h|x_j|e_j) + F(x)}{h^2|x_k||x_j|}$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} = \frac{F(x + h|x_k|e_k) - 2F(x) + F(x - h|x_k|e_k)}{h^2|x_k|^2}$$

When evaluating the performance of the Nelder-Mead method, we have decided to compute the execution time, the final value of the function, and the number of iterations needed to reach convergence for all initial conditions, and then compute the mean of these quantities.
Whilst for the truncated Newton performance evaluation we have decided to consider for each case:

- Experimental order of convergence (plotted for the last steps)
- Number of successful runs out of the 11 determined by the 11 distinct starting points. Successful runs are those ones that have converged (we are not considering neither runs stopped because the maximum number of iterations was reached nor runs stopped because the Armijo condition was never reached in the backtracking process)
- Average minimum function value found
- Average number of iterations to reach convergence
- Average execution time
- Average number of times the negative curvature condition is satisfied (forcing the CG iterations to stop)
- Average of the average number of CG iterations (inner loops)
- Average of the average number of backtracking iterations

Note that whenever an average is computed, it is done just considering the successful runs.
Observe that each case is determined by the dimension, the forcing term and the type of derivative, i.e. exact, approximated classical finite differences or approximated with finite differences with increment depending on the point at which they are computed. Furthermore the cases with the approximated derivatives are also defined by the increment value.

5.1 FUNCTION 79

$$F(x) = \frac{1}{2}\sum_{k=1}^{n} f_k^2(x)$$
$$f_k(x) = \left(3 - \frac{x_k}{10}\right) x_k + 1 - x_{k-1} - 2x_{k+1}, \quad 1 \le k \le n$$
$$x_0 = x_{n+1} = 0, \quad x_l = -1, \quad l \ge 1$$

Exact derivatives

$$\frac{\partial F(x)}{\partial x_k} = -2f_{k-1}(x) + (3 - \frac{1}{5}x_k)f_k(x) - f_{k+1}(x) \quad \forall k = 2...n-1$$
$$\frac{\partial F(x)}{\partial x_1} = (3 - \frac{1}{5}x_1)f_1(x) - f_2(x)$$
$$\frac{\partial F(x)}{\partial x_n} = (3 - \frac{1}{5}x_n)f_n(x) - 2f_{n-1}(x)$$
$$\frac{\partial^2 F(x)}{\partial^2 x_k} = 5 - \frac{1}{5}f_k(x) + (3 - \frac{1}{5}x_k)^2 \quad \forall k = 1...n$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} = -2(3 - \frac{1}{5}x_k) - (3 - \frac{1}{5}x_{k+1}) \quad \forall k = 1...n-1$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} = 0 \quad |k-j| \ge 2$$

Finite differences type 1

$$\frac{\partial F(x)}{\partial x_k} \approx -2f_{k-1}(x) + (3 - \frac{1}{5}x_k)f_k(x) - f_{k+1}(x) - \frac{1}{10}(3 - \frac{1}{5}x_k)h^2 \quad \forall k = 2...n-1$$
$$\frac{\partial F(x)}{\partial x_1} \approx (3 - \frac{1}{5}x_1)f_1(x) - f_2(x) - \frac{1}{10}(3 - \frac{1}{5}x_1)h^2$$
$$\frac{\partial F(x)}{\partial x_n} \approx -2f_{n-1}(x) + (3 - \frac{1}{5}x_n)f_n(x) - \frac{1}{10}(3 - \frac{1}{5}x_n)h^2$$
$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx 5 - \frac{1}{5}f_k(x) + (3 - \frac{1}{5}x_k)^2 + \frac{1}{100}h^2 \quad \forall k = 1...n$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx -2(3 - \frac{1}{5}x_k) - (3 - \frac{1}{5}x_{k+1}) + \frac{3}{10}h \quad \forall k = 1...n-1$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 0 \quad |k-j| \ge 2$$

Finite differences type 2

$$\frac{\partial F(x)}{\partial x_k} \approx -2f_{k-1}(x) + (3 - \frac{1}{5}x_k)f_k(x) - f_{k+1}(x) - \frac{1}{10}(3 - \frac{1}{5}x_k)h^2|x_k|^2 \quad \forall k = 1...n-1$$
$$\frac{\partial F(x)}{\partial x_1} \approx (3 - \frac{1}{5}x_1)f_1(x) - f_2(x) - \frac{1}{10}(3 - \frac{1}{5}x_1)h^2|x_1|^2$$
$$\frac{\partial F(x)}{\partial x_n} \approx -2f_{n-1}(x) + (3 - \frac{1}{5}x_n)f_n(x) - \frac{1}{10}(3 - \frac{1}{5}x_n)h^2|x_n|^2$$
$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx 5 - \frac{1}{5}f_k(x) + (3 - \frac{1}{5}x_k)^2 + \frac{1}{100}h^2|x_k|^2 \quad \forall k = 1...n$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx -2(3 - \frac{1}{5}x_k) - (3 - \frac{1}{5}x_{k+1}) + \frac{h|x_k|}{5} + \frac{h|x_{k+1}|}{10}$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx 0 \quad |k-j| \ge 2$$

NEW VERSION

```
if fin_dif_2 % version of finite differences with abs(xj)
    z1=zeros(n,1);
    z2=zeros(n,1);
    z3=zeros(n,1);
    z1(1:3:end)=ones(length(1:3:end),1);
    z2(2:3:end)=ones(length(2:3:end),1);
    z3(3:3:end)=ones(length(3:3:end),1);
    z1=h*z1.*abs(x);
    z2=h*z2.*abs(x);
    z3=h*z3.*abs(x);
    F1=(JF(x+z1)-JF(x-z1));
    F2=(JF(x+z2)-JF(x-z2));
    F3=(JF(x+z3)-JF(x-z3));
    F1=[0;F1];
    rem_n=rem(n,3);
    if rem_n==0
        F3=[F3;0];
    elseif rem_n==1
        F1=[F1;0];
    else
        F2=[F2;0];
    end
    m1=reshape(F1,3,[])';
    m2=reshape(F2,3,[])';
    m3=reshape(F3,3,[])';
    M=zeros(n,3);
    M(1:3:end,:)=m1;
    M(2:3:end,:)=m2;
    M(3:3:end,:)=m3;
    for col=1:3
        M(:,col)=M(:,col).*(1./abs(x))/h;
    end
    if sparse % sparse
        HF=spdiags(M,[1,0,-1],n,n);
    else % NOT sparse
        HF=diag(M(:,2))+diag(M(2:end,1),1)+diag(M(1:end-1,1),-1);
    end
else % classic version of finite differences
    z1=zeros(n,1);
    z2=zeros(n,1);
    z3=zeros(n,1);
    z1(1:3:end)=ones(length(1:3:end),1);
    z2(2:3:end)=ones(length(2:3:end),1);
    z3(3:3:end)=ones(length(3:3:end),1);
    F1=(JF(x+h*z1)-JF(x-h*z1))/(2*h);
    F2=(JF(x+h*z2)-JF(x-h*z2))/(2*h);
    F3=(JF(x+h*z3)-JF(x-h*z3))/(2*h);
    F1=[0;F1];
    rem_n=rem(n,3);
    if rem_n==0
        F3=[F3;0];
    elseif rem_n==1
        F1=[F1;0];
    else
        F2=[F2;0];
    end
    m1=reshape(F1,3,[])';
    m2=reshape(F2,3,[])';
    m3=reshape(F3,3,[])';
    M=zeros(n,3);
    M(1:3:end,:)=m1;
    M(2:3:end,:)=m2;
    M(3:3:end,:)=m3;
    if sparse % sparse
        HF=spdiags(M,[1,0,-1],n,n);
    else % NOT sparse
        HF=diag(M(:,2))+diag(M(2:end,1),1)+diag(M(1:end-1,1),-1);
    end
end
```

5.1.1 Nelder-Mead method

Result with $n = 10$

Initial condition	Time	FinalValue	Iterations
x0	0.0625167	9.70898E-14	2594
x1	0.0429871	2.22575E-13	2107
x2	0.0559888	2.9783E-13	2027
x3	0.0462386	1.30708E-13	1946
x4	0.0442741	1.8956E-13	2065
x5	0.1135344	1.50051E-13	5378
x6	0.0531327	1.07706E-13	2368
x7	0.0564274	1.43151E-13	2361
x8	0.0521129	8.03728E-14	2128
x9	0.0667942	1.70411E-13	2409
x10	0.0618143	1.92916E-13	2477
Mean	**0.059620109**	**1.62034E-13**	**2532.727273**

Table 7: Results of F79 n=10 Nelder-Mead

Result with $n = 25$

Problem	Time	FinalValue	Iterations
x0	0.3383135	3,99908828056904	12959
x1	0.3771915	3,99908828056908	12841
x2	0.4607917	3,99908828056902	14656
x3	0.3749015	3,99908828056901	11845
x4	0.6225806	3,87482796806862	19900
x5	0.4023339	3,87482796806879	13069
x6	0.4107565	3,99908828056925	14013
x7	0.3174045	3,99908828056897	10841
x8	0.3461964	3,99908828056912	11905
x9	0.3649025	3,87482796806866	12066
x10	0.3609082	3,87482796806875	12294
Mean	**0.397843709**	**3.953902712**	**13308.09091**

Table 8: Results of F79 n=25 Nelder-Mead

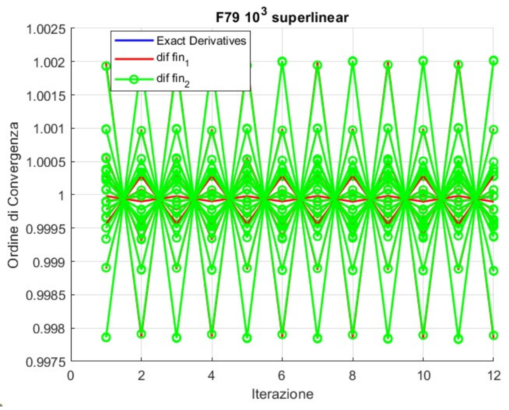
Result with $n = 50$

Problem	Time	FinalValue	Iterations
x0	1.7384367	4.001778886	51181
x1	1.8373300	4.001748508	53583
x2	1.5385742	4.001794267	48027
x3	1.7497013	4.001756596	52733
x4	1.4795829	4.001746977	45651
x5	1.5160436	4.001726586	47966
x6	1.7476550	4.001783654	55301
x7	1.5199245	4.00172154	47575
x8	1.9863490	4.001720556	62899
x9	1.6320707	4.00178723	50887
x10	2.1579486	4.001903271	69891
Mean	**1.718510591**	**4.001769825**	**53244.90909**

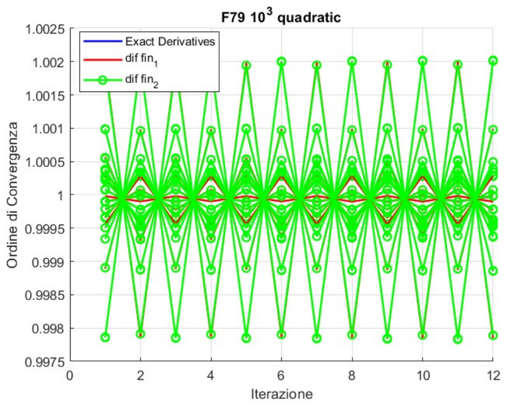
Table 9: Results of F79 n=50 Nelder-Mead

5.1.2 Truncated Newton method

Result with $n = 10^3$

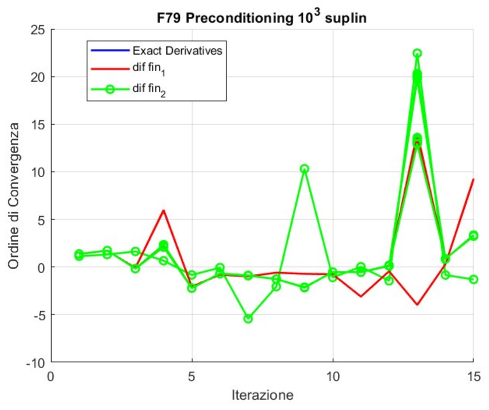


(a) The last 12 value of experimental rate of convergence F79 $n = 10^3$ superlinear

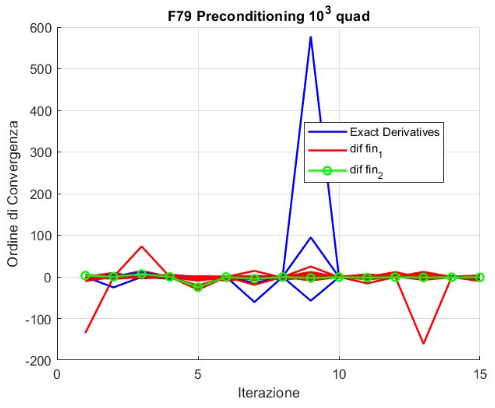


(b) The last 12 values of experimental rate of convergence F79 $n = 10^3$ quadratic

Figure 9: The last 12 values of experimental rate of convergence F79 $n = 10^3$



(a) The last 15 value of experimental rate of convergence F79 $n = 10^3$ superlinear with preconditioning



(b) The last 15 values of experimental rate of convergence F79 $n = 10^3$ quadratic with preconditioning

Figure 10: The last 15 values of experimental rate of convergence F79 $n = 10^3$ with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	11	11	11	11	11	11	11
FD2	11	11	11	11	11	11	11

Table 10: Number of converged processes out of 11 initial conditions $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	11	11	11	11	11	11	11
FD2	11	11	11	11	11	11	11

Table 11: Number of converged processes out of 11 initial conditions $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 12: Number of converged processes out of 11 initial conditions $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 13: Number of converged processes out of 11 initial conditions $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64849E-06	2,28281E-13	1,30477E-13	1,30417E-13	1,30417E-13	1,30417E-13	1,30417E-13
FD2	0,000163591	2,5676E-12	1,30524E-13	1,30417E-13	1,30417E-13	1,30417E-13	1,30417E-13

Table 14: Average function minimum value found $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64849E-06	2,28281E-13	1,30477E-13	1,30417E-13	1,30417E-13	1,30417E-13	1,30417E-13
FD2	0,000163591	2,5676E-12	1,30524E-13	1,30417E-13	1,30417E-13	1,30417E-13	1,30417E-13

Table 15: Average function minimum value found $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64831E-06	1,32162E-13	5,49696E-14	4,821E-14	5,2043E-14	4,97111E-14	4,49272E-14
FD2	0,000163589	2,32282E-12	4,84975E-14	5,29555E-14	5,03848E-14	5,77076E-14	4,49272E-14

Table 16: Average function minimum value found $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64831E-06	1,32162E-13	5,49696E-14	4,821E-14	5,2043E-14	4,97111E-14	4,49272E-14
FD2	1,64831E-06	1,32162E-13	5,49696E-14	4,821E-14	5,2043E-14	4,97111E-14	4,49272E-14

Table 17: Average function minimum value found $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1073,727273	1073,636364	1072,272727	1072,272727	1072,272727	1072,272727	1072,2727273
FD2	1071,272727	1072	1072,272727	1072,272727	1072,272727	1072,272727	1072,272727

Table 18: Average number of iterations $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1073,727273	1073,636364	1072,272727	1072,272727	1072,272727	1072,272727	1072,272727
FD2	1071,272727	1072	1072,272727	1072,272727	1072,272727	1072,272727	1072,272727

Table 19: Average number of iterations $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	202,5	245,8	247,9	248,2	247,2	248,6	249,1
FD2	190,5	235,1	248,3	247,8	247,8	247,5	249,1

Table 20: Average number of iterations $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	202,5	245,8	247,9	248,2	247,2	248,6	249,1
FD2	202,5	245,8	247,9	248,2	247,2	248,6	249,1

Table 21: Average number of iterations $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,262818573	0,260690582	0,256824645	0,257514645	0,2605777	0,261408164	0,159791818
FD2	0,272187991	0,265809909	0,265973945	0,268696164	0,270382309	0,269784882	0,159791818

Table 22: Average execution time $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,254398809	0,252861064	0,253173064	0,252259055	0,254536318	0,253803318	0,1489014
FD2	0,261852427	0,262889118	0,261912027	0,262374836	0,263047145	0,262542418	0,1489014

Table 23: Average execution time $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,15196421	0,19158078	0,19621516	0,1796628	0,18207084	0,19176677	0,1210501
FD2	0,15162157	0,19568357	0,19455265	0,18882074	0,19208563	0,18924941	0,1210501

Table 24: Average execution time $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,13812773	0,16844514	0,1717329	0,1738393	0,17084201	0,17081557	0,11063079
FD2	0,00000162	0,00000144	0,00000157	0,0000016	0,00000157	0,00000151	0,11063079

Table 25: Average execution time $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1070	1069,636364	1068,272727	1068,272727	1068,272727	1068,272727	1068,272727
FD2	1067,272727	1067,909091	1068,272727	1068,272727	1068,272727	1068,272727	1068,272727

Table 26: Average times negative curvature condition satisfied $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1070	1069,636364	1068,272727	1068,272727	1068,272727	1068,272727	1068,272727
FD2	1067,272727	1067,909091	1068,272727	1068,272727	1068,272727	1068,272727	1068,272727

Table 27: Average times negative curvature condition satisfied $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	200,5	243,6	245,7	246	245	246,4	246,9
FD2	188,3	232,9	246,1	245,6	245,6	245,3	246,9

Table 28: Average times negative curvature condition satisfied $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	200,5	243,6	245,7	246	245	246,4	246,9
FD2	200,5	243,6	245,7	246	245	246,4	246,9

Table 29: Average times negative curvature condition satisfied $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,007733453	0,007905386	0,008096966	0,008096966	0,008096966	0,008096966	0,008096966
FD2	0,008295999	0,008188702	0,008096966	0,008096966	0,008096966	0,008096966	0,008096966

Table 30: Average of the average number of CG iterations (inner loops) $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,007733453	0,007905386	0,008096966	0,008096966	0,008096966	0,008096966	0,008096966
FD2	0,008295999	0,008188702	0,008096966	0,008096966	0,008096966	0,008096966	0,008096966

Table 31: Average of the average number of CG iterations (inner loops) $n = 10^3$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,015816957	0,013882223	0,013780478	0,013749719	0,0138059	0,013716199	0,013677897
FD2	0,017889354	0,014485968	0,013761598	0,013760557	0,013781087	0,013765173	0,013677897

Table 32: Average of the average number of CG iterations (inner loops) $n = 10^3$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,015816957	0,013882223	0,013780478	0,013749719	0,0138059	0,013716199	0,013677897
FD2	0,015816957	0,013882223	0,013780478	0,013749719	0,0138059	0,013716199	0,013677897

Table 33: Average of the average number of CG iterations (inner loops) $n = 10^3$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000341789	0,00060193	0,000520514	0,000520514	0,000520514	0,000520514	0,000520514
FD2	0,000521449	0,000609452	0,000520514	0,000520514	0,000520514	0,000520514	0,000520514

Table 34: Average of the average number of backtracking iterations $n = 10^3$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000341789	0,00060193	0,000520514	0,000520514	0,000520514	0,000520514	0,000520514
FD2	0,000521449	0,000609452	0,000520514	0,000520514	0,000520514	0,000520514	0,000520514

Table 35: Average of the average number of backtracking iterations $n = 10^3$ quadratic without preconditioning

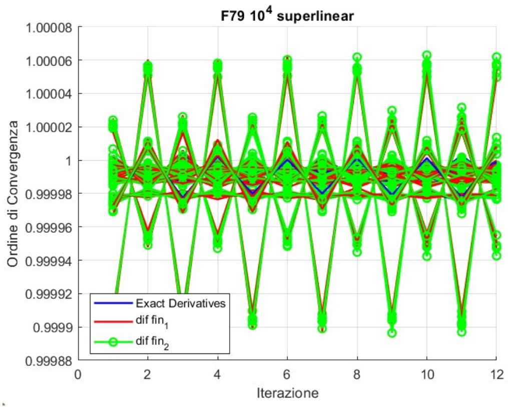
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586
FD2	0,050985357	0,081279835	0,096544335	0,094205248	0,095829323	0,093804506	0,092815586

Table 36: Average of the average number of backtracking iterations $n = 10^3$ superlinear with preconditioning

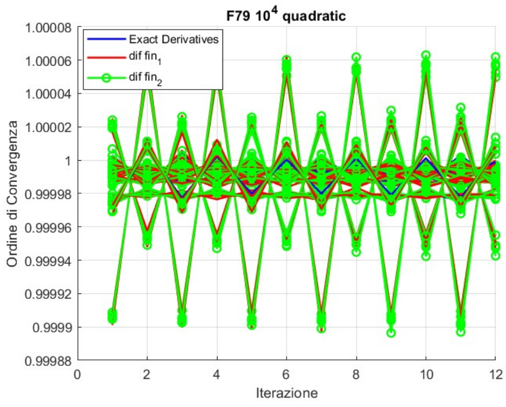
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586
FD2	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586

Table 37: Average of the average number of backtracking iterations $n = 10^3$ quadratic with preconditioning

Result with $n = 10^4$

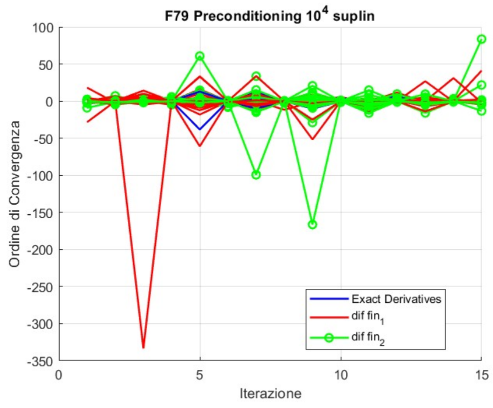


(a) The last 12 value of experimental rate of convergence F79 $n = 10^4$ superlinear

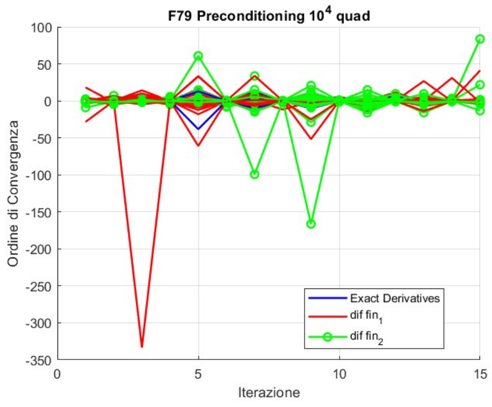


(b) The last 12 values of experimental rate of convergence F79 $n = 10^4$ quadratic

Figure 11: The last 12 values of experimental rate of convergence F79 $n = 10^4$



(a) The last 15 value of experimental rate of convergence F79 $n = 10^4$ superlinear with preconditioning



(b) The last 15 values of experimental rate of convergence F79 $n = 10^4$ quadratic with preconditioning

Figure 12: The last 15 values of experimental rate of convergence F79 $n = 10^4$ with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 38: Number of converged processes out of 11 initial conditions $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 39: Number of converged processes out of 11 initial conditions $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 40: Number of converged processes out of 11 initial conditions $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 41: Number of converged processes out of 11 initial conditions $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64952E-05	5,72495E-13	1,27663E-13	1,27636E-13	1,27636E-13	1,27636E-13	1,27636E-13
FD2	0,001647018	1,94039E-11	1,27915E-13	1,27636E-13	1,27636E-13	1,27636E-13	1,27636E-13

Table 42: Average function minimum value found $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64952E-05	5,72495E-13	1,27663E-13	1,27636E-13	1,27636E-13	1,27636E-13	1,27636E-13
FD2	0,001647018	1,94039E-11	1,27915E-13	1,27636E-13	1,27636E-13	1,27636E-13	1,27636E-13

Table 43: Average function minimum value found $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64945E-05	4,43041E-13	5,58975E-14	4,94455E-14	5,17299E-14	5,41583E-14	5,66234E-14
FD2	0,00164701	1,85524E-11	5,05451E-14	4,84265E-14	5,0759E-14	5,49415E-14	5,66234E-14

Table 44: Average function minimum value found $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,64945E-05	4,43041E-13	5,58975E-14	4,94455E-14	5,17299E-14	5,41583E-14	5,66234E-14
FD2	0,00164701	1,85524E-11	5,05451E-14	4,84265E-14	5,0759E-14	5,49415E-14	5,66234E-14

Table 45: Average function minimum value found $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1134,8	1134	1134	1134	1134	1134	1134
FD2	1130,6	1134	1134	1134	1134	1134	1134

Table 46: Average number of iterations $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1134,8	1134	1134	1134	1134	1134	1134
FD2	1130,6	1134	1134	1134	1134	1134	1134

Table 47: Average number of iterations $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	210,6	256,5	263,7	264,7	264	263,9	263,8
FD2	200,3	245,4	264,3	264,8	265,6	263,6	263,8

Table 48: Average number of iterations $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	210,6	256,5	263,7	264,7	264	263,9	263,8
FD2	200,3	245,4	264,3	264,8	265,6	263,6	263,8

Table 49: Average number of iterations $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	2,76876088	2,88121	2,91783852	2,91327214	2,9214707	2,93277786	1,7251081
FD2	2,98634329	3,01241734	3,01942235	3,02614068	3,04076341	3,0319359	1,7251081

Table 50: Average execution time $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	2,83464851	2,88809497	2,89634924	2,88306139	2,90498264	2,89429229	1,72022804
FD2	2,96925532	2,99972357	2,99437455	2,99620357	3,01965132	3,00720703	1,72022804

Table 51: Average execution time $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	4,0949378	4,24745524	4,38996612	4,7513206	4,35869642	4,20794768	3,36667142
FD2	3,58946374	4,0294938	4,82114118	4,51312471	4,35712136	4,62912884	3,36667142

Table 52: Average execution time $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	3,45212792	4,00931231	4,17802566	4,33683535	4,42266159	4,31398013	2,94954536
FD2	3,25473483	4,17854053	4,35395646	4,47720204	4,67192816	4,36589619	2,94954536

Table 53: Average execution time $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1130,8	1129,9	1129,9	1129,9	1129,9	1129,9	1129,9
FD2	1126,4	1129,9	1129,9	1129,9	1129,9	1129,9	1129,9

Table 54: Average times negative curvature condition satisfied $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1130,8	1129,9	1129,9	1129,9	1129,9	1129,9	1129,9
FD2	1126,4	1129,9	1129,9	1129,9	1129,9	1129,9	1129,9

Table 55: Average times negative curvature condition satisfied $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	208,6	254,5	261,7	262,7	262	261,9	261,8
FD2	198,3	243,4	262,3	262,8	263,6	261,6	261,8

Table 56: Average times negative curvature condition satisfied $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	208,6	254,5	261,7	262,7	262	261,9	261,8
FD2	198,3	243,4	262,3	262,8	263,6	261,6	261,8

Table 57: Average times negative curvature condition satisfied $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,00704971	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005
FD2	0,007716595	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005

Table 58: Average of the average number of CG iterations (inner loops) $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,00704971	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005
FD2	0,007716595	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005	0,007232005

Table 59: Average of the average number of CG iterations (inner loops) $n = 10^4$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,014246828	0,01169776	0,011377698	0,011334716	0,011364226	0,011369095	0,011373067
FD2	0,014980909	0,012227256	0,011351292	0,011329913	0,011295823	0,011382155	0,011373067

Table 60: Average of the average number of CG iterations (inner loops) $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,014246828	0,01169776	0,011377698	0,011334716	0,011364226	0,011369095	0,011373067
FD2	0,014980909	0,012227256	0,011351292	0,011329913	0,011295823	0,011382155	0,011373067

Table 61: Average of the average number of CG iterations (inner loops) $n = 10^4$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957
FD2	0,000639854	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957

Table 62: Average of the average number of backtracking iterations $n = 10^4$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957	0,0002659573
FD2	0,000639854	0,000265957	0,000265957	0,000265957	0,000265957	0,000265957	0,0002659573

Table 63: Average of the average number of backtracking iterations $n = 10^4$ quadratic without preconditioning

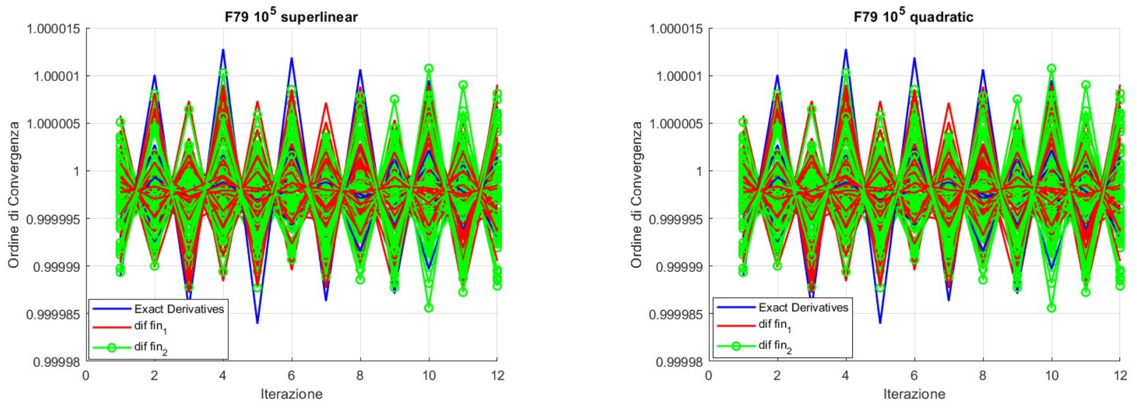
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exac**
FD1	0,059887902	0,08157586	0,089931923	0,092971752	0,090164806	0,089053094	0,088755
FD2	0,046500061	0,079162711	0,090823135	0,092170126	0,084739867 0,091098594	0,088755284	

Table 64: Average of the average number of backtracking iterations $n = 10^4$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,059887902	0,08157586	0,089931923	0,092971752	0,090164806	0,089053094	0,088755284
FD2	0,046500061	0,079162711	0,090823135	0,092170126	0,084739867	0,091098594	0,088755284

Table 65: Average of the average number of backtracking iterations $n = 10^4$ quadratic with preconditioning

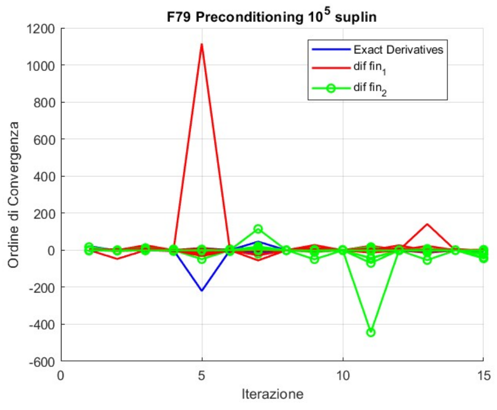
Result with $n = 10^5$



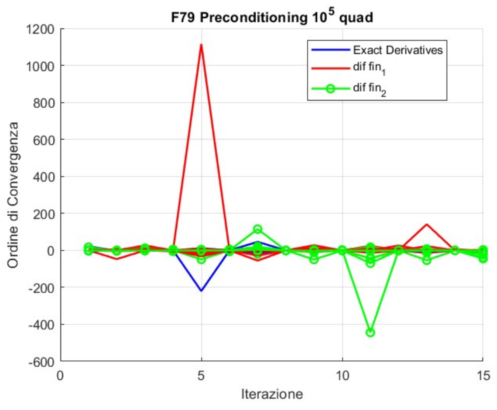
(a) The last 12 value of experimental rate of convergence F79 $n = 10^5$ superlinear

(b) The last 12 values of experimental rate of convergence F79 $n = 10^5$ quadratic

Figure 13: The last 12 values of experimental rate of convergence F79 $n = 10^5$



(a) The last 15 value of experimental rate of convergence F79 $n = 10^5$ superlinear with preconditioning



(b) The last 15 values of experimental rate of convergence F79 $n = 10^5$ quadratic with preconditioning

Figure 14: The last 15 values of experimental rate of convergence F79 $n = 10^5$ with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 66: Number of converged processes out of 11 initial conditions $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	10	10	10	10	10	10	10

Table 67: Number of converged processes out of 11 initial conditions $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	11	11	11	11	11	11	10

Table 68: Number of converged processes out of 11 initial conditions $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	10	10	10	10	10
FD2	11	11	11	11	11	11	10

Table 69: Number of converged processes out of 11 initial conditions $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000164949	2,66634E-12	1,26483E-13	1,26394E-13	1,26394E-13	1,26394E-13	1,26394E-13
FD2	0,016481148	1,73953E-10	1,27286E-13	1,26394E-13	1,26394E-13	1,26394E-13	1,26394E-13

Table 70: Average function minimum value found $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000164949	2,66634E-12	1,26483E-13	1,26394E-13	1,26394E-13	1,26394E-13	1,26394E-13
FD2	0,016481148	1,73953E-10	1,27286E-13	1,26394E-13	1,26394E-13	1,26394E-13	1,26394E-13

Table 71: Average function minimum value found $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000164947	2,37541E-12	5,3821E-14	5,14858E-14	5,21974E-14	4,63964E-14	5,19072E-14
FD2	3681,814983	3681,8	3681,8	3681,8	3681,8	3681,8	5,19072E-14

Table 72: Average function minimum value found $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,000164947	2,37541E-12	5,3821E-14	5,14858E-14	5,21974E-14	4,63964E-14	5,19072E-14
FD2	3681,814983	3681,8	3681,8	3681,8	3681,8	3681,8	5,19072E-14

Table 73: Average function minimum value found $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1197,3	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5
FD2	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5

Table 74: Average number of iterations $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1197,3	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5
FD2	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5	1197,5

Table 75: Average number of iterations $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	220,3	266,7	279,6	278,6	278,5	279,6	279,1
FD2	227,8181818	268	289,4545455	289,0909091	290	290,4545455	279,1

Table 76: Average number of iterations $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	220,3	266,7	279,6	278,6	278,5	279,6	279,1
FD2	227,8181818	268	289,4545455	289,0909091	290	290,4545455	279,1

Table 77: Average number of iterations $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	32,47818541	32,37461464	32,38209421	32,32290573	32,29777017	32,38602828	19,21847459
FD2	33,48180861	33,39968532	33,41472301	33,39566559	33,50999861	33,4293263	19,21847459

Table 78: Average execution time $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	32,32581964	32,26278849	32,28085333	32,30582055	32,2885162	32,34913174	19,25721946
FD2	33,38395965	33,35463102	33,31423121	33,32538161	33,32875581	33,36192088	19,25721946

Table 79: Average execution time $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	41,25697461	44,74935331	47,87824201	50,82696964	47,82574632	49,6072814	36,45314351
FD2	56,46326012	59,42785285	64,41934805	67,66200601	63,79790558	64,98710046	36,45314351

Table 80: Average execution time $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	40,8662184	45,38316313	46,6642369	51,39837812	46,58480493	48,08694101	34,03909018
FD2	58,67760532	60,90014943	64,47887666	64,21147615	66,3672601	64,23011453	34,03909018

Table 81: Average execution time $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1193,3	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5
FD2	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5

Table 82: Average times negative curvature condition satisfied $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1193,3	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5
FD2	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5	1193,5

Table 83: Average times negative curvature condition satisfied $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	218,3	264,7	277,6	276,6	276,5	277,6	277,1
FD2	190	230,1818182	251,6363636	251,2727273	252,1818182	252,6363636	277,1

Table 84: Average times negative curvature condition satisfied $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	218,3	264,7	277,6	276,6	276,5	277,6	277,1
FD2	190	230,1818182	251,6363636	251,2727273	252,1818182	252,6363636	277,1

Table 85: Average times negative curvature condition satisfied $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,006681703	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586
FD2	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586

Table 86: Average of the average number of CG iterations (inner loops) $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,006681703	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586
FD2	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586	0,006680586

Table 87: Average of the average number of CG iterations (inner loops) $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,013619193	0,011249532	0,010730634	0,010768999	0,010772471	0,010730166	0,010749232
FD2	0,103835476	0,101597561	0,10069152	0,100705562	0,100670872	0,100653547	0,010749232

Table 88: Average of the average number of CG iterations (inner loops) $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,013619193	0,011249532	0,010730634	0,010768999	0,010772471	0,010730166	0,010749232
FD2	0,103835476	0,101597561	0,100691521	0,100705562	0,100670872	0,100653547	0,010749232

Table 89: Average of the average number of CG iterations (inner loops) $n = 10^5$ quadratic with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	0	0	0	0	0	0
FD2	0	0	0	0	0	0	0

Table 90: Average of the average number of backtracking iterations $n = 10^5$ superlinear without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	0	0	0	0	0	0
FD2	0	0	0	0	0	0	0

Table 91: Average of the average number of backtracking iterations $n = 10^5$ quadratic without preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,055400451	0,080648559	0,086945848	0,092990958	0,09371735	0,092285477	0,088874018
FD2	4,586875068	4,610745165	4,627199036	4,626673638	4,628992578	4,625238069	0,088874018

Table 92: Average of the average number of backtracking iterations $n = 10^5$ superlinear with preconditioning

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,055400451	0,080648559	0,086945848	0,092990958	0,09371735	0,092285477	0,088874018
FD2	4,586875068	4,610745165	4,627199036	4,626673638	4,628992578	4,625238069	0,088874018

Table 93: Average of the average number of backtracking iterations $n = 10^5$ quadratic with preconditioning

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,159791818	1,7251081	19,21847459
Average Iter	1072,272727	1134	1197,5
Average fval	1,30417E-13	1,27636E-13	1,26394E-13
Violation	1068,272727	1129,9	1193,5
Average iter Bt	0,000520514	0,000265957	0
Average iter cg	0,008096966	0,007232005	0,006680586
N converged	11	10	10

Table 94: Results of exact derivatives of F79 superlinear without preconditioning

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,1489014	1,72022804	19,25721946
Average Iter	1072,272727	1134	1197,5
Average fval	1,30417E-13	1,27636E-13	1,26394E-13
Violation	1068,272727	1129,9	1193,5
Average iter Bt	0,000520514	0,000265957	0
Average iter cg	0,008096966	0,007232005	0,006680586
N converged	11	10	10

Table 95: Results of exact derivatives of F79 quadratic without preconditioning

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,1210501	3,36667142	36,45314351
Average Iter	249,1	263,8	279,1
Average fval	4,49272E-14	5,66234E-14	5,19072E-14
Violation	246,9	261,8	277,1
Average iter Bt	0,092815586	0,088755284	0,088874018
Average iter cg	0,013677897	0,011373067	0,010749232
N converged	10	10	10

Table 96: Results of exact derivatives of F79 superlinear with preconditioning

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,11063079	2,94954536	34,03909018
Average Iter	249,1	263,8	279,1
Average fval	4,49272E-14	5,66234E-14	5,19072E-14
Violation	246,9	261,8	277,1
Average iter Bt	0,092815586	0,088755284	0,088874018
Average iter cg	0,013677897	0,011373067	0,010749232
N converged	10	10	10

Table 97: Results of exact derivatives of F79 quadratic with preconditioning

5.2 FUNCTION 27

$$\begin{aligned}
F(x) &= \frac{1}{2}\sum_{k=1}^{n+1} f_k^2(x) \\
f_k(x) &= \frac{1}{\sqrt{100000}}(x_k - 1), \quad 1 \le k \le n \\
f_{n+1}(x) &= \sum_{i=1}^{n} x_i^2 - \frac{1}{4} \\
x_l &= l, \quad l \ge 1
\end{aligned}$$

Exact derivatives

$$\begin{aligned}
\frac{\partial F(x)}{\partial x_k} &= \frac{1}{2}\left(\frac{1}{100000}(2x_k - 2) + 4x_k\left(\sum_{i=1}^{n} x_i^2\right) - x_k\right) \\
\frac{\partial^2 F(x)}{\partial^2 x_k} &= \frac{1}{2}\left(\frac{2}{100000} + 4\left(\sum_{i=1}^{n} x_i^2\right) + 8x_k^2 - 1\right) \\
\frac{\partial^2 F(x)}{\partial x_k \partial x_j} &= \frac{\partial^2 F(x)}{\partial x_j \partial x_k} = 4x_k x_j
\end{aligned}$$

Finite differences type 1

$$\frac{\partial F(x)}{\partial x_k} \approx \frac{1}{2}\left(\frac{1}{100000}(2x_k-2)+4x_k\left(\sum_{i=1}^{n} x_i^2\right)-x_k+4h^2x_k\right)$$
$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx \frac{1}{2}\left(\frac{2}{100000}+4\left(\sum_{i=1}^{n} x_i^2\right)+8x_k^2-1+2h^2\right)$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 4x_kx_j+h^2+2hx_j+2hx_k$$

Finite differences type 2

$$\frac{\partial F(x)}{\partial x_k} \approx \frac{1}{2}\left(\frac{1}{100000}(2x_k-2)+4x_k\left(\sum_{i=1}^{n} x_i^2\right)-x_k+4h^2|x_k|^2x_k\right)$$
$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx \frac{1}{2}\left(\frac{2}{100000}+4\left(\sum_{i=1}^{n} x_i^2\right)+8x_k^2-1+2h^2|x_k|^2\right)$$
$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 4x_kx_j+h^2|x_k||x_j|+2hx_j|x_k|+2hx_k|x_j|$$

NEW VERSION

```

    % Initialization of zj and j
    zj = z0;
    j= 0;

    if ~exact %approximation with finite difference (not exact)
        if fin_dif_2
            Azj= (gradf(xk+h.*abs(xk).*zj)-gradk)./(h*abs(xk));
        else
            Azj= (gradf(xk+h*zj)-gradk)/h;
        end
    end

       if ~exact %approximation with finite difference (not exact)

           if fin_dif_2
               z= (gradf(xk+h.*abs(xk).*p)-gradk)./(h*abs(xk));
           else
                z=(gradf(xk+h*p)-gradk)/h;
           end
       end

       if ~exact %approximation with finite difference (not exact)
            if fin_dif_2
                z_new= ((gradf(xk+h.*abs(xk).*p)-gradk)./(h*abs(xk)))';
            else
                z_new=((gradf(xk+h*p)-gradk)/h)';
            end
       end
```

5.2.1 Nelder-Mead method

Result with $n = 10$

Problem	Time	FinalValue	Iterations
x0	0,0156353	4,35423E-05	826
x1	0,0155162	4,95445E-05	1298
x2	0,0147933	4,4262E-05	1151
x3	0,0141024	4,36357E-05	1221
x4	0,0087644	4,49094E-05	983
x5	0,0095868	4,18928E-05	950
x6	0,0143182	4,26716E-05	1664
x7	0,0344026	3,92334E-05	1370
x8	0,0142046	5,40643E-05	876
x9	0,017337	4,0536E-05	1478
x10	0,009324	4,51286E-05	1125
Mean	**0,015271345**	**4,44928E-05**	**1176,545455**

Table 98: Result of F27, n=10, Nelder-Mead

Result with $n = 25$

Problem	Time	FinalValue	Iterations
x0	0,1334736	0,000110988	6833
x1	0,0586358	0,000111837	6381
x2	0,0802748	0,000110878	7972
x3	0,0616168	0,000108868	6745
x4	0,0884021	0,000110707	8692
x5	0,0637203	0,000109901	6836
x6	0,0731952	0,000114476	7820
x7	0,0589386	0,000110833	6528
x8	0,0573529	0,000111574	6062
x9	0,088446	0,000112699	9800
x10	0,0511614	0,000110347	5640
Mean	**0,074110682**	**0,000111192**	**7209,909091**

Table 99: Result of F27, n=25, Nelder-Mead

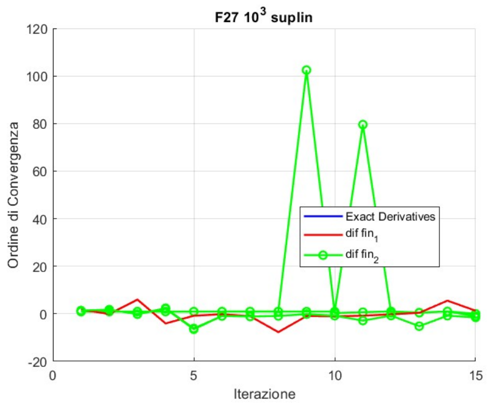
Result with $n = 50$

Problem	Time	FinalValue	Iterations
x0	0,2270877	0,00023083	16226
x1	0,3836984	0,00022999	28313
x2	0,2326935	0,000228422	18140
x3	0,3804489	0,000234437	27507
x4	0,304543	0,000228956	21933
x5	0,2797645	0,000232844	18730
x6	0,2734104	0,000226026	16162
x7	0,4140496	0,000227525	19645
x8	0,649336	0,000232526	25173
x9	0,6481402	0,000230922	21532
x10	0,7279965	0,000231636	21260
Mean	**0,411015336**	**0,000230374**	**21329,18182**

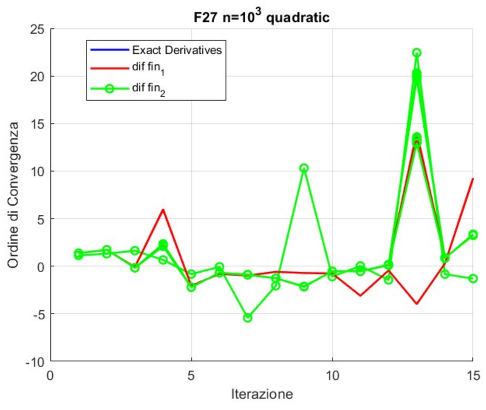
Table 100: Result of F27, n=50, Nelder-Mead

5.2.2 Truncated Newton method

Result with $n = 10^3$



(a) The last 15 value of experimental rate of convergence F27 $n = 10^3$ superlinear



(b) The last 15 values of experimental rate of convergence F27 $n = 10^3$ quadratic

Figure 15: The last 15 values of experimental rate of convergence F27 $n = 10^3$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	0	11	11	11	11	11	11

Table 101: Number of converged processes out of 11 initial conditions $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	1	11	11	11	11	11	11

Table 102: Number of converged processes out of 11 initial conditions $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088
FD2	Nan	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088

Table 103: Average function minimum value found $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088
FD2	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088	0,004843088

Table 104: Average function minimum value found $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	42	41	41	41	41	41
FD2	Nan	41	41	41	41	41	41

Table 105: Average number of iterations $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	41	41	41	41	41	41
FD2	40	41	41	41	41	41	41

Table 106: Average number of iterations $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,012942436	0,011721	0,010156264	0,012622455	0,010907036	0,021423909
FD2	Nan	0,015114364	0,013620936	0,014552109	0,013211882	0,0134954	0,021423909

Table 107: Average execution time $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,0132438	0,010158645	0,007151345	0,008478936	0,008292427	0,048330218
FD2	0,0347983	0,0183122	0,012602718	0,010286091	0,011775491	0,010540691	0,048330218

Table 108: Average execution time $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	Nan	0	0	0	0	0	0

Table 109: Average times negative curvature condition satisfied $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	8	0,818181818	0	0	0	0	0

Table 110: Average times negative curvature condition satisfied $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,261904762	1,219512195	1,219512195	1,219512195	1,219512195	1,219512195
FD2	Nan	1,219512195	1,219512195	1,219512195	1,219512195	1,219512195	1,219512195

Table 111: Average of the average number of CG iterations (inner loops)$n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,365853659	1,317073171	1,317073171	1,317073171	1,317073171	1,317073171
FD2	5,625	1,625277162	1,341463415	1,317073171	1,317073171	1,317073171	1,317073171

Table 112: Average of the average number of CG iterations (inner loops) $n = 10^3$ quadratic

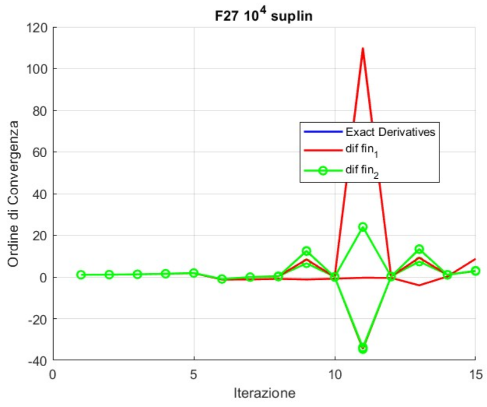
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,095238095	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976
FD2	Nan	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976

Table 113: Average of the average number of backtracking iterations $n = 10^3$ superlinear

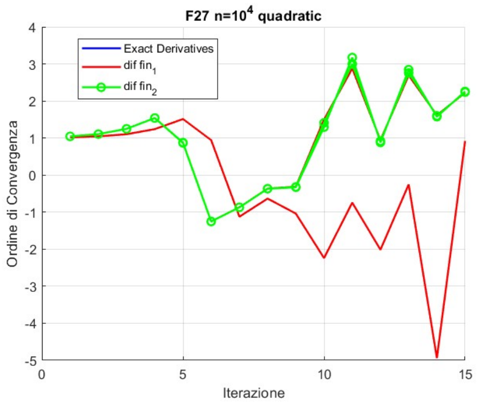
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488
FD2	0,5	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488

Table 114: Average of the average number of backtracking iterations $n = 10^3$ quadratic

Result with $n = 10^4$



(a) The last 15 value of experimental rate of convergence F27 $n = 10^4$ superlinear



(b) The last 15 values of experimental rate of convergence F27 $n = 10^4$ quadratic

Figure 16: The last 15 values of experimental rate of convergence F27 $n = 10^4$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	0	11	11	11	11	11	11

Table 115: Number of converged process out of 11 initial conditions $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	0	11	11	11	11	11	11

Table 116: Number of converged process out of 11 initial conditions $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756
FD2	Nan	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756

Table 117: Average function minimum value found $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756
FD2	Nan	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756	0,049500756

Table 118: Average function minimum value found $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	47	47	47	47	47	47
FD2	Nan	47	47	47	47	47	47

Table 119: Average number of iterations $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	46	47	47	47	47	47
FD2	Nan	47	47	47	47	47	47

Table 120: Average number of iterations $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,239780991	0,163524355	0,189020209	0,185570673	0,179425827	0,167516736
FD2	Nan	0,235294727	0,209697264	0,235221345	0,233327491	0,241269191	0,167516736

Table 121: Average execution time $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,2225699	0,201023482	0,201411464	0,192743164	0,175465664	0,143637027
FD2	Nan	0,259827591	0,275391973	0,266899527	0,240828273	0,230133973	0,143637027

Table 122: Average execution time $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	Nan	0	0	0	0	0	0

Table 123: Average times negative curvature condition satisfied $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	Nan	1	1	0	0	0	0

Table 124: Average times negative curvature condition satisfied $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,170212766	1,212765957	1,212765957	1,212765957	1,212765957	1,212765957
FD2	Nan	1,234042553	1,212765957	1,212765957	1,212765957	1,212765957	1,212765957

Table 125: Average of the average number of CG iterations (inner loops) $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,217391304	1,234042553	1,234042553	1,234042553	1,234042553	1,234042553
FD2	Nan	1,319148936	1,234042553	1,234042553	1,234042553	1,234042553	1,234042553

Table 126: Average of the average number of CG iterations (inner loops) $n = 10^4$ quadratic

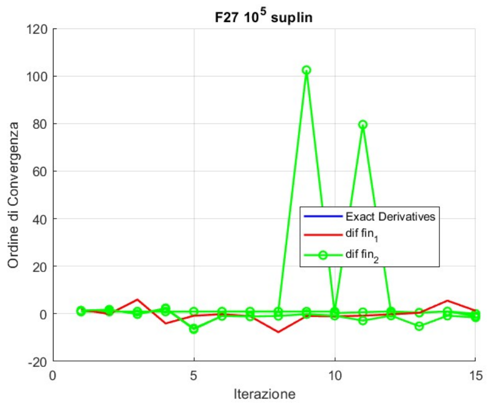
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,063829787	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191
FD2	Nan	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191

Table 127: Average of the average number of backtracking iterations $n = 10^4$ superlinear

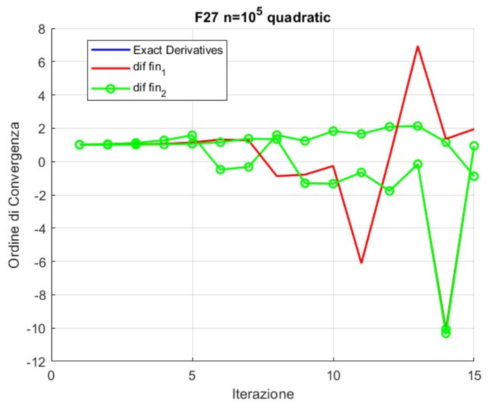
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,02173913	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191
FD2	Nan	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191

Table 128: Average of the average number of backtracking iterations $n = 10^4$ quadratic

Result with $n = 10^5$



(a) The last 15 value of experimental rate of convergence F27 $n = 10^5$ superlinear



(b) The last 15 values of experimental rate of convergence F27 $n = 10^5$ quadratic

Figure 17: The last 15 values of experimental rate of convergence F27 $n = 10^5$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	0	11	11	11	11	11	11

Table 129: Number of converged processes out of 11 initial conditions $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0	11	11	11	11	11	11
FD2	11	11	11	11	11	11	11

Table 130: Number of converged processes out of 11 initial conditions $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158
FD2	Nan	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158

Table 131: Average function minimum value found $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158
FD2	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158	0,498415158

Table 132: Average function minimum value found $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	54	53	53	53	53	53
FD2	Nan	53	53	53	53	53	53

Table 133: Average number of iterations $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	53	52	52	52	52	52
FD2	55	52	52	52	52	52	52

Table 134: Average number of iterations $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,539437664	1,262051645	1,304349464	1,312664036	1,235337036	1,037782909
FD2	Nan	1,447233882	1,582913873	1,610615727	1,5469376	1,534066991	1,037782909

Table 135: Average execution time $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,118114336	1,071511009	1,021770664	1,072485282	1,029379373	0,844162873
FD2	1,682657736	1,457743	1,339080236	1,332187864	1,327400382	1,2827277	0,844162873

Table 136: Average execution time $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	Nan	0	0	0	0	0	0

Table 137: Average times negative curvature condition satisfied $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	4	1	0	0	0	0	0

Table 138: Average times negative curvature condition satisfied $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,12962963	1,150943396	1,150943396	1,150943396	1,150943396	1,150943396
FD2	Nan	1,150943396	1,150943396	1,150943396	1,150943396	1,150943396	1,150943396

Table 139: Average of the average number of CG iterations (inner loops) $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	1,169811321	1,153846154	1,153846154	1,153846154	1,153846154	1,153846154
FD2	1,272727273	1,153846154	1,153846154	1,153846154	1,153846154	1,153846154	1,153846154

Table 140: Average of the average number of CG iterations (inner loops) $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0,018867925	0,018867925	0,018867925	0,018867925	0,018867925
FD2	Nan	0,018867925	0,018867925	0,018867925	0,018867925	0,018867925	0,018867925

Table 141: Average of the average number of backtracking iterations $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	Nan	0	0	0	0	0	0
FD2	0	0	0	0	0	0	0

Table 142: Average of the average number of backtracking iterations $n = 10^5$ quadratic

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,021423909	0,167516736	1,037782909
Average Iter	41	47	53
Average fval	0,004843088	0,049500756	0,498415158
Violation	0	0	0
Average iter Bt	0,097560976	0,042553191	0,018867925
Average iter cg	1,219512195	1,212765957	1,150943396
N converged	11	11	11

Table 143: Average of results with exact derivatives $n = 10^3, 10^4, 10^5$ superlinear

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,048330218	0,143637027	0,844162873
Average Iter	41	47	52
Average fval	0,004843088	0,049500756	0,498415158
Violation	0	0	0
Average iter Bt	0,048780488	0,042553191	0
Average iter cg	1,317073171	1,234042553	1,153846154
N converged	11	11	11

Table 144: Average of results with exact derivatives $n = 10^3, 10^4, 10^5$ quadratic

5.3 FUNCTION 16 - BANDED TRIGONOMETRIC PROBLEM

$$\begin{aligned} F(x) &= \sum_{i=1}^{n} i\left[(1-\cos(x_i)) + \sin(x_i) - 1 - \sin(x_{i+1})\right] \\ x_0 &= x_{n+1} = 0, \quad x_i = 1, \quad i \geq 1 \end{aligned}$$

Exact derivatives

$$\begin{aligned} \frac{\partial F(x)}{\partial x_k} &= k \sin x_k - 2\cos x_k \quad \forall k = 1...n-1 \\ \frac{\partial F(x)}{\partial x_n} &= n \sin x_n + (n-1)\cos x_n \\ \frac{\partial^2 F(x)}{\partial^2 x_k} &= k \cos x_k + 2\sin x_k \quad \forall k = 1...n-1 \\ \frac{\partial^2 F(x)}{\partial^2 x_n} &= (1-n)\sin x_n + n\cos x_n \\ \frac{\partial^2 F(x)}{\partial x_k \partial x_j} &= 0 \quad \forall k = 1...n \end{aligned}$$

Finite differences type 1

$$\begin{aligned} \frac{\partial F(x)}{\partial x_k} &\approx \frac{\sin(h)}{h}\left(2\cos(x_k) + k\sin(x_k)\right) \quad \forall k = 1...n-1 \\ \frac{\partial F(x)}{\partial x_n} &\approx \frac{\sin(h)}{h}\left(n\sin(x_k) - (n-1)\cos(x_k)\right) \\ \frac{\partial^2 F(x)}{\partial^2 x_k} &\approx \left(\frac{h^2}{4*3} - 1\right)\left(2\sin(x_k) - k\cos(x_k)\right) \quad \forall k = 1...n-1 \\ \frac{\partial^2 F(x)}{\partial^2 x_n} &\approx \left(1 - \frac{h^2}{4*3}\right)\left((n-1)\sin(x_n) + n\cos(x_n)\right) \\ \frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} &\approx 0 \quad \forall k, j = 1...n \end{aligned}$$

Finite differences type 2

$$
\begin{aligned}
\frac{\partial F(x)}{\partial x_k} &\approx \frac{\sin(h|x_k|)}{h|x_k|}\left(2\cos(x_k)+k\sin(x_k)\right) \quad \forall k=1...n-1\\
\frac{\partial F(x)}{\partial x_n} &\approx \frac{\sin(h|x_n|)}{h|x_n|}\left(n\sin(x_k)-(n-1)\cos(x_k)\right)\\
\frac{\partial^2 F(x)}{\partial^2 x_k} &\approx \left(\frac{h|x_k|^2}{4*3}-1\right)\left(2\sin(x_k)-k\cos(x_k)\right) \quad \forall k=1...n-1\\
\frac{\partial^2 F(x)}{\partial^2 x_n} &\approx \left(1-\frac{h|x_n|^2}{4*3}\right)\left((n-1)\sin(x_n)+n\cos(x_n)\right)\\
\frac{\partial^2 F(x)}{\partial x_k \partial x_j} &\approx 0 \quad \forall k,j=1...n
\end{aligned}
$$

NEW VERSION

```
function HF=HF16New(x,sparse,exact,fin_dif_2,h,JF)
% Function that computes the Hessian of function 79
% sparse= bool. True= computes the sparse version
% exact= bool. True= computes the exact version, False= computes the approximated version
   with finite differences
% fin_dif_2= bool. True if exact=false and finite differences are computed using as
  increment h*abs(x_j) for the derivative with respect to j
% h= increment for the approximated version (if exact=true put h=0)
% JF= function handle of the gradient

n=length(x);
if exact %exact version
    indices=(1:n)';
    D=indices.*cos(x)-2*sin(x);
    if sparse %sparse version
        HF=spdiags(D,0,n,n);
    else % NOT sparse version
        HF=diag(D);
    end
    HF(n,n)=(n-1)*sin(x(n))+n*cos(x(n));
else  %approximation with finite difference (not exact)
    if fin_dif_2 % version of finite differences with abs(xj)
          d=(JF(x+h.*abs(x).*ones(n,1))-JF(x-h.*abs(x).*ones(n,1)))./(2*h*abs(x));
          if sparse %sparse version
              HF =spdiags(d,0,n,n);
          else % NOT sparse
              HF=diag(d);
          end
    else % classic version of finite differences
          d=(JF(x+h*ones(n,1))-JF(x-h*ones(n,1)))/(2*h);
          if sparse %sparse version
              HF =spdiags(d,0,n,n);
          else % NOT sparse
              HF=diag(d);
          end
    end
end

end
```

5.3.1 Nelder-Mead method

Result with $n = 10$

Initial condition	**Time**	**FinalValue**	**Iterations**
x0	0.0257791	-8.051392105006307	573
x1	0.0186579	-8.05139210506313	900
x2	0.0152838	-8.05139210506308	696
x3	0.0203209	-8.05139210506315	815
x4	0.0144454	-8.05139210506306	582
x5	0.0228020	-8.05139210506309	676
x6	0.0181076	-8.05139210506312	703
x7	0.0163092	-8.05139210506305	632
x8	0.0141219	-8.05139210506292	601
x9	0.0151389	-8.05139210506305	730
x10	0.0147382	-8.05139210506298	702
Mean	**0.017791355**	**-8.05139210506306**	**691.8181818**

Table 145: Results of F16 n=10 Nelder-Mead

Result with $n = 25$

Problem	**Time**	**FinalValue**	**Iterations**
x0	0.1227638	-16.1379269689142	6360
x1	0.1110022	-16.1379269689143	6141
x2	0.1061365	-16.1379269689143	6246
x3	0.1136934	-16.1379269689143	7119
x4	0.1202319	-16.1379269689143	6213
x5	0.1273512	-16.1379269689142	5737
x6	0.0932860	-16.1379269689143	5547
x7	0.1167377	-16.1379269689142	7367
x8	0.1205107	-16.1379269689143	7041
x9	0.0990386	-16.1379269689143	5601
x10	0.1253703	-16.1379269689143	7374
Mean	**0.114192936**	**-16.1379269689143**	**6431.454545**

Table 146: Results of F16 n=25 Nelder-Mead

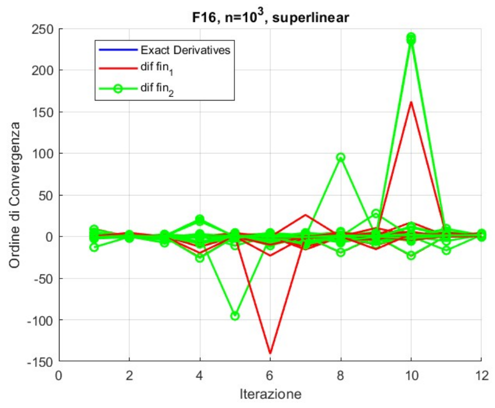
Result with $n = 50$

Problem	**Time**	**FinalValue**	**Iterations**
x0	1.0077609	-27.8948622787264	38856
x1	1.4635351	-27.8948622787267	38738
x2	1.4855457	-27.8948622787263	38223
x3	2.7477621	-27.8948622787267	75595
x4	1.3581615	-27.8948622787243	36980
x5	1.9837365	-27.8948622787261	48035
x6	1.6769416	-27.8948622787268	38812
x7	3.5937880	-27.8948622787266	60807
x8	1.9009525	-27.8948622787263	41588
x9	2.5503011	-27.8948622787261	47869
x10	2.3369653	-27.8948622787269	38145
Mean	**2.009586391**	**-27.8948622787263**	**45786.18182**

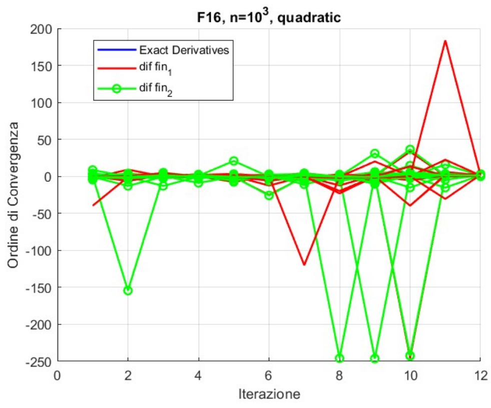
Table 147: Results of F16 n=50 Nelder-Mead

5.3.2 Truncated Newton Method

Result with $n = 10^3$



(a) The last 12 value of experimental rate of convergence F16 $n = 10^3$ superlinear



(b) The last 12 values of experimental rate of convergence F16 $n = 10^3$ quadratic

Figure 18: The last 12 values of experimental rate of convergence F16 $n = 10^3$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	10	8	9	9	9	9
FD2	7	9	10	10	9	9	9

Table 148: Number of converged processes out of 11 initial conditions $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	9	10	10	10	10	10
FD2	8	10	10	10	10	10	10

Table 149: Number of converged processes out of 11 initial conditions $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764
FD2	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764

Table 150: Average function minimum value found $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764
FD2	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764	-427.4044764

Table 151: Average function minimum value found $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	22.5	22.2	22.0	22.33333333	22.33333333	22.33333333	22.33333333
FD2	22.42857143	22.22222222	22.1	22.2	22.33333333	22.33333333	22.33333333

Table 152: Average number of iterations $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	20.6	20.44444444	21.0	20.8	20.8	20.8	20.8
FD2	20.375	20.9	20.3	20.5	20.8	20.8	20.8

Table 153: Average number of iterations $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0.00661609	0.00660261	0.006798163	0.006580044	0.006635511	0.006544878	0.014478189
FD2	0.034171786	0.032698156	0.03230913	0.03244219	0.033012233	0.0327274	0.014478189

Table 154: Average execution time $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0.0055531	0.006017667	0.00614935	0.00606408	0.00570859	0.00548137	0.00597978
FD2	0.029210463	0.03075712	0.02896457	0.02921496	0.03052635	0.02971453	0.00597978

Table 155: Average execution time $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	4.000000000	3.700000000	3.625000000	3.666666667	3.666666667	3.666666667	3.666666667
FD2	3.857142857	3.777777778	3.700000000	3.700000000	3.666666667	3.666666667	3.666666667

Table 156: Average times negative curvature condition satisfied $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	3.600000000	3.666666667	3.700000000	3.700000000	3.700000000	3.700000000	3.700000000
FD2	3.750000000	3.700000000	3.700000000	3.700000000	3.700000000	3.700000000	3.700000000

Table 157: Average times negative curvature condition satisfied $n = 10^3$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	13,21628845	13,13441519	13,24576402	13,56649997	13,56649997	13,56649997	13,56649997
FD2	13,02899196	12,85397843	13,22039109	13,47969281	13,56649997	13,56649997	13,56649997

Table 158: Average of the average number of CG iterations (inner loops) $n = 10^3$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	14,4599335	13,79607009	14,52293651	14,46338137	14,46338137	14,46338137	14,46338137
FD2	13,73568354	14,44545291	13,76197691	14,08873851	14,46338137	14,46338137	14,46338137

Table 159: Average of the average number of CG iterations (inner loops) $n = 10^3$ quadratic

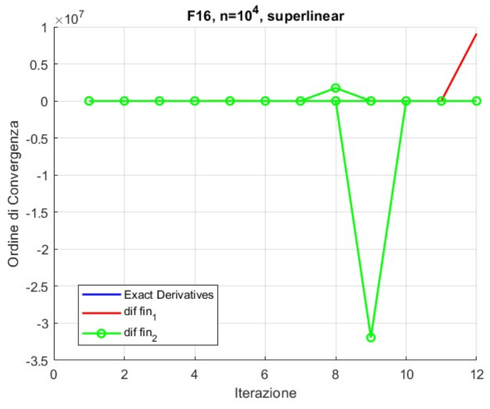
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,501573081	0,431463247	0,368521605	0,461148594	0,461148594	0,461148594	0,461148594
FD2	0,425185529	0,41514874	0,416245856	0,450235598	0,461148594	0,461148594	0,461148594

Table 160: Average of the average number of backtracking iterations $n = 10^3$ superlinear

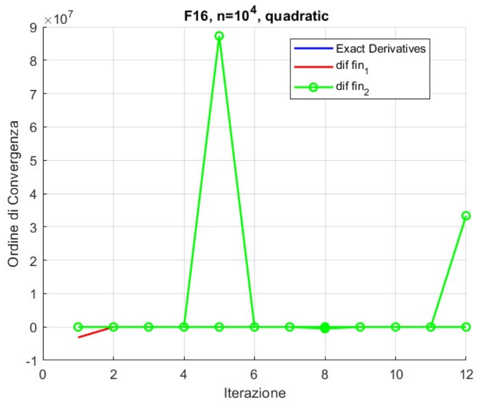
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,453264948	0,425322061	0,99746633	0,448700919	0,448700919	0,448700919	0,448700919
FD2	0,461958152	0,471916055	0,441742424	0,434685767	0,448700919	0,448700919	0,448700919

Table 161: Average of the average number of backtracking iterations $n = 10^3$ quadratic

Result with $n = 10^4$



(a) The last 12 value of experimental rate of convergence F16 $n = 10^4$ superlinear



(b) The last 12 values of experimental rate of convergence F16 $n = 10^4$ quadratic

Figure 19: The last 12 values of experimental rate of convergence F16 $n = 10^4$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7	8	10	7	7	7	7
FD2	10	7	8	9	7	7	7

Table 162: Number of converged processes out of 11 initial conditions $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	9	9	9	9	9	9	9
FD2	10	8	10	9	9	9	9

Table 163: Number of converged processes out of 11 initial conditions $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448
FD2	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448

Table 164: Average function minimum value found $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448
FD2	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448	-4159.932448

Table 165: Average function minimum value found $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	30.71428571	30.5	31.2	28.14285714	28.14285714	28.14285714	28.14285714
FD2	30.9	30.71428571	32.75	29.66666667	28.14285714	28.14285714	28.14285714

Table 166: Average number of iterations $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	32.33333333	29.22222222	29.44444444	29.55555556	29.55555556	29.55555556	29.55555556
FD2	32.9	28.875	39.6	29.33333333	29.55555556	29.55555556	29.55555556

Table 167: Average number of iterations $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,111040086	0,107913063	0,12545102	0,102662686	0,098604643	0,096207814	0,103340243
FD2	0,46977598	0,451592057	0,50932155	0,436318722	0,409451029	0,412165	0,103340243

Table 168: Average execution time $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,155196778	0,109654889	0,113168667	0,113572856	0,111070889	0,113361089	0,109789589
FD2	0,55448594	0,434380763	0,74857033	0,436595844	0,443202633	0,444925844	0,109789589

Table 169: Average execution time $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7,285714286	7,875	6,8	5,571428571	5,571428571	5,571428571	5,571428571
FD2	6,7	7	6,5	6,555555556	5,571428571	5,571428571	5,571428571

Table 170: Average times negative curvature condition satisfied $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7,222222222	7,444444444	7,111111111	6,888888889	6,888888889	6,888888889	6,888888889
FD2	7,1	6,5	6,7	7,555555556	6,888888889	6,888888889	6,888888889

Table 171: Average times negative curvature condition satisfied $n = 10^4$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	29,70578186	27,52531806	29,22165798	27,96220849	27,96220849	27,96220849	27,96220849
FD2	29,83583197	27,32313105	32,97042508	27,21293409	27,96220849	27,96220849	27,96220849

Table 172: Average of the average number of CG iterations (inner loops) $n = 10^4$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	33,91543736	29,34801524	30,57613089	30,3267502	30,3267502	30,3267502	30,3267502
FD2	33,9474141	29,25566471	40,44665986	28,09979699	30,3267502	30,3267502	30,3267502

Table 173: Average of the average number of CG iterations (inner loops) $n = 10^4$ quadratic

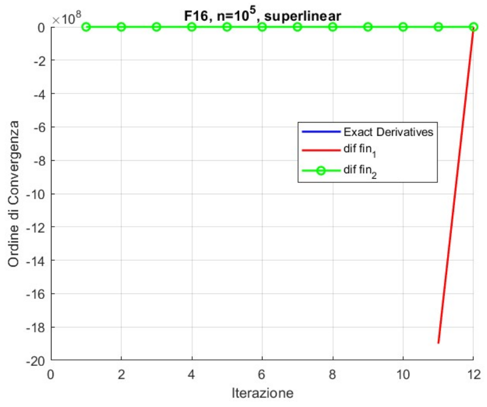
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,43710686	0,364964526	0,91230475	0,274938323	0,274938323	0,274938323	0,274938323
FD2	0,78700118	0,369798728	1,11886688	0,347550951	0,274938323	0,274938323	0,274938323

Table 174: Average of the average number of backtracking iterations $n = 10^4$ superlinear

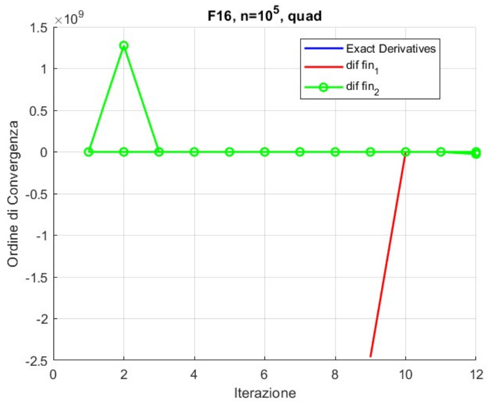
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,573693933	0,392162411	0,581851996	0,373341243	0,373341243	0,373341243	0,373341243
FD2	1,800345009	0,362367185	4,289922801	0,389773489	0,373341243	0,373341243	0,373341243

Table 175: Average of the average number of backtracking iterations $n = 10^4$ quadratic

Result with $n = 10^5$



(a) The last 12 value of experimental rate of convergence F16 $n = 10^5$ superlinear



(b) The last 12 values of experimental rate of convergence F16 $n = 10^5$ quadratic

Figure 20: The last 12 values of experimental rate of convergence F16 $n = 10^5$

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	11	10	11	10	10	10	10
FD2	10	11	11	9	10	10	10

Table 176: Number of converged processes out of 11 initial conditions $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	10	9	9	11	11	11	11
FD2	9	11	11	9	11	11	11

Table 177: Number of converged processes out of 11 initial conditions $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831
FD2	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831

Table 178: Average function minimum value found $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831
FD2	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831	-41443,75831

Table 179: Average function minimum value found $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	46,18181818	48,3	88,72727273	71,7	71,7	71,7	71,7
FD2	46,6	49,27272727	156,8181818	47,55555556	71,7	71,7	71,7

Table 180: Average number of iterations $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	87,9	64,22222222	46,11111111	112,6363636	112,6363636	112,6363636	112,6363636
FD2	138,1111111	50,72727273	48	96,55555556	112,6363636	112,6363636	112,6363636

Table 181: Average number of iterations $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	1,805642291	1,82645465	7,522901136	5,09686661	5,1365726	5,08613145	5,09064755
FD2	6,84923564	7,250678091	33,50863898	7,038724822	12,94518934	12,87035541	5,09064755

Table 182: Average execution time $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7,44131593	4,124659544	1,790884411	10,72272155	10,80732529	10,73784363	10,71965694
FD2	29,60867177	7,697176027	7,160017991	19,09137196	23,05572195	23,13530151	10,71965694

Table 183: Average execution time $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7	7,6	6,818181818	7,5	7,5	7,5	7,5
FD2	6	7,454545455	6,818181818	7	7,5	7,5	7,5

Table 184: Average times negative curvature condition satisfied $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	7,4	6,222222222	5,888888889	6,818181818	6,818181818	6,818181818	6,818181818
FD2	5,555555556	7,454545455	6,818181818	5,555555556	6,818181818	6,818181818	6,818181818

Table 185: Average times negative curvature condition satisfied $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	52,37982844	52,88016219	57,93746012	56,48235789	56,48235789	56,48235789	56,48235789
FD2	53,57423435	54,00620316	60,23810531	54,1258101	56,48235789	56,48235789	56,48235789

Table 186: Average of the average number of CG iterations (inner loops) $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	59,56973675	59,89891512	56,01536435	60,62053027	60,62053027	60,62053027	60,62053027
FD2	59,72343733	56,12745288	56,65852298	62,27479657	60,62053027	60,62053027	60,62053027

Table 187: Average of the average number of CG iterations (inner loops) $n = 10^5$ quadratic

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	0,111160835	0,099096348	2,901269123	3,47442024	3,47442024	3,47442024	3,47442024
FD2	0,101583846	0,122831622	4,388417883	0,103586084	3,47442024	3,47442024	3,47442024

Table 188: Average of the average number of backtracking iterations $n = 10^5$ superlinear

Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	**Exact**
FD1	3,61276951	2,987962831	0,080283777	3,682467125	3,682467125	3,682467125	3,682467125
FD2	3,920093584	0,737906641	0,116674267	3,66585017	3,682467125	3,682467125	3,682467125

Table 189: Average of the average number of backtracking iterations $n = 10^5$ quadratic

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,014478189	0,103340243	5,09064755
Average Iter	22,33333333	28,14285714	71,7
Average fval	-427,4044764	-4159,932448	-41443,75831
Violation	3,666666667	5,571428571	7,5
Average iter Bt	0,461148594	0,274938323	3,47442024
Average iter cg	13,56649997	27,96220849	56,48235789
N converged	9	7	10

Table 190: Results of exact derivatives of F16 superlinear

Row	$n = 10^3$	$n = 10^4$	$n = 10^5$
Average Time	0,00597978	0,109789589	10,71965694
Average Iter	20,8	29,55555556	112,6363636
Average fval	-427,4044764	-4159,932448	-41443,75831
Violation	3,7	6,888888889	6,818181818
Average iter Bt	0,448700919	0,373341243	3,682467125
Average iter cg	14,46338137	30,3267502	60,62053027
N converged	10	9	11

Table 191: Results of exact derivatives of F16 quadratic

6 Comments

6.1 Nelder-Mead method

6.1.1 F79

As can be seen from table 7, for $n = 10$, the algorithm reaches convergence for all starting points and the function minimum values are comparable. This means that the simplex contracts around the same area or in two different local minima with the same function value. The average execution time is very low but sensitive to the starting point: for x_5, it is 0,1135344, while for x_1, it is only 0.0429871. The average number of iterations is approximately 2532.

Regarding $n = 25$, it can be observed from table 8 that the average execution time and the number of iterations have significantly increased compared to $n = 10$, but overall, the execution time remains very low. In this case, the algorithm converges to two different local minima: for $x_0, x_1, x_2, x_3, x_6, x_7, x_8$, it reaches 3.99908828056904 with a tolerance of 10^{-13}, while for the other starting points, the algorithm converges to 3.87482796806862, which are evidently both local minima of the function.

For $n = 50$, a significant increase in execution time and the number of iterations can again be observed in table 9. In this case, for all starting points, the algorithm stops in the same region of space but each time at a different point: the point with the minimum function value found through the simplex exploration.

6.1.2 F27

As can be seen from the tables 98, 99, 100, the regions of the minimum values found in the three dimensions are consistent with the dimension: for $n = 10$, the average is 4.44928×10^{-5}; for $n = 25$, it is 0.000111192; and for $n = 50$, it is 0.000230374. These results are also consistent with the minima found by the Truncated Newton algorithm for the same problem in higher dimensions. This suggests that the algorithm likely reaches a region of global minimum.

Observing the number of iterations as n increases, it can be seen that there is more than linear growth and that, even for $n = 10$, the number of iterations is significant, reaching an average of 21329 iterations for the largest n.

In any case, comparing the averages obtained in this problem with those of F79, it can be observed that the number of iterations is approximately half the times of the F79's one in the three dimensions, and that the execution time of F79 is significantly higher, exceeding that required for F27 by more than an order of magnitude.

Although the final function value is not significantly influenced by the starting point, the execution time and the number of iterations, for the three dimensions, are quite dependent on it.

6.1.3 F16

In this case, for $n = 10^3$, $n = 10^4$, and $n = 10^5$, all initial conditions converge to the same minimum value of the function. It can be observed that the execution time for $n = 10$ is on the order of 10^{-2} seconds, for $n = 25$ it increases by one order of magnitude, and for $n = 50$, it doubles that order of magnitude.

Regarding the number of iterations, the trend between $n = 10$ and $n = 25$ follows the same pattern, increasing by one order of magnitude, from an average of 691.8 to an average of 6432, while for $n = 50$, the average is 45786.

For $n = 10$, despite all initial conditions converging to the same minimum value, the algorithm appears to be sensitive to the starting point in terms of the number of iterations, with cases requiring up to 900 iterations and others only 582. The same holds for $n = 50$, where the number of iterations varies from 755595 to 36980 depending on the starting point.

6.2 Truncated Newton method

While dealing with this algorithm we had to always consider storing the Hessian matrix. As we knew that we would have worked with large dimensions, we decided to always store the matrix as sparse whenever it was possible. This way we were able to save a lot of memory space. In fact, for F16 both the Hessian and its approximated version are diagonal matrices and for F79 they are all tridiagonal matrices: saving them as dense would have been a huge waste of efficiency.
Moreover all finite differences were directly computed for each function in order to work with a specific and efficient implementation that could perform well in every dimension.

6.2.1 F79

We start analyzing the results obtained without using preconditioning.

It can be easily observed in the presented plots 9, 11, 13 that for every dimension the experimental rate of convergence goes to about 1 in the last iterations: this means that for every dimension this method converges with a linear rate. It can also be watched that the behavior does not change while varying the forcing term: as we said in the first part of the report, the order of convergence is well-known with the chosen η_k just while working under suitable assumptions with the basic inexact Newton method and not with the truncated version we are studying.

Then it can be seen that the method works extremely well on the function: at least 10 out of 11 runs are always successful, with both forcing terms in all dimensions. Furthermore, from all the initial points it converges into a global minimum since the minimum function value is always almost zero, as shown in tables 14,15,42,43,70,71, and F79 is a sum of squares, so it is non negative for construction and cannot be smaller than the values found.

It is interesting to point out that with both forcing terms while the dimension grows larger the number of iterations does not increase much (as displayed in tables 18,19,46,47,74,75 they are around 1072 when $n = 10^3$ and around 1197 when $n = 10^5$). However, there is a huge difference in the needed execution time that grows slightly more than of one order of magnitude from one dimension to another, as can be read in tables22,23,50,51,78,79. While comparing the execution times it can be also seen that the performance with the exact derivatives is always better (around 2/3) than the one obtained with the approximated version, even though the number of steps is the same, and that the behavior with the two different approximations is almost the same in terms of time and iterations needed.

Examining tables 26,27,54,55,82,83, it comes out that the number of average violation is close to the one of the total iterations, meaning that in most of the steps the iterations of the CG method are stopped due to the satisfaction of the negative curvature condition. Furthermore, comparing this result with tables 30,31,58,59,86,87, it becomes obvious that in most of the iterations the descent direction chosen is the steepest descent one. In fact, the CG iterations' average number is almost zero, meaning that with few exceptions the number of inner iterations is null, due to the curvature condition as just said, and so the solution of the linear system is its residual, i.e. the steepest descent direction.

Lastly, considering tables 34,35,62,63,90,91, it can be observed that also the average number of backtrackings is almost zero in every dimension, meaning that in most of the iterations the Armijo condition is already satisfied by $\alpha = 1$ (step of the basic Newton method) and there is no need to exploit the inexact linesearch method to find a suitable steplength. Besides it can be read in tables 90 and 91 that for $n = 10^5$ the number of backtrackings for every step is exactly zero.

It is significant to highlight that all the above written observations are true for both forcing terms and for both types of finite differences.

Recalling that the this function's Hessian matrix is tridiagonal and symmetric, we have decided to choose the Gauss-Seidel preconditioning which works with M, that is built as the lower-triangular matrix with the same lower-triangular and diagonal elements of the original one and the null upper-triangle.
Convergence property is lost when applying the preconditioning to the Hessian matrix: as we can see in the plots 10, 12, 14 there is no asymptotic value at which the experimental rate of convergence stabilizes, so we cannot discuss the order of convergence.

When applying the preconditioning the method keeps working well as readable in tables 12,13,40,41,68,69: 10 out of 11 runs converge in all the shown cases. Furthermore, the method keeps converging to global minimum, as it can be deduced by the almost null minimum values presented in tables 16,17,44,45,72,73.

What changes significantly is the number of iterations needed to reach convergence that goes to approximately 250 for $n = 10^3$ and slightly increases while n grows large, but always staying below 300, see tables 20,21,48,49,76,77.

Nonetheless, the execution times do not decrease: to be more precise they increase too in higher dimensions, see tables 24,25,52,53,80,81. Furthermore we can observe that, using the preconditioning, the two types of finite differences start behaving differently: the approximation with h_i takes longer execution times with respect to the classical finite differences with the same increment h. By the way, tests with both finite differences keep taking longer execution times than the same ones done using exact derivatives.
This behavior, compared to the one observed without the preconditioning application, can be explained by the introduction of new operations in every iteration due to the preconditioning: in fact, for every step k there are $j_k + 2$ more linear systems to solve, where j_k is the number of inner iterations (CG iterations) of that step. Even though these systems have M as coefficient-matrix and so are easy to be solved and the number of inner iterations is on average almost null (see tables 32,33,60,61,88,89), this still leads to higher computational times.

What has been already observed about the average number of times the negative curvature condition is satisfied is still true (see tables 28,29,56,57,84,85), and the same can be said about the number of CG iterations (see tables 32,33,60,61,88,89) and the backtracking behavior (see tables 36,37,64,65,92,93). We can summarize it reminding that also with preconditioning in most iterations the descent direction used is the steepest descent one that satisfies the negative curvature condition, causing the CG method to stop prematurely, and that the step length chosen is directly $\alpha = 1$ with no need of backtracking.

Reading tables 92 and 93 an exception can be noticed: with $n = 10^5$ and derivatives approximated with the second type of finite differences the average of the mean number of backtracking iterations is around 4.6, with both forcing terms. In that case it is also witnessed that the method does not converge to a global minimum (found with the exact derivatives and with classical finite differences), as it can be read in tables 72 and 73, where it figures an exceptional minimum value: 3681,8. This can be explained with the transition through a region where smaller steps are needed to satisfy the Armijo condition and with the convergence to a local minimum of the process starting in the only initial point that leads to a failing run in all the other cases. In fact, these cases are the only ones where convergence is reached with all the initial points (see tables 68,69). This would also explain why the average number of iterations and the average execution times are higher for $n = 10^5$ with the second type of finite differences, as can be noticed in tables 76,77,80,81. From those tables, it can be affirmed that also with preconditioning with both forcing terms the performance with exact derivatives is better.

6.2.2 F27

Since the Hessian matrix of the function F27 is not sparse, storing it in memory becomes unfeasible for large dimension (10^5) problems. However, since all operations in the algorithm just involve the product of the Hessian matrix with a vector, we opted to compute and store only the resulting product H*z instead of the full matrix. Similarly, we only saved the Hessian-vector products instead of the entire matrix, while using the approximated version computed with both types of finite differences. For this reason, we did not apply preconditioning to the Hessian matrix of F27.

As shown in the tables 101 115 129 102, 116 130, except for the finite differences method with $h = 10^{-2}$, all other cases converge. The only case with $h = 10^{-2}$ that converges is when $n = 10^5$ with the quadratic forcing term and second-type finite differences, table 130.

In all three dimensions, the algorithm reaches a plausible minimum, which is likely the global minimum since the function is a sum of squares. However, this sum cannot be zero, as the first n terms vanish for

$x_i = 1$ while the (n+1)-th term, cancels out only if $\sum_{i=1}^{n} x_i^2 = \frac{1}{4}$. The tables 103, 104, 117, 118, 131, 132 contain the minimum value of the function in the different cases.

Considering only the cases where the algorithm successfully converged, we observe that the number of iterations grows linearly across the three dimensions, increasing by approximately six outer iterations as n increases 105,119,133, 106,120,134.

The average time per iteration grows by an order of magnitude as n increases. However, even for 10^5, the maximum average time per iteration remains around one second, unlike other functions where it can reach up to one minute. 107,108, 121, 122,135,136

Overall, it can be observed that the negative curvature condition is almost never violated, so every time the CG method terminates, it is because it has found the solution to the system 109,110, 123, 124,137,138.

Additionally, the average number of backtracking iterations remains very low (< 0.1), meaning the first step is almost always accepted, i.e most of the times an unitary steplength $\alpha_k = 1$ is used 113,114, 127,128,141,142. Similarly, the average number of iterations in the conjugate gradient method is also low (1.3)111,112, 125,126,139,140. The combination of all these factors ensure a very low execution time.

It can be observed that for $n = 10^4, 10^5$, the algorithm using the quadratic forcing term appears to be slightly faster for both exact derivatives and finite differences, whereas for 10^3 , the superlinear approach is more advantageous.

It is not possible to analyze the convergence rate using the experimental convergence rate, as the number of iterations performed by the algorithm is too small for it to stabilize, as can be seen from the figures 15, 16, 17.

6.2.3 F16

Before analyzing the obtained results, it has to be mentioned that for testing F16 in the different dimensions we had to reduce the tolerance *tolgrad* while increasing n: $tolgrad = 5e-7$ when $n = 10^3$, $tolgrad = 1e-5$ when $n = 10^4$ and $tolgrad = 5e-4$ when $n = 10^5$. For this reason it will not be so interesting to compare the results obtained in one dimension with the ones obtained in the others. Hence, the analysis will be mainly focused on the behavior in the single dimension.

For this function, as said for the previous one, the order of convergence cannot be commented as the experimental one plotted in figures 18, 19,20 never stabilizes.

In general, the method works quite well: in the worst scenarios there are 7 successful runs out of the total 11 (tables 148,149,162,163,176,177) and they all converge to the same minimum function value, as can be seen in tables 150,151,164,165,178,179. where it can be also seen that for F16 the minimum value changes while varying the dimension: the increase of one order of magnitude of n corresponds to a decrease of the same size of the minimum value.

Looking at tables 152,153,166,167,180,181 it can be affirmed that in every dimension the convergence is reached with few iterations: around 22 with $n = 10^3$, 30 with $n = 10^4$ and less than 100 with $n = 10^5$ (there is only one exception with the second type finite differences and $h = 10^{-6}$). It can be established that with $n = 10^3$ and $n = 10^4$ exact derivatives and their approximations lead to similar numbers of iterations, while with $n = 10^5$ the average number of iterations with approximations converge to that one with the exact derivatives for smaller h values (i.e. for h that goes to zero), while for bigger ones the behavior is variable (sometimes the convergence is reached with less iterations than for the exact case other times with far more). While comparing the execution times presented in tables 154,155,168,169,182,183 it can be remarked that for $n = 10^3$ and $n = 10^4$ exact derivatives and approximated ones with classical finite differences lead to similar execution times, lower (5 times lower to be precise) than the ones obtained with the second type finite differences. When considering $n = 10^5$ (tables 182,183, 180,181) it might be showed that in general classical finite differences work better than the other version and that better results are obtained with η_k superlinear, with the only exception of $h = 10^{-6}$, as can be seen monitoring both execution times and average number of outer iterations.

While n increases, the average of the average number of inner (CG) iterations doubles: this aspect can be compared as does not depend on *tolgrad*. At this point it should also be remarked that, testing the truncated Newton method, we had to use, for definition, the CG method to solve all the linear systems whose coefficient-matrix was F16's Hessian. However, this matrix is always a diagonal one, as already showed, and so the systems would be solved faster exploiting directly vector component-wise divisions (i.e. using the Hessian inverse matrix to directly compute the solution as $p^k = -(\nabla^2 f(x^k))^{-1}\nabla f(x^k)$, where p^k i-th component is $-\frac{(\nabla f(x^k))_i}{(\nabla^2 f(x^k))_{ii}}$). This means that probably the trucated Newton method is not the most efficient among the optimization ones while dealing with F16.

At this point it can also be justified the reason why there is no test with preconditioning for F16: the Hessian is already a diagonal matrix and, using the Gauss-Seidel preconditioning illustrated above, the preconditioning matrix obtained would be the Hessian itself. This would just lead to solving more linear systems with this coefficient-matrix and, so it makes no sense. As already specified, this linear system should not be solved with CG if seeking efficiency.

Conclusion

Analyzing the general performance of the optimization algorithms on the three chosen problems, it can be observed that Nelder-Mead is a very fast algorithm for small dimensions, but it requires many more iterations compared to the Truncated Newton method. Furthermore, it is highly sensitive to the choice of the starting point, as the minimum it converges to might be a local minimum, from which it cannot escape because it is not capable of adequately exploring the search space.

For the Truncated Newton method, the use of the Conjugate Gradient (CG) method to solve the linear system within the algorithm was not always found to be useful. In fact, for problems 79 and 16, since the Hessian matrices were respectively tridiagonal and diagonal, it would have made more sense to solve the linear system directly with a direct method, rather than using an iterative method. This would have resulted in a shorter execution time by eliminating the inner iterations of the CG method.

It was possible to study a sensible convergence order of 1 only in the case of problem F79 without preconditioning, where the algorithm takes a large number of iterations, allowing the order of convergence to stabilize. In all the other cases with less iterations the order of convergence never stabilized.

7 Appendix: MATLAB codes

Functions codes

```matlab
function F1=F16(x)
%function F16 with vector operations
n=length(x);
term1=1-cos(x);
sin_prec=[0; x(1:end-1)];
sin_next=[x(2:end); 0];
term2=sin(sin_prec)-sin(sin_next);
indici=(1:n)';
F1=sum(indici.*(term1+term2));
end
```

```matlab
function JF=JF16(x,exact,fin_dif_2,h)
% Function that computes the gradient of function 27
% exact= bool. True= computes the exact version, False= computes the approximated version
     with finite differences
% fin_dif_2= bool. True if exact=false and finite differences are computed using as
    increment h*abs(x_j) for the derivative with respect to j
% h= increment for the approximated version (if exact=true put h=0)

n=length(x);

indices=(1:n)';
JF=indices.*sin(x)+2*cos(x);
JF(n)=(1-n)*cos(x(n))+n*sin(x(n));
if ~exact %approximation with finite difference (not exact)
   if fin_dif_2 % version of finite differences with abs(xj)
        JF=(sin(h*abs(x))./(h*abs(x))).*JF;
   else % classic version of finite differences
        JF=sin(h)/h*JF;
   end
end

end
```

```matlab
function HF=HF16(x,sparse,exact,fin_dif_2,h)
% Function that computes the Hessian of function 79
% sparse= bool. True= computes the sparse version
% exact= bool. True= computes the exact version, False= computes the approximated version
    with finite differences
% fin_dif_2= bool. True if exact=false and finite differences are computed using as
    increment h*abs(x_j) for the derivative with respect to j
% h= increment for the approximated version (if exact=true put h=0)

n=length(x);
indices=(1:n)';
D=indices.*cos(x)-2*sin(x);
if sparse %sparse version
     HF=spdiags(D,0,n,n);%exact version
    if ~exact %approximation with finite difference (not exact)
        if fin_dif_2 % version of finite differences with abs(xj)
           %vec_diag=((1-cos(h*abs(x)))./(x.^2))/(h^2); % no taylor expansion
           vec_diag= 1-(h^2)*(x.^2)/12 ; % with taylor expansion
           new_diag=vec_diag.*diag(HF,0);
           HF=spdiags(new_diag,0,n,n);
        else % classic version of finite differences
           HF=(1-(h^2)/12)*HF ;  % as h is small i use cos(h) taylor expansion to avoid
               numerical cancellation
        end
        % elementi non diagonali nulli anche con differenze finite
    end
else % NOT sparse version
    HF=diag(D); %exact version
    if ~exact %approximation with finite difference (not exact)
       if fin_dif_2 % version of finite differences with abs(xj)
           % vec_diag=((1-cos(h*abs(x)))./(x.^2))/(h^2); % no taylor expansion
           vec_diag= 1-(h^2)*(x.^2)/12 ; % with taylor expansion
           HF=diag(vec_diag.*diag(HF));
       else % classic version of finite differences
           HF=(1-(h^2)/12)*HF; % as h is small i use cos(h) taylor expansion to avoid
               numerical cancellation
       end
    end
end

HF(n,n)=(n-1)*sin(x(n))+n*cos(x(n));
if ~exact  %approximation with finite difference (not exact)
    HF(n,n)=(1-(h^2)/12)*HF(n,n);
end

end
```

```matlab
function F=F27(x)
%function F27 with vector operations
fk=(x-1)/sqrt(100000);
F=sum(fk.^2);
F=F+(sum(x.^2)-0.25)^2;
F=F/2;
end
```

```matlab
function JF=JF27(x,exact,fin_dif_2,h)
% Function that computes the gradient of function 27
% exact= bool. True= computes the exact version, False= computes the approximated version
    with finite differences
% h= increment for the approximated version (if exact=true put h=0)

n=length(x);
s=sum(x.^2);
JF=((2*x-2)/100000 + 4*s*x-x)/2;
if ~exact %approximation with finite difference (not exact)
   if fin_dif_2
       JF=JF+2*(h^2).*abs(x).*x;
   else
       JF=JF+2*(h^2)*x;
end
end
```

```
function F=F79(x)
%function F79 with vector operations
term1=(3-x/10).*x;
x_next=[x(2:end);0];
x_prev=[0;x(1:end-1)];
term2=1-x_prev-2*x_next;
F=sum((term1+term2).^2);
F=F/2;
end
```

```
function JF=JF79(x,exact,fin_dif_2,h)
% Function that computes the gradient of function 79
% exact= bool. True= computes the exact version, False= computes the approximated version
    with finite differences
% fin_dif_2= bool. True if exact=false and finite differences are computed using as
    increment h*abs(x_j) for the derivative with respect to j
% h= increment for the approximated version (if exact=true put h=0)

x_next=[x(2:end);0];
x_prev=[0;x(1:end-1)];

f=(3-x/10).*x+1-x_prev-2*x_next;
f_next=[f(2:end);0];
f_prev=[0;f(1:end-1)];

JF=-2*f_prev+f.*(3-x/5)-f_next;
if ~exact %approximation with finite difference (not exact)
   if fin_dif_2 % version of finite differences with abs(xj)
       JF=JF-(3-x/5).*(x.^2)*(h^2)/10;
   else % classic version of finite differences
       JF=JF-(3-x/5)*(h^2)/10;
   end
end
end
```

```
function HF=HF79(x,sparse,exact,fin_dif_2,h)
% Function that computes the Hessian of function 79
% sparse= bool. True= computes the sparse version
% exact= bool. True= computes the exact version, False= computes the approximated version
    with finite differences
% fin_dif_2= bool. True if exact=false and finite differences are computed using as
    increment h*abs(x_j) for the derivative with respect to j
% h= increment for the approximated version (if exact=true put h=0)

x_next=[x(2:end);0];
x_prev=[0;x(1:end-1)];
n= length(x);

f=(3-x/10).*x+1-x_prev-2*x_next;
diag0=5-f/5+(3-x/5).^2;
diag_1= -2*(3-x(1:end-1)/5)-(3-x_next(1:end-1)/5);
if sparse % sparse
    diag_up=[0;diag_1] ;
    diag_down=[diag_1 ; 0];
    HF=spdiags([diag_down, diag0,diag_up],[-1,0,1],n,n); %exact version
    if ~exact  %approximation with finite difference (not exact)
        if fin_dif_2 % version of finite differences with abs(xj)
            HF= HF + (h^2)*spdiags(x.^2,0,n,n)/100 ;
            vec_diag= abs(x(1:end-1))/5 + abs(x(2:end))/10;
            HF= HF + spdiags(h*[0;vec_diag],1,n,n) +spdiags(h*[vec_diag;0],-1,n,n);

        else % classic version of finite differences
            HF= HF + spdiags((h^2)*ones(n,1)/100,0,n,n);
            new_coef=3*h/10;
            HF= HF + spdiags(new_coef*ones(n,1),1,n,n) +spdiags(new_coef*ones(n,1),-1,n,n
                );
        end
    end

else % NOT sparse
    HF=diag(diag0)+diag(diag_1,1)+diag(diag_1,-1); %exact version
    if ~exact  %approximation with finite difference (not exact)
```

```

        if fin_dif_2 % version of finite differences with abs(xj)
            HF= HF + (h^2)*diag(x.^2)/100 ;
            vec_diag= abs(x(1:end-1))/5 + abs(x(2:end))/10;
            HF= HF + diag(h*vec_diag,1) +diag(h*vec_diag,-1);

        else % classic version of finite differences
            HF= HF + (h^2)*eye(n)/100 ;
            new_coef=3*h/10;
            HF= HF + diag(new_coef*ones(n-1,1),1) +diag(new_coef*ones(n-1,1),-1);
    end
end

end
```

Nelder-Mead codes

```
function [x, f_k, n_iter]=Nelder_mead(x0,f,rho,mu, gamma, sigma, tol, max_iter, Delta)
n=length(x0);
% NOTATION
%n = dimension of vectors
% S = Simplex = matrix n x n+1 --> n+1 vectors of lenght n
% f_val_S = row vector, lenght=n+1 containing the values of the function in
% the n+1 point of the simplex

% Function that performs the Nelder-Mead optimization method, for a
% given function f

% INPUTS:
% x0 = n-dimensional column vector. Initial point;
% f = function handle that describes a function R^n->R;
% rho = reflection parameter
% mu= expansion parameter
% gamma = contraction parameter
% sigma = shrinking parameter
% tol= tolerance for stopping criteria
% max_iter= maximum number of iteration permitted;

% OUTPUTS:
% xk = the sequence of the best xk ad every iteration;
% fk = the sequence of f(xk) ad every iteration value;
% n_iter = number of iteration

x=x0;
f_k=f(x0);

S = [x0, x0 + Delta * eye(n)];


%f_values in the vertices of the simplex
f_val_S=zeros(1,n+1);
for i = 1:n+1
   f_val_S(i) = f(S(:,i));
end


idx=1:n+1;
iter=0;

[f_val_S, sort_idx] = sort(f_val_S); %vector with ordered index eith respect to the
    values of the function
idx=idx(sort_idx);

while iter < max_iter && abs(f_val_S(n+1) - f_val_S(1)) > tol

    %---------REFLECTION PHASE ----------------------
    %computation of barycenter point
    x_bar=mean(S(:, idx(1:n)),2); %

    %computation and evaluation of reflection point
    x_r= x_bar + rho*(x_bar - S(:,idx(n+1)));
    f_r=f(x_r);

    if f_r < f_val_S(1)
        %-----------EXPANSION-----------
        %computation and evaluation of expansion point
        x_e=x_bar + mu*(x_r-x_bar);
        f_e=f(x_e);

        if f_e < f_r
            %hold expansion point
            S(:,idx(n+1))=x_e;
            f_val_S(n+1)=f_e;
        else
            % hold reflection point
            S(:, idx(n+1)) = x_r;
            f_val_S(n+1) = f_r;
        end

    elseif f_r < f_val_S(n)

        %hold reflexion point x_r
        S(:, idx(n+1)) = x_r;
        f_val_S(n+1) = f_r;

    else
        %-------------------CONTRACTION------------
        %calculate x_c with the best point between x_r, x_n+1
        if f_r < f_val_S(n+1)
            x_c= x_bar + gamma *(x_bar - x_r);
        else
            x_c= x_bar + gamma *(x_bar - S(:,idx(n+1)));
        end
        f_c=f(x_c);
        if f_c < f_val_S(n+1)
            %hold x_c
            S(:, idx(n+1)) = x_c;
            f_val_S(n+1) = f_c;

        else
            %----------------------SHRINKAGE------------
            for i = 2:n+1
                    S(:, idx(i)) = S(:, idx(1)) + sigma * (S(:, idx(i)) - S(:,idx(1) ));
                    f_val_S(i) = f(S(:, idx(i)));
            end

        end

    end
    [f_val_S, sort_idx] = sort(f_val_S);
    idx=idx(sort_idx);

    iter = iter + 1;

    x=[x, S(:,idx(1))];
    f_k=[f_k;f_val_S(1)];
end

n_iter=iter;

end
```

```
rng(345989);
format short;

% Rosenbrock function
f = @(x) 100*(x(2,:) - x(1,:).^2).^2 + (1 - x(1,:)).^2;

% initial points
x0 = [1.2; 1.2];
x1 = [-1.2; 1];
tol = 1e-14;  %tolerance
max_iter = 1e5; % Number of max iteration

```

```matlab
% parameters with some tuning
parametri=[
    1,   2,   0.5, 0.5, 0.1;
    1.2, 4,   0.7, 0.3, 0.1;
    1.2, 4,   0.7, 0.3, 0.5;
    1.4, 5,   0.8, 0.2, 0.1;
    1.65, 4.55, 0.95, 0.15, 0.1;
    2,   4,   0.7, 0.5, 0.5;
    ];
% Cration of a table
results = table;

%test the function with several parameteres in the two different inital
%points
for i = 1:size(parametri, 1)
    rho = parametri(i, 1);
    chi = parametri(i, 2);
    gamma = parametri(i, 3);
    sigma = parametri(i, 4);
    Delta = parametri(i, 5);

    tic;
    [x_0, f_0, n_iter_0] = Nelder_mead(x0, f, rho, chi, gamma, sigma, tol, max_iter,
        Delta);
    tempo_x0=toc;
    tic;
    [x_1, f_1, n_iter_1] = Nelder_mead(x1, f, rho, chi, gamma, sigma, tol, max_iter,
        Delta);
    tempo_x1=toc;

    % Add the result in the table
    results = [results; table(rho, chi, gamma, sigma, Delta, ...
                x_0(1,end), x_0(2,end), f_0(end), n_iter_0, tempo_x0, ...
                x_1(1,end), x_1(2,end), f_1(end), n_iter_1, tempo_x1)];
end

% columns' name
results.Properties.VariableNames = {'Rho', 'Chi', 'Gamma', 'Sigma', 'Delta', ...
    'X_0(1)', 'X_0(2)', 'F_0', 'Iter_0','Time_0' ...
    'X_1(1)', 'X_1(2)', 'F_1', 'Iter_1', 'Time_1'};


% disp table
disp('Table with the results of Nelder-Mead method with Rosenbrock function ');
disp(results);

% save the results ona cvs file
writetable(results, 'Nelder_Rosenbrock_with_Delta.xlsx', 'WriteRowNames', true);
```

```
%% FUNCTION F16 n=10
% setting parameters
format long
rng(345989);
n = 10;
tol = 1e-13;       %tollerance
max_iter = 1e06;   %max iteration
rho = 1.1;         %reflection parameter
mu = 2.5;          %expansion parameter
gamma = 0.8;       %contraction parameter
sigma = 0.9;       %shirinking parameter
delta = 1;         %Initialization of the simplex

% function
F = @(x) F16(x);


N=10; %number of starting points
x0 = ones(n, 1);  % starting point
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5); %random matrix between [-1,1]
Mat_points=Mat_points + rand_mat; %starting points
%vector for saving times
times_10=zeros(1,N+1);
%vector for saving minimum point
vec_10=zeros(1,N+1);
%vector for saving iteration
vec_iter_10=zeros(1,N+1);



for j = 1:N+1
    %applying the function F16 to the 11 strarting points
    tic;
    [xk_16_10, fk_16_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
    %saving results
    times_10(j) = toc;
    vec_10(j) = fk_16_10(end);
    vec_iter_10(j) = n_iter;

end
%creation of a table with the results
results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_10', vec_10', vec_iter_10', ...
                     'VariableNames', {'Initial␣condition', 'Time', 'FinalValue', '
                         Iterations'});
% Computation of mean of the three values saved
mean_time = mean(results_n10.Time);
mean_final_value = mean(results_n10.FinalValue);
mean_iterations = mean(results_n10.Iterations);

% Insert the mean in the tables
mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n10.Properties.VariableNames);
results_n10 = [results_n10; mean_row];

% Display the table
disp(results_n10);
% Creation an excel table
writetable(results_n10, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_10');


%% FUNCTION F16 n=25
%The same structure of n=10
n = 25;
tol = 1e-13;
max_iter = 1e06;
rho = 1.1;
mu = 1.8;
gamma = 0.8;
sigma = 0.9;
```

```matlab
delta = 0.1;

F = @(x) F16(x);

x0 = ones(n, 1);
Mat_points = repmat(x0,1,N+1) + 2*(rand([n, N+1]) - 0.5);

times_25 = zeros(1,N+1);
vec_25 = zeros(1,N+1);
vec_iter_25 = zeros(1,N+1);

for j = 1:N+1
    tic;
    [xk_16_25, fk_16_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
    times_25(j) = toc;
    vec_25(j) = fk_16_25(end);
    vec_iter_25(j) = n_iter;

end
results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_25', vec_25', vec_iter_25', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
mean_time = mean(results_n25.Time);
mean_final_value = mean(results_n25.FinalValue);
mean_iterations = mean(results_n25.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n25.Properties.VariableNames);

results_n25 = [results_n25; mean_row];

disp(results_n25);

writetable(results_n25, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_25');


%% FUNCTION F16 n=50
% The same stucture of n=10
n = 50;
tol = 1e-13;
max_iter = 1e06;
rho = 1.1;
mu = 1.8;
gamma = 0.8;
sigma = 0.9;
delta = 0.1;

F = @(x) F16(x);

x0 = ones(n, 1);
Mat_points = repmat(x0,1,N+1) + 2*(rand([n, N+1]) - 0.5);

times_50 = zeros(1,N+1);
vec_50 = zeros(1,N+1);
vec_iter_50 = zeros(1,N+1);

for j = 1:N+1
    tic;
    [xk_16_50, fk_16_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
    times_50(j) = toc;
    vec_50(j) = fk_16_50(end);
    vec_iter_50(j) = n_iter;
end

results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_50', vec_50', vec_iter_50', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
mean_time = mean(results_n50.Time);
mean_final_value = mean(results_n50.FinalValue);
```

```
mean_iterations = mean(results_n50.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n50.Properties.VariableNames);

results_n50 = [results_n50; mean_row];
disp(results_n50);
writetable(results_n50, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_50');
```

```
%% FUNZIONE F27 n=10
format long
rng(345989);
%setting parameters
n = 10;
tol = 1e-13;       %tollerance
max_iter = 1e06;   %max iteration
rho = 1.1;         %reflection parameter
mu = 2.5;          %expansion parameter
gamma = 0.8;       %contraction parameter
sigma = 0.9;       %shirinking parameter
delta = 1;         %Initialization of the simplex

%function
F = @(x) F27(x);

N = 10; %number of starting points
x0 = ones(n, 1); %startinh point
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5); %random matrix between [-1,1]
Mat_points=Mat_points + rand_mat; %starting points
times_10 = zeros(1,N+1); %vector for saving times
vec_10 = zeros(1,N+1); %vector for saving minimum point
vec_iter_10 = zeros(1,N+1); %vector for saving iteration

for j = 1:N+1
    %applying the function F16 to the 11 strarting points
    tic;
    [xk_27_10, fk_27_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
      %saving results
    times_10(j) = toc;
    vec_10(j) = fk_27_10(end);
    vec_iter_10(j) = n_iter;

end
%creation of a table with the results
results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_10', vec_10', vec_iter_10', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});

% Computation of mean of the three values saved
mean_time = mean(results_n10.Time);
mean_final_value = mean(results_n10.FinalValue);
mean_iterations = mean(results_n10.Iterations);

% Insert the mean in the tables
mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n10.Properties.VariableNames);
results_n10 = [results_n10; mean_row];

% Display the table
disp(results_n10);
% Creation an excel table
writetable(results_n10, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_10');

%% FUNZIONE F27 n=25
%The same structure of n=10
n = 25;
tol = 1e-14;
max_iter = 1e08;
rho = 1.1;
mu = 1.8;
```

```
gamma = 0.8;
sigma = 0.9;
delta = 0.1;

F = @(x) F27(x);

x0 = ones(n, 1);
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5);
Mat_points=Mat_points + rand_mat;

times_25 = zeros(1,N+1);
vec_25 = zeros(1,N+1);
vec_iter_25 = zeros(1,N+1);

for j = 1:N+1
    tic;
    [xk_27_25, fk_27_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
    times_25(j) = toc;
    vec_25(j) = fk_27_25(end);
    vec_iter_25(j) = n_iter;

end

results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_25', vec_25', vec_iter_25', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});

mean_time = mean(results_n25.Time);
mean_final_value = mean(results_n25.FinalValue);
mean_iterations = mean(results_n25.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n25.Properties.VariableNames);

results_n25 = [results_n25; mean_row];

disp(results_n25);

writetable(results_n25, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_25');

%% FUNZIONE F27 n=50
%The same structure of n=10
n = 50;
tol = 1e-13;
max_iter = 1e06;
rho = 1.1;
mu = 1.8;
gamma = 0.8;
sigma = 0.9;
delta = 0.1;

F = @(x) F27(x);

x0 = ones(n, 1);
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5);
Mat_points=Mat_points + rand_mat;

times_50 = zeros(1,N+1);
vec_50 = zeros(1,N+1);
vec_iter_50 = zeros(1,N+1);

for j = 1:N+1
    tic;
    [xk_27_50, fk_27_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter, delta);
    times_50(j) = toc;
    vec_50(j) = fk_27_50(end);
    vec_iter_50(j) = n_iter;

```

```
end

results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_50', vec_50', vec_iter_50', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});

mean_time = mean(results_n50.Time);
mean_final_value = mean(results_n50.FinalValue);
mean_iterations = mean(results_n50.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n50.Properties.VariableNames);

results_n50 = [results_n50; mean_row];

disp(results_n50);

writetable(results_n50, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_50');

disp('Tutti␣i␣risultati␣sono␣stati␣salvati␣in␣Risultati_F27.xlsx.');
```

```
%% FUNZIONE F79
% setting parameters
format long
rng(345989);
n = 10;
tol = 1e-13;       %tollerance
max_iter = 1e06;   %max iteration
rho = 1.1;         %reflection parameter
mu = 2.5;          %expansion parameter
gamma = 0.8;       %contraction parameter
sigma = 0.9;       %shirinking parameter
delta = 1;         %Initialization of the simplex

% function
F = @(x) F79(x);

N=10; %number of starting points
x0 = ones(n, 1);  % starting point
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5); % random matrix between [-1,1]
Mat_points=Mat_points + rand_mat; % starting points
%vector for saving times
times_10=zeros(1,N+1);
%vector for saving minimum point
vec_10=zeros(1,N+1);
%vector for saving iterations
vec_iter_10=zeros(1,N+1);

for j =1:N+1
    %applying the function F79 to the 11 starting points
    tic;
    [xk_79_10, fk_79_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter,delta);
    %saving results
    times_10(j)=toc;
    vec_10(j)=fk_79_10(end);
    vec_iter_10(j)=n_iter;
end

% Creation of a table with the results
results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_10', vec_10', vec_iter_10', ...
                     'VariableNames', {'Initial␣condition', 'Time', 'FinalValue', '
                         Iterations'});
% Computation of mean of the three values saved
mean_time = mean(results_n10.Time);
mean_final_value = mean(results_n10.FinalValue);
mean_iterations = mean(results_n10.Iterations);

% Insert the mean in the tables
```

```
mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n10.Properties.VariableNames);
results_n10 = [results_n10; mean_row];

% Display the table
disp(results_n10);
% Creation an excel table
writetable(results_n10, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_10');




%% FUNZIONE F79 n=25
%the same structure of n=10
format long
rng(345989);
n = 25;
tol = 1e-13;
max_iter = 1e06;
rho = 1.1;
mu = 1.9;
gamma = 0.8;
sigma = 0.9;
delta = 0.1;

F = @(x) F79(x);

N=10;
x0 = ones(n, 1);
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5);
Mat_points=Mat_points + rand_mat;
times_25=zeros(1,N+1);
vec_25=zeros(1,N+1);
vec_iter_25=zeros(1,N+1);

for j =1:N+1
    tic;
    [xk_79_25, fk_79_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter,delta);
    times_25(j)=toc;
    vec_25(j)=fk_79_25(end);
    vec_iter_25(j)=n_iter;
end

results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_25', vec_25', vec_iter_25', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});

mean_time = mean(results_n25.Time);
mean_final_value = mean(results_n25.FinalValue);
mean_iterations = mean(results_n25.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n25.Properties.VariableNames);

results_n25 = [results_n25; mean_row];

disp(results_n25);

writetable(results_n25, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_25');


%% FUNZIONE F79 n=50
% The same structure of n=10
format long
rng(345989);
n = 50;
tol = 1e-13;
max_iter = 1e06;
rho = 1.1;
mu = 1.8;
```

```matlab
gamma = 0.8;
sigma = 0.9;
delta = 0.1;

F = @(x) F79(x);

N=10;
x0 = ones(n, 1);
Mat_points=repmat(x0,1,N+1);
rand_mat=2*(rand([n, N+1]) - 0.5);
Mat_points=Mat_points + rand_mat;
times_50=zeros(1,N+1);
vec_50=zeros(1,N+1);
vec_iter_50=zeros(1,N+1);

for j =1:N+1
    tic;
    [xk_79_50, fk_79_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
         tol, max_iter,delta);
    times_50(j)=toc;
    vec_50(j)=fk_79_50(end);
    vec_iter_50(j)=n_iter;
end

results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
                     times_50', vec_50', vec_iter_50', ...
                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});

mean_time = mean(results_n50.Time);
mean_final_value = mean(results_n50.FinalValue);
mean_iterations = mean(results_n50.Iterations);

mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
                 'VariableNames', results_n50.Properties.VariableNames);

results_n50 = [results_n50; mean_row];

disp(results_n50);

writetable(results_n50, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_50');

disp('All␣the␣results␣have␣been␣saved␣in␣Risultati_F79_Nelder.xlsx.');
```

Truncated Newton codes

```matlab
function [xk, fk, gradfk_norm, k, xseq, btseq,cgiterseq,convergence_order,flag, converged
    , violations] = truncated_newton(x0, f, gradf, Hessf, kmax, tolgrad, ftol, cg_maxit,
    z0, c1, rho, btmax)
% Function that performs the truncated Newton optimization method, for a
% given function f, with backtracking. This version can use both the exact derivatives
    and the approximated version.
%
% INPUTS:
% x0 = n-dimensional column vector. Initial point;
% f = function handle that describes a function R^n->R;
% gradf = function handle that describes the gradient of f;
% Hessf= function handle that describes the Hessian of f;
% kmax = maximum number of iterations permitted;
% tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
% ftol= function handle of relative tolerance depending on the norm of the gradient (for
    coniugate gradient method)
% cg_maxit = maximum number of iterations of coniugate gradient method
% c1= factor for the Armijo condition in (0,1);
% rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
% backtracking;
% btmax= maximum number of backtracks permitted;

%
% OUTPUTS:
% xk = the last x computed by the function;
% fk = the value f(xk);
% gradfk_norm = value of the norm of gradf(xk)
% k = index of the last iteration performed
% xseq = n-by-k matrix where the columns are the elements xk of the sequence
% btseq = row vector with the number of backtracks done at every iteration
% cgiterseq=
% convergence_order = estimated order of convergence
% flag= string that says how the method has ended
% converged= bool. True if the method has converged
% violations=number of violations of positive curvature condition


% Initializations
xseq = zeros(length(x0), kmax);
cgiterseq = zeros(1, kmax);
btseq = zeros(1,kmax);
convergence_order=zeros(1,kmax);

xk = x0; % assigning the initial point
k = 0;
gradk= gradf(xk); % assigning the initial gradient of f(xk)
gradfk_norm = norm(gradk); % assigning the initial gradient norm
flag=nan;
violations=0;


while k < kmax && gradfk_norm > tolgrad

    %%% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%%
    % Hessf(xk)=A, pk=z, -gradk=b
    A=Hessf(xk); % computing Hessian (if A sparse products with dense vectors will be dense)
    % Initialization of zj and j
    zj = z0;
    j= 0;

    % Initialization of relative residual and of descent direction
    res = -gradk - A*zj; % initialize relative residual res=b-Ax
    p = res; % initialize descent direction
    norm_b = gradfk_norm; % norm(b) where b=-gradk
    norm_r = norm(res); % norm of the residual

    while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the norm of the gradient
       z = A*p; %product of A and descent direction
       a = (res'*p)/(p'*z); % update exact step along the direction
       zj = zj+ a*p; % update solution
       res = res - a*z; %update residual
       beta = -(res'*z)/(p'*z);
       p = res + beta*p; % update descent direction

       sign_curve=sign(p'*A*p);
       if sign_curve ~= 1 % negative curvature condition  p'*A*p <= 0
           violations =violations+1;
           break;
       end

       norm_r = norm(res);
       j = j+1;

    end

    pk=zj; % descent direction computed (considering the negative curvature condition)


    % Backtracking to compute the steplength
    bt=0;
    alpha=1; % initial steplenght=1
    xnew = xk + alpha * pk; % Compute the new value for x with alpha
    while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
        alpha=rho*alpha;
        xnew = xk + alpha * pk; % Compute the new value for x with alpha
        bt = bt +1;
    end

    if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))  % Break if armijo not satisfied
        flag='Procedure␣stopped␣because␣the␣Armijo␣condition␣was␣NOT␣satisfied';
        converged=false;
        break;
    else
        xk=xnew;
    end

    gradk= gradf(xk); % assigning the initial gradient of f(xk)
    gradfk_norm = norm(gradk); % assigning the initial gradient norm
    k = k + 1; % Increase the step by one

    xseq(:, k) = xk; % Store current xk in xseq
    btseq(k)=bt; % Store number of backtracking iterations
    cgiterseq(k)=j; % Store coniugate gradient iterations in pcgiterseq
    if k>3
        convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1))/norm(xseq(:, k-1)-xseq(:,
            k-2)))/log(norm(xseq(:, k-1)-xseq(:, k-2))/norm(xseq(:, k-2)-xseq(:, k-3)));
    end
end

if isnan(flag)
    if k==kmax && gradfk_norm > tolgrad
        flag='Procedure␣stopped␣because␣the␣maximum␣number␣of␣iterations␣was␣reached';
        converged=false;
    else
        flag=['Procedure␣stopped␣in␣',num2str(k), '␣steps,␣with␣gradient␣norm␣', num2str(
            gradfk_norm)];
        converged=true;
    end
end
fk = f(xk); % Compute f(xk)

xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
xseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
btseq = btseq(1:k); % "Cut" btseq to the correct size
cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
convergence_order=convergence_order(1:k); % "Cut" convergence order


end
```

```
function [xk, fk, gradfk_norm, k, xseq, btseq,cgiterseq,convergence_order,flag, converged
    , violations] = truncated_newton_precond_79(x0, f, gradf, Hessf, kmax, tolgrad, ftol,
    cg_maxit,z0, c1, rho, btmax)
% Function that performs the truncated Newton optimization method, for a
% given function f, with backtracking. This version can use both the exact derivatives
    and the approximated version.
%
% INPUTS:
% x0 = n-dimensional column vector. Initial point;
% f = function handle that describes a function R^n->R;
% gradf = function handle that describes the gradient of f;
% Hessf= function handle that describes the Hessian of f;
% kmax = maximum number of iterations permitted;
% tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
% ftol= function handle of relative tolerance depending on the norm of the gradient (for
    coniugate gradient method)
% cg_maxit = maximum number of iterations of coniugate gradient method
% c1= factor for the Armijo condition in (0,1);
% rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
% backtracking;
% btmax= maximum number of backtracks permitted;
%
% OUTPUTS:
% xk = the last x computed by the function;
% fk = the value f(xk);
% gradfk_norm = value of the norm of gradf(xk)
% k = index of the last iteration performed
% xseq = n-by-k matrix where the columns are the elements xk of the sequence
% btseq = row vector with the number of backtracks done at every iteration
% cgiterseq=
% convergence_order = estimated order of convergence
% flag= string that says how the method has ended
% converged= bool. True if the method has converged
% violations=number of violations of positive curvature condition


% Initializations
xseq = zeros(length(x0), kmax);
cgiterseq = zeros(1, kmax);
btseq = zeros(1,kmax);
convergence_order=zeros(1,kmax);

xk = x0; % assigning the initial point
k = 0;
gradk= gradf(xk); % assigning the initial gradient of f(xk)
gradfk_norm = norm(gradk); % assigning the initial gradient norm
flag=nan;

violations=0;

while k < kmax && gradfk_norm > tolgrad
%%% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%%
    % Hessf(xk)=A, pk=z, -gradk=b
    A=Hessf(xk); % computing Hessian (if A sparse products with dense vectors will be
        dense)
    % Initialization of zj and j
    zj = z0;
    j= 0;
    %Initialization of relative residuals and of decent direction
    res = A*zj+gradk ; % initialize relative residual res=b-Ax


    D = diag(diag(A));  % Diagonal Matrix (D)
    L = tril(A, -1);    % Triangula inferior (L)
    M=D+L;              % Preconditioning Matrix M

    y=M\res;
    p = -y; % initialize descent direction

    norm_b = gradfk_norm; % norm(b) where b=-gradk
    norm_r = norm(res); % norm of the residual


    while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the
        norm of the gradient
       z = A*p; %product of A and descent direction
       a = (res'*y)/(p'*z); % update exact step along the direction
       zj = zj+ a*p; % update solution
       res1 = res + a*z; %update residual

       %solve the system Myk+1=rk+1
       y1=M\res1;

       beta = (res1'*y1)/(res'*y);
       p = -y1 + beta*p; % update descent direction

       sign_curve=sign(p'*A*p);
       if sign_curve ~= 1 % negative curvature condition  p'*A*p <= 0
           violations =violations+1;
           break;
       end

       res=res1;
       y=y1;

       norm_r = norm(res);
       j = j+1;

    end

    pk=zj; % descent direction computed (considering the negative curvature condition)

```

```matlab

    % Backtracking to compute the steplength
    bt=0;
    alpha=1; % initial steplenght=1
    xnew = xk + alpha * pk; % Compute the new value for x with alpha
    while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
        alpha=rho*alpha;
        xnew = xk + alpha * pk; % Compute the new value for x with alpha
        bt = bt +1;
    end

    if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))  % Break if armijo not satisfied
        flag='Procedure␣stopped␣because␣the␣Armijo␣condition␣was␣NOT␣satisfied';
        converged=false;
        break;
    else
        xk=xnew;
    end

    gradk= gradf(xk); % assigning the initial gradient of f(xk)
    gradfk_norm = norm(gradk); % assigning the initial gradient norm
    k = k + 1; % Increase the step by one

    xseq(:, k) = xk; % Store current xk in xseq
    btseq(k)=bt; % Store number of backtracking iterations
    cgiterseq(k)=j; % Store coniugate gradient iterations in pcgiterseq
    if k>3
        convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1))/norm(xseq(:, k-1)-xseq(:,
            k-2)))/log(norm(xseq(:, k-1)-xseq(:, k-2))/norm(xseq(:, k-2)-xseq(:, k-3)));
    end
end

if isnan(flag)
    if k==kmax && gradfk_norm > tolgrad
        flag='Procedure␣stopped␣because␣the␣maximum␣number␣of␣iterations␣was␣reached';
        converged=false;
    else
        flag=['Procedure␣stopped␣in␣',num2str(k), '␣steps,␣with␣gradient␣norm␣', num2str(
            gradfk_norm)];
        converged=true;
    end
end
fk = f(xk); % Compute f(xk)

xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
xseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
btseq = btseq(1:k); % "Cut" btseq to the correct size
cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
convergence_order=convergence_order(1:k); % "Cut" convergence order


end
```

```matlab
function [xk, fk, gradfk_norm, k, xseq, btseq,cgiterseq,convergence_order,flag, converged
    , violations] = truncated_newton_27(x0, f, gradf,exact, fin_dif_2, h, kmax, tolgrad,
    ftol, cg_maxit,z0, c1, rho, btmax)
% Function that performs the truncated Newton optimization method, for for function F27,
    with backtracking.
% INPUTS:
% x0 = n-dimensional column vector. Initial point;
% f = function handle that describes a function R^n->R;
% gradf = function handle that describes the gradient of f;
% exact = bool. True if exact version of the hessian. False= approximated version with
    finite differences
% h= increment for finite differences. if exact=true put h=0.
% kmax = maximum number of iterations permitted;
% tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
% ftol= function handle of relative tolerance depending on the norm of the gradient (for
    coniugate gradient method)
% cg_maxit = maximum number of iterations of coniugate gradient method
% c1= factor for the Armijo condition in (0,1);
% rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
% backtracking;
% btmax= maximum number of backtracks permitted;
%
% OUTPUTS:
% xk = the last x computed by the function;
% fk = the value f(xk);
% gradfk_norm = value of the norm of gradf(xk)
% k = index of the last iteration performed
% xseq = n-by-k matrix where the columns are the elements xk of the sequence
% btseq = row vector with the number of backtracks done at every iteration
% cgiterseq=
% convergence_order = estimated order of convergence
% flag= string that says how the method has ended
% converged= bool. True if the method has converged
% violations=number of violations of positive curvature condition

% Initializations
xseq = zeros(length(x0), kmax);
cgiterseq = zeros(1, kmax);
btseq = zeros(1,kmax);
convergence_order=zeros(1,kmax);

xk = x0; % assigning the initial point
k = 0;
gradk= gradf(xk); % assigning the initial gradient of f(xk)
gradfk_norm = norm(gradk); % assigning the initial gradient norm
flag=nan;

violations=0;

while k < kmax && gradfk_norm > tolgrad

    %%% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%%
    % Hessf(xk)=A, pk=z, -gradk=b
    % Hessf27(x)(i,j)= 4*x_i*x_j
    % Hessf27(x)(i,i)= (2/100000 -1+ 4*(sum(x.^2)) + 8*x_i^2)/2
    % the matrix is NOT sparse BUT with large n cannot be stored. So we
    % compute directly the matrix vector products.
    % EXACT VERSION: Hessf27(x)*z= 4*s*v1 -4*v2 +v3   with s=sum(x), v1=x.*z, v2= (x.^2)
        .*z, v3=diag(Hessf27(x)).*z
    % diag(Hessf27(x))= (2/100000 -1+ 4*s + 8*x.^2)/2;
    % APPROXIMATED VERSION:  Hessf27_approx(x,h)*z= Hessf27(x)*z + 2*n*(h^2)*z

    %with finite difference 1: Hessf27(x)(i,j)= 4*x_i*x_j + h^2 +2hx_j+2hx_i
    %APPROXIMATED VERSION: Hessf27(x)(i,i)= (2/100000 -1+ 4*(sum(x.^2)) + 8*x_i^2 + 2*h
        ^2)/2
    %APPROXIMATED VERSION:  Hessf27_approx(x,h)*z= Hessf27(x)*z +(h^2)*sum_z*ones(n,1) -
        (h^2)*z + 2*h*sum_z*x + 2*h*(x'*z)-4*h*(x.*z)

    %with finite difference 2: Hessf27(x)(i,j)= 4*x_i*x_j*mod(x_i)*mod(x_j)
    %+ h^2*mod(x_i)^2*mod(x_j)^2
    %+2hx_j*mod(x_i)^2*mod(x_j)+2hx_i*mod(x_j)^2*mod(x_i)

    diagA=(2/100000 -1+ 4*sum(xk.^2) + 8*xk.^2)/2;

    % Initialization of zj and j
    zj = z0;
    j= 0;

    % Initialization of relative residual and of descent direction
    Azj= 4*(xk'*zj)*xk-4*(xk.^2).*zj+diagA.*zj; % A*zj
    sum_z=sum(zj);
    if ~exact %approximation with finite difference (not exact)
        if fin_dif_2
            Azj= (diagA.*zj +(h^2.*abs(xk).^2)) +( h^2*abs(xk).*(abs(xk)'*zj) - h^2*abs(
                xk).^2.*zj )+(2*h*xk.*(abs(xk)'*zj) - 4*h*xk.*abs(xk).*zj) +(2*h*abs(xk)
                .*(xk'*zj)) + (4*xk.*(xk'*zj)-4*(xk.^2).*zj);
        else
            Azj=Azj+ (h^2)*sum_z*ones(length(x0),1) - (h^2)*zj + 2*h*sum_z*xk + 2*h*(xk'*
                zj)-4*h*(xk.*zj);
        end
    end

    res = -gradk - Azj; % initialize relative residual res=b-Ax
    p = res; % initialize descent direction
    norm_b = gradfk_norm; % norm(b) where b=-gradk
    norm_r = norm(res); % norm of the residual

    while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the norm of the gradient
       z= 4*(xk'*p)*xk-4*(xk.^2).*p+diagA.*p; % A*p : product of A and descent direction
       sum_p=sum(p);
       if ~exact %approximation with finite difference (not exact)

           if fin_dif_2
                z= (diagA.*p +(h^2.*abs(xk).^2)) +( h^2*abs(xk).*(abs(xk)'*p) - h^2*abs(xk).^2.*p )+(2*h*xk.*(abs(xk)'*p) - 4*h*xk.*abs(xk).*p) +(2*h*abs(xk).*(xk'*p)) + (4*xk.*(xk'*p)-4*(xk.^2).*p);
           else

                z=z+ (h^2)*sum_p*ones(length(x0),1) - (h^2)*p + 2*h*sum_p*xk + 2*h*(xk'*p)-4*h*(xk.*p);

           end
       end
       a = (res'*p)/(p'*z); % update exact step along the direction
       zj = zj+ a*p; % update solution
       res = res - a*z; %update residual
       beta = -(res'*z)/(p'*z);
       p = res + beta*p; % update descent direction


       z_new=4*(xk'*p)*xk'-4*((xk.^2).*p)'+(diagA.*p)'; % p'*A (as A*p because A symmetric but as a row vector)  --> needed for curvature condition
       sum_p=sum(p);
       if ~exact %approximation with finite difference (not exact)
            if fin_dif_2
                z_new= ((diagA.*p +(h^2.*abs(xk).^2)) +( h^2*abs(xk).*(abs(xk)'*p) - h^2*abs(xk).^2.*p )+(2*h*xk.*(abs(xk)'*p) - 4*h*xk.*abs(xk).*p) +(2*h*abs(xk).*(xk'*p)) + (4*xk.*(xk'*p)-4*(xk.^2).*p))';
            else
                z_new=z_new+ ((h^2)*sum_p*ones(length(x0),1))' - ((h^2)*p+ 2*h*sum_p*xk)' + (2*h*(xk'*p)-4*h*(xk.*p))';
            end
       end
       sign_curve=sign(z_new*p);
       if sign_curve ~= 1 % negative curvature condition  p'*A*p <= 0
           violations =violations+1;
           break;
       end

       norm_r = norm(res);
       j = j+1;

    end

    pk=zj; % descent direction computed (considering the negative curvature condition)


    % Backtracking to compute the steplength
    bt=0;
    alpha=1; % initial steplenght=1
    xnew = xk + alpha * pk; % Compute the new value for x with alpha
    while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
        alpha=rho*alpha;
        xnew = xk + alpha * pk; % Compute the new value for x with alpha
        bt = bt +1;
    end

    if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))  % Break if armijo not satisfied
        flag='Procedure␣stopped␣because␣the␣Armijo␣condition␣was␣NOT␣satisfied';
        converged=false;
        break;
    else
        xk=xnew;
    end

    gradk= gradf(xk); % assigning the initial gradient of f(xk)
    gradfk_norm = norm(gradk); % assigning the initial gradient norm
    k = k + 1; % Increase the step by one

    xseq(:, k) = xk; % Store current xk in xseq
    btseq(k)=bt; % Store number of backtracking iterations
    cgiterseq(k)=j; % Store coniugate gradient iterations in pcgiterseq
    if k>3
        convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1))/norm(xseq(:, k-1)-xseq(:,
            k-2)))/log(norm(xseq(:, k-1)-xseq(:, k-2))/norm(xseq(:, k-2)-xseq(:, k-3)));
    end
end

if isnan(flag)
    if k==kmax && gradfk_norm > tolgrad
        flag='Procedure␣stopped␣because␣the␣maximum␣number␣of␣iterations␣was␣reached';
        converged=false;
    else
        flag=['Procedure␣stopped␣in␣',num2str(k), '␣steps,␣with␣gradient␣norm␣', num2str(
            gradfk_norm)];
        converged=true;
    end
end
fk = f(xk); % Compute f(xk)

xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
xseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
btseq = btseq(1:k); % "Cut" btseq to the correct size
cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
convergence_order=convergence_order(1:k); % "Cut" convergence order


end
```

```
%% ROSENBROCK FUNCTION
addpath("C:\Users\sofia\Documents\Numerical-Optimization")
rng(345989);

f_Rosen = @(x) 100*(x(2,:)-x(1,:).^2).^2+(1-x(1,:)).^2 ;
gradf_Rosen= @(x) [400*x(1,:).^3+(2-400*x(2,:)).*x(1,:)-2;
                    200*(x(2,:)-x(1,:).^2)];
Hessf_Rosen=@(x) [1200*x(1,:).^2-400*x(2,:)+2, -400*x(1,:) ;
                    -400*x(1,:), 200];
x0_a=[1.2;1.2];
x0_b=[-1.2;1];

load forcing_terms.mat

%% TEST of Trucated Newton with x0=[1.2;1.2]

kmax=500;
tolgrad=5e-7;
cg_maxit=50;

z0=zeros(2,1);
c1=1e-4;
rho=0.5;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

% Superlinear term of convergence
tic
[x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1,flag1,converged1,violations1
    ] = truncated_newton(x0_a, f_Rosen, gradf_Rosen, Hessf_Rosen, kmax, tolgrad,
    fterms_suplin, cg_maxit,z0, c1, rho, btmax);
time1=toc;
disp('Test␣on␣Rosenbrock␣function␣with␣x0=[1.2;1.2]␣and␣superlinear␣term␣for␣the␣adaptive
    ␣tolerance:␣')
disp(flag1)
disp(['Elapsed␣time:␣',num2str(time1)])
disp(['Function␣value␣in␣the␣point␣found:␣',num2str(f1)])
disp(['Number␣of␣violations␣of␣curvature␣condition:␣', num2str(violations1)])
last_bt1=sum(btseq1)/k1;
last_cg1=sum(cgiterseq1)/k1;
```

```matlab

%plot
plot_iterative_optimization_results(f_Rosen,xseq1, btseq1);
figure;
hold on
plot(1:length(conv_ord1), conv_ord1, 'Color', 'b', 'LineWidth', 1.5)
title('Rosenbrock␣truncated␣Newton␣method␣[1.2,␣1.2]␣superlinear');
hold off;

% Quadratic term of convergence
tic
[x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2,flag2,converged2,violations2] = truncated_newton(x0_a, f_Rosen, gradf_Rosen, Hessf_Rosen, kmax, tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
time2=toc;
disp('Test␣on␣Rosenbrock␣function␣with␣x0=[1.2;1.2]␣and␣quadratic␣term␣for␣the␣adaptive␣tolerance:␣')
disp(flag2)
disp(['Elapsed␣time:␣',num2str(time2)])
disp(['Function␣value␣in␣the␣point␣found:␣',num2str(f2)])
disp(['Number␣of␣violations␣of␣curvature␣condition:␣', num2str(violations2)])
last_bt2=sum(btseq2)/k2;
last_cg2=sum(cgiterseq2)/k2;

%plot
plot_iterative_optimization_results(f_Rosen,xseq2, btseq2);
figure;
hold on
plot(1:length(conv_ord2), conv_ord2, 'Color', 'b', 'LineWidth', 1.5)
title('Rosenbrock␣truncated␣Newton␣method␣[1.2,␣1.2]␣quadratic');
hold off;


%% TEST of Trucated Newton  with x0=[-1.2;1]

kmax=500;
tolgrad=5e-7;
cg_maxit=50;

z0=zeros(2,1);
c1=1e-4;
rho=0.5;
btmax=50;
% rho=0.8;
% btmax=155;

% Superlinear term of convergence
tic
[x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3,flag3,converged3,violations3] = truncated_newton(x0_b, f_Rosen, gradf_Rosen, Hessf_Rosen, kmax, tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
time3=toc;
disp('Test␣on␣Rosenbrock␣function␣with␣x0=[-1.2;1]␣and␣superlinear␣term␣for␣the␣adaptive␣tolerance:␣')
disp(flag3)
disp(['Elapsed␣time:␣',num2str(time3)])
disp(['Function␣value␣in␣the␣point␣found:␣',num2str(f3)])
disp(['Number␣of␣violations␣of␣curvature␣condition:␣', num2str(violations3)])
last_bt3=sum(btseq3)/k3;
last_cg3=sum(cgiterseq3)/k3;

%plot
plot_iterative_optimization_results(f_Rosen,xseq3, btseq3);
figure;
hold on
plot(1:length(conv_ord3), conv_ord3, 'Color', 'b', 'LineWidth', 1.5)
title('Rosenbrock␣truncated␣Newton␣method␣[-1.2,␣1]␣superlinear');
hold off;

% Quadratic term of convergence
tic
[x4, f4, gradf_norm4, k4, xseq4, btseq4,cgiterseq4,conv_ord4,flag4,converged4,violations4] = truncated_newton(x0_b, f_Rosen, gradf_Rosen, Hessf_Rosen, kmax, tolgrad,
    fterms_quad, cg_maxit,z0, c1, rho, btmax);
time4=toc;
disp('Test␣on␣Rosenbrock␣function␣with␣x0=[-1.2;1]␣and␣quadratic␣term␣for␣the␣adaptive␣tolerance:␣')
disp(flag4)
disp(['Elapsed␣time:␣',num2str(time4)])
disp(['Function␣value␣in␣the␣point␣found:␣',num2str(f4)])
disp(['Number␣of␣violations␣of␣curvature␣condition:␣', num2str(violations4)])
last_bt4=sum(btseq4)/k4;
last_cg4=sum(cgiterseq4)/k4;

%plot
figure;
hold on
plot(1:length(conv_ord4), conv_ord4, 'Color', 'b', 'LineWidth', 1.5)
title('Rosenbrock␣truncated␣Newton␣method␣[-1.2,␣1]␣quadratic');
hold off;
plot_iterative_optimization_results(f_Rosen,xseq4, btseq4);


%% Table with results

results_table = table({'[1.2;1.2]'; '[1.2;1.2]'; '[-1.2;1]'; '[-1.2;1]'}, ...
                      {'Superlineare'; 'Quadratica'; 'Superlineare'; 'Quadratica'}, ...
                      [f1; f2; f3; f4], [k1; k2; k3; k4], ...
                      [time1; time2; time3; time4], ...
                      [violations1; violations2; violations3; violations4],...
                      [last_cg1; last_cg2; last_cg3; last_cg4],...
                      [last_bt1; last_bt2; last_bt3; last_bt4],...
                      'VariableNames', {'Starting␣point','␣Forcing␣terms', 'f_x', 'Number␣of␣iterations', 'Executing␣time', 'Violation␣of␣curvature␣conditions', 'Average␣of␣cg␣iterations', 'Average␣of␣bt␣iterations'});
writetable(results_table, 'rosenbrock_truncated.xlsx','WriteRowNames', true);
```

```
%% FUNCTION 79  (with different initial points)- with exact derivatives and finite differences

sparse=true;

F = @(x) F79(x);  % Defining F79 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function handle
HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
```

```
% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;
    last_vals = conv_ord1_ex(max(end-11,1):end);
    mat_conv1_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;
    last_vals = conv_ord1_df1(max(end-11,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;
    last_vals = conv_ord1_df2(max(end-11,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};

    end
end


%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

```

```
% title and legend
title('F79␣10^3␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣superlinear');
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
```

```

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣suplin');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣suplin');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣superlinear');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
```

```
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    superlinear');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    superlinear');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^3,␣superlinear');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f79_suplin.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f79_suplin.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f79_suplin.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
writetable(T11, 'results_f79_suplin.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f79_suplin.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);


%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(12,N+1); %matrix with che last 12 values of rate of convergence for the starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
```

```
     starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;
    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;
    last_vals = conv_ord2_ex(max(end-11,1):end);
    mat_conv2_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;
    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;
    last_vals = conv_ord2_df1(max(end-11,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;
    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-11,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};


    end
end


%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79␣10^4␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    superlinear');
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
```

```
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation iteration table (only for successful runs): F79, n=10^4, 
    superlinear');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F79, n=10^4, 
    superlinear');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```matlab
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    suplinear');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    superlinear');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    superlinear');
disp(T16);

```

```
%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^4,␣superlinear');
disp(T17);
%save the table in a file xlsx
writetable(T4, 'results_f79_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f79_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f79_suplin.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
writetable(T15, 'results_f79_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f79_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);

%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(12:N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
```

```
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-11,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-11,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-11,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};


    end
end


%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79␣10^5␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
```

```
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
```

```

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣superlinear');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣superlinear');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
```

```
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^5,␣superlinear');
disp(T21);
%save the tables
writetable(T7, 'results_f79_suplin.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f79_suplin.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f79_suplin.xlsx', 'Sheet', 'viol_5','WriteRowNames', true);
writetable(T19, 'results_f79_suplin.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f79_suplin.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);

%% table with the resulta of the exact derivatives

data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval','Violation','Average␣iter␣Bt',
    'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f79_suplin.xlsx', 'Sheet', 'ExactComparison', '
    WriteRowNames', true);
```

```
%% FUNCTION 79  (with different initial points)- with exact derivatives and finite
    differences- QUADRATIC TERM OF CONVERGENCE

sparse=true;

F = @(x) F79(x);  % Defining F79 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
    handle
HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
    function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);
```

```matlab

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6, N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
```

```
    disp(['Condizione iniziale n. ',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times1_ex(j)=toc;

    disp(['Exact derivatives: ',flag1])
    vec_converged1_ex(j)=converged1;
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;
    last_vals = conv_ord1_ex(max(end-11,1):end);
    mat_conv1_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;
    last_vals = conv_ord1_df1(max(end-11,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;
    last_vals = conv_ord1_df2(max(end-11,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};


    end
end

%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;
```

```

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F79␣10^3␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣quadratic'
    );
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
```

```

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    quadratic');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    quadratic');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
```

```
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    quadratic');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    quadratic');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^3,␣
    quadratic');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
```

```matlab

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^3,␣quadratic');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f79_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f79_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f79_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f79_quad.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
writetable(T11, 'results_f79_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f79_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f79_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);


%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(12,N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
```

```
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;
    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;
    last_vals = conv_ord2_ex(max(end-11,1):end);
    mat_conv2_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;
    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;
    last_vals = conv_ord2_df1(max(end-11,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;
    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-11,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};

    end
end


%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79␣␣10^4␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;



%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
```

```
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣quadratic'
    );
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    quadratic');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    quadratic');
disp(T6);

%% VIOLATION
```

```

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79, n=10^4, 
    quadratic');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79, n=10^4, 
    quadratic');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

```

```matlab
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^4,␣
    quadratic');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^4,␣quadratic');
disp(T17);
%save the table if a file xlsx
writetable(T4, 'results_f79_quad.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f79_quad.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f79_quad.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f79_quad.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
writetable(T15, 'results_f79_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f79_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f79_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);


%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
```

```
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(12:N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-11,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-11,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-11,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};

    end
end

%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79␣10^5␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
```

```
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣quadratic'
    );
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    quadratic');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

```

```
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F79, n=10^5, 
    quadratic');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79, n=10^5, 
    quadratic');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79, n=10^5, 
    quadratic');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
```

```
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    quadratic');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^5,␣quadratic');
disp(T21);
%save the tables

writetable(T7, 'results_f79_quad.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f79_quad.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f79_quad.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f79_quad.xlsx', 'Sheet', 'viol_5','WriteRowNames', true);
writetable(T19, 'results_f79_quad.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f79_quad.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f79_quad.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% table with the result of the exact derivatives
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval','Violation','Average␣iter␣Bt',
    'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f79_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames
    ', true);
```

```matlab
%% FUNCTION 79  PRECONDITIOINING (with different initial points)- with exact derivatives
    and finite differences

sparse=true;

F = @(x) F79(x);  % Defining F79 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
    handle
HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
    function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-3;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for the
     starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv_fd1=cell(6, N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
```

```
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;

    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;

    last_vals = conv_ord1_ex(max(end-14,1):end);
    mat_conv_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    %
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;


    last_vals = conv_ord1_df1(max(end-14,1):end);
    mat_conv_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;

    last_vals = conv_ord1_df2(max(end-14,1):end);
    mat_conv_fd2(i/2, j) = {last_vals};


    end
end


%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F79P 10^3 superlinear');
xlabel('Iterazione');
ylabel('Ordine di Convergenza');
legend({'Exact Derivatives', 'dif fin_1', 'dif fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
```

```
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣
    superlinear');
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣
    suplin');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

```

```
T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F79P, n=10^3, 
    suplin');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79P, n=10^3, 
    superlinear');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79P, n=10^3, 
    superlinear');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

```

```
mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣
    superlinear');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79P,␣n=10^3,␣superlinear');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f79P_suplin.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f79P_suplin.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
writetable(T11, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);



%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
```

```
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;

    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;

    last_vals = conv_ord2_ex(max(end-14,1):end);
    mat_conv2_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;

    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;


    last_vals = conv_ord2_df1(max(end-14,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;

    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;

    last_vals = conv_ord2_df2(max(end-14,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};


    end
end

%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1 , '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2 , '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79P␣10^4␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    superlinear');
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
```

```matlab
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    superlinear');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    superlinear');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    suplinear');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
```

```

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79P, n=10^4, 
    superlinear');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F79P, n=10^4, 
    superlinear');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number of converged : F79P, n=10^4, superlinear');
disp(T17);
%save the table in a file xlsx
writetable(T4, 'results_f79P_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f79P_suplin.xlsx', 'Sheet', 'v_4','WriteRowNames', true);
writetable(T15, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);



%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(15:N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
   condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
   condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
   x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
```

```
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;

    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;

    last_vals = conv_ord3_ex(max(end-14,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;


    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;

    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;


    last_vals = conv_ord3_df1(max(end-14,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag3])
    mat_converged3_fd2(i/2,j)=converged2;

    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;

    last_vals = conv_ord3_df2(max(end-14,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};


    end
end

%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79P 10^5 superlinear');
xlabel('Iterazione');
ylabel('Ordine di Convergenza');
legend({'Exact Derivatives', 'dif fin_1', 'dif fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation times table (only for successful runs): F79P, n=10^5, 
    superlinear');
```

```
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation iteration table (only for successful runs): F79P, n=10^5, 
    superlinear');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F79P, n=10^5, 
    superlinear');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
```

```
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79P, n=10^5, 
    superlinear');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79P, n=10^5, 
    superlinear');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

```

```
T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F79P, n=10^5, superlinear');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number of converged : F79P, n=10^5, superlinear');
disp(T21);

writetable(T7, 'results_f79P_suplin.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f79P_suplin.xlsx', 'Sheet', 'v_5','WriteRowNames', true);
writetable(T19, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% Creation of the table with the result of exact derivatives
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt', 'Average iter cg', 'N converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f79P_suplin.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames', true);
```

```
%% FUNCTION 79 PRECONDITIONING QUAD (with different initial points)- with exact derivatives and finite differences

sparse=true;

F = @(x) F79(x);  % Defining F79 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function handle
HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm
```

```

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-3;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for the
     starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv_fd1=cell(6, N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;

    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;

    last_vals = conv_ord1_ex(max(end-14,1):end);
    mat_conv_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;

    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;


    last_vals = conv_ord1_df1(max(end-14,1):end);
    mat_conv_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;


    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;

    last_vals = conv_ord1_df2(max(end-14,1):end);
    mat_conv_fd2(i/2, j) = {last_vals};


    end
end
```

```matlab


%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F79P␣10^3␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣quadratic');
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
```

```matlab
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣
    quadratic');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^3,␣
    quadratic');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
```

```
%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79P, n=10^3, 
    quadratic');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F79P, n=10^3, 
    quadratic');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F79P, n=10^3, 
    quadratic');
```

```
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79P,␣n=10^3,␣quadratic');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f79P_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f79P_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f79P_quad.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
writetable(T11, 'results_f79P_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f79P_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);





%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

```

```
% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;

    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;

    last_vals = conv_ord2_ex(max(end-14,1):end);
    mat_conv2_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;

    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;


    last_vals = conv_ord2_df1(max(end-14,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;

    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;

    last_vals = conv_ord2_df2(max(end-14,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};


    end
end


%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79P␣␣10^4␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;



%% Execution time

% Exact derivative
```

```
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣quadratic
    ');
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    quadratic');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

```

```
mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    quadratic');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    quadratic');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];
```

```

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    quadratic');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^4,␣
    quadratic');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79P,␣n=10^4,␣quadratic');
disp(T17);
%save the table if a file xlsx
writetable(T4, 'results_f79P_quad.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f79P_quad.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f79P_quad.xlsx', 'Sheet', 'v_4','WriteRowNames', true);
writetable(T15, 'results_f79P_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f79P_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);




%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm
```

```
cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=-1*ones(n,1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(15:N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF_ex, HF_ex, kmax
        , tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;

    last_vals = conv_ord3_ex(max(end-14,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;


    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;


    last_vals = conv_ord3_df1(max(end-14,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton_precond_79(Mat_points(:,j), F, JF, HF, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged2;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;

    last_vals = conv_ord3_df2(max(end-14,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};


    end
end

```

```
%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79P␣10^5␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^5,␣quadratic
    ');
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation iteration table (only for successful runs): F79P, n=10^5, quadratic');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F79P, n=10^5, quadratic');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F79P, n=10^5, quadratic');
disp(T18);

```

```
%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^5,␣
    quadratic');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79P,␣n=10^5,␣
    quadratic');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
```

```
disp('Number␣of␣converged␣:␣F79P,␣n=10^5,␣quadratic');
disp(T21);
%save the tables
writetable(T7, 'results_f79P_quad.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f79P_quad.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f79P_quad.xlsx', 'Sheet', 'v_5','WriteRowNames', true);
writetable(T19, 'results_f79P_quad.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f79P_quad.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% table with the result of the exact derivatives
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval', 'Violation', 'Average␣iter␣Bt
    ', 'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f79P_quad.xlsx', 'Sheet', 'ExactComparison', '
    WriteRowNames', true);
```

```
%% FUNCTION 27  (with different initial points)- with exact derivatives and finite
    differences

F = @(x) F27(x);  % Defining F27 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF27(x,exact,fin_dif2,h); % Defining JF27 as function
    handle

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
```

```
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(15,N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true,false,0, kmax,
         tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;

    last_vals = conv_ord1_ex(max(end-14,1):end);
    mat_conv1_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;

    last_vals = conv_ord1_df1(max(end-14,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;

    last_vals = conv_ord1_df2(max(end-14,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};



    end
end

%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F27␣10^3␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;
```

```

%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣
    superlinear');
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣suplin
    ');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
```

```
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣
    suplin');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣
    superlinear');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
```

```

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣superlinear');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣superlinear');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F27,␣n=10^3,␣superlinear');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f27_suplin.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f27_suplin.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f27_suplin.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
writetable(T11, 'results_f27_suplin.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f27_suplin.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);



%% n=10^4 (1e4)

```

```
rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(15,N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);


% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true,false,0, kmax,
         tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;

    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;

    last_vals = conv_ord2_ex(max(end-14,1):end);
    mat_conv2_ex(:, j) = last_vals;



    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;

    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;

    last_vals = conv_ord2_df1(max(end-14,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};

    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);

    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;

    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-14,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};
    end
end


%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F27␣10^4␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣
    superlinear');
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
```

```

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣
    superlinear');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣
    superlinear');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
```

```matlab
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F27, n=10^4, 
    suplinear');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F27, n=10^4, 
    superlinear');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F27, n=10^4, 
    superlinear');
disp(T16);

%% Number of initial point converged
```

```
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F27,␣n=10^4,␣superlinear');
disp(T17);
%save the table in a file xlsx
writetable(T4, 'results_f27_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f27_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f27_suplin.xlsx', 'Sheet', 'v_4','WriteRowNames', true);
writetable(T15, 'results_f27_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f27_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);



%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(15:N+1);%matrix with che last 15 values of rate of convergence for the
     starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
```

```
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);


for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true, false,0, kmax
        , tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);

    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-14,1):end);
    mat_conv3_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;

    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-14,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);

    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
        tolgrad, fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-14,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};

    end
end


%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F27␣10^5␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';
```

```

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation times table (only for successful runs): F27, n=10^5,
    superlinear');
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation iteration table (only for successful runs): F27, n=10^5,
    superlinear');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F27, n=10^5,
    superlinear');
disp(T9);

%% VIOLATION

```

```
vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F27, n=10^5, 
    superlinear');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F27, n=10^5, 
    superlinear');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
```

```
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^5,␣superlinear');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F27,␣n=10^5,␣superlinear');
disp(T21);
%save the tables
writetable(T7, 'results_f27_suplin.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f27_suplin.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f27_suplin.xlsx', 'Sheet', 'v_5','WriteRowNames', true);
writetable(T19, 'results_f27_suplin.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f27_suplin.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% table with the resulta of the exact derivatives

data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval', 'Violation', 'Average␣iter␣Bt', 'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};


T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f27_suplin.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames', true);
```

```
%% FUNCTION 27  QUADRATIC (with different initial points)- with exact derivatives and finite differences


F = @(x) F27(x);  % Defining F27 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF27(x,exact,fin_dif2,h); % Defining JF27 as function handle

load forcing_terms.mat % possible terms for adaptive tolerance

```

```
%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(15,N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);

for j =1:N+1
```

```
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true,false,0, kmax,
         tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;
    %conv_ord1(end-10:end) %aggiustare
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;

    last_vals = conv_ord1_ex(max(end-14,1):end);
    mat_conv1_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;
    last_vals = conv_ord1_df1(max(end-14,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    tic;

    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;
    last_vals = conv_ord1_df2(max(end-14,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};

    end
end
```

```
%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F27␣10^3␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^3,␣quadratic'
    );
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
```

```
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation iteration table (only for successful runs): F27, n=10^3, 
    quadratic');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F27, n=10^3, 
    quadratic');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```
%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F27, n=10^3, 
    quadratic');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F27, n=10^3, 
    quadratic');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F27, n=10^3, 
    quadratic');
disp(T12);
```

```
%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F27,␣n=10^3,␣quadratic');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f27_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f27_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f27_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f27_quad.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
writetable(T11, 'results_f27_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f27_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f27_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);



%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(15,N+1);%matrix with che last 15 values of rate of convergence for the
     starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
```

```
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);


% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2, violations2] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true,false,0, kmax, tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;
    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;

    last_vals = conv_ord2_ex(max(end-14,1):end);
    mat_conv2_ex(:, j) = last_vals;



    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2, violations2] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax, tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;
    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;

    last_vals = conv_ord2_df1(max(end-14,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};

    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);

    tic;

    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2, violations2] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax, tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;
    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-14,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};

    end
end

%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F27␣␣10^4␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;



%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
```

```
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣quadratic'
    );
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣
    quadratic');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
```

```
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣quadratic');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣quadratic');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣quadratic');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
```

```matlab
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^4,␣
    quadratic');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F27,␣n=10^4,␣quadratic');
disp(T17);
%save the table if a file xlsx
writetable(T4, 'results_f27_quad.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f27_quad.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f27_quad.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f27_quad.xlsx', 'Sheet', 'v_4','WriteRowNames', true);
writetable(T15, 'results_f27_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f27_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f27_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);



%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0=(1:n)';  % Initial point
N=10; % number of initial points to be generated
```

```

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(15:N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);


% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
     starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);


% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);


for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true, false,0, kmax
        , tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);

    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;

    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-14,1):end);
    mat_conv3_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);

    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF, false,false,h, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;


    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-14,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);

    tic;

    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
        tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-14,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};

    end
end

%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F27␣10^5␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^5,␣quadratic'
    );
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^5,␣
    quadratic');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
```

```
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^5,␣quadratic');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F27,␣n=10^5,␣quadratic');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
```

```matlab

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F27, n=10^5, 
    quadratic');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation cg iteration table (only for successful runs): F27, n=10^5, 
    quadratic');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number of converged : F27, n=10^5, quadratic');
disp(T21);
%save the tables

writetable(T7, 'results_f27_quad.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f27_quad.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f27_quad.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f27_quad.xlsx', 'Sheet', 'v_5','WriteRowNames', true);
writetable(T19, 'results_f27_quad.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f27_quad.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f27_quad.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% table with the result of the exact derivatives
```

```
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt
    ', 'Average iter cg', 'N converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};


T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f27_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames
    ', true);
```

```
%% FUNCTION 16  (with different initial points)- with exact derivatives and finite
    differences - QUADRATIC TERM OF CONVERGENCE

sparse=true;

F = @(x) F16(x);  % Defining F16 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF16(x,exact,fin_dif2,h); % Defining JF16 as function
    handle
HF_gen= @(x,exact,fin_dif2,h) HF16(x,sparse,exact,fin_dif2,h); % Defining HF16 as
    function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1); % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);
```

```matlab

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6, N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione iniziale n. ',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times1_ex(j)=toc;

    disp(['Exact derivatives: ',flag1])
    vec_converged1_ex(j)=converged1;
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;
    last_vals = conv_ord1_ex(max(end-11,1):end);
    mat_conv1_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;
    last_vals = conv_ord1_df1(max(end-11,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;
    last_vals = conv_ord1_df2(max(end-11,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};

    end
end


%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F16␣10^3␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

```

```
% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣quadratic'
    );
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    quadratic');
disp(T2);

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```matlab
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation fmin value table (only for successful runs): F16, n=10^3, 
    quadratic');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation violation  table (only for successful runs): F16, n=10^3, 
    quadratic');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average computation bt iteration table (only for successful runs): F16, n=10^3, 
    quadratic');
disp(T11);
```

```
%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    quadratic');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F16,␣n=10^3,␣quadratic');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f16_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f16_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f16_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f16_quad.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
writetable(T11, 'results_f16_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f16_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f16_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);



%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=1e-5; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
```

```
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(12,N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;
    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;
    last_vals = conv_ord2_ex(max(end-11,1):end);
    mat_conv2_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;
    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;
    last_vals = conv_ord2_df1(max(end-11,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};



    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;
    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-11,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};


    end
end

%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
```

```matlab
        hold on;
    end
end

title('F16␣␣10^4␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;



%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣quadratic'
    );
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

```

```matlab
T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    quadratic');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    quadratic');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    quadratic');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
```

```
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    quadratic');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    quadratic');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F16,␣n=10^4,␣quadratic');
disp(T17);
%save the table if a file xlsx
writetable(T4, 'results_f16_quad.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f16_quad.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f16_quad.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f16_quad.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
writetable(T15, 'results_f16_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f16_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f16_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);



%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-4; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(12:N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
```

```
mat_conv3_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-11,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-11,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_quad, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-11,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};

    end
end
%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F16␣10^5␣quadratic');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣quadratic'
    );
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

```

```
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣quadratic');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣quadratic');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣
    quadratic');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣
    quadratic');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];

T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^5,␣
    quadratic');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
```

```
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F16,␣n=10^5,␣quadratic');
disp(T21);
%save the tables

writetable(T7, 'results_f16_quad.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f16_quad.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f16_quad.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f16_quad.xlsx', 'Sheet', 'viol_5','WriteRowNames', true);
writetable(T19, 'results_f16_quad.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f16_quad.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f16_quad.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);


%% table with the result of the exact derivatives
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval','Violation','Average␣iter␣Bt',
    'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f16_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames
    ', true);
```

```
%% FUNCTION 16  (with different initial points)- with exact derivatives and finite
    differences

sparse=true;

F = @(x) F16(x);  % Defining F16 as function handle
JF_gen = @(x,exact,fin_dif2,h) JF16(x,exact,fin_dif2,h); % Defining JF16 as function
    handle
HF_gen= @(x,exact,fin_dif2,h) HF16(x,sparse,exact,fin_dif2,h); % Defining HF16 as
    function handle (sparse version)

load forcing_terms.mat % possible terms for adaptive tolerance

%% n=10^3 (1e3)

rng(345989);

n=1e3;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-7; % tolerance on gradient norm

cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
    system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1); % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
```

```
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times1_ex=zeros(1,N+1); % vector with execution times
vec_val1_ex=zeros(1,N+1); %vector with minimal values found
vec_grad1_ex=zeros(1,N+1); %vector with final gradient
vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv1_ex=zeros(12, N+1);  %matrix with che last 12 values of rate of convergence for
    the starting point
vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times1_fd1=zeros(6,N+1); % matrix with execution times
mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd1=cell(6, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times1_fd2=zeros(6,N+1); % matrix with execution times
mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv1_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
     starting point
mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_ex,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times1_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag1])
    vec_converged1_ex(j)=converged1;
    vec_val1_ex(j)=f1;
    vec_grad1_ex(j)=gradf_norm1;
    vec_iter1_ex(j)=k1;
    vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
    vec_bt1_ex(j)=sum(btseq1)/k1;
    vec_violations1_ex(j)=violations1;
    last_vals = conv_ord1_ex(max(end-11,1):end);
    mat_conv1_ex(:, j) = last_vals;


    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df1,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd1(i/2,j)=converged1;
    mat_val1_fd1(i/2,j)=f1;
    mat_grad1_fd1(i/2,j)=gradf_norm1;
    mat_iter1_fd1(i/2,j)=k1;
    mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd1(i/2,j)=violations1;
    last_vals = conv_ord1_df1(max(end-11,1):end);
    mat_conv1_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x1, f1, gradf_norm1, k1, xseq1, btseq1,cgiterseq1,conv_ord1_df2,flag1, converged1,
        violations1] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times1_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag1])
    mat_converged1_fd2(i/2,j)=converged1;
    mat_val1_fd2(i/2,j)=f1;
    mat_grad1_fd2(i/2,j)=gradf_norm1;
    mat_iter1_fd2(i/2,j)=k1;
    mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
    mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
    mat_violations1_fd2(i/2,j)=violations1;
    last_vals = conv_ord1_df2(max(end-11,1):end);
    mat_conv1_fd2(i/2, j) = {last_vals};

    end
end



%% Plot of the last 12 values of experimentale rate of convergence
num_initial_points = N + 1;
figure;
hold on;

% Plot for every initial condition
for j = 1:num_initial_points
    conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
        conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

% title and legend
title('F16␣10^3␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
```

```
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution Time

% Exact Derivative
vec_times_ex_clean = vec_times1_ex; %a copy of the vector
vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean

% FD1
mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean

% FD2
mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table costruction with exact for both the row
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

% visualization
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    superlinear');
disp(T1);


%% All the tables has the same structure
%% Iteration

vec_times_ex_clean = vec_iter1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];

T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣suplin
    ');
disp(T2);
```

```

%% F value

vec_times_ex_clean = vec_val1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];

T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    suplin');
disp(T3);

%% VIOLATION

vec_times_ex_clean = vec_violations1_ex;
vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations1_fd1;
mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations1_fd2;
mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

%
fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];

T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    superlinear');
disp(T10);


%% BT-SEQ
vec_bt_ex_clean = vec_bt1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt1_fd2;
```

```matlab
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];

T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    superlinear');
disp(T11);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter1_ex;
vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter1_fd1;
mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter1_fd2;
mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];

T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^3,␣
    superlinear');
disp(T12);

%% Number of starting point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged1_fd1,2)';
fd2_vals = sum(mat_converged1_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];

T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F16,␣n=10^3,␣superlinear');
disp(T13);
%save the table in a file xlsx
writetable(T1, 'results_f16_suplin.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
writetable(T2, 'results_f16_suplin.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
writetable(T3, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
writetable(T10, 'results_f16_suplin.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
writetable(T11, 'results_f16_suplin.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
writetable(T12, 'results_f16_suplin.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
writetable(T13, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);
```

```
%% n=10^4 (1e4)

rng(345989);

n=1e4;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=1e-5; % tolerance on gradient norm %%%%%%%%%%%%%%%%%%% decide if we want to keep
     the tolerance 5e-7

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times2_ex=zeros(1,N+1); % vector with execution times
vec_val2_ex=zeros(1,N+1); %vector with minimal values found
vec_grad2_ex=zeros(1,N+1); %vector with final gradient
vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv2_ex=zeros(12,N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times2_fd1=zeros(6,N+1); % matrix with execution times
mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times2_fd2=zeros(6,N+1); % matrix with execution times
mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv2_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
```

```
mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_ex,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times2_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag2])
    vec_converged2_ex(j)=converged2;
    vec_val2_ex(j)=f2;
    vec_grad2_ex(j)=gradf_norm2;
    vec_iter2_ex(j)=k2;
    vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
    vec_bt2_ex(j)=sum(btseq2)/k2;
    vec_violations2_ex(j)=violations2;
    last_vals = conv_ord2_ex(max(end-11,1):end);
    mat_conv2_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd1(i/2,j)=converged2;
    mat_val2_fd1(i/2,j)=f2;
    mat_grad2_fd1(i/2,j)=gradf_norm2;
    mat_iter2_fd1(i/2,j)=k2;
    mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd1(i/2,j)=violations2;
    last_vals = conv_ord2_df1(max(end-11,1):end);
    mat_conv2_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
        violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times2_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag2])
    mat_converged2_fd2(i/2,j)=converged2;
    mat_val2_fd2(i/2,j)=f2;
    mat_grad2_fd2(i/2,j)=gradf_norm2;
    mat_iter2_fd2(i/2,j)=k2;
    mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
    mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
    mat_violations2_fd2(i/2,j)=violations2;
    last_vals = conv_ord2_df2(max(end-11,1):end);
    mat_conv2_fd2(i/2, j) = {last_vals};

    end
end
```

```

%% The Plot has the same structure
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv2_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv2_fd1{i, j};
        conv_ord_fd2 = mat_conv2_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F16␣10^4␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;


%% Execution time

% Exact derivative
vec_times_ex_clean = vec_times2_ex; %a copy of the vector
vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan');  % computation of the mean

% FD1
mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean

% FD2
mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean

% Creation of the labels
h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

% Table creation
rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
%display the table
disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    superlinear');
disp(T4);

%% Iteration

vec_times_ex_clean = vec_iter2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
```

```

mat_times_fd2_clean = mat_iter2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];

T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    superlinear');
disp(T5);

%% Function value

vec_times_ex_clean = vec_val2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];

T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    superlinear');
disp(T6);

%% VIOLATION

vec_times_ex_clean = vec_violations2_ex;
vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations2_fd1;
mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_violations2_fd2;
mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
```

```matlab
data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];

T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    suplinear');
disp(T14);

%% BT-SEQ

vec_bt_ex_clean = vec_bt2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];

T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    superlinear');
disp(T15);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter2_ex;
vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter2_fd1;
mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter2_fd2;
mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];

T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F16,␣n=10^4,␣
    superlinear');
disp(T16);

%% Number of initial point converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```
fd1_vals = sum(mat_converged2_fd1,2)';
fd2_vals = sum(mat_converged2_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];

T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F16,␣n=10^4,␣superlinear');
disp(T17);
%save the table in a file xlsx
writetable(T4, 'results_f16_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
writetable(T5, 'results_f16_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
writetable(T6, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
writetable(T14, 'results_f16_suplin.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
writetable(T15, 'results_f16_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
writetable(T16, 'results_f16_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
writetable(T17, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);


%% n=10^5 (1e5)

rng(345989);

n=1e5;

kmax=1.5e3; % maximum number of iterations of Newton method
tolgrad=5e-4; % tolerance on gradient norm

cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
     system)
z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)

% Backtracking parameters
c1=1e-4;
rho=0.50;
btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)

x0 = ones(n, 1);  % initial point
N=10; % number of initial points to be generated

% Initial points:
Mat_points=repmat(x0,1,N+1);
rand_mat=2*rand(n, N)-1;
Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

% Structure for EXACT derivatives
vec_times3_ex=zeros(1,N+1); % vector with execution times
vec_val3_ex=zeros(1,N+1); %vector with minimal values found
vec_grad3_ex=zeros(1,N+1); %vector with final gradient
vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
mat_conv3_ex=zeros(12:N+1);%matrix with che last 12 values of rate of convergence for the
     starting point
vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method

JF_ex = @(x) JF_gen(x,true,false,0);
HF_ex = @(x) HF_gen(x,true,false,0);

% Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
mat_times3_fd1=zeros(6,N+1); % matrix with execution times
mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
```

```
mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd1 = @(x,h) JF_gen(x,false,false,h);
HF_fd1 = @(x,h) HF_gen(x,false,false,h);

% Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
mat_times3_fd2=zeros(6,N+1); % matrix with execution times
mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
mat_conv3_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method

JF_fd2 = @(x,h) JF_gen(x,false,true,h);
HF_fd2 = @(x,h) HF_gen(x,false,true,h);

for j =1:N+1
    disp(['Condizione␣iniziale␣n.␣',num2str(j)])

    % EXACT DERIVATIVES
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_ex,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF_ex, HF_ex, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    vec_times3_ex(j)=toc;

    disp(['Exact␣derivatives:␣',flag3])
    vec_converged3_ex(j)=converged3;
    vec_val3_ex(j)=f3;
    vec_grad3_ex(j)=gradf_norm3;
    vec_iter3_ex(j)=k3;
    vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
    vec_bt3_ex(j)=sum(btseq3)/k3;
    vec_violations3_ex(j)=violations3;
    last_vals = conv_ord3_ex(max(end-11,1):end);
    mat_conv3_ex(:, j) = last_vals;

    for i=2:2:12
    h=10^(-i);

    % FINITE DIFFERENCES 1
    JF=@(x)JF_fd1(x,h);
    HF=@(x)HF_fd1(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df1,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd1(i/2,j)=toc;

    disp(['Finite␣differences␣(classical␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd1(i/2,j)=converged3;
    mat_val3_fd1(i/2,j)=f3;
    mat_grad3_fd1(i/2,j)=gradf_norm3;
    mat_iter3_fd1(i/2,j)=k3;
    mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd1(i/2,j)=violations3;
    last_vals = conv_ord3_df1(max(end-11,1):end);
    mat_conv3_fd1(i/2, j) = {last_vals};


    % FINITE DIFFERENCES 2
    JF=@(x) JF_fd2(x,h);
    HF=@(x) HF_fd2(x,h);
    tic;
    [x3, f3, gradf_norm3, k3, xseq3, btseq3,cgiterseq3,conv_ord3_df2,flag3, converged3,
        violations3] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
        fterms_suplin, cg_maxit,z0, c1, rho, btmax);
    mat_times3_fd2(i/2,j)=toc;

    disp(['Finite␣differences␣(new␣version)␣with␣h=1e-',num2str(i),'␣:␣',flag3])
    mat_converged3_fd2(i/2,j)=converged3;
    mat_val3_fd2(i/2,j)=f3;
    mat_grad3_fd2(i/2,j)=gradf_norm3;
    mat_iter3_fd2(i/2,j)=k3;
    mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
    mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
    mat_violations3_fd2(i/2,j)=violations3;
    last_vals = conv_ord3_df2(max(end-11,1):end);
    mat_conv3_fd2(i/2, j) = {last_vals};


    end
end


%% The plot has the same structure as n=10^3
num_initial_points = N + 1;
figure;
hold on;

for j = 1:num_initial_points
    conv_ord_ex = mat_conv3_ex(:,j);
    plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
    hold on;
    for i =1:6
        conv_ord_fd1 = mat_conv3_fd1{i, j};
        conv_ord_fd2 = mat_conv3_fd2{i, j};
        plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
        hold on;
        plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
        hold on;
    end
end

title('F79␣10^5␣superlinear');
xlabel('Iterazione');
ylabel('Ordine␣di␣Convergenza');
legend({'Exact␣Derivatives', 'dif␣fin_1', 'dif␣fin_2'}, 'Location', 'Best');
grid on;
hold off;

%% Time

vec_times_ex_clean = vec_times3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_times3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_times3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];

T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣times␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T7);

%% Iteration

vec_times_ex_clean = vec_iter3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_iter3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_iter3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];

T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T8);

%% function value

vec_times_ex_clean = vec_val3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_val3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

mat_times_fd2_clean = mat_val3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];

T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣fmin␣value␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T9);

%% VIOLATION

vec_times_ex_clean = vec_violations3_ex;
vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

mat_times_fd1_clean = mat_violations3_fd1;
mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

```

```
mat_times_fd2_clean = mat_violations3_fd2;
mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];

T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣violation␣␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T18);

%% BT-SEQ

vec_bt_ex_clean = vec_bt3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_bt3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_bt3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];

T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣bt␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣
    superlinear');
disp(T19);

%% CG-SEQ

vec_bt_ex_clean = vec_cg_iter3_ex;
vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');

mat_bt_fd1_clean = mat_cg_iter3_fd1;
mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');

mat_bt_fd2_clean = mat_cg_iter3_fd2;
mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = avg_fd1';
fd2_vals = avg_fd2';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];
```

```
T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Average␣computation␣cg␣iteration␣table␣(only␣for␣successful␣runs):␣F79,␣n=10^5,␣superlinear');
disp(T20);

%% Number of initial condition converged

h_exponents = [2, 4, 6, 8, 10, 12];
h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

fd1_vals = sum(mat_converged3_fd1,2)';
fd2_vals = sum(mat_converged3_fd2,2)';

rowNames = {'FD1', 'FD2'};
columnNames = [ h_labels,'Exact'];
data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];

T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

disp('Number␣of␣converged␣:␣F79,␣n=10^5,␣superlinear');
disp(T21);
%save the tables
writetable(T7, 'results_f16_suplin.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
writetable(T8, 'results_f16_suplin.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
writetable(T9, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
writetable(T18, 'results_f16_suplin.xlsx', 'Sheet', 'viol_5','WriteRowNames', true);
writetable(T19, 'results_f16_suplin.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
writetable(T20, 'results_f16_suplin.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
writetable(T21, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);



%% table with the resulta of the exact derivatives
data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
        avg_exact_i1, avg_exact_i2, avg_exact_i3;
        avg_exact_f1, avg_exact_f2, avg_exact_f3;
        avg_exact_v1, avg_exact_v2, avg_exact_v3;
        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
        sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];

rowNames = {'Average␣Time', 'Average␣Iter', 'Average␣fval','Violation','Average␣iter␣Bt', 'Average␣iter␣cg', 'N␣converged'};
columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};

T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
disp(T_compare)

writetable(T_compare, 'results_f16_suplin.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames', true);
```