

Numerical Optimization for Large Scale Problems Assignment

Sofia Bernardoni 347900

Gaia Giulio 345989

February 2025

Index

1 Introduction: Assignment on Unconstrained Optimization	2
2 Methods	2
2.1 Nelder Mead Method	2
2.2 Truncated Newton Method	4
3 Test on Rosenbrock function	5
3.1 Nelder Mead method	5
3.2 Truncated Newton method	6
4 Chosen problems	9
5 Tables with results	10
5.1 FUNCTION 79	11
5.1.1 Nelder-Mead method	13
5.1.2 Truncated Newton method	13
5.2 FUNCTION 27	27
5.2.1 Nelder-Mead method	28
5.2.2 Truncated Newton method	29
5.3 FUNCTION 16 - BANDED TRIGONOMETRIC PROBLEM	36
5.3.1 Nelder-Mead method	37
5.3.2 Truncated Newton Method	38
6 Comments	45
6.1 Nelder-Mead method	45
6.1.1 F79	45
6.1.2 F27	45
6.1.3 F16	46
6.2 Truncated Newton method	46
6.2.1 F79	46
6.2.2 F27	47
6.2.3 F16	48
7 Appendix: MATLAB codes	49

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 $x(0) \in [\bar{x}_1 - 1, \bar{x}_1 + 1] \times \dots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$

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

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

where $\bar{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ 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

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

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

¹https://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 :

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

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:

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

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

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

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 be found as a stationary point for $m_k(p)$; then it satisfies

$$\nabla m_k(p) = 0 \implies \underbrace{\nabla^2 f(x^k)}_A \underbrace{p^k}_z = \underbrace{-\nabla f(x^k)}_b \quad (1)$$

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

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

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	<u>0</u>	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 Rosenbrock 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

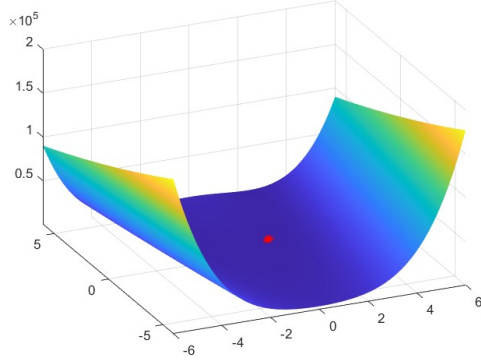
- $0 < \gamma < 1$ Contraction parameter
- $0 < \sigma < 1$ Shrinking parameter
- $\delta > 0$ the parameter for the initial simplex generated by the Nelder-Mead method

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.

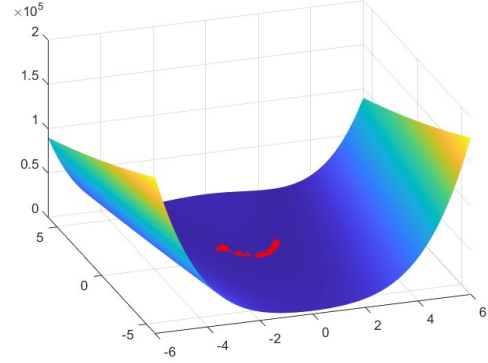
3.2 Truncated Newton method

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

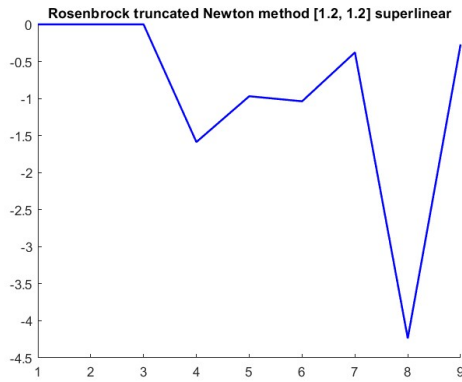


(a) 3D plot of the Rosenbrock function with the sequence found by the algorithm starting from $x_0 = [1.2, 1.2]$

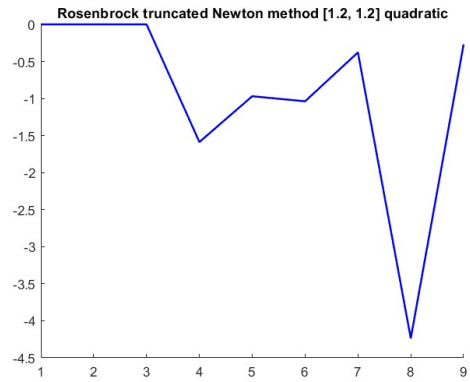


(b) 3D plot of the Rosenbrock function with the sequence found by the algorithm starting from $x_1 = [-1.2, 1]$

Figure 1: 3D plot of Rosenbrock function

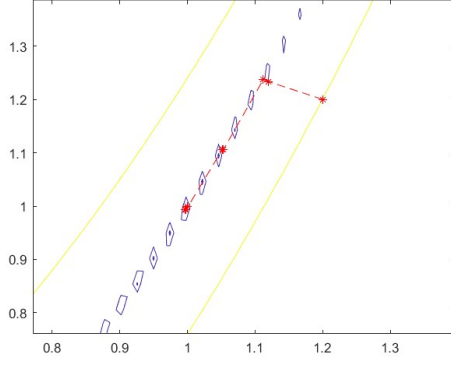


(a) Experimental rate of convergence starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

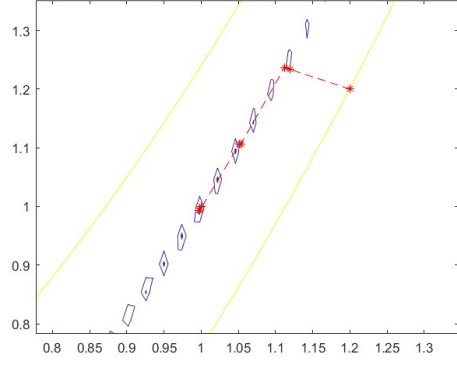


(b) Experimental rate of convergence starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 2: Experimental rate of convergence starting from x_0

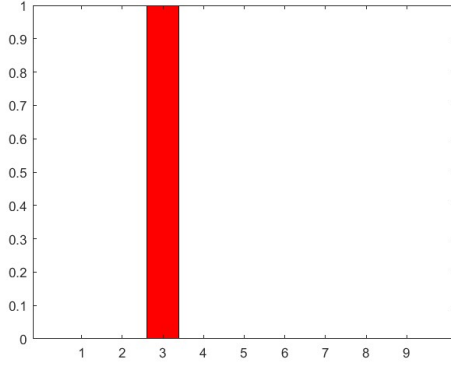


(a) Contour line of the function value and the sequence generated by the algorithm starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

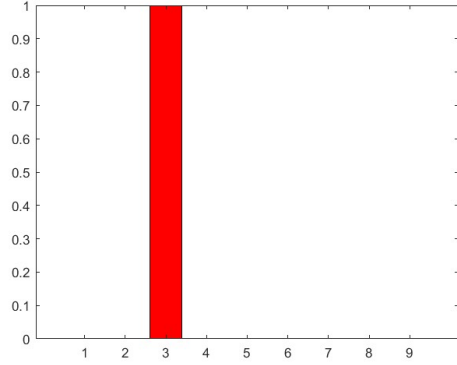


(b) Contour line of the function value and the sequence generated by the algorithm starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 3: Contour line starting from x_0

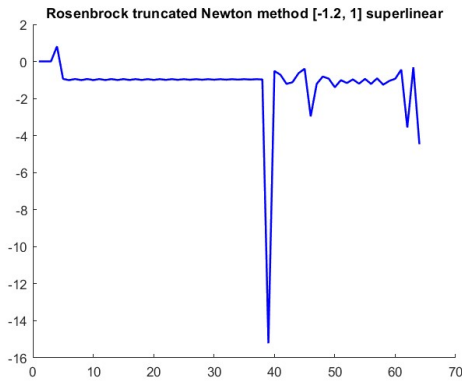


(a) Number of backtracking iterations required at every outer iteration starting from $x_0 = [1.2, 1.2]$ with a superlinear forcing term of tolerance

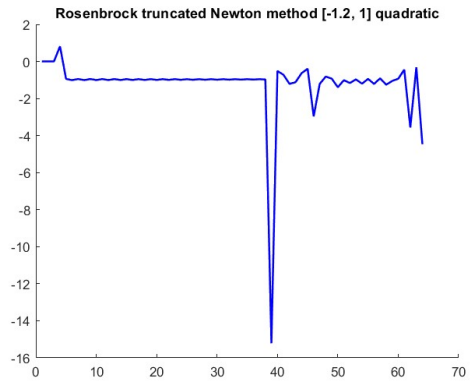


(b) Number of backtracking iterations required at every outer iteration starting from $x_0 = [1.2, 1.2]$ with a quadratic forcing term of tolerance

Figure 4: Backtracking starting from x_0

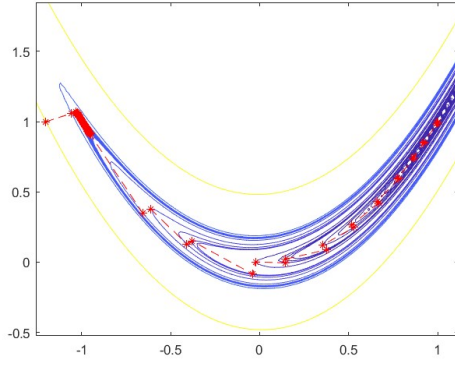


(a) Experimental rate of convergence starting from $x_1 = [-1.2, 1]$ with a superlinear forcing term of tolerance

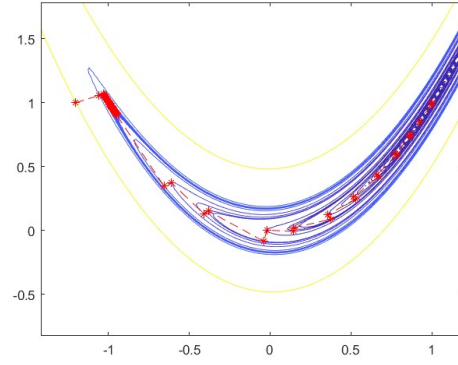


(b) Experimental rate of convergence starting from $x_1 = [-1.2, 1]$ with a quadratic forcing term of tolerance

Figure 5: Experimental rate of convergence starting from x_1

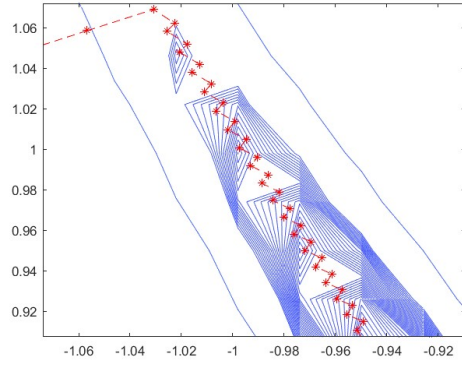


(a) Contour line of the function value and the sequence generated by the algorithm starting from $x_1 = [-1.2, 1]$ with a superlinear forcing term of tolerance

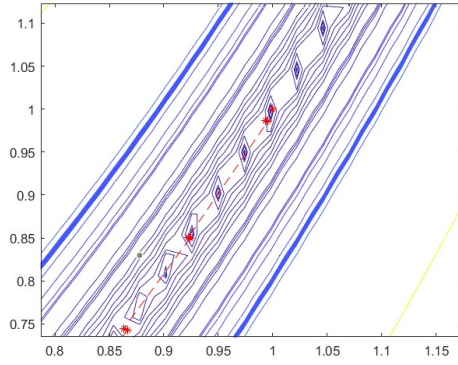


(b) Contour line of the function value and the sequence generated by the algorithm starting from $x_1 = [-1.2, 1]$ with a quadratic forcing term of tolerance

Figure 6: Contour line starting from x_1

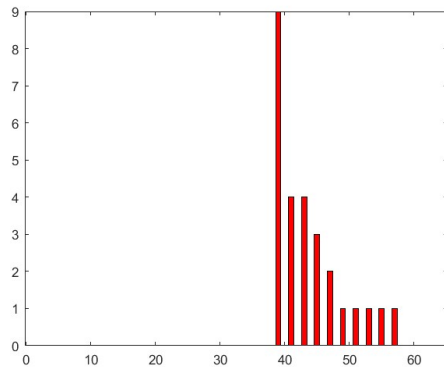


(a) The first iterations of the algorithm starting from $x_1 = [-1.2, 1]$

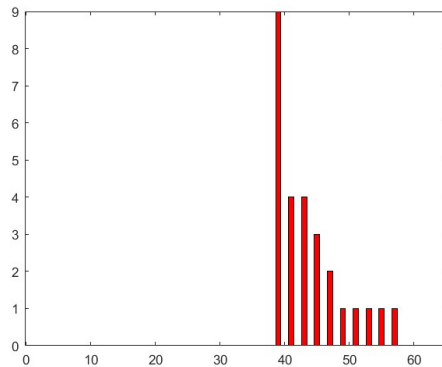


(b) The last iterations of the algorithm starting from $x_1 = [-1.2, 1]$

Figure 7: Zoom of contour line starting from x_1



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



(b) Number of backtracking iterations required at 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:

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

2. F27:

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

3. F16:

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

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

$$F(x) = \frac{1}{2} \sum_{k=1}^n f_k^2(x)$$

$$f_k(x) = \left(3 - \frac{x_k}{5}\right) x_k + 1 - x_{k-1} - 2x_{k+1}, \quad 1 \leq k \leq n$$

$$x_0 = x_{n+1} = 0, \quad x_l = -1, \quad l \geq 1$$

Exact derivatives

$$\frac{\partial F(x)}{\partial x_k} = -2f_{k-1}(x) + \left(3 - \frac{1}{5}x_k\right)f_k(x) - f_{k+1}(x) \quad \forall k = 2 \dots n-1$$

$$\frac{\partial F(x)}{\partial x_1} = \left(3 - \frac{1}{5}x_1\right)f_1(x) - f_2(x)$$

$$\frac{\partial F(x)}{\partial x_n} = \left(3 - \frac{1}{5}x_n\right)f_n(x) - 2f_{n-1}(x)$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} = 5 - \frac{1}{5}f_k(x) + \left(3 - \frac{1}{5}x_k\right)^2 \quad \forall k = 1 \dots n$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} = -2\left(3 - \frac{1}{5}x_k\right) - \left(3 - \frac{1}{5}x_{k+1}\right) \quad \forall k = 1 \dots n-1$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} = 0 \quad |k - j| \geq 2$$

Finite differences type 1

$$\frac{\partial F(x)}{\partial x_k} \approx -2f_{k-1}(x) + \left(3 - \frac{1}{5}x_k\right)f_k(x) - f_{k+1}(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_k\right)h^2 \quad \forall k = 2 \dots n-1$$

$$\frac{\partial F(x)}{\partial x_1} \approx \left(3 - \frac{1}{5}x_1\right)f_1(x) - f_2(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_1\right)h^2$$

$$\frac{\partial F(x)}{\partial x_n} \approx -2f_{n-1}(x) + \left(3 - \frac{1}{5}x_n\right)f_n(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_n\right)h^2$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx 5 - \frac{1}{5}f_k(x) + \left(3 - \frac{1}{5}x_k\right)^2 + \frac{1}{100}h^2 \quad \forall k = 1 \dots n$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx -2\left(3 - \frac{1}{5}x_k\right) - \left(3 - \frac{1}{5}x_{k+1}\right) + \frac{3}{10}h \quad \forall k = 1 \dots n-1$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 0 \quad |k - j| \geq 2$$

Finite differences type 2

$$\frac{\partial F(x)}{\partial x_k} \approx -2f_{k-1}(x) + \left(3 - \frac{1}{5}x_k\right)f_k(x) - f_{k+1}(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_k\right)h^2|x_k|^2 \quad \forall k = 1 \dots n-1$$

$$\frac{\partial F(x)}{\partial x_1} \approx \left(3 - \frac{1}{5}x_1\right)f_1(x) - f_2(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_1\right)h^2|x_1|^2$$

$$\frac{\partial F(x)}{\partial x_n} \approx -2f_{n-1}(x) + \left(3 - \frac{1}{5}x_n\right)f_n(x) - \frac{1}{10}\left(3 - \frac{1}{5}x_n\right)h^2|x_n|^2$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx 5 - \frac{1}{5}f_k(x) + \left(3 - \frac{1}{5}x_k\right)^2 + \frac{1}{100}h^2|x_k|^2 \quad \forall k = 1 \dots n$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx -2\left(3 - \frac{1}{5}x_k\right) - \left(3 - \frac{1}{5}x_{k+1}\right) + \frac{h|x_k|}{5} + \frac{h|x_{k+1}|}{10}$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx 0 \quad |k - j| \geq 2$$

NEW VERSION

```

1  if fin_dif_2 % version of finite differences with abs(xj)
2      z1=zeros(n,1);
3      z2=zeros(n,1);
4      z3=zeros(n,1);
5      z1(1:3:end)=ones(length(1:3:end),1);
6      z2(2:3:end)=ones(length(2:3:end),1);
7      z3(3:3:end)=ones(length(3:3:end),1);
8      z1=h*z1.*abs(x);
9      z2=h*z2.*abs(x);
10     z3=h*z3.*abs(x);
11     F1=(JF(x+z1)-JF(x-z1));
12     F2=(JF(x+z2)-JF(x-z2));
13     F3=(JF(x+z3)-JF(x-z3));
14     F1=[0;F1];
15     rem_n=rem(n,3);
16     if rem_n==0
17         F3=[F3;0];
18     elseif rem_n==1
19         F1=[F1;0];
20     else
21         F2=[F2;0];
22     end
23     m1=reshape(F1,3,[])';
24     m2=reshape(F2,3,[])';
25     m3=reshape(F3,3,[])';
26     M=zeros(n,3);
27     M(1:3:end,:)=m1;
28     M(2:3:end,:)=m2;
29     M(3:3:end,:)=m3;
30     for col=1:3
31         M(:,col)=M(:,col).*(1./abs(x))/h;
32     end
33     if sparse % sparse
34         HF=spdiags(M,[1,0,-1],n,n);
35     else % NOT sparse
36         HF=diag(M(:,2))+diag(M(2:end,1),1)+diag(M(1:end-1,1),-1);
37     end
38 else % classic version of finite differences
39     z1=zeros(n,1);
40     z2=zeros(n,1);
41     z3=zeros(n,1);
42     z1(1:3:end)=ones(length(1:3:end),1);
43     z2(2:3:end)=ones(length(2:3:end),1);
44     z3(3:3:end)=ones(length(3:3:end),1);
45     F1=(JF(x+h*z1)-JF(x-h*z1))/(2*h);
46     F2=(JF(x+h*z2)-JF(x-h*z2))/(2*h);
47     F3=(JF(x+h*z3)-JF(x-h*z3))/(2*h);
48     F1=[0;F1];
49     rem_n=rem(n,3);
50     if rem_n==0
51         F3=[F3;0];
52     elseif rem_n==1
53         F1=[F1;0];
54     else
55         F2=[F2;0];
56     end
57     m1=reshape(F1,3,[])';
58     m2=reshape(F2,3,[])';
59     m3=reshape(F3,3,[])';
60     M=zeros(n,3);
61     M(1:3:end,:)=m1;
62     M(2:3:end,:)=m2;
63     M(3:3:end,:)=m3;
64     if sparse % sparse
65         HF=spdiags(M,[1,0,-1],n,n);
66     else % NOT sparse
67         HF=diag(M(:,2))+diag(M(2:end,1),1)+diag(M(1:end-1,1),-1);
68     end
69 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

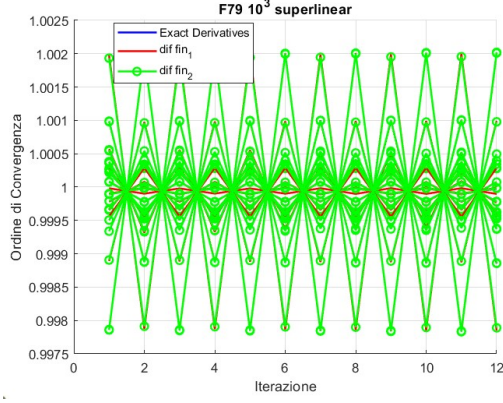
Result with $n = 50$

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

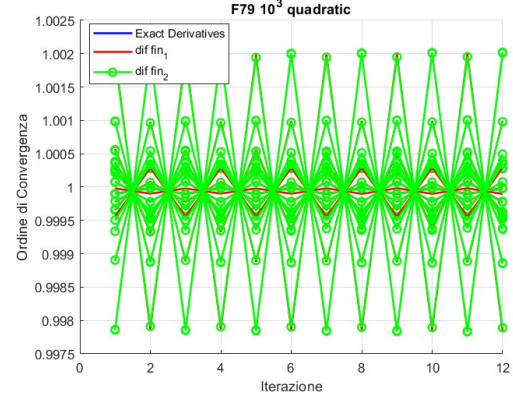
Table 9: Results of F79 n=50 Nelder-Mead

5.1.2 Truncated Newton method

Result with $n = 10^3$

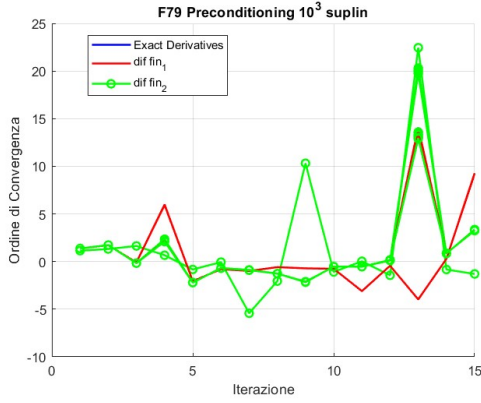


(a) The last 12 value of experimental rate of convergence F79 $n = 10^3$ superlinear

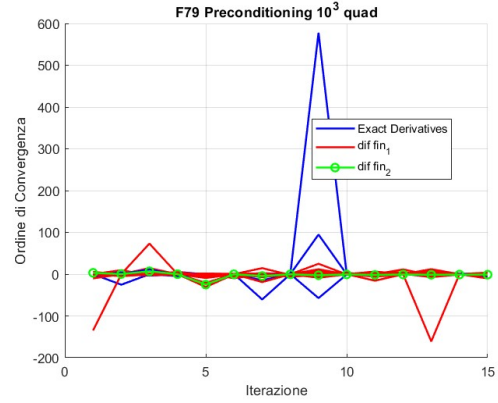


(b) The last 12 values of experimental rate of convergence F79 $n = 10^3$ quadratic

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

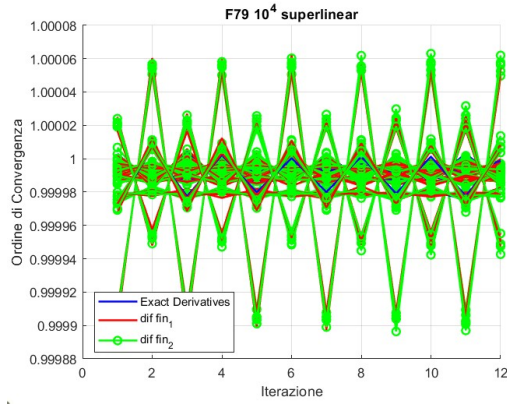
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586
FD2	0,050985357	0,081279835	0,096544335	0,094205248	0,095829323	0,093804506	0,092815586

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

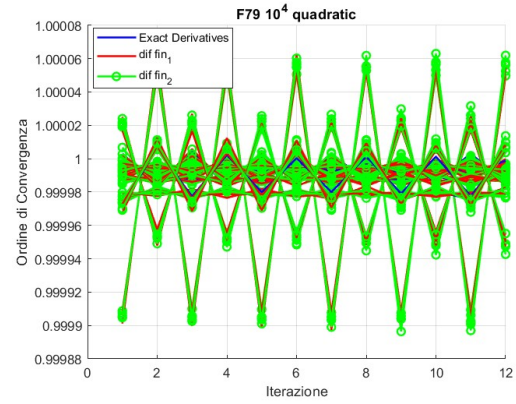
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586
FD2	0,058307492	0,094892594	0,09259999	0,093216783	0,098875207	0,092227912	0,092815586

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

Result with $n = 10^4$

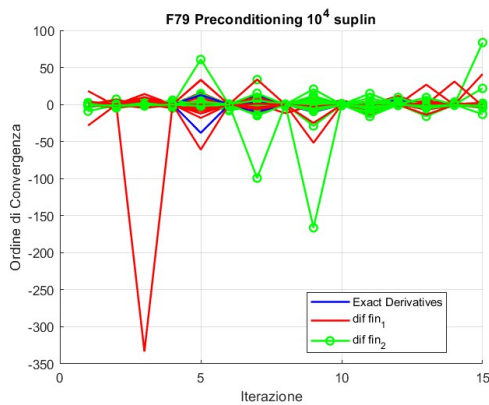


(a) The last 12 value of experimental rate of convergence F79 $n = 10^4$ superlinear

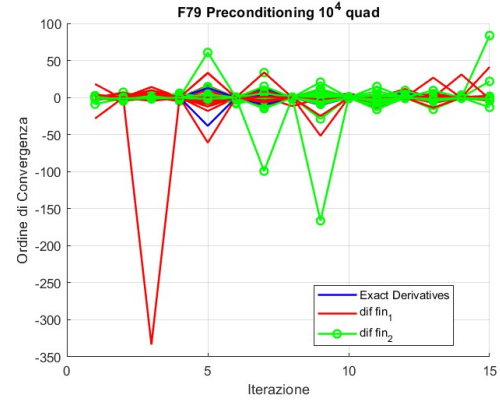


(b) The last 12 values of experimental rate of convergence F79 $n = 10^4$ quadratic

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

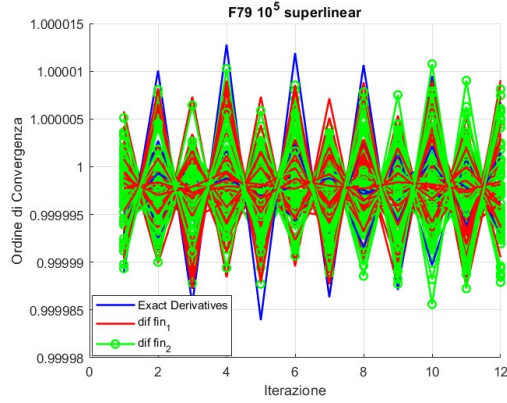
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,059887902	0,08157586	0,089931923	0,092971752	0,090164806	0,089053094	0,088755284
FD2	0,046500061	0,079162711	0,090823135	0,092170126	0,084739867	0,091098594	0,088755284

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

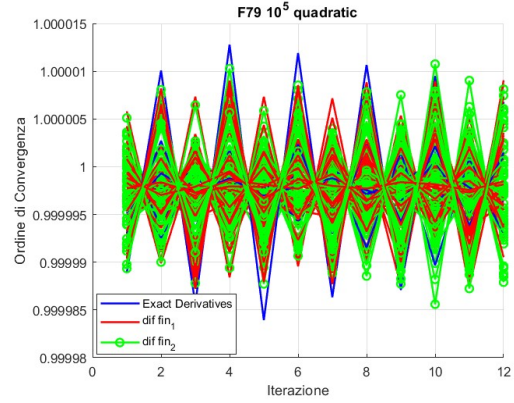
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,059887902	0,08157586	0,089931923	0,092971752	0,090164806	0,089053094	0,088755284
FD2	0,046500061	0,079162711	0,090823135	0,092170126	0,084739867	0,091098594	0,088755284

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

Result with $n = 10^5$

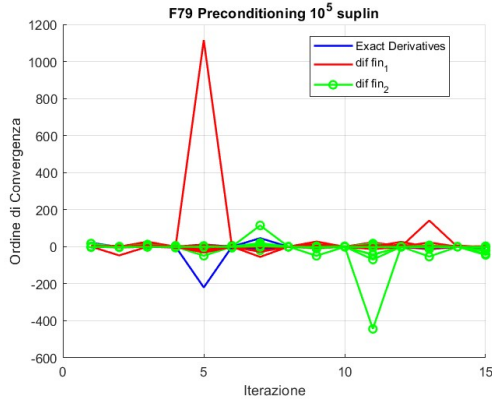


(a) The last 12 value of experimental rate of convergence F79 $n = 10^5$ superlinear

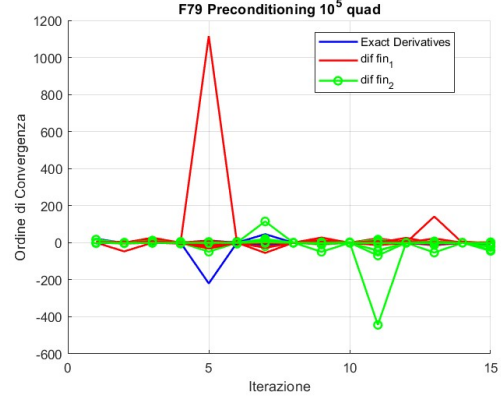


(b) The last 12 values of experimental rate of convergence F79 $n = 10^5$ quadratic

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

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

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

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

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

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 4x_k x_j + h^2 + 2hx_j + 2hx_k$$

Finite differences type 2

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

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

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} \approx 4x_k x_j + h^2 |x_k| |x_j| + 2hx_j |x_k| + 2hx_k |x_j|$$

NEW VERSION

```

1
2 % Initialization of zj and j
3 zj = z0;
4 j= 0;
5
6 if ~exact %approximation with finite difference (not exact)
7     if fin_dif_2
8         Azj= (gradf(xk+h.*abs(xk).*zj)-gradk)/(h*abs(xk));
9     else
10        Azj= (gradf(xk+h*zj)-gradk)/h;
11    end
12 end
13
14 if ~exact %approximation with finite difference (not exact)
15
16     if fin_dif_2
17         z= (gradf(xk+h.*abs(xk).*p)-gradk)/(h*abs(xk));
18     else
19         z=(gradf(xk+h*p)-gradk)/h;
20     end
21 end
22
23 if ~exact %approximation with finite difference (not exact)
24     if fin_dif_2
25         z_new= ((gradf(xk+h.*abs(xk).*p)-gradk)/(h*abs(xk)))';
26     else
27         z_new=((gradf(xk+h*p)-gradk)/h)';
28     end
29 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

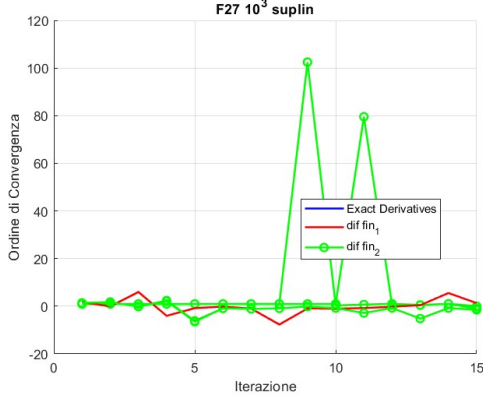
Result with $n = 50$

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

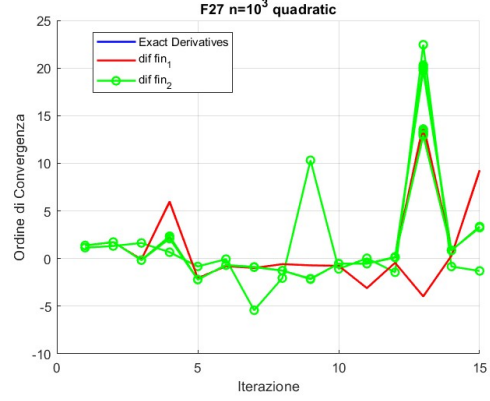
Table 100: Result of F27, n=50, Nelder-Mead

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

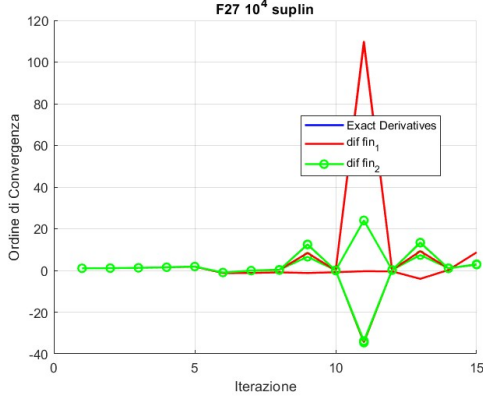
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	Nan	0,095238095	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976
FD2	Nan	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976	0,097560976

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

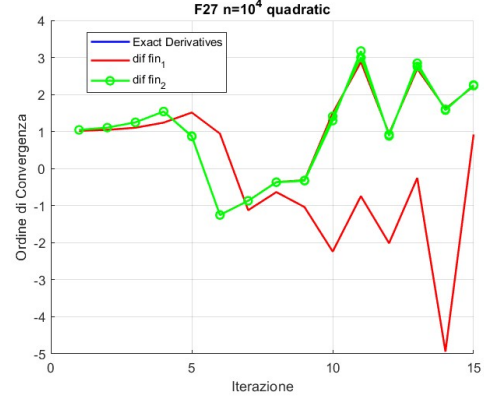
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	Nan	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488
FD2	0,5	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488	0,048780488

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

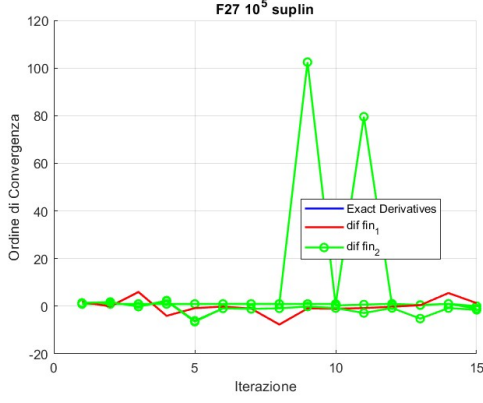
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	Nan	0,063829787	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191
FD2	Nan	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191

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

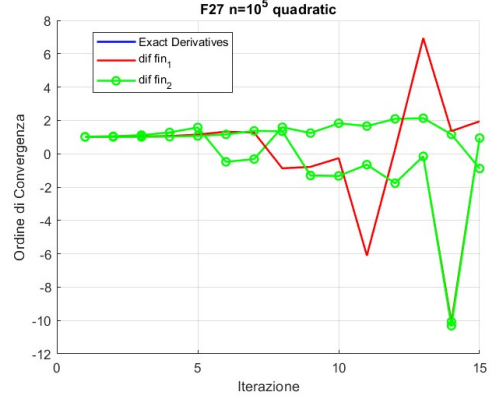
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	Nan	0,02173913	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191
FD2	Nan	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191	0,042553191

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

$$F(x) = \sum_{i=1}^n i [(1 - \cos(x_i)) + \sin(x_i) - 1 - \sin(x_{i+1})]$$

$$x_0 = x_{n+1} = 0, \quad x_i = 1, \quad i \geq 1$$

Exact derivatives

$$\frac{\partial F(x)}{\partial x_k} = k \sin x_k - 2 \cos x_k \quad \forall k = 1 \dots n - 1$$

$$\frac{\partial F(x)}{\partial x_n} = n \sin x_n + (n - 1) \cos x_n$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} = k \cos x_k + 2 \sin x_k \quad \forall k = 1 \dots n - 1$$

$$\frac{\partial^2 F(x)}{\partial^2 x_n} = (1 - n) \sin x_n + n \cos x_n$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_j} = 0 \quad \forall k = 1 \dots n$$

Finite differences type 1

$$\frac{\partial F(x)}{\partial x_k} \approx \frac{\sin(h)}{h} (2 \cos(x_k) + k \sin(x_k)) \quad \forall k = 1 \dots n - 1$$

$$\frac{\partial F(x)}{\partial x_n} \approx \frac{\sin(h)}{h} (n \sin(x_n) - (n - 1) \cos(x_n))$$

$$\frac{\partial^2 F(x)}{\partial^2 x_k} \approx \left(\frac{h^2}{4 * 3} - 1 \right) (2 \sin(x_k) - k \cos(x_k)) \quad \forall k = 1 \dots n - 1$$

$$\frac{\partial^2 F(x)}{\partial^2 x_n} \approx \left(1 - \frac{h^2}{4 * 3} \right) ((n - 1) \sin(x_n) + n \cos(x_n))$$

$$\frac{\partial^2 F(x)}{\partial x_k \partial x_{k+1}} \approx 0 \quad \forall k, j = 1 \dots n$$

Finite differences type 2

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

NEW VERSION

```

1 function HF=HF16New(x,sparse,exact,fin_dif_2,h,JF)
2 % Function that computes the Hessian of function 79
3 % sparse= bool. True= computes the sparse version
4 % exact= bool. True= computes the exact version, False= computes the approximated version
   with finite differences
5 % 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
6 % h= increment for the approximated version (if exact=true put h=0)
7 % JF= function handle of the gradient
8
9 n=length(x);
10 if exact %exact version
11     indices=(1:n)';
12     D=indices.*cos(x)-2*sin(x);
13     if sparse %sparse version
14         HF=spdiags(D,0,n,n);
15     else % NOT sparse version
16         HF=diag(D);
17     end
18     HF(n,n)=(n-1)*sin(x(n))+n*cos(x(n));
19 else %approximation with finite difference (not exact)
20     if fin_dif_2 %version of finite differences with abs(xj)
21         d=(JF(x+h.*abs(x).*ones(n,1))-JF(x-h.*abs(x).*ones(n,1)))/(2*h*abs(x));
22         if sparse %sparse version
23             HF =spdiags(d,0,n,n);
24         else % NOT sparse
25             HF=diag(d);
26         end
27     else % classic version of finite differences
28         d=(JF(x+h*ones(n,1))-JF(x-h*ones(n,1)))/(2*h);
29         if sparse %sparse version
30             HF =spdiags(d,0,n,n);
31         else % NOT sparse
32             HF=diag(d);
33         end
34     end
35 end
36
37 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

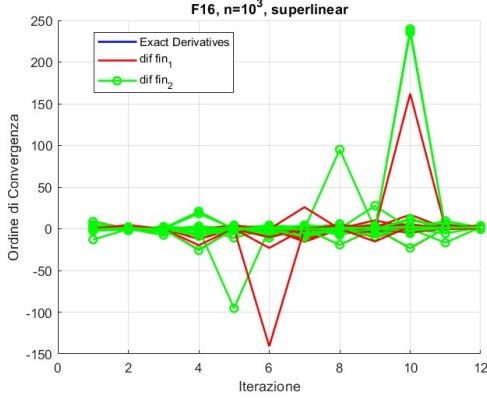
Result with $n = 50$

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

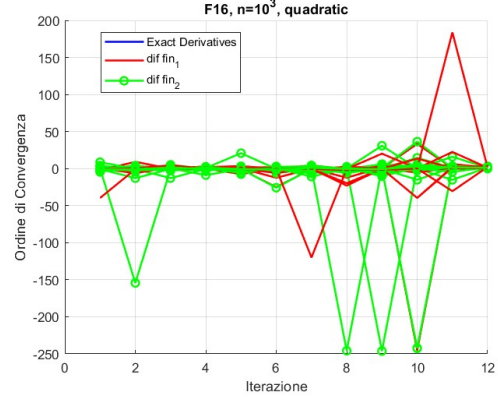
Table 147: Results of F16 n=50 Nelder-Mead

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

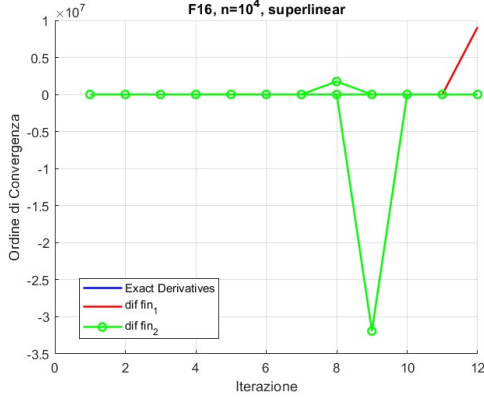
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,501573081	0,431463247	0,368521605	0,461148594	0,461148594	0,461148594	0,461148594
FD2	0,425185529	0,41514874	0,416245856	0,450235598	0,461148594	0,461148594	0,461148594

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

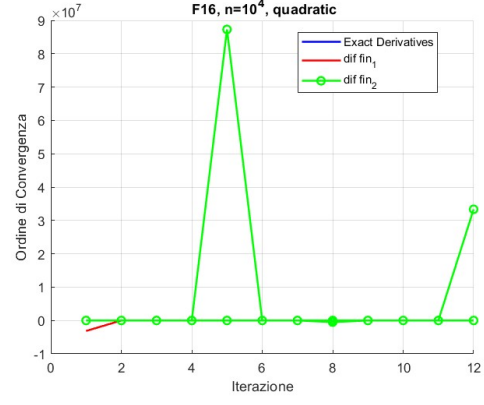
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,453264948	0,425322061	0,99746633	0,448700919	0,448700919	0,448700919	0,448700919
FD2	0,461958152	0,471916055	0,441742424	0,434685767	0,448700919	0,448700919	0,448700919

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

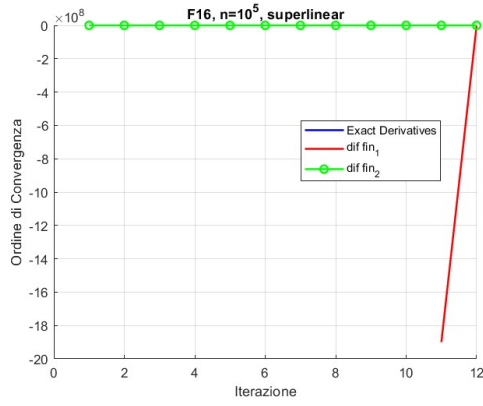
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	0,43710686	0,364964526	0,91230475	0,274938323	0,274938323	0,274938323	0,274938323
FD2	0,78700118	0,369798728	1,11886688	0,347550951	0,274938323	0,274938323	0,274938323

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

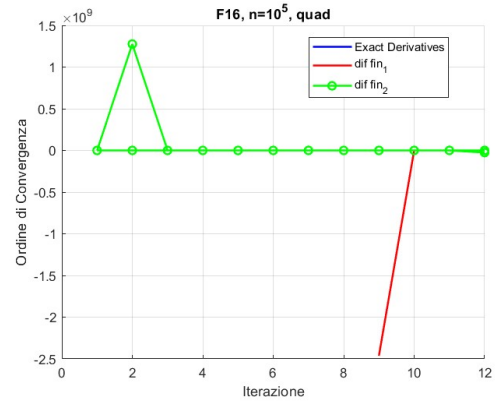
Row	$h = 10^{-2}$	$h = 10^{-4}$	$h = 10^{-6}$	$h = 10^{-8}$	$h = 10^{-10}$	$h = 10^{-12}$	Exact
FD1	1,573693933	0,392162411	0,581851996	0,373341243	0,373341243	0,373341243	0,373341243
FD2	1,800345009	0,362367185	4,289922801	0,389773489	0,373341243	0,373341243	0,373341243

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 tables [22](#), [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 back-trackings 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 truncated 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

```

1 function F1=F16(x)
2 %function F16 with vector operations
3 n=length(x);
4 term1=1-cos(x);
5 sin_prec=[0; x(1:end-1)];
6 sin_next=[x(2:end); 0];
7 term2=sin(sin_prec)-sin(sin_next);
8 indici=(1:n)';
9 F1=sum(indici.*(term1+term2));
10 end

```

```

1 function JF=JF16(x,exact,fin_dif_2,h)
2 % Function that computes the gradient of function 27
3 % exact= bool. True= computes the exact version, False= computes the approximated version
  with finite differences
4 % 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
5 % h= increment for the approximated version (if exact=true put h=0)
6
7 n=length(x);
8
9 indices=(1:n)';
10 JF=indices.*sin(x)+2*cos(x);
11 JF(n)=(1-n)*cos(x(n))+n*sin(x(n));
12 if ~exact %approximation with finite difference (not exact)
13     if fin_dif_2 % version of finite differences with abs(xj)
14         JF=(sin(h*abs(x))./(h*abs(x))).*JF;
15     else % classic version of finite differences
16         JF=sin(h)/h*JF;
17     end
18 end
19
20 end

```

```

1 function HF=HF16(x,sparse,exact,fin_dif_2,h)
2 % Function that computes the Hessian of function 79
3 % sparse= bool. True= computes the sparse version
4 % exact= bool. True= computes the exact version, False= computes the approximated version
   with finite differences
5 % 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
6 % h= increment for the approximated version (if exact=true put h=0)
7
8 n=length(x);
9 indices=(1:n)';
10 D=indices.*cos(x)-2*sin(x);
11 if sparse %sparse version
12     HF=spdiags(D,0,n,n);%exact version
13     if ~exact %approximation with finite difference (not exact)
14         if fin_dif_2 % version of finite differences with abs(xj)
15             %vec_diag=((1-cos(h*abs(x)))./(x.^2))/(h^2); % no taylor expansion
16             vec_diag= 1-(h^2)*(x.^2)/12 ; % with taylor expansion
17             new_diag=vec_diag.*diag(HF,0);
18             HF=spdiags(new_diag,0,n,n);
19         else % classic version of finite differences
20             HF=(1-(h^2)/12)*HF ; % as h is small i use cos(h) taylor expansion to avoid
               numerical cancellation
21         end
22         % elementi non diagonali nulli anche con differenze finite
23     end
24 else % NOT sparse version
25     HF=diag(D); %exact version
26     if ~exact %approximation with finite difference (not exact)
27         if fin_dif_2 % version of finite differences with abs(xj)
28             % vec_diag=((1-cos(h*abs(x)))./(x.^2))/(h^2); % no taylor expansion
29             vec_diag= 1-(h^2)*(x.^2)/12 ; % with taylor expansion
30             HF=diag(vec_diag.*diag(HF));
31         else % classic version of finite differences
32             HF=(1-(h^2)/12)*HF; % as h is small i use cos(h) taylor expansion to avoid
               numerical cancellation
33         end
34     end
35 end
36
37 HF(n,n)=(n-1)*sin(x(n))+n*cos(x(n));
38 if ~exact %approximation with finite difference (not exact)
39     HF(n,n)=(1-(h^2)/12)*HF(n,n);
40 end
41
42 end

```

```

1 function F=F27(x)
2 %function F27 with vector operations
3 fk=(x-1)/sqrt(100000);
4 F=sum(fk.^2);
5 F=F+(sum(x.^2)-0.25)^2;
6 F=F/2;
7 end

```

```

1 function JF=JF27(x,exact,fin_dif_2,h)
2 % Function that computes the gradient of function 27
3 % exact= bool. True= computes the exact version, False= computes the approximated version
   with finite differences
4 % h= increment for the approximated version (if exact=true put h=0)
5
6 n=length(x);
7 s=sum(x.^2);
8 JF=((2*x-2)/100000 + 4*s*x-x)/2;
9 if ~exact %approximation with finite difference (not exact)
10     if fin_dif_2
11         JF=JF+2*(h^2).*abs(x).*x;
12     else
13         JF=JF+2*(h^2)*x;
14     end
15 end

```

```

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

```

```

1 function JF=JF79(x,exact,fin_dif_2,h)
2 % Function that computes the gradient of function 79
3 % exact= bool. True= computes the exact version, False= computes the approximated version
  with finite differences
4 % 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
5 % h= increment for the approximated version (if exact=true put h=0)
6
7 x_next=[x(2:end);0];
8 x_prev=[0;x(1:end-1)];
9
10 f=(3-x/10).*x+1-x_prev-2*x_next;
11 f_next=[f(2:end);0];
12 f_prev=[0;f(1:end-1)];
13
14 JF=-2*f_prev+f.*(3-x/5)-f_next;
15 if ~exact %approximation with finite difference (not exact)
16     if fin_dif_2 % version of finite differences with abs(xj)
17         JF=JF-(3-x/5).*(x.^2)*(h^2)/10;
18     else % classic version of finite differences
19         JF=JF-(3-x/5)*(h^2)/10;
20     end
21 end
22 end

```

```

1 function HF=HF79(x,sparse,exact,fin_dif_2,h)
2 % Function that computes the Hessian of function 79
3 % sparse= bool. True= computes the sparse version
4 % exact= bool. True= computes the exact version, False= computes the approximated version
  with finite differences
5 % 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
6 % h= increment for the approximated version (if exact=true put h=0)
7
8 x_next=[x(2:end);0];
9 x_prev=[0;x(1:end-1)];
10 n= length(x);
11
12 f=(3-x/10).*x+1-x_prev-2*x_next;
13 diag0=5-f/5+(3-x/5).^2;
14 diag_1= -2*(3-x(1:end-1)/5)-(3-x_next(1:end-1)/5);
15 if sparse % sparse
16     diag_up=[0;diag_1];
17     diag_down=[diag_1; 0];
18     HF=spdiags([diag_down, diag0, diag_up],[-1,0,1],n,n); %exact version
19     if ~exact %approximation with finite difference (not exact)
20         if fin_dif_2 % version of finite differences with abs(xj)
21             HF= HF + (h^2)*spdiags(x.^2,0,n,n)/100;
22             vec_diag= abs(x(1:end-1))/5 + abs(x(2:end))/10;
23             HF= HF + spdiags(h*[0;vec_diag],1,n,n) +spdiags(h*[vec_diag;0],-1,n,n);
24
25         else % classic version of finite differences
26             HF= HF + spdiags((h^2)*ones(n,1)/100,0,n,n);
27             new_coef=3*h/10;
28             HF= HF + spdiags(new_coef*ones(n,1),1,n,n) +spdiags(new_coef*ones(n,1),-1,n,n);
29         end
30     end
31
32 else % NOT sparse
33     HF=diag(diag0)+diag(diag_1,1)+diag(diag_1,-1); %exact version
34     if ~exact %approximation with finite difference (not exact)

```

```

35
36     if fin_dif_2 % version of finite differences with abs(xj)
37         HF= HF + (h^2)*diag(x.^2)/100 ;
38         vec_diag= abs(x(1:end-1))/5 + abs(x(2:end))/10;
39         HF= HF + diag(h*vec_diag,1) +diag(h*vec_diag,-1);
40
41     else % classic version of finite differences
42         HF= HF + (h^2)*eye(n)/100 ;
43         new_coef=3*h/10;
44         HF= HF + diag(new_coef*ones(n-1,1),1) +diag(new_coef*ones(n-1,1),-1);
45     end
46 end
47
48 end

```

Nelder-Mead codes

```

1  function [x, f_k, n_iter]=Nelder_mead(x0,f,rho,mu, gamma, sigma, tol, max_iter, Delta)
2  n=length(x0);
3  % NOTATION
4  %n = dimension of vectors
5  % S = Simplex = matrix n x n+1 --> n+1 vectors of lenght n
6  % f_val_S = row vector, lenght=n+1 containing the values of the function in
7  % the n+1 point of the simplex
8
9  % Function that performs the Nelder-Mead optimization method, for a
10 % given function f
11
12 % INPUTS:
13 % x0 = n-dimensional column vector. Initial point;
14 % f = function handle that describes a function R^n->R;
15 % rho = reflection parameter
16 % mu= expansion parameter
17 % gamma = contraction parameter
18 % sigma = shrinking parameter
19 % tol= tolerance for stopping criteria
20 % max_iter= maximum number of iteration permitted;
21
22 % OUTPUTS:
23 % xk = the sequence of the best xk ad every iteration;
24 % fk = the sequence of f(xk) ad every iteration value;
25 % n_iter = number of iteration
26
27 x=x0;
28 f_k=f(x0);
29
30 S = [x0, x0 + Delta * eye(n)];
31
32
33 %f_values in the vertices of the simplex
34 f_val_S=zeros(1,n+1);
35 for i = 1:n+1
36     f_val_S(i) = f(S(:,i));
37 end
38
39
40 idx=1:n+1;
41 iter=0;
42
43 [f_val_S, sort_idx] = sort(f_val_S); %vector with ordered index eith respect to the
44 % values of the function
45 idx=idx(sort_idx);
46
47 while iter < max_iter && abs(f_val_S(n+1) - f_val_S(1)) > tol
48     %-----REFLECTION PHASE -----
49     %computation of barycenter point
50     x_bar=mean(S(:, idx(1:n)),2); %
51
52     %computation and evaluation of reflection point
53     x_r= x_bar + rho*(x_bar - S(:,idx(n+1)));

```

```

54     f_r=f(x_r);
55
56     if f_r < f_val_S(1)
57         %-----EXPANSION-----
58         %computation and evaluation of expansion point
59         x_e=x_bar + mu*(x_r-x_bar);
60         f_e=f(x_e);
61
62         if f_e < f_r
63             %hold expansion point
64             S(:,idx(n+1))=x_e;
65             f_val_S(n+1)=f_e;
66         else
67             % hold reflection point
68             S(:, idx(n+1)) = x_r;
69             f_val_S(n+1) = f_r;
70         end
71
72     elseif f_r < f_val_S(n)
73
74         %hold reflexion point x_r
75         S(:, idx(n+1)) = x_r;
76         f_val_S(n+1) = f_r;
77
78     else
79         %-----CONTRACTION-----
80         %calculate x_c with the best point between x_r, x_n+1
81         if f_r < f_val_S(n+1)
82             x_c= x_bar + gamma *(x_bar - x_r);
83         else
84             x_c= x_bar + gamma *(x_bar - S(:,idx(n+1)));
85         end
86         f_c=f(x_c);
87         if f_c < f_val_S(n+1)
88             %hold x_c
89             S(:, idx(n+1)) = x_c;
90             f_val_S(n+1) = f_c;
91
92         else
93             %-----SHRINKAGE-----
94             for i = 2:n+1
95                 S(:, idx(i)) = S(:, idx(1)) + sigma * (S(:, idx(i)) - S(:,idx(1) ));
96                 f_val_S(i) = f(S(:, idx(i)));
97             end
98
99         end
100
101     end
102     [f_val_S, sort_idx] = sort(f_val_S);
103     idx=idx(sort_idx);
104
105     iter = iter + 1;
106
107     x=[x, S(:,idx(1))];
108     f_k=[f_k;f_val_S(1)];
109 end
110
111 n_iter=iter;
112
113 end

```

```

1  rng(345989);
2  format short;
3
4  % Rosenbrock function
5  f = @(x) 100*(x(2,:)- x(1,:).^2).^2 + (1 - x(1,:)).^2;
6
7  % initial points
8  x0 = [1.2; 1.2];
9  x1 = [-1.2; 1];
10 tol = 1e-14; %tolerance
11 max_iter = 1e5; % Number of max iteration
12

```

```

13 % parameters with some tuning
14 parametri=[
15     1,    2,    0.5, 0.5, 0.1;
16     1.2, 4,    0.7, 0.3, 0.1;
17     1.2, 4,    0.7, 0.3, 0.5;
18     1.4, 5,    0.8, 0.2, 0.1;
19     1.65, 4.55, 0.95, 0.15, 0.1;
20     2,    4,    0.7, 0.5, 0.5;
21 ];
22 % Cration of a table
23 results = table;
24
25 %test the function with several parameteres in the two different initial
26 %points
27 for i = 1:size(parametri, 1)
28     rho = parametri(i, 1);
29     chi = parametri(i, 2);
30     gamma = parametri(i, 3);
31     sigma = parametri(i, 4);
32     Delta = parametri(i, 5);
33
34     tic;
35     [x_0, f_0, n_iter_0] = Nelder_mead(x0, f, rho, chi, gamma, sigma, tol, max_iter,
36         Delta);
37     tempo_x0=toc;
38     tic;
39     [x_1, f_1, n_iter_1] = Nelder_mead(x1, f, rho, chi, gamma, sigma, tol, max_iter,
40         Delta);
41     tempo_x1=toc;
42
43     % Add the result in the table
44     results = [results; table(rho, chi, gamma, sigma, Delta, ...
45         x_0(1,end), x_0(2,end), f_0(end), n_iter_0, tempo_x0, ...
46         x_1(1,end), x_1(2,end), f_1(end), n_iter_1, tempo_x1)];
47 end
48
49 % columns' name
50 results.Properties.VariableNames = {'Rho', 'Chi', 'Gamma', 'Sigma', 'Delta', ...
51     'X_0(1)', 'X_0(2)', 'F_0', 'Iter_0', 'Time_0' ...
52     'X_1(1)', 'X_1(2)', 'F_1', 'Iter_1', 'Time_1'};
53
54 % disp table
55 disp('Table with the results of Nelder-Mead method with Rosenbrock function');
56 disp(results);
57
58 % save the results ona cvs file
59 writetable(results, 'Nelder_Rosenbrock_with_Delta.xlsx', 'WriteRowNames', true);

```

```

1  %% FUNCTION F16 n=10
2  % setting parameters
3  format long
4  rng(345989);
5  n = 10;
6  tol = 1e-13;          %tollerance
7  max_iter = 1e06;      %max iteration
8  rho = 1.1;           %reflection parameter
9  mu = 2.5;            %expansion parameter
10 gamma = 0.8;          %contraction parameter
11 sigma = 0.9;          %shirinking parameter
12 delta = 1;            %Initialization of the simplex
13
14 % function
15 F = @(x) F16(x);
16
17
18 N=10; %number of starting points
19 x0 = ones(n, 1); % starting point
20 Mat_points= repmat(x0,1,N+1);
21 rand_mat=2*(rand([n, N+1]) - 0.5); %random matrix between [-1,1]
22 Mat_points=Mat_points + rand_mat; %starting points
23 %vector for saving times
24 times_10=zeros(1,N+1);
25 %vector for saving minimum point
26 vec_10=zeros(1,N+1);
27 %vector for saving iteration
28 vec_iter_10=zeros(1,N+1);
29
30
31
32 for j = 1:N+1
33     %applying the function F16 to the 11 strarting points
34     tic;
35     [xk_16_10, fk_16_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
36         tol, max_iter, delta);
37     %saving results
38     times_10(j) = toc;
39     vec_10(j) = fk_16_10(end);
40     vec_iter_10(j) = n_iter;
41 end
42 %creation of a table with the results
43 results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
44     ...
45     times_10', vec_10', vec_iter_10', ...
46     'VariableNames', {'Initial_condition', 'Time', 'FinalValue', '
47     Iterations'});
48
49 % Computation of mean of the three values saved
50 mean_time = mean(results_n10.Time);
51 mean_final_value = mean(results_n10.FinalValue);
52 mean_iterations = mean(results_n10.Iterations);
53
54 % Insert the mean in the tables
55 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
56     'VariableNames', results_n10.Properties.VariableNames);
57 results_n10 = [results_n10; mean_row];
58
59 % Display the table
60 disp(results_n10);
61 % Creation an excel table
62 writetable(results_n10, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_10');
63
64 %% FUNCTION F16 n=25
65 %The same structure of n=10
66 n = 25;
67 tol = 1e-13;
68 max_iter = 1e06;
69 rho = 1.1;
70 mu = 1.8;
71 gamma = 0.8;
72 sigma = 0.9;

```

```

71 delta = 0.1;
72
73 F = @(x) F16(x);
74
75 x0 = ones(n, 1);
76 Mat_points = repmat(x0,1,N+1) + 2*(rand([n, N+1]) - 0.5);
77
78 times_25 = zeros(1,N+1);
79 vec_25 = zeros(1,N+1);
80 vec_iter_25 = zeros(1,N+1);
81
82 for j = 1:N+1
83     tic;
84     [xk_16_25, fk_16_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
85         tol, max_iter, delta);
86     times_25(j) = toc;
87     vec_25(j) = fk_16_25(end);
88     vec_iter_25(j) = n_iter;
89 end
90 results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
91     ...
92     times_25', vec_25', vec_iter_25', ...
93     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
94 mean_time = mean(results_n25.Time);
95 mean_final_value = mean(results_n25.FinalValue);
96 mean_iterations = mean(results_n25.Iterations);
97
98 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
99     'VariableNames', results_n25.Properties.VariableNames);
100 results_n25 = [results_n25; mean_row];
101
102 disp(results_n25);
103
104 writetable(results_n25, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_25');
105
106
107 %% FUNCTION F16 n=50
108 % The same structure of n=10
109 n = 50;
110 tol = 1e-13;
111 max_iter = 1e06;
112 rho = 1.1;
113 mu = 1.8;
114 gamma = 0.8;
115 sigma = 0.9;
116 delta = 0.1;
117
118 F = @(x) F16(x);
119
120 x0 = ones(n, 1);
121 Mat_points = repmat(x0,1,N+1) + 2*(rand([n, N+1]) - 0.5);
122
123 times_50 = zeros(1,N+1);
124 vec_50 = zeros(1,N+1);
125 vec_iter_50 = zeros(1,N+1);
126
127 for j = 1:N+1
128     tic;
129     [xk_16_50, fk_16_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
130         tol, max_iter, delta);
131     times_50(j) = toc;
132     vec_50(j) = fk_16_50(end);
133     vec_iter_50(j) = n_iter;
134 end
135 results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
136     ...
137     times_50', vec_50', vec_iter_50', ...
138     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
139 mean_time = mean(results_n50.Time);
140 mean_final_value = mean(results_n50.FinalValue);

```



```

140 mean_iterations = mean(results_n50.Iterations);
141
142 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
143                 'VariableNames', results_n50.Properties.VariableNames);
144
145 results_n50 = [results_n50; mean_row];
146 disp(results_n50);
147 writetable(results_n50, 'Risultati_F16_Nelder.xlsx', 'Sheet', 'n_50');

```

```

1 %% FUNZIONE F27 n=10
2 format long
3 rng(345989);
4 %setting parameters
5 n = 10;
6 tol = 1e-13; %tollerance
7 max_iter = 1e06; %max iteration
8 rho = 1.1; %reflection parameter
9 mu = 2.5; %expansion parameter
10 gamma = 0.8; %contraction parameter
11 sigma = 0.9; %shirinking parameter
12 delta = 1; %Initialization of the simplex
13
14 %function
15 F = @(x) F27(x);
16
17 N = 10; %number of starting points
18 x0 = ones(n, 1); %startinh point
19 Mat_points= repmat(x0,1,N+1);
20 rand_mat=2*(rand([n, N+1]) - 0.5); %random matrix between [-1,1]
21 Mat_points=Mat_points + rand_mat; %starting points
22 times_10 = zeros(1,N+1); %vector for saving times
23 vec_10 = zeros(1,N+1); %vector for saving minimum point
24 vec_iter_10 = zeros(1,N+1); %vector for saving iteration
25
26 for j = 1:N+1
27     %applying the function F16 to the 11 strarting points
28     tic;
29     [xk_27_10, fk_27_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
30         tol, max_iter, delta);
31     %saving results
32     times_10(j) = toc;
33     vec_10(j) = fk_27_10(end);
34     vec_iter_10(j) = n_iter;
35
36 end
37 %creation of a table with the results
38 results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
39                     ...
40                     times_10', vec_10', vec_iter_10', ...
41                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
42
43 % Computation of mean of the three values saved
44 mean_time = mean(results_n10.Time);
45 mean_final_value = mean(results_n10.FinalValue);
46 mean_iterations = mean(results_n10.Iterations);
47
48 % Insert the mean in the tables
49 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
50                 'VariableNames', results_n10.Properties.VariableNames);
51 results_n10 = [results_n10; mean_row];
52
53 % Display the table
54 disp(results_n10);
55 % Creation an excel table
56 writetable(results_n10, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_10');
57
58 %% FUNZIONE F27 n=25
59 %The same structure of n=10
60 n = 25;
61 tol = 1e-14;
62 max_iter = 1e08;
63 rho = 1.1;
64 mu = 1.8;

```

```

63 gamma = 0.8;
64 sigma = 0.9;
65 delta = 0.1;
66
67 F = @(x) F27(x);
68
69 x0 = ones(n, 1);
70 Mat_points=repmat(x0,1,N+1);
71 rand_mat=2*(rand([n, N+1]) - 0.5);
72 Mat_points=Mat_points + rand_mat;
73
74 times_25 = zeros(1,N+1);
75 vec_25 = zeros(1,N+1);
76 vec_iter_25 = zeros(1,N+1);
77
78 for j = 1:N+1
79     tic;
80     [xk_27_25, fk_27_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
81         tol, max_iter, delta);
82     times_25(j) = toc;
83     vec_25(j) = fk_27_25(end);
84     vec_iter_25(j) = n_iter;
85 end
86
87 results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
88     ...
89     times_25', vec_25', vec_iter_25', ...
90     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
91
92 mean_time = mean(results_n25.Time);
93 mean_final_value = mean(results_n25.FinalValue);
94 mean_iterations = mean(results_n25.Iterations);
95
96 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
97     'VariableNames', results_n25.Properties.VariableNames);
98
99 results_n25 = [results_n25; mean_row];
100
101 disp(results_n25);
102
103 writetable(results_n25, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_25');
104
105 %% FUNZIONE F27 n=50
106 %The same structure of n=10
107 n = 50;
108 tol = 1e-13;
109 max_iter = 1e06;
110 rho = 1.1;
111 mu = 1.8;
112 gamma = 0.8;
113 sigma = 0.9;
114 delta = 0.1;
115
116 F = @(x) F27(x);
117
118 x0 = ones(n, 1);
119 Mat_points=repmat(x0,1,N+1);
120 rand_mat=2*(rand([n, N+1]) - 0.5);
121 Mat_points=Mat_points + rand_mat;
122
123 times_50 = zeros(1,N+1);
124 vec_50 = zeros(1,N+1);
125 vec_iter_50 = zeros(1,N+1);
126
127 for j = 1:N+1
128     tic;
129     [xk_27_50, fk_27_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
130         tol, max_iter, delta);
131     times_50(j) = toc;
132     vec_50(j) = fk_27_50(end);
133     vec_iter_50(j) = n_iter;

```

```

133 end
134
135 results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
136                     times_50', vec_50', vec_iter_50', ...
137                     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
138
139 mean_time = mean(results_n50.Time);
140 mean_final_value = mean(results_n50.FinalValue);
141 mean_iterations = mean(results_n50.Iterations);
142
143 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
144                 'VariableNames', results_n50.Properties.VariableNames);
145
146 results_n50 = [results_n50; mean_row];
147
148 disp(results_n50);
149
150 writetable(results_n50, 'Risultati_F27_Nelder.xlsx', 'Sheet', 'n_50');
151
152 disp('Tutti i risultati sono stati salvati in Risultati_F27.xlsx.');
```

```

1 %% FUNZIONE F79
2 % setting parameters
3 format long
4 rng(345989);
5 n = 10;
6 tol = 1e-13; %tollerance
7 max_iter = 1e06; %max iteration
8 rho = 1.1; %reflection parameter
9 mu = 2.5; %expansion parameter
10 gamma = 0.8; %contraction parameter
11 sigma = 0.9; %shrinking parameter
12 delta = 1; %Initialization of the simplex
13
14 % function
15 F = @(x) F79(x);
16
17 N=10; %number of starting points
18 x0 = ones(n, 1); % starting point
19 Mat_points=repmat(x0,1,N+1);
20 rand_mat=2*(rand([n, N+1]) - 0.5); % random matrix between [-1,1]
21 Mat_points=Mat_points + rand_mat; % starting points
22 %vector for saving times
23 times_10=zeros(1,N+1);
24 %vector for saving minimum point
25 vec_10=zeros(1,N+1);
26 %vector for saving iterations
27 vec_iter_10=zeros(1,N+1);
28
29 for j =1:N+1
30     %applying the function F79 to the 11 starting points
31     tic;
32     [xk_79_10, fk_79_10, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
33         tol, max_iter,delta);
34     %saving results
35     times_10(j)=toc;
36     vec_10(j)=fk_79_10(end);
37     vec_iter_10(j)=n_iter;
38 end
39
40 % Creation of a table with the results
41 results_n10 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
    ...
42                     times_10', vec_10', vec_iter_10', ...
43                     'VariableNames', {'Initial condition', 'Time', 'FinalValue', '
44                                     Iterations'});
45
46 % Computation of mean of the three values saved
47 mean_time = mean(results_n10.Time);
48 mean_final_value = mean(results_n10.FinalValue);
49 mean_iterations = mean(results_n10.Iterations);
50
51 % Insert the mean in the tables
```

```

49 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
50                 'VariableNames', results_n10.Properties.VariableNames);
51 results_n10 = [results_n10; mean_row];
52
53 % Display the table
54 disp(results_n10);
55 % Creation an excel table
56 writetable(results_n10, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_10');
57
58
59
60
61 %% FUNZIONE F79 n=25
62 %the same structure of n=10
63 format long
64 rng(345989);
65 n = 25;
66 tol = 1e-13;
67 max_iter = 1e06;
68 rho = 1.1;
69 mu = 1.9;
70 gamma = 0.8;
71 sigma = 0.9;
72 delta = 0.1;
73
74 F = @(x) F79(x);
75
76 N=10;
77 x0 = ones(n, 1);
78 Mat_points= repmat(x0,1,N+1);
79 rand_mat=2*(rand([n, N+1]) - 0.5);
80 Mat_points=Mat_points + rand_mat;
81 times_25=zeros(1,N+1);
82 vec_25=zeros(1,N+1);
83 vec_iter_25=zeros(1,N+1);
84
85 for j =1:N+1
86     tic;
87     [xk_79_25, fk_79_25, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
88         tol, max_iter,delta);
89     times_25(j)=toc;
90     vec_25(j)=fk_79_25(end);
91     vec_iter_25(j)=n_iter;
92 end
93 results_n25 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
94     ...
95     times_25', vec_25', vec_iter_25', ...
96     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
97
98 mean_time = mean(results_n25.Time);
99 mean_final_value = mean(results_n25.FinalValue);
100 mean_iterations = mean(results_n25.Iterations);
101
102 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
103                 'VariableNames', results_n25.Properties.VariableNames);
104 results_n25 = [results_n25; mean_row];
105
106 disp(results_n25);
107
108 writetable(results_n25, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_25');
109
110
111 %% FUNZIONE F79 n=50
112 % The same structure of n=10
113 format long
114 rng(345989);
115 n = 50;
116 tol = 1e-13;
117 max_iter = 1e06;
118 rho = 1.1;
119 mu = 1.8;

```

```

120 gamma = 0.8;
121 sigma = 0.9;
122 delta = 0.1;
123
124 F = @(x) F79(x);
125
126 N=10;
127 x0 = ones(n, 1);
128 Mat_points= repmat(x0,1,N+1);
129 rand_mat=2*(rand([n, N+1]) - 0.5);
130 Mat_points=Mat_points + rand_mat;
131 times_50=zeros(1,N+1);
132 vec_50=zeros(1,N+1);
133 vec_iter_50=zeros(1,N+1);
134
135 for j =1:N+1
136     tic;
137     [xk_79_50, fk_79_50, n_iter] = Nelder_mead(Mat_points(:,j), F, rho, mu, gamma, sigma,
138         tol, max_iter,delta);
139     times_50(j)=toc;
140     vec_50(j)=fk_79_50(end);
141     vec_iter_50(j)=n_iter;
142 end
143 results_n50 = table(["x0"; "x1"; "x2"; "x3"; "x4"; "x5"; "x6"; "x7"; "x8"; "x9"; "x10"],
144     ...
145     times_50', vec_50', vec_iter_50', ...
146     'VariableNames', {'Problem', 'Time', 'FinalValue', 'Iterations'});
147
148 mean_time = mean(results_n50.Time);
149 mean_final_value = mean(results_n50.FinalValue);
150 mean_iterations = mean(results_n50.Iterations);
151
152 mean_row = table("Mean", mean_time, mean_final_value, mean_iterations, ...
153     'VariableNames', results_n50.Properties.VariableNames);
154
155 results_n50 = [results_n50; mean_row];
156
157 disp(results_n50);
158
159 writetable(results_n50, 'Risultati_F79_Nelder.xlsx', 'Sheet', 'n_50');
160
161 disp('All the results have been saved in Risultati_F79_Nelder.xlsx.');
```

Truncated Newton codes

```

1 function [xk, fk, gradfk_norm, k, xseq, btseq, cgiterseq, convergence_order, flag, converged
2     , violations] = truncated_newton(x0, f, gradf, Hessf, kmax, tolgrad, ftol, cg_maxit,
3     z0, c1, rho, btmax)
4
5 % Function that performs the truncated Newton optimization method, for a
6 % given function f, with backtracking. This version can use both the exact derivatives
7 % and the approximated version.
8
9 %
10 % INPUTS:
11 % x0 = n-dimensional column vector. Initial point;
12 % f = function handle that describes a function R^n->R;
13 % gradf = function handle that describes the gradient of f;
14 % Hessf= function handle that describes the Hessian of f;
15 % kmax = maximum number of iterations permitted;
16 % tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
17 % ftol= function handle of relative tolerance depending on the norm of the gradient (for
18 % conjugate gradient method)
19 % cg_maxit = maximum number of iterations of conjugate gradient method
20 % c1= factor for the Armijo condition in (0,1);
21 % rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
22 % backtracking;
23 % btmax= maximum number of backtracks permitted;
24
25 %
26 % OUTPUTS:
27 % xk = the last x computed by the function;
```

```

22 % fk = the value f(xk);
23 % gradfk_norm = value of the norm of gradf(xk)
24 % k = index of the last iteration performed
25 % xseq = n-by-k matrix where the columns are the elements xk of the sequence
26 % btseq = row vector with the number of backtracks done at every iteration
27 % cgiterseq=
28 % convergence_order = estimated order of convergence
29 % flag= string that says how the method has ended
30 % converged= bool. True if the method has converged
31 % violations=number of violations of positive curvature condition
32
33
34 % Initializations
35 xseq = zeros(length(x0), kmax);
36 cgiterseq = zeros(1, kmax);
37 btseq = zeros(1,kmax);
38 convergence_order=zeros(1,kmax);
39
40 xk = x0; % assigning the initial point
41 k = 0;
42 gradk= gradf(xk); % assigning the initial gradient of f(xk)
43 gradfk_norm = norm(gradk); % assigning the initial gradient norm
44 flag=nan;
45 violations=0;
46
47
48 while k < kmax && gradfk_norm > tolgrad
49
50     %% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%
51     % Hessf(xk)=A, pk=z, -gradk=b
52     A=Hessf(xk); % computing Hessian (if A sparse products with dense vectors will be
                    % dense)
53     % Initialization of zj and j
54     zj = z0;
55     j= 0;
56
57     % Initialization of relative residual and of descent direction
58     res = -gradk - A*zj; % initialize relative residual res=b-Ax
59     p = res; % initialize descent direction
60     norm_b = gradfk_norm; % norm(b) where b=-gradk
61     norm_r = norm(res); % norm of the residual
62
63     while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the
                    % norm of the gradient
64         z = A*p; %product of A and descent direction
65         a = (res'*p)/(p'*z); % update exact step along the direction
66         zj = zj+ a*p; % update solution
67         res = res - a*z; %update residual
68         beta = -(res'*z)/(p'*z);
69         p = res + beta*p; % update descent direction
70
71         sign_curve=sign(p'*A*p);
72         if sign_curve ~= 1 % negative curvature condition p'*A*p <= 0
73             violations =violations+1;
74             break;
75         end
76
77         norm_r = norm(res);
78         j = j+1;
79
80     end
81
82     pk=zj; % descent direction computed (considering the negative curvature condition)
83
84
85     % Backtracking to compute the steplength
86     bt=0;
87     alpha=1; % initial steplenght=1
88     xnew = xk + alpha * pk; % Compute the new value for x with alpha
89     while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
90         alpha=rho*alpha;
91         xnew = xk + alpha * pk; % Compute the new value for x with alpha
92         bt = bt +1;

```

```

93     end
94
95     if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk)) % Break if armijo not satisfied
96         flag='Procedure stopped because the Armijo condition was NOT satisfied';
97         converged=false;
98         break;
99     else
100         xk=xnew;
101     end
102
103     gradk= gradf(xk); % assigning the initial gradient of f(xk)
104     gradfk_norm = norm(gradk); % assigning the initial gradient norm
105     k = k + 1; % Increase the step by one
106
107     xseq(:, k) = xk; % Store current xk in xseq
108     btseq(k)=bt; % Store number of backtracking iterations
109     cgiterseq(k)=j; % Store conjugate gradient iterations in pcgiterseq
110     if k>3
111         convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1)))/log(norm(xseq(:, k-1)-xseq(:, k-2)))/log(norm(xseq(:, k-2)-xseq(:, k-3)));
112     end
113 end
114
115 if isnan(flag)
116     if k==kmax && gradfk_norm > tolgrad
117         flag='Procedure stopped because the maximum number of iterations was reached';
118         converged=false;
119     else
120         flag=['Procedure stopped in ', num2str(k), ' steps, with gradient norm ', num2str(
            gradfk_norm)];
121         converged=true;
122     end
123 end
124 fk = f(xk); % Compute f(xk)
125
126 xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
127 xseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
128 btseq = btseq(1:k); % "Cut" btseq to the correct size
129 cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
130 convergence_order=convergence_order(1:k); % "Cut" convergence order
131
132
133 end

```

```

1 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)
2
3 % Function that performs the truncated Newton optimization method, for a
4 % given function f, with backtracking. This version can use both the exact derivatives
5 % and the approximated version.
6
7 % INPUTS:
8 % x0 = n-dimensional column vector. Initial point;
9 % f = function handle that describes a function R^n->R;
10 % gradf = function handle that describes the gradient of f;
11 % Hessf= function handle that describes the Hessian of f;
12 % kmax = maximum number of iterations permitted;
13 % tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
14 % ftol= function handle of relative tolerance depending on the norm of the gradient (for
    conjugate gradient method)
15 % cg_maxit = maximum number of iterations of conjugate gradient method
16 % c1= factor for the Armijo condition in (0,1);
17 % rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
    backtracking;
18 % btmax= maximum number of backtracks permitted;
19
20 % OUTPUTS:
21 % xk = the last x computed by the function;
22 % fk = the value f(xk);
23 % gradfk_norm = value of the norm of gradf(xk)
24 % k = index of the last iteration performed
25 % 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

```

```

26 % cgiterseq=
27 % convergence_order = estimated order of convergence
28 % flag= string that says how the method has ended
29 % converged= bool. True if the method has converged
30 % violations=number of violations of positive curvature condition
31
32
33 % Initializations
34 xseq = zeros(length(x0), kmax);
35 cgiterseq = zeros(1, kmax);
36 btseq = zeros(1,kmax);
37 convergence_order=zeros(1,kmax);
38
39 xk = x0; % assigning the initial point
40 k = 0;
41 gradk= gradf(xk); % assigning the initial gradient of f(xk)
42 gradfk_norm = norm(gradk); % assigning the initial gradient norm
43 flag=nan;
44
45 violations=0;
46
47 while k < kmax && gradfk_norm > tolgrad
48   %%% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%%
49   % Hessf(xk)=A, pk=z, -gradk=b
50   A=Hessf(xk); % computing Hessian (if A sparse products with dense vectors will be
51   % dense)
52   % Initialization of zj and j
53   zj = z0;
54   j= 0;
55   %Initialization of relative residuals and of decent direction
56   res = A*zj+gradk ; % initialize relative residual res=b-Ax
57
58   D = diag(diag(A)); % Diagonal Matrix (D)
59   L = tril(A, -1); % Triangula inferior (L)
60   M=D+L; % Preconditioning Matrix M
61
62   y=M\res;
63   p = -y; % initialize descent direction
64
65   norm_b = gradfk_norm; % norm(b) where b=-gradk
66   norm_r = norm(res); % norm of the residual
67
68
69   while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the
70   % norm of the gradient
71   z = A*p; %product of A and descent direction
72   a = (res'*y)/(p'*z); % update exact step along the direction
73   zj = zj+ a*p; % update solution
74   res1 = res + a*z; %update residual
75
76   %solve the system Myk+1=rk+1
77   y1=M\res1;
78
79   beta = (res1'*y1)/(res'*y);
80   p = -y1 + beta*p; % update descent direction
81
82   sign_curve=sign(p'*A*p);
83   if sign_curve ~= 1 % negative curvature condition p'*A*p <= 0
84     violations =violations+1;
85     break;
86   end
87
88   res=res1;
89   y=y1;
90
91   norm_r = norm(res);
92   j = j+1;
93
94 end
95
96 pk=zj; % descent direction computed (considering the negative curvature condition)

```



```

97
98 % Backtracking to compute the steplength
99 bt=0;
100 alpha=1; % initial steplength=1
101 xnew = xk + alpha * pk; % Compute the new value for x with alpha
102 while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
103     alpha=rho*alpha;
104     xnew = xk + alpha * pk; % Compute the new value for x with alpha
105     bt = bt +1;
106 end
107
108 if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk)) % Break if armijo not satisfied
109     flag='Procedure stopped because the Armijo condition was NOT satisfied';
110     converged=false;
111     break;
112 else
113     xk=xnew;
114 end
115
116 gradk= gradf(xk); % assigning the initial gradient of f(xk)
117 gradfk_norm = norm(gradk); % assigning the initial gradient norm
118 k = k + 1; % Increase the step by one
119
120 xseq(:, k) = xk; % Store current xk in xseq
121 btseq(k)=bt; % Store number of backtracking iterations
122 cgiterseq(k)=j; % Store conjugate gradient iterations in pcgiterseq
123 if k>3
124     convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1))/norm(xseq(:, k-1)-xseq(:,
125         k-2)))/log(norm(xseq(:, k-1)-xseq(:, k-2))/norm(xseq(:, k-2)-xseq(:, k-3)));
126 end
127 end
128
129 if isnan(flag)
130     if k==kmax && gradfk_norm > tolgrad
131         flag='Procedure stopped because the maximum number of iterations was reached';
132         converged=false;
133     else
134         flag=['Procedure stopped in ', num2str(k), ' steps, with gradient norm ', num2str(
135             gradfk_norm)];
136         converged=true;
137     end
138 end
139
140 fk = f(xk); % Compute f(xk)
141
142 xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
143 xseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
144 btseq = btseq(1:k); % "Cut" btseq to the correct size
145 cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
146 convergence_order=convergence_order(1:k); % "Cut" convergence order
147
148 end

```

```

1 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)
2 % Function that performs the truncated Newton optimization method, for for function F27,
with backtracking.
3 % INPUTS:
4 % x0 = n-dimensional column vector. Initial point;
5 % f = function handle that describes a function R^n->R;
6 % gradf = function handle that describes the gradient of f;
7 % exact = bool. True if exact version of the hessian. False= approximated version with
finite differences
8 % h= increment for finite differences. if exact=true put h=0.
9 % kmax = maximum number of iterations permitted;
10 % tolgrad = value used as stopping criterion w.r.t. the norm of the gradient
11 % ftol= function handle of relative tolerance depending on the norm of the gradient (for
conjugate gradient method)
12 % cg_maxit = maximum number of iterations of conjugate gradient method
13 % c1= factor for the Armijo condition in (0,1);
14 % rho= fixed (for simplicity) factor less than 1 used to reduce alpha in
15 % backtracking;

```

```

16 % btmax= maximum number of backtracks permitted;
17 %
18 % OUTPUTS:
19 % xk = the last x computed by the function;
20 % fk = the value f(xk);
21 % gradfk_norm = value of the norm of gradf(xk)
22 % k = index of the last iteration performed
23 % xseq = n-by-k matrix where the columns are the elements xk of the sequence
24 % btseq = row vector with the number of backtracks done at every iteration
25 % cgiterseq=
26 % convergence_order = estimated order of convergence
27 % flag= string that says how the method has ended
28 % converged= bool. True if the method has converged
29 % violations=number of violations of positive curvature condition
30
31 % Initializations
32 xseq = zeros(length(x0), kmax);
33 cgiterseq = zeros(1, kmax);
34 btseq = zeros(1,kmax);
35 convergence_order=zeros(1,kmax);
36
37 xk = x0; % assigning the initial point
38 k = 0;
39 gradk= gradf(xk); % assigning the initial gradient of f(xk)
40 gradfk_norm = norm(gradk); % assigning the initial gradient norm
41 flag=nan;
42
43 violations=0;
44
45 while k < kmax && gradfk_norm > tolgrad
46
47     %% Compute pk solving Hessf(xk)pk=-gradk with Coniugate Gradient method. %%
48     % Hessf(xk)=A, pk=z, -gradk=b
49     % Hessf27(x)(i,j)= 4*x_i*x_j
50     % Hessf27(x)(i,i)= (2/100000 -1+ 4*(sum(x.^2)) + 8*x_i^2)/2
51     % the matrix is NOT sparse BUT with large n cannot be stored. So we
52     % compute directly the matrix vector products.
53     % EXACT VERSION: Hessf27(x)*z= 4*s*v1 -4*v2 +v3 with s=sum(x), v1=x.*z, v2= (x.^2)
54     % .*z, v3=diag(Hessf27(x)).*z
55     % diag(Hessf27(x))= (2/100000 -1+ 4*s + 8*x.^2)/2;
56     % APPROXIMATED VERSION: Hessf27_approx(x,h)*z= Hessf27(x)*z + 2*n*(h^2)*z
57
58     %with finite difference 1: Hessf27(x)(i,j)= 4*x_i*x_j + h^2 +2hx_j+2hx_i
59     %APPROXIMATED VERSION: Hessf27(x)(i,i)= (2/100000 -1+ 4*(sum(x.^2)) + 8*x_i^2 + 2*h
60     %^2)/2
61     %APPROXIMATED VERSION: Hessf27_approx(x,h)*z= Hessf27(x)*z +(h^2)*sum_z*ones(n,1) -
62     % (h^2)*z + 2*h*sum_z*x + 2*h*(x'*z)-4*h*(x.*z)
63
64     %with finite difference 2: Hessf27(x)(i,j)= 4*x_i*x_j*mod(x_i)*mod(x_j)
65     %+ h^2*mod(x_i)^2*mod(x_j)^2
66     %+2hx_j*mod(x_i)^2*mod(x_j)+2hx_i*mod(x_j)^2*mod(x_i)
67
68     diagA=(2/100000 -1+ 4*sum(xk.^2) + 8*xk.^2)/2;
69
70     % Initialization of zj and j
71     zj = z0;
72     j= 0;
73
74     % Initialization of relative residual and of descent direction
75     Azj= 4*(xk'*zj)*xk-4*(xk.^2).*zj+diagA.*zj; % A*zj
76     sum_z=sum(zj);
77     if ~exact %approximation with finite difference (not exact)
78         if fin_dif_2
79             Azj= (diagA.*zj +(h^2.*abs(xk).^2)) +( h^2*abs(xk).*(abs(xk)'*zj) - h^2*abs(
80                 xk).^2.*zj )+(2*h*xk.*(abs(xk)'*zj) - 4*h*xk.*abs(xk).*zj) +(2*h*abs(xk)
81                 .*(xk'*zj)) + (4*xk.*(xk'*zj)-4*(xk.^2).*zj);
82         else
83             Azj=Azj+ (h^2)*sum_z*ones(length(x0),1) - (h^2)*zj + 2*h*sum_z*xk + 2*h*(xk'*
84                 zj)-4*h*(xk.*zj);
85         end
86     end
87
88     res = -gradk - Azj; % initialize relative residual res=b-Ax

```

```

83 p = res; % initialize descent direction
84 norm_b = gradfk_norm; % norm(b) where b=-gradk
85 norm_r = norm(res); % norm of the residual
86
87 while (j<cg_maxit && norm_r>ftol(j,norm_b)*norm_b ) %adaptive tolerance based on the
    norm of the gradient
88     z= 4*(xk'*p)*xk-4*(xk.^2).*p+diagA.*p; % A*p : product of A and descent direction
89     sum_p=sum(p);
90     if ~exact %approximation with finite difference (not exact)
91
92         if fin_dif_2
93             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);
94
95         else
96             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);
97
98         end
99     end
100     a = (res'*p)/(p'*z); % update exact step along the direction
101     zj = zj+ a*p; % update solution
102     res = res - a*z; %update residual
103     beta = -(res'*z)/(p'*z);
104     p = res + beta*p; % update descent direction
105
106
107     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
108     sum_p=sum(p);
109     if ~exact %approximation with finite difference (not exact)
110         if fin_dif_2
111             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))';
112
113         else
114             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))';
115
116         end
117     end
118     sign_curve=sign(z_new*p);
119     if sign_curve ~= 1 % negative curvature condition p'*A*p <= 0
120         violations =violations+1;
121         break;
122     end
123
124     norm_r = norm(res);
125     j = j+1;
126
127 end
128
129 pk=zj; % descent direction computed (considering the negative curvature condition)
130
131 % Backtracking to compute the steplength
132 bt=0;
133 alpha=1; % initial steplength=1
134 xnew = xk + alpha * pk; % Compute the new value for x with alpha
135 while bt<btmax && (f(xnew)>(f(xk)+c1*alpha*(gradk'*pk))) % Armijo condition
136     alpha=rho*alpha;
137     xnew = xk + alpha * pk; % Compute the new value for x with alpha
138     bt = bt +1;
139 end
140
141 if bt==btmax && f(xnew)>(f(xk)+c1*alpha*(gradk'*pk)) % Break if armijo not satisfied
142     flag='Procedure stopped because the Armijo condition was NOT satisfied';
143     converged=false;
144     break;
145 else
146     xk=xnew;
147 end

```

```

148     gradk= gradf(xk); % assigning the initial gradient of f(xk)
149     gradfk_norm = norm(gradk); % assigning the initial gradient norm
150     k = k + 1; % Increase the step by one
151
152     xseq(:, k) = xk; % Store current xk in xseq
153     btseq(k)=bt; % Store number of backtracking iterations
154     cgiterseq(k)=j; % Store conjugate gradient iterations in pcgiterseq
155     if k>3
156         convergence_order(k)=log(norm(xseq(:, k)-xseq(:, k-1))/norm(xseq(:, k-1)-xseq(:,
157             k-2)))/log(norm(xseq(:, k-1)-xseq(:, k-2))/norm(xseq(:, k-2)-xseq(:, k-3)));
158     end
159 end
160 if isnan(flag)
161     if k==kmax && gradfk_norm > tolgrad
162         flag='Procedure stopped because the maximum number of iterations was reached';
163         converged=false;
164     else
165         flag=['Procedure stopped in ', num2str(k), ' steps, with gradient norm ', num2str(
166             gradfk_norm)];
167         converged=true;
168     end
169 end
170 fk = f(xk); % Compute f(xk)
171
172 xseq = xseq(:, 1:k); % "Cut" xseq to the correct size
173 btseq = [x0, xseq]; % "Add" x0 at the beginning of xseq (otherwise the first el. is x1)
174 btseq = btseq(1:k); % "Cut" btseq to the correct size
175 cgiterseq = cgiterseq(1:k); % "Cut" cgiterseq to the correct size
176 convergence_order=convergence_order(1:k); % "Cut" convergence order
177
178 end

```

```

1 %% ROSENBROCK FUNCTION
2 addpath('C:\Users\sofia\Documents\Numerical-Optimization')
3 rng(345989);
4
5 f_Rosen = @(x) 100*(x(2,:)-x(1,:).^2).^2+(1-x(1,:)).^2 ;
6 gradf_Rosen= @(x) [400*x(1,:).^3+(2-400*x(2,:)).*x(1,:)-2;
7     200*(x(2,:)-x(1,:).^2)];
8 Hessf_Rosen=@(x) [1200*x(1,:).^2-400*x(2,:)+2, -400*x(1,:);
9     -400*x(1,:), 200];
10 x0_a=[1.2;1.2];
11 x0_b=[-1.2;1];
12
13 load forcing_terms.mat
14
15 %% TEST of Truncated Newton with x0=[1.2;1.2]
16
17 kmax=500;
18 tolgrad=5e-7;
19 cg_maxit=50;
20
21 z0=zeros(2,1);
22 c1=1e-4;
23 rho=0.5;
24 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
25
26 % Superlinear term of convergence
27 tic
28 [x1, f1, gradf_norm1, k1, xseq1, btseq1, cgiterseq1, conv_ord1, flag1, converged1, violations1
29     ] = truncated_newton(x0_a, f_Rosen, gradf_Rosen, Hessf_Rosen, kmax, tolgrad,
30         fterms_suplin, cg_maxit, z0, c1, rho, btmax);
31 time1=toc;
32 disp('Test on Rosenbrock function with x0=[1.2;1.2] and superlinear term for the adaptive
33     tolerance: ');
34 disp(flag1)
35 disp(['Elapsed time: ', num2str(time1)])
36 disp(['Function value in the point found: ', num2str(f1)])
37 disp(['Number of violations of curvature condition: ', num2str(violations1)])
38 last_bt1=sum(btseq1)/k1;
39 last_cg1=sum(cgiterseq1)/k1;

```

```

37
38 %plot
39 plot_iterative_optimization_results(f_Rosen,xseq1, btseq1);
40 figure;
41 hold on
42 plot(1:length(conv_ord1), conv_ord1, 'Color', 'b', 'LineWidth', 1.5)
43 title('Rosenbrock_truncated_Newton_method[1.2,1.2]_superlinear');
44 hold off;
45
46 % Quadratic term of convergence
47 tic
48 [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);
49 time2=toc;
50 disp('Test_on_Rosenbrock_function_with_x0=[1.2;1.2]_and_quadratic_term_for_the_adaptive_
    tolerance: ')
51 disp(flag2)
52 disp(['Elapsed_time:', num2str(time2)])
53 disp(['Function_value_in_the_point_found:', num2str(f2)])
54 disp(['Number_of_violations_of_curvature_condition:', num2str(violations2)])
55 last_bt2=sum(btseq2)/k2;
56 last_cg2=sum(cgiterseq2)/k2;
57
58 %plot
59 plot_iterative_optimization_results(f_Rosen,xseq2, btseq2);
60 figure;
61 hold on
62 plot(1:length(conv_ord2), conv_ord2, 'Color', 'b', 'LineWidth', 1.5)
63 title('Rosenbrock_truncated_Newton_method[1.2,1.2]_quadratic');
64 hold off;
65
66
67 %% TEST of Trucated Newton with x0=[-1.2;1]
68
69 kmax=500;
70 tolgrad=5e-7;
71 cg_maxit=50;
72
73 z0=zeros(2,1);
74 c1=1e-4;
75 rho=0.5;
76 btmax=50;
77 % rho=0.8;
78 % btmax=155;
79
80 % Superlinear term of convergence
81 tic
82 [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);
83 time3=toc;
84 disp('Test_on_Rosenbrock_function_with_x0=[-1.2;1]_and_superlinear_term_for_the_adaptive_
    tolerance: ')
85 disp(flag3)
86 disp(['Elapsed_time:', num2str(time3)])
87 disp(['Function_value_in_the_point_found:', num2str(f3)])
88 disp(['Number_of_violations_of_curvature_condition:', num2str(violations3)])
89 last_bt3=sum(btseq3)/k3;
90 last_cg3=sum(cgiterseq3)/k3;
91
92 %plot
93 plot_iterative_optimization_results(f_Rosen,xseq3, btseq3);
94 figure;
95 hold on
96 plot(1:length(conv_ord3), conv_ord3, 'Color', 'b', 'LineWidth', 1.5)
97 title('Rosenbrock_truncated_Newton_method[-1.2,1]_superlinear');
98 hold off;
99
100 % Quadratic term of convergence
101 tic
102 [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,

```

```

103     fterms_quad, cg_maxit,z0, c1, rho, btmax);
104 time4=toc;
105 disp('Test on Rosenbrock function with x0=[-1.2;1] and quadratic term for the adaptive
106     tolerance: ')
107 disp(flag4)
108 disp(['Elapsed time:', num2str(time4)])
109 disp(['Function value in the point found:', num2str(f4)])
110 disp(['Number of violations of curvature condition:', num2str(violations4)])
111 last_bt4=sum(btseq4)/k4;
112 last_cg4=sum(cgiterseq4)/k4;
113
114 %plot
115 figure;
116 hold on
117 plot(1:length(conv_ord4), conv_ord4, 'Color', 'b', 'LineWidth', 1.5)
118 title('Rosenbrock truncated Newton method [-1.2,1] quadratic');
119 hold off;
120 plot_iterative_optimization_results(f_Rosen,xseq4, btseq4);
121
122 %% Table with results
123 results_table = table({'[1.2;1.2]'; '[1.2;1.2]'; '[-1.2;1]'; '[-1.2;1]'}, ...
124     {'Superlineare'; 'Quadratica'; 'Superlineare'; 'Quadratica'}, ...
125     [f1; f2; f3; f4], [k1; k2; k3; k4], ...
126     [time1; time2; time3; time4], ...
127     [violations1; violations2; violations3; violations4],...
128     [last_cg1; last_cg2; last_cg3; last_cg4],...
129     [last_bt1; last_bt2; last_bt3; last_bt4],...
130     'VariableNames', {'Starting point', 'Forcing terms', 'f_x', 'Number
131         of iterations', 'Executing time', 'Violation of curvature
132         conditions', 'Average of cg iterations', 'Average of bt
133         iterations'});
134 writetable(results_table, 'rosenbrock_truncated.xlsx','WriteRowNames', true);

```

```

1 %% FUNCTION 79 (with different initial points)- with exact derivatives and finite
2     differences
3
4 sparse=true;
5
6 F = @(x) F79(x); % Defining F79 as function handle
7 JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
8     handle
9 HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
10     function handle (sparse version)
11
12 load forcing_terms.mat % possible terms for adaptive tolerance
13
14 %% n=10^3 (1e3)
15
16 rng(345989);
17
18 n=1e3;
19
20 kmax=1.5e3; % maximum number of iterations of Newton method
21 tolgrad=5e-7; % tolerance on gradient norm
22
23 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
24     system)
25 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
26
27 % Backtracking parameters
28 c1=1e-4;
29 rho=0.50;
30 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
31
32 x0=-1*ones(n,1); % initial point
33 N=10; % number of initial points to be generated
34
35 % Initial points:
36 Mat_points= repmat(x0,1,N+1);
37 rand_mat=2*rand(n, N)-1;
38 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points

```

```

35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
41 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
42 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
43 mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
44 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
46
47 JF_ex = @(x) JF_gen(x,true,false,0);
48 HF_ex = @(x) HF_gen(x,true,false,0);
49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv1_fd1=cell(6, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv1_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1
79     disp(['Condizione_iniziale_n.',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83
84     [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);
85
86     vec_times1_ex(j)=toc;
87
88     disp(['Exact_derivatives:',flag1])
89     vec_converged1_ex(j)=converged1;
90     vec_val1_ex(j)=f1;
91     vec_grad1_ex(j)=gradf_norm1;
92     vec_iter1_ex(j)=k1;
93     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
94     vec_bt1_ex(j)=sum(btseq1)/k1;
95     vec_violations1_ex(j)=violations1;
96     last_vals = conv_ord1_ex(max(end-11,1):end);
97     mat_conv1_ex(:, j) = last_vals;
98

```

```

99
100     for i=2:2:12
101         h=10-(i);
102
103         % FINITE DIFFERENCES 1
104         JF=@(x) JF_fd1(x,h);
105         HF=@(x) HF_fd1(x,h);
106         tic;
107
108         [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);
109
110         mat_times1_fd1(i/2,j)=toc;
111
112         disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag1])
113         mat_converged1_fd1(i/2,j)=converged1;
114         mat_val1_fd1(i/2,j)=f1;
115         mat_grad1_fd1(i/2,j)=gradf_norm1;
116         mat_iter1_fd1(i/2,j)=k1;
117         mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
118         mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
119         mat_violations1_fd1(i/2,j)=violations1;
120         last_vals = conv_ord1_df1(max(end-11,1):end);
121         mat_conv1_fd1(i/2, j) = {last_vals};
122
123
124         % FINITE DIFFERENCES 2
125         JF=@(x) JF_fd2(x,h);
126         HF=@(x) HF_fd2(x,h);
127         tic;
128
129         [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);
130
131         mat_times1_fd2(i/2,j)=toc;
132
133         disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
134         mat_converged1_fd2(i/2,j)=converged1;
135         mat_val1_fd2(i/2,j)=f1;
136         mat_grad1_fd2(i/2,j)=gradf_norm1;
137         mat_iter1_fd2(i/2,j)=k1;
138         mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
139         mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
140         mat_violations1_fd2(i/2,j)=violations1;
141         last_vals = conv_ord1_df2(max(end-11,1):end);
142         mat_conv1_fd2(i/2, j) = {last_vals};
143
144     end
145 end
146
147
148 %% Plot of the last 12 values of experimentale rate of convergence
149 num_initial_points = N + 1;
150 figure;
151 hold on;
152
153 % Plot for every initial condition
154 for j = 1:num_initial_points
155     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
156     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
157     hold on;
158     for i =1:6
159         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
160         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
161         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
162         hold on;
163         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
164         hold on;
165     end
166 end
167

```



```

168 % title and legend
169 title('F79_10^3_superlinear');
170 xlabel('Iterazione');
171 ylabel('Ordine di Convergenza');
172 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
173 grid on;
174 hold off;
175
176
177 %% Execution Time
178
179 % Exact Derivative
180 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
181 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
182 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
183
184 % FD1
185 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
186 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
187 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
188
189 % FD2
190 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
191 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
192 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
193
194 % Creation of the labels
195 h_exponents = [2, 4, 6, 8, 10, 12];
196 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
197
198 fd1_vals = avg_fd1';
199 fd2_vals = avg_fd2';
200
201 % Table construction with exact for both the row
202 rowNames = {'FD1', 'FD2'};
203 columnNames = [h_labels, 'Exact'];
204 data = [fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
205 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
206
207 % visualization
208 disp('Average computation times table (only for successful runs): F79, n=10^3,
    superlinear');
209 disp(T1);
210
211
212 %% All the tables has the same structure
213 %% Iteration
214
215 vec_times_ex_clean = vec_iter1_ex;
216 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
217 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
218
219 mat_times_fd1_clean = mat_iter1_fd1;
220 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
221 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
222
223 mat_times_fd2_clean = mat_iter1_fd2;
224 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
225 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
226
227 h_exponents = [2, 4, 6, 8, 10, 12];
228 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
229
230 fd1_vals = avg_fd1';
231 fd2_vals = avg_fd2';
232
233 rowNames = {'FD1', 'FD2'};
234 columnNames = [h_labels, 'Exact'];
235 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];
236
237 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

```

```

238
239 disp('Average computation iteration table (only for successful runs):_F79,n=10^3,suplin
    ');
240 disp(T2);
241
242 %% F value
243
244 vec_times_ex_clean = vec_val1_ex;
245 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
246 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
247
248 mat_times_fd1_clean = mat_val1_fd1;
249 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
250 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
251
252 mat_times_fd2_clean = mat_val1_fd2;
253 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
254 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
255
256 h_exponents = [2, 4, 6, 8, 10, 12];
257 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
258
259 fd1_vals = avg_fd1';
260 fd2_vals = avg_fd2';
261
262 rowNames = {'FD1', 'FD2'};
263 columnNames = [h_labels, 'Exact'];
264 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1];
265
266 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
267
268 disp('Average computation fmin value table (only for successful runs):_F79,n=10^3,
    suplin');
269 disp(T3);
270
271 %% VIOLATION
272
273 vec_times_ex_clean = vec_violations1_ex;
274 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
275 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
276
277 mat_times_fd1_clean = mat_violations1_fd1;
278 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
279 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
280
281 mat_times_fd2_clean = mat_violations1_fd2;
282 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
283 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
284
285 h_exponents = [2, 4, 6, 8, 10, 12];
286 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
287
288 %
289 fd1_vals = avg_fd1';
290 fd2_vals = avg_fd2';
291
292 rowNames = {'FD1', 'FD2'};
293 columnNames = [h_labels, 'Exact'];
294 data = [fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1];
295
296 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
297
298 disp('Average computation violation table (only for successful runs):_F79,n=10^3,
    superlinear');
299 disp(T10);
300
301
302 %% BT-SEQ
303 vec_bt_ex_clean = vec_bt1_ex;
304 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
305 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
306
307 mat_bt_fd1_clean = mat_bt1_fd1;

```

```

308 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
309 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
310
311 mat_bt_fd2_clean = mat_bt1_fd2;
312 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
313 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
314
315 h_exponents = [2, 4, 6, 8, 10, 12];
316 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
317
318 fd1_vals = avg_fd1';
319 fd2_vals = avg_fd2';
320
321 rowNames = {'FD1', 'FD2'};
322 columnNames = [h_labels, 'Exact'];
323 data = [fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1];
324
325 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
326
327 disp('Average computation bt iteration table (only for successful runs): F79, n=10^3,
superlinear');
328 disp(T11);
329
330 %% CG-SEQ
331
332 vec_bt_ex_clean = vec_cg_iter1_ex;
333 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
334 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
335
336 mat_bt_fd1_clean = mat_cg_iter1_fd1;
337 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
338 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
339
340 mat_bt_fd2_clean = mat_cg_iter1_fd2;
341 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
342 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
343
344 h_exponents = [2, 4, 6, 8, 10, 12];
345 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
346
347 fd1_vals = avg_fd1';
348 fd2_vals = avg_fd2';
349
350 rowNames = {'FD1', 'FD2'};
351 columnNames = [h_labels, 'Exact'];
352 data = [fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1];
353
354 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
355
356 disp('Average computation cg iteration table (only for successful runs): F79, n=10^3,
superlinear');
357 disp(T12);
358
359 %% Number of starting point converged
360
361 h_exponents = [2, 4, 6, 8, 10, 12];
362 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
363
364 fd1_vals = sum(mat_converged1_fd1, 2)';
365 fd2_vals = sum(mat_converged1_fd2, 2)';
366
367 rowNames = {'FD1', 'FD2'};
368 columnNames = [h_labels, 'Exact'];
369 data = [fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex)];
370
371 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
372
373 disp('Number of converged: F79, n=10^3, superlinear');
374 disp(T13);
375 %save the table in a file.xlsx
376 writetable(T1, 'results_f79_suplin.xlsx', 'Sheet', 'time_3', 'WriteRowNames', true);
377 writetable(T2, 'results_f79_suplin.xlsx', 'Sheet', 'niter_3', 'WriteRowNames', true);
378 writetable(T3, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_3', 'WriteRowNames', true);

```

```

379 writetable(T10, 'results_f79_suplin.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
380 writetable(T11, 'results_f79_suplin.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
381 writetable(T12, 'results_f79_suplin.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
382 writetable(T13, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);
383
384 %% n=10^4 (1e4)
385
386 rng(345989);
387
388 n=1e4;
389
390 kmax=1.5e3; % maximum number of iterations of Newton method
391 tolgrad=5e-7; % tolerance on gradient norm
392
393 cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
394 system)
395 z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)
396
397 % Backtracking parameters
398 c1=1e-4;
399 rho=0.50;
400 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
401
402 x0=-1*ones(n,1); % initial point
403 N=10; % number of initial points to be generated
404
405 % Initial points:
406 Mat_points=repmat(x0,1,N+1);
407 rand_mat=2*rand(n, N)-1;
408 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
409
410 % Structure for EXACT derivatives
411 vec_times2_ex=zeros(1,N+1); % vector with execution times
412 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
413 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
414 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
415 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
416 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
417 mat_conv2_ex=zeros(12,N+1); %matrix with che last 12 values of rate of convergence for
the starting point
418 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
419 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
condition in Newton method
420
421 JF_ex = @(x) JF_gen(x,true,false,0);
422 HF_ex = @(x) HF_gen(x,true,false,0);
423
424 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
425 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
426 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
427 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
428 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
429 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
430 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
431 mat_conv2_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
starting point
432 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
433 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
condition in Newton method
434
435 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
436 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
437
438 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
x_j) as increment)
439 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
440 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
441 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
442 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
443 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
444 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
445 mat_conv2_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the

```

```

    starting point
446 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
447 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
448
449 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
450 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
451
452 for j =1:N+1
453     disp(['Condizione_iniziale_n.',num2str(j)])
454
455     % EXACT DERIVATIVES
456     tic;
457     [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);
458     vec_times2_ex(j)=toc;
459
460     disp(['Exact_derivatives:',flag2])
461     vec_converged2_ex(j)=converged2;
462     vec_val2_ex(j)=f2;
463     vec_grad2_ex(j)=gradf_norm2;
464     vec_iter2_ex(j)=k2;
465     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
466     vec_bt2_ex(j)=sum(btseq2)/k2;
467     vec_violations2_ex(j)=violations2;
468     last_vals = conv_ord2_ex(max(end-11,1):end);
469     mat_conv2_ex(:, j) = last_vals;
470
471
472     for i=2:2:12
473         h=10^(-i);
474
475         % FINITE DIFFERENCES 1
476         JF=@(x) JF_fd1(x,h);
477         HF=@(x) HF_fd1(x,h);
478         tic;
479         [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);
480         mat_times2_fd1(i/2,j)=toc;
481
482         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag2])
483         mat_converged2_fd1(i/2,j)=converged2;
484         mat_val2_fd1(i/2,j)=f2;
485         mat_grad2_fd1(i/2,j)=gradf_norm2;
486         mat_iter2_fd1(i/2,j)=k2;
487         mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
488         mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
489         mat_violations2_fd1(i/2,j)=violations2;
490         last_vals = conv_ord2_df1(max(end-11,1):end);
491         mat_conv2_fd1(i/2, j) = {last_vals};
492
493
494         % FINITE DIFFERENCES 2
495         JF=@(x) JF_fd2(x,h);
496         HF=@(x) HF_fd2(x,h);
497         tic;
498         [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);
499         mat_times2_fd2(i/2,j)=toc;
500
501         disp(['Finite_differences_(new_version)_with_h=1e-',num2str(i),'_',flag2])
502         mat_converged2_fd2(i/2,j)=converged2;
503         mat_val2_fd2(i/2,j)=f2;
504         mat_grad2_fd2(i/2,j)=gradf_norm2;
505         mat_iter2_fd2(i/2,j)=k2;
506         mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
507         mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
508         mat_violations2_fd2(i/2,j)=violations2;
509         last_vals = conv_ord2_df2(max(end-11,1):end);
510         mat_conv2_fd2(i/2, j) = {last_vals};

```

```

511
512
513     end
514 end
515
516
517 %% The Plot has the same structure
518 num_initial_points = N + 1;
519 figure;
520 hold on;
521
522 for j = 1:num_initial_points
523     conv_ord_ex = mat_conv2_ex(:,j);
524     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
525     hold on;
526     for i = 1:6
527         conv_ord_fd1 = mat_conv2_fd1{i, j};
528         conv_ord_fd2 = mat_conv2_fd2{i, j};
529         plot(1:12,conv_ord_fd1, '-','Color', 'r', 'LineWidth', 1.5);
530         hold on;
531         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
532         hold on;
533     end
534 end
535
536 title('F79_10^4_superlinear');
537 xlabel('Iterazione');
538 ylabel('Ordine di Convergenza');
539 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
540 grid on;
541 hold off;
542
543
544 %% Execution time
545
546 % Exact derivative
547 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
548 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
549 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
550
551 % FD1
552 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
553 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
554 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
555
556 % FD2
557 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
558 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
559 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
560
561 % Creation of the labels
562 h_exponents = [2, 4, 6, 8, 10, 12];
563 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
564
565 fd1_vals = avg_fd1';
566 fd2_vals = avg_fd2';
567
568 % Table creation
569 rowNames = {'FD1', 'FD2'};
570 columnNames = [h_labels, 'Exact'];
571 data = [fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2];
572 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
573 %display the table
574 disp('Average computation times table (only for successful runs): F79, n=10^4,
    superlinear');
575 disp(T4);
576
577 %% Iteration
578
579 vec_times_ex_clean = vec_iter2_ex;
580 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;

```

```

581 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
582
583 mat_times_fd1_clean = mat_iter2_fd1;
584 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
585 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
586
587 mat_times_fd2_clean = mat_iter2_fd2;
588 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
589 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
590
591 h_exponents = [2, 4, 6, 8, 10, 12];
592 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
593
594 fd1_vals = avg_fd1';
595 fd2_vals = avg_fd2';
596
597 rowNames = {'FD1', 'FD2'};
598 columnNames = [h_labels, 'Exact'];
599 data = [fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2];
600
601 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
602
603 disp('Average computation iteration table (only for successful runs): F79, n=10^4,
604      superlinear');
605 disp(T5);
606
607 %% Function value
608
609 vec_times_ex_clean = vec_val2_ex;
610 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
611 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
612
613 mat_times_fd1_clean = mat_val2_fd1;
614 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
615 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
616
617 mat_times_fd2_clean = mat_val2_fd2;
618 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
619 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
620
621 h_exponents = [2, 4, 6, 8, 10, 12];
622 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
623
624 fd1_vals = avg_fd1';
625 fd2_vals = avg_fd2';
626
627 rowNames = {'FD1', 'FD2'};
628 columnNames = [h_labels, 'Exact'];
629 data = [fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2];
630
631 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
632
633 disp('Average computation fmin value table (only for successful runs): F79, n=10^4,
634      superlinear');
635 disp(T6);
636
637 %% VIOLATION
638
639 vec_times_ex_clean = vec_violations2_ex;
640 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
641 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
642
643 mat_times_fd1_clean = mat_violations2_fd1;
644 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
645 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
646
647 mat_times_fd2_clean = mat_violations2_fd2;
648 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
649 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
650
651 h_exponents = [2, 4, 6, 8, 10, 12];
652 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```

652 fd1_vals = avg_fd1';
653 fd2_vals = avg_fd2';
654
655 rowNames = {'FD1', 'FD2'};
656 columnNames = [ h_labels, 'Exact'];
657 data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];
658
659 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
660
661 disp('Average computation violation table (only for successful runs): F79, n=10^4,
suplinear');
662 disp(T14);
663
664 %% BT-SEQ
665
666 vec_bt_ex_clean = vec_bt2_ex;
667 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
668 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
669
670 mat_bt_fd1_clean = mat_bt2_fd1;
671 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
672 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
673
674 mat_bt_fd2_clean = mat_bt2_fd2;
675 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
676 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
677
678 h_exponents = [2, 4, 6, 8, 10, 12];
679 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
680
681 fd1_vals = avg_fd1';
682 fd2_vals = avg_fd2';
683
684 rowNames = {'FD1', 'FD2'};
685 columnNames = [ h_labels, 'Exact'];
686 data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];
687
688 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
689
690 disp('Average computation bt iteration table (only for successful runs): F79, n=10^4,
superlinear');
691 disp(T15);
692
693 %% CG-SEQ
694
695 vec_bt_ex_clean = vec_cg_iter2_ex;
696 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
697 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
698
699 mat_bt_fd1_clean = mat_cg_iter2_fd1;
700 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
701 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
702
703 mat_bt_fd2_clean = mat_cg_iter2_fd2;
704 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
705 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
706
707 h_exponents = [2, 4, 6, 8, 10, 12];
708 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
709
710 fd1_vals = avg_fd1';
711 fd2_vals = avg_fd2';
712
713 rowNames = {'FD1', 'FD2'};
714 columnNames = [ h_labels, 'Exact'];
715 data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];
716
717 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
718
719 disp('Average computation cg iteration table (only for successful runs): F79, n=10^4,
superlinear');
720 disp(T16);
721

```



```

722 %% Number of initial point converged
723
724 h_exponents = [2, 4, 6, 8, 10, 12];
725 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
726
727 fd1_vals = sum(mat_converged2_fd1,2)';
728 fd2_vals = sum(mat_converged2_fd2,2)';
729
730 rowNames = {'FD1', 'FD2'};
731 columnNames = [ h_labels, 'Exact'];
732 data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];
733
734 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
735
736 disp('Number of converged: F79, n=10^4, superlinear');
737 disp(T17);
738 %save the table in a file.xlsx
739 writetable(T4, 'results_f79_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
740 writetable(T5, 'results_f79_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
741 writetable(T6, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
742 writetable(T14, 'results_f79_suplin.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
743 writetable(T15, 'results_f79_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
744 writetable(T16, 'results_f79_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
745 writetable(T17, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);
746
747 %% n=10^5 (1e5)
748
749 rng(345989);
750
751 n=1e5;
752
753 kmax=1.5e3; % maximum number of iterations of Newton method
754 tolgrad=5e-7; % tolerance on gradient norm
755
756 cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
757 system)
758 z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)
759
760 % Backtracking parameters
761 c1=1e-4;
762 rho=0.50;
763 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
764
765 x0=-1*ones(n,1); % initial point
766 N=10; % number of initial points to be generated
767
768 % Initial points:
769 Mat_points= repmat(x0,1,N+1);
770 rand_mat=2*rand(n, N)-1;
771 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
772
773 % Structure for EXACT derivatives
774 vec_times3_ex=zeros(1,N+1); % vector with execution times
775 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
776 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
777 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
778 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
779 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
780 mat_conv3_ex=zeros(12:N+1); %matrix with the last 12 values of rate of convergence for
781 the starting point
782 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
783 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
784 condition in Newton method
785
786 JF_ex = @(x) JF_gen(x,true,false,0);
787 HF_ex = @(x) HF_gen(x,true,false,0);
788
789 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
790 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
791 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
792 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
793 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
794 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations

```

```

792 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
793 mat_conv3_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
794 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
795 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
796
797 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
798 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
799
800 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
801 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
802 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
803 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
804 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
805 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
806 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
807 mat_conv3_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
808 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
809 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
810
811 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
812 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
813
814 for j =1:N+1
815     disp(['Condizione_iniziale_n.',num2str(j)])
816
817     % EXACT DERIVATIVES
818     tic;
819     [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);
820     vec_times3_ex(j)=toc;
821
822     disp(['Exact_derivatives:',flag3])
823     vec_converged3_ex(j)=converged3;
824     vec_val3_ex(j)=f3;
825     vec_grad3_ex(j)=gradf_norm3;
826     vec_iter3_ex(j)=k3;
827     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
828     vec_bt3_ex(j)=sum(btseq3)/k3;
829     vec_violations3_ex(j)=violations3;
830     last_vals = conv_ord3_ex(max(end-11,1):end);
831     mat_conv3_ex(:, j) = last_vals;
832
833     for i=2:2:12
834         h=10^(-i);
835
836         % FINITE DIFFERENCES 1
837         JF=@(x) JF_fd1(x,h);
838         HF=@(x) HF_fd1(x,h);
839         tic;
840         [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);
841         mat_times3_fd1(i/2,j)=toc;
842
843         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),':',flag3])
844         mat_converged3_fd1(i/2,j)=converged3;
845         mat_val3_fd1(i/2,j)=f3;
846         mat_grad3_fd1(i/2,j)=gradf_norm3;
847         mat_iter3_fd1(i/2,j)=k3;
848         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
849         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
850         mat_violations3_fd1(i/2,j)=violations3;
851         last_vals = conv_ord3_df1(max(end-11,1):end);
852         mat_conv3_fd1(i/2, j) = {last_vals};
853
854
855

```

```

856 % FINITE DIFFERENCES 2
857 JF=@(x) JF_fd2(x,h);
858 HF=@(x) HF_fd2(x,h);
859 tic;
860 [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);
861 mat_times3_fd2(i/2,j)=toc;
862
863 disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag3])
864 mat_converged3_fd2(i/2,j)=converged3;
865 mat_val3_fd2(i/2,j)=f3;
866 mat_grad3_fd2(i/2,j)=gradf_norm3;
867 mat_iter3_fd2(i/2,j)=k3;
868 mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
869 mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
870 mat_violations3_fd2(i/2,j)=violations3;
871 last_vals = conv_ord3_df2(max(end-11,1):end);
872 mat_conv3_fd2(i/2, j) = {last_vals};
873
874
875 end
876 end
877
878
879 %% The plot has the same structure as n=10^3
880 num_initial_points = N + 1;
881 figure;
882 hold on;
883
884 for j = 1:num_initial_points
885     conv_ord_ex = mat_conv3_ex(:,j);
886     plot(1:12, conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
887     hold on;
888     for i = 1:6
889         conv_ord_fd1 = mat_conv3_fd1{i, j};
890         conv_ord_fd2 = mat_conv3_fd2{i, j};
891         plot(1:12, conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
892         hold on;
893         plot(1:12, conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
894         hold on;
895     end
896 end
897
898 title('F79_10^5_superlinear');
899 xlabel('Iterazione');
900 ylabel('Ordine di Convergenza');
901 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
902 grid on;
903 hold off;
904
905 %% Time
906
907 vec_times_ex_clean = vec_times3_ex;
908 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
909 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
910
911 mat_times_fd1_clean = mat_times3_fd1;
912 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
913 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
914
915 mat_times_fd2_clean = mat_times3_fd2;
916 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
917 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
918
919 h_exponents = [2, 4, 6, 8, 10, 12];
920 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
921
922 fd1_vals = avg_fd1';
923 fd2_vals = avg_fd2';
924
925 rowNames = {'FD1', 'FD2'};
926 columnNames = [h_labels, 'Exact'];

```

```

927 data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];
928
929 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
930
931 disp('Average computation times table (only for successful runs): F79, n=10^5,
superlinear');
932 disp(T7);
933
934 %% Iteration
935
936 vec_times_ex_clean = vec_iter3_ex;
937 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
938 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
939
940 mat_times_fd1_clean = mat_iter3_fd1;
941 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
942 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
943
944 mat_times_fd2_clean = mat_iter3_fd2;
945 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
946 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
947
948 h_exponents = [2, 4, 6, 8, 10, 12];
949 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
950
951 fd1_vals = avg_fd1';
952 fd2_vals = avg_fd2';
953
954 rowNames = {'FD1', 'FD2'};
955 columnNames = [ h_labels, 'Exact'];
956 data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];
957
958 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
959
960 disp('Average computation iteration table (only for successful runs): F79, n=10^5,
superlinear');
961 disp(T8);
962
963 %% function value
964
965 vec_times_ex_clean = vec_val3_ex;
966 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
967 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
968
969 mat_times_fd1_clean = mat_val3_fd1;
970 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
971 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
972
973 mat_times_fd2_clean = mat_val3_fd2;
974 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
975 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
976
977 h_exponents = [2, 4, 6, 8, 10, 12];
978 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
979
980 fd1_vals = avg_fd1';
981 fd2_vals = avg_fd2';
982
983 rowNames = {'FD1', 'FD2'};
984 columnNames = [ h_labels, 'Exact'];
985 data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];
986
987 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
988
989 disp('Average computation fmin value table (only for successful runs): F79, n=10^5,
superlinear');
990 disp(T9);
991
992 %% VIOLATION
993
994 vec_times_ex_clean = vec_violations3_ex;
995 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
996 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');

```

```

997
998 mat_times_fd1_clean = mat_violations3_fd1;
999 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1000 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1001
1002 mat_times_fd2_clean = mat_violations3_fd2;
1003 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1004 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1005
1006 h_exponents = [2, 4, 6, 8, 10, 12];
1007 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1008
1009 fd1_vals = avg_fd1';
1010 fd2_vals = avg_fd2';
1011
1012 rowNames = {'FD1', 'FD2'};
1013 columnNames = [ h_labels, 'Exact'];
1014 data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];
1015
1016 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1017
1018 disp('Average computation violation table (only for successful runs): F79, n=10^5,
1019      superlinear');
1019 disp(T18);
1020
1021 %% BT-SEQ
1022
1023 vec_bt_ex_clean = vec_bt3_ex;
1024 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1025 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1026
1027 mat_bt_fd1_clean = mat_bt3_fd1;
1028 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1029 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1030
1031 mat_bt_fd2_clean = mat_bt3_fd2;
1032 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1033 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1034
1035 h_exponents = [2, 4, 6, 8, 10, 12];
1036 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1037
1038 fd1_vals = avg_fd1';
1039 fd2_vals = avg_fd2';
1040
1041 rowNames = {'FD1', 'FD2'};
1042 columnNames = [ h_labels, 'Exact'];
1043 data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];
1044
1045 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1046
1047 disp('Average computation bt iteration table (only for successful runs): F79, n=10^5,
1048      superlinear');
1048 disp(T19);
1049
1050 %% CG-SEQ
1051
1052 vec_bt_ex_clean = vec_cg_iter3_ex;
1053 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1054 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1055
1056 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1057 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1058 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1059
1060 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1061 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1062 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1063
1064 h_exponents = [2, 4, 6, 8, 10, 12];
1065 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1066
1067 fd1_vals = avg_fd1';

```

```

1068 fd2_vals = avg_fd2';
1069
1070 rowNames = {'FD1', 'FD2'};
1071 columnNames = [ h_labels, 'Exact'];
1072 data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1073
1074 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1075
1076 disp('Average computation cg iteration table (only for successful runs): F79, n=10^5,
    superlinear');
1077 disp(T20);
1078
1079 %% Number of initial condition converged
1080
1081 h_exponents = [2, 4, 6, 8, 10, 12];
1082 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1083
1084 fd1_vals = sum(mat_converged3_fd1,2)';
1085 fd2_vals = sum(mat_converged3_fd2,2)';
1086
1087 rowNames = {'FD1', 'FD2'};
1088 columnNames = [ h_labels, 'Exact'];
1089 data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1090
1091 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1092
1093 disp('Number of converged: F79, n=10^5, superlinear');
1094 disp(T21);
1095 %save the tables
1096 writetable(T7, 'results_f79_suplin.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1097 writetable(T8, 'results_f79_suplin.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1098 writetable(T9, 'results_f79_suplin.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1099 writetable(T18, 'results_f79_suplin.xlsx', 'Sheet', 'viol_5', 'WriteRowNames', true);
1100 writetable(T19, 'results_f79_suplin.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1101 writetable(T20, 'results_f79_suplin.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1102 writetable(T21, 'results_f79_suplin.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1103
1104 %% table with the results of the exact derivatives
1105
1106 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)];
1113
1114 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
    'Average iter cg', 'N converged'};
1115 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1116
1117 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1118 disp(T_compare)
1119
1120 writetable(T_compare, 'results_f79_suplin.xlsx', 'Sheet', 'ExactComparison', '
    WriteRowNames', true);

```

```

1 %% FUNCTION 79 (with different initial points)- with exact derivatives and finite
    differences- QUADRATIC TERM OF CONVERGENCE
2
3 sparse=true;
4
5 F = @(x) F79(x); % Defining F79 as function handle
6 JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
    handle
7 HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
    function handle (sparse version)
8
9 load forcing_terms.mat % possible terms for adaptive tolerance
10
11 %% n=10^3 (1e3)
12
13 rng(345989);

```

```

14
15 n=1e3;
16
17 kmax=1.5e3; % maximum number of iterations of Newton method
18 tolgrad=5e-7; % tolerance on gradient norm
19
20 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
    system)
21 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
22
23 % Backtracking parameters
24 c1=1e-4;
25 rho=0.50;
26 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
27
28 x0=-1*ones(n,1); % initial point
29 N=10; % number of initial points to be generated
30
31 % Initial points:
32 Mat_points= repmat(x0,1,N+1);
33 rand_mat=2*rand(n, N)-1;
34 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
41 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
42 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
43 mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
44 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
46
47 JF_ex = @(x) JF_gen(x,true,false,0);
48 HF_ex = @(x) HF_gen(x,true,false,0);
49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv1_fd1=cell(6, N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv1_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1

```

```

79     disp(['Condizione_iniziale_n.',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83     [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);
84     vec_times1_ex(j)=toc;
85
86     disp(['Exact_derivatives:',flag1])
87     vec_converged1_ex(j)=converged1;
88     vec_val1_ex(j)=f1;
89     vec_grad1_ex(j)=gradf_norm1;
90     vec_iter1_ex(j)=k1;
91     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
92     vec_bt1_ex(j)=sum(btseq1)/k1;
93     vec_violations1_ex(j)=violations1;
94     last_vals = conv_ord1_ex(max(end-11,1):end);
95     mat_conv1_ex(:, j) = last_vals;
96
97
98     for i=2:2:12
99         h=10^(-i);
100
101         % FINITE DIFFERENCES 1
102         JF=@(x) JF_fd1(x,h);
103         HF=@(x) HF_fd1(x,h);
104         tic;
105         [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);
106         mat_times1_fd1(i/2,j)=toc;
107
108         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag1])
109         mat_converged1_fd1(i/2,j)=converged1;
110         mat_val1_fd1(i/2,j)=f1;
111         mat_grad1_fd1(i/2,j)=gradf_norm1;
112         mat_iter1_fd1(i/2,j)=k1;
113         mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
114         mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
115         mat_violations1_fd1(i/2,j)=violations1;
116         last_vals = conv_ord1_df1(max(end-11,1):end);
117         mat_conv1_fd1(i/2, j) = {last_vals};
118
119
120         % FINITE DIFFERENCES 2
121         JF=@(x) JF_fd2(x,h);
122         HF=@(x) HF_fd2(x,h);
123         tic;
124         [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);
125         mat_times1_fd2(i/2,j)=toc;
126
127         disp(['Finite_differences_(new_version)_with_h=1e-',num2str(i),'_',flag1])
128         mat_converged1_fd2(i/2,j)=converged1;
129         mat_val1_fd2(i/2,j)=f1;
130         mat_grad1_fd2(i/2,j)=gradf_norm1;
131         mat_iter1_fd2(i/2,j)=k1;
132         mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
133         mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
134         mat_violations1_fd2(i/2,j)=violations1;
135         last_vals = conv_ord1_df2(max(end-11,1):end);
136         mat_conv1_fd2(i/2, j) = {last_vals};
137
138
139     end
140 end
141
142 %% Plot of the last 12 values of experimentale rate of convergence
143 num_initial_points = N + 1;
144 figure;
145 hold on;

```



```

146
147 % Plot for every initial condition
148 for j = 1:num_initial_points
149     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
150     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
151     hold on;
152     for i = 1:6
153         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
154         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
155         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
156         hold on;
157         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
158         hold on;
159     end
160 end
161
162 % title and legend
163 title('F79_10^3_quadratic');
164 xlabel('Iterazione');
165 ylabel('Ordine di Convergenza');
166 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
167 grid on;
168 hold off;
169
170
171 %% Execution Time
172
173 % Exact Derivative
174 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
175 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
176 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
177
178 % FD1
179 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
180 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
181 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
182
183 % FD2
184 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
185 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
186 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
187
188 % Creation of the labels
189 h_exponents = [2, 4, 6, 8, 10, 12];
190 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
191
192 fd1_vals = avg_fd1';
193 fd2_vals = avg_fd2';
194
195 % Table costruction with exact for both the row
196 rowNames = {'FD1', 'FD2'};
197 columnNames = [ h_labels, 'Exact'];
198 data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
199 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
200
201 % visualization
202 disp('Average computation times table (only for successful runs): F79, n=10^3, quadratic'
    );
203 disp(T1);
204
205
206 %% All the tables has the same structure
207 %% Iteration
208
209 vec_times_ex_clean = vec_iter1_ex;
210 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
211 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
212
213 mat_times_fd1_clean = mat_iter1_fd1;
214 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
215 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

```

```

216
217 mat_times_fd2_clean = mat_iter1_fd2;
218 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
219 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
220
221 h_exponents = [2, 4, 6, 8, 10, 12];
222 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
223
224 fd1_vals = avg_fd1';
225 fd2_vals = avg_fd2';
226
227 rowNames = {'FD1', 'FD2'};
228 columnNames = [h_labels, 'Exact'];
229 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];
230
231 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
232
233 disp('Average computation iteration table (only for successful runs): F79, n=10^3, quadratic');
234 disp(T2);
235
236 %% F value
237
238 vec_times_ex_clean = vec_val1_ex;
239 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
240 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
241
242 mat_times_fd1_clean = mat_val1_fd1;
243 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
244 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
245
246 mat_times_fd2_clean = mat_val1_fd2;
247 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
248 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
249
250 h_exponents = [2, 4, 6, 8, 10, 12];
251 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
252
253 fd1_vals = avg_fd1';
254 fd2_vals = avg_fd2';
255
256 rowNames = {'FD1', 'FD2'};
257 columnNames = [h_labels, 'Exact'];
258 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];
259
260 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
261
262 disp('Average computation fmin value table (only for successful runs): F79, n=10^3, quadratic');
263 disp(T3);
264
265 %% VIOLATION
266
267 vec_times_ex_clean = vec_violations1_ex;
268 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
269 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
270
271 mat_times_fd1_clean = mat_violations1_fd1;
272 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
273 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
274
275 mat_times_fd2_clean = mat_violations1_fd2;
276 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
277 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
278
279 h_exponents = [2, 4, 6, 8, 10, 12];
280 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
281
282 %
283 fd1_vals = avg_fd1';
284 fd2_vals = avg_fd2';
285
286 rowNames = {'FD1', 'FD2'};

```

```

287 columnNames = [ h_labels, 'Exact'];
288 data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];
289
290 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
291
292 disp('Average computation violation table (only for successful runs): F79, n=10^3,
quadratic');
293 disp(T10);
294
295
296 %% BT-SEQ
297 vec_bt_ex_clean = vec_bt1_ex;
298 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
299 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
300
301 mat_bt_fd1_clean = mat_bt1_fd1;
302 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
303 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
304
305 mat_bt_fd2_clean = mat_bt1_fd2;
306 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
307 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
308
309 h_exponents = [2, 4, 6, 8, 10, 12];
310 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
311
312 fd1_vals = avg_fd1';
313 fd2_vals = avg_fd2';
314
315 rowNames = {'FD1', 'FD2'};
316 columnNames = [ h_labels, 'Exact'];
317 data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
318
319 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
320
321 disp('Average computation bt iteration table (only for successful runs): F79, n=10^3,
quadratic');
322 disp(T11);
323
324 %% CG-SEQ
325
326 vec_bt_ex_clean = vec_cg_iter1_ex;
327 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
328 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
329
330 mat_bt_fd1_clean = mat_cg_iter1_fd1;
331 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
332 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
333
334 mat_bt_fd2_clean = mat_cg_iter1_fd2;
335 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
336 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
337
338 h_exponents = [2, 4, 6, 8, 10, 12];
339 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
340
341 fd1_vals = avg_fd1';
342 fd2_vals = avg_fd2';
343
344 rowNames = {'FD1', 'FD2'};
345 columnNames = [ h_labels, 'Exact'];
346 data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
347
348 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
349
350 disp('Average computation cg iteration table (only for successful runs): F79, n=10^3,
quadratic');
351 disp(T12);
352
353 %% Number of starting point converged
354
355 h_exponents = [2, 4, 6, 8, 10, 12];
356 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```

357 fd1_vals = sum(mat_converged1_fd1,2)';
358 fd2_vals = sum(mat_converged1_fd2,2)';
359
360 rowNames = {'FD1', 'FD2'};
361 columnNames = [ h_labels, 'Exact'];
362 data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex)];
363
364 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
365
366 disp('Number of converged F79, n=10^3, quadratic');
367 disp(T13);
368
369 %save the table in a file.xlsx
370 writetable(T1, 'results_f79_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
371 writetable(T2, 'results_f79_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
372 writetable(T3, 'results_f79_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
373 writetable(T10, 'results_f79_quad.xlsx', 'Sheet', 'viol_3','WriteRowNames', true);
374 writetable(T11, 'results_f79_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
375 writetable(T12, 'results_f79_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
376 writetable(T13, 'results_f79_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);
377
378
379 %% n=10^4 (1e4)
380
381 rng(345989);
382
383 n=1e4;
384
385 kmax=1.5e3; % maximum number of iterations of Newton method
386 tolgrad=5e-7; % tolerance on gradient norm
387
388 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
389 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
390
391 % Backtracking parameters
392 c1=1e-4;
393 rho=0.50;
394 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
395
396 x0=-1*ones(n,1); % initial point
397 N=10; % number of initial points to be generated
398
399 % Initial points:
400 Mat_points=repmat(x0,1,N+1);
401 rand_mat=2*rand(n, N)-1;
402 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
403
404 % Structure for EXACT derivatives
405 vec_times2_ex=zeros(1,N+1); % vector with execution times
406 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
407 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
408 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
409 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
410 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
411 mat_conv2_ex=zeros(12,N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
412 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
413 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
414
415 JF_ex = @(x) JF_gen(x,true,false,0);
416 HF_ex = @(x) HF_gen(x,true,false,0);
417
418 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
419 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
420 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
421 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
422 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
423 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
424 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
425 mat_conv2_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point

```

```

426 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
427 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
428
429 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
430 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
431
432 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
433 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
434 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
435 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
436 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
437 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
438 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
439 mat_conv2_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
440 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
441 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
442
443 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
444 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
445
446 for j =1:N+1
447     disp(['Condizione_iniziale_n.',num2str(j)])
448
449     % EXACT DERIVATIVES
450     tic;
451     [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);
452     vec_times2_ex(j)=toc;
453
454     disp(['Exact_derivatives:',flag2])
455     vec_converged2_ex(j)=converged2;
456     vec_val2_ex(j)=f2;
457     vec_grad2_ex(j)=gradf_norm2;
458     vec_iter2_ex(j)=k2;
459     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
460     vec_bt2_ex(j)=sum(btseq2)/k2;
461     vec_violations2_ex(j)=violations2;
462     last_vals = conv_ord2_ex(max(end-11,1):end);
463     mat_conv2_ex(:, j) = last_vals;
464
465     for i=2:2:12
466         h=10^(-i);
467
468         % FINITE DIFFERENCES 1
469         JF=@(x) JF_fd1(x,h);
470         HF=@(x) HF_fd1(x,h);
471         tic;
472         [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);
473         mat_times2_fd1(i/2,j)=toc;
474
475         disp(['Finite_differences(classical_version)_with_h=1e-',num2str(i),'_',flag2])
476         mat_converged2_fd1(i/2,j)=converged2;
477         mat_val2_fd1(i/2,j)=f2;
478         mat_grad2_fd1(i/2,j)=gradf_norm2;
479         mat_iter2_fd1(i/2,j)=k2;
480         mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
481         mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
482         mat_violations2_fd1(i/2,j)=violations2;
483         last_vals = conv_ord2_df1(max(end-11,1):end);
484         mat_conv2_fd1(i/2, j) = {last_vals};
485
486
487
488         % FINITE DIFFERENCES 2
489         JF=@(x) JF_fd2(x,h);
490         HF=@(x) HF_fd2(x,h);

```

```

491     tic;
492     [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);
493     mat_times2_fd2(i/2,j)=toc;
494
495     disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag2])
496     mat_converged2_fd2(i/2,j)=converged2;
497     mat_val2_fd2(i/2,j)=f2;
498     mat_grad2_fd2(i/2,j)=gradf_norm2;
499     mat_iter2_fd2(i/2,j)=k2;
500     mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
501     mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
502     mat_violations2_fd2(i/2,j)=violations2;
503     last_vals = conv_ord2_df2(max(end-11,1):end);
504     mat_conv2_fd2(i/2, j) = {last_vals};
505
506     end
507 end
508
509 %% The Plot has the same structure
510 num_initial_points = N + 1;
511 figure;
512 hold on;
513
514 for j = 1:num_initial_points
515     conv_ord_ex = mat_conv2_ex(:,j);
516     plot(1:12, conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
517     hold on;
518     for i = 1:6
519         conv_ord_fd1 = mat_conv2_fd1{i, j};
520         conv_ord_fd2 = mat_conv2_fd2{i, j};
521         plot(1:12, conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
522         hold on;
523         plot(1:12, conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
524         hold on;
525     end
526 end
527
528 title('F79 10^4 quadratic');
529 xlabel('Iterazione');
530 ylabel('Ordine di Convergenza');
531 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
532 grid on;
533 hold off;
534
535
536
537
538 %% Execution time
539
540 % Exact derivative
541 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
542 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
543 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
544
545 % FD1
546 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
547 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
548 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
549
550 % FD2
551 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
552 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
553 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
554
555 % Creation of the labels
556 h_exponents = [2, 4, 6, 8, 10, 12];
557 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
558
559 fd1_vals = avg_fd1';

```

```

560 fd2_vals = avg_fd2';
561
562 % Table creation
563 rowNames = {'FD1', 'FD2'};
564 columnNames = [ h_labels, 'Exact'];
565 data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
566 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
567 %display the table
568 disp('Average computation times table (only for successful runs): F79, n=10^4, quadratic'
);
569 disp(T4);
570
571 %% Iteration
572
573 vec_times_ex_clean = vec_iter2_ex;
574 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
575 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
576
577 mat_times_fd1_clean = mat_iter2_fd1;
578 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
579 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
580
581 mat_times_fd2_clean = mat_iter2_fd2;
582 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
583 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
584
585 h_exponents = [2, 4, 6, 8, 10, 12];
586 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
587
588 fd1_vals = avg_fd1';
589 fd2_vals = avg_fd2';
590
591 rowNames = {'FD1', 'FD2'};
592 columnNames = [ h_labels, 'Exact'];
593 data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];
594
595 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
596
597 disp('Average computation iteration table (only for successful runs): F79, n=10^4,
quadratic');
598 disp(T5);
599
600 %% Function value
601
602 vec_times_ex_clean = vec_val2_ex;
603 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
604 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
605
606 mat_times_fd1_clean = mat_val2_fd1;
607 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
608 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
609
610 mat_times_fd2_clean = mat_val2_fd2;
611 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
612 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
613
614 h_exponents = [2, 4, 6, 8, 10, 12];
615 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
616
617 fd1_vals = avg_fd1';
618 fd2_vals = avg_fd2';
619
620 rowNames = {'FD1', 'FD2'};
621 columnNames = [ h_labels, 'Exact'];
622 data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];
623
624 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
625
626 disp('Average computation fmin value table (only for successful runs): F79, n=10^4,
quadratic');
627 disp(T6);
628
629 %% VIOLATION

```

```

630
631 vec_times_ex_clean = vec_violations2_ex;
632 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
633 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
634
635 mat_times_fd1_clean = mat_violations2_fd1;
636 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
637 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
638
639 mat_times_fd2_clean = mat_violations2_fd2;
640 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
641 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
642
643 h_exponents = [2, 4, 6, 8, 10, 12];
644 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
645
646 fd1_vals = avg_fd1';
647 fd2_vals = avg_fd2';
648
649 rowNames = {'FD1', 'FD2'};
650 columnNames = [h_labels, 'Exact'];
651 data = [fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2];
652
653 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
654
655 disp('Average computation violation table (only for successful runs): F79, n=10^4, quadratic');
656 disp(T14);
657
658 %% BT-SEQ
659
660 vec_bt_ex_clean = vec_bt2_ex;
661 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
662 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
663
664 mat_bt_fd1_clean = mat_bt2_fd1;
665 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
666 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
667
668 mat_bt_fd2_clean = mat_bt2_fd2;
669 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
670 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
671
672 h_exponents = [2, 4, 6, 8, 10, 12];
673 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
674
675 fd1_vals = avg_fd1';
676 fd2_vals = avg_fd2';
677
678 rowNames = {'FD1', 'FD2'};
679 columnNames = [h_labels, 'Exact'];
680 data = [fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2];
681
682 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
683
684 disp('Average computation bt iteration table (only for successful runs): F79, n=10^4, quadratic');
685 disp(T15);
686
687 %% CG-SEQ
688
689 vec_bt_ex_clean = vec_cg_iter2_ex;
690 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
691 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
692
693 mat_bt_fd1_clean = mat_cg_iter2_fd1;
694 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
695 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
696
697 mat_bt_fd2_clean = mat_cg_iter2_fd2;
698 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
699 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
700

```



```

701 h_exponents = [2, 4, 6, 8, 10, 12];
702 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
703
704 fd1_vals = avg_fd1';
705 fd2_vals = avg_fd2';
706
707 rowNames = {'FD1', 'FD2'};
708 columnNames = [ h_labels, 'Exact'];
709 data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];
710
711 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
712
713 disp('Average computation cg iteration table (only for successful runs): F79, n=10^4, quadratic');
714 disp(T16);
715
716 %% Number of initial point converged
717
718 h_exponents = [2, 4, 6, 8, 10, 12];
719 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
720
721 fd1_vals = sum(mat_converged2_fd1,2)';
722 fd2_vals = sum(mat_converged2_fd2,2)';
723
724 rowNames = {'FD1', 'FD2'};
725 columnNames = [ h_labels, 'Exact'];
726 data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];
727
728 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
729
730 disp('Number of converged: F79, n=10^4, quadratic');
731 disp(T17);
732 %save the table if a file.xlsx
733 writetable(T4, 'results_f79_quad.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
734 writetable(T5, 'results_f79_quad.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
735 writetable(T6, 'results_f79_quad.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
736 writetable(T14, 'results_f79_quad.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
737 writetable(T15, 'results_f79_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
738 writetable(T16, 'results_f79_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
739 writetable(T17, 'results_f79_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);
740
741
742 %% n=10^5 (1e5)
743
744 rng(345989);
745
746 n=1e5;
747
748 kmax=1.5e3; % maximum number of iterations of Newton method
749 tolgrad=5e-7; % tolerance on gradient norm
750
751 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear system)
752 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
753
754 % Backtracking parameters
755 c1=1e-4;
756 rho=0.50;
757 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
758
759 x0=-1*ones(n,1); % initial point
760 N=10; % number of initial points to be generated
761
762 % Initial points:
763 Mat_points= repmat(x0,1,N+1);
764 rand_mat=2*rand(n, N)-1;
765 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
766
767 % Structure for EXACT derivatives
768 vec_times3_ex=zeros(1,N+1); % vector with execution times
769 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
770 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
771 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations

```

```

772 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
773 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
774 mat_conv3_ex=zeros(12:N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
775 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
776 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
777
778 JF_ex = @(x) JF_gen(x,true,false,0);
779 HF_ex = @(x) HF_gen(x,true,false,0);
780
781 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
782 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
783 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
784 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
785 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
786 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
787 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
788 mat_conv3_fd1=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
789 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
790 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
791
792 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
793 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
794
795 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
796 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
797 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
798 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
799 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
800 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
801 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
802 mat_conv3_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
803 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
804 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
805
806 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
807 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
808
809 for j =1:N+1
810     disp(['Condizione_iniziale_n.',num2str(j)])
811
812     % EXACT DERIVATIVES
813     tic;
814     [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);
815     vec_times3_ex(j)=toc;
816
817     disp(['Exact_derivatives:',num2str(j)])
818     vec_converged3_ex(j)=converged3;
819     vec_val3_ex(j)=f3;
820     vec_grad3_ex(j)=gradf_norm3;
821     vec_iter3_ex(j)=k3;
822     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
823     vec_bt3_ex(j)=sum(btseq3)/k3;
824     vec_violations3_ex(j)=violations3;
825     last_vals = conv_ord3_ex(max(end-11,1):end);
826     mat_conv3_ex(:, j) = last_vals;
827
828     for i=2:2:12
829         h=10^(-i);
830
831         % FINITE DIFFERENCES 1
832         JF=@(x) JF_fd1(x,h);
833         HF=@(x) HF_fd1(x,h);
834         tic;
835         [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);
836 mat_times3_fd1(i/2,j)=toc;
837
838 disp(['Finite differences (classical version) with h=1e-', num2str(i), ' : ', flag3])
839 mat_converged3_fd1(i/2,j)=converged3;
840 mat_val3_fd1(i/2,j)=f3;
841 mat_grad3_fd1(i/2,j)=gradf_norm3;
842 mat_iter3_fd1(i/2,j)=k3;
843 mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
844 mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
845 mat_violations3_fd1(i/2,j)=violations3;
846 last_vals = conv_ord3_df1(max(end-11,1):end);
847 mat_conv3_fd1(i/2, j) = {last_vals};
848
849
850
851 % FINITE DIFFERENCES 2
852 JF=@(x) JF_fd2(x,h);
853 HF=@(x) HF_fd2(x,h);
854 tic;
855 [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);
856 mat_times3_fd2(i/2,j)=toc;
857
858 disp(['Finite differences (new version) with h=1e-', num2str(i), ' : ', flag3])
859 mat_converged3_fd2(i/2,j)=converged3;
860 mat_val3_fd2(i/2,j)=f3;
861 mat_grad3_fd2(i/2,j)=gradf_norm3;
862 mat_iter3_fd2(i/2,j)=k3;
863 mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
864 mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
865 mat_violations3_fd2(i/2,j)=violations3;
866 last_vals = conv_ord3_df2(max(end-11,1):end);
867 mat_conv3_fd2(i/2, j) = {last_vals};
868
869     end
870 end
871
872 %% The plot has the same structure as n=10^3
873 num_initial_points = N + 1;
874 figure;
875 hold on;
876
877 for j = 1:num_initial_points
878     conv_ord_ex = mat_conv3_ex(:,j);
879     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
880     hold on;
881     for i =1:6
882         conv_ord_fd1 = mat_conv3_fd1{i, j};
883         conv_ord_fd2 = mat_conv3_fd2{i, j};
884         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
885         hold on;
886         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
887         hold on;
888     end
889 end
890
891 title('F79_10^5 quadratic');
892 xlabel('Iterazione');
893 ylabel('Ordine di Convergenza');
894 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
895 grid on;
896 hold off;
897
898 %% Time
899
900 vec_times_ex_clean = vec_times3_ex;
901 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
902 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
903
904 mat_times_fd1_clean = mat_times3_fd1;

```

```

905 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
906 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
907
908 mat_times_fd2_clean = mat_times3_fd2;
909 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
910 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
911
912 h_exponents = [2, 4, 6, 8, 10, 12];
913 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
914
915 fd1_vals = avg_fd1';
916 fd2_vals = avg_fd2';
917
918 rowNames = {'FD1', 'FD2'};
919 columnNames = [h_labels, 'Exact'];
920 data = [fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3];
921
922 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
923
924 disp('Average computation times table (only for successful runs): F79, n=10^5, quadratic');
925 disp(T7);
926
927 %% Iteration
928
929 vec_times_ex_clean = vec_iter3_ex;
930 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
931 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
932
933 mat_times_fd1_clean = mat_iter3_fd1;
934 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
935 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
936
937 mat_times_fd2_clean = mat_iter3_fd2;
938 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
939 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
940
941 h_exponents = [2, 4, 6, 8, 10, 12];
942 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
943
944 fd1_vals = avg_fd1';
945 fd2_vals = avg_fd2';
946
947 rowNames = {'FD1', 'FD2'};
948 columnNames = [h_labels, 'Exact'];
949 data = [fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3];
950
951 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
952
953 disp('Average computation iteration table (only for successful runs): F79, n=10^5, quadratic');
954 disp(T8);
955
956 %% function value
957
958 vec_times_ex_clean = vec_val3_ex;
959 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
960 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
961
962 mat_times_fd1_clean = mat_val3_fd1;
963 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
964 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
965
966 mat_times_fd2_clean = mat_val3_fd2;
967 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
968 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
969
970 h_exponents = [2, 4, 6, 8, 10, 12];
971 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
972
973 fd1_vals = avg_fd1';
974 fd2_vals = avg_fd2';
975

```

```

976 rowNames = {'FD1', 'FD2'};
977 columnNames = [ h_labels, 'Exact'];
978 data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];
979
980 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
981
982 disp('Average computation fmin value table (only for successful runs): F79, n=10^5, quadratic');
983 disp(T9);
984
985 %% VIOLATION
986
987 vec_times_ex_clean = vec_violations3_ex;
988 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
989 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
990
991 mat_times_fd1_clean = mat_violations3_fd1;
992 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
993 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
994
995 mat_times_fd2_clean = mat_violations3_fd2;
996 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
997 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
998
999 h_exponents = [2, 4, 6, 8, 10, 12];
1000 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1001
1002 fd1_vals = avg_fd1';
1003 fd2_vals = avg_fd2';
1004
1005 rowNames = {'FD1', 'FD2'};
1006 columnNames = [ h_labels, 'Exact'];
1007 data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];
1008
1009 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1010
1011 disp('Average computation violation table (only for successful runs): F79, n=10^5, quadratic');
1012 disp(T18);
1013
1014 %% BT-SEQ
1015
1016 vec_bt_ex_clean = vec_bt3_ex;
1017 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1018 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1019
1020 mat_bt_fd1_clean = mat_bt3_fd1;
1021 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1022 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1023
1024 mat_bt_fd2_clean = mat_bt3_fd2;
1025 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1026 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1027
1028 h_exponents = [2, 4, 6, 8, 10, 12];
1029 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1030
1031 fd1_vals = avg_fd1';
1032 fd2_vals = avg_fd2';
1033
1034 rowNames = {'FD1', 'FD2'};
1035 columnNames = [ h_labels, 'Exact'];
1036 data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];
1037
1038 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1039
1040 disp('Average computation bt iteration table (only for successful runs): F79, n=10^5, quadratic');
1041 disp(T19);
1042
1043 %% CG-SEQ
1044
1045 vec_bt_ex_clean = vec_cg_iter3_ex;

```

```

1046 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1047 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1048
1049 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1050 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1051 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1052
1053 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1054 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1055 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1056
1057 h_exponents = [2, 4, 6, 8, 10, 12];
1058 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1059
1060 fd1_vals = avg_fd1';
1061 fd2_vals = avg_fd2';
1062
1063 rowNames = {'FD1', 'FD2'};
1064 columnNames = [h_labels, 'Exact'];
1065 data = [fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1066
1067 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1068
1069 disp('Average computation cg iteration table (only for successful runs): F79, n=10^5, u
    quadratic');
1070 disp(T20);
1071
1072 %% Number of initial condition converged
1073
1074 h_exponents = [2, 4, 6, 8, 10, 12];
1075 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1076
1077 fd1_vals = sum(mat_converged3_fd1, 2)';
1078 fd2_vals = sum(mat_converged3_fd2, 2)';
1079
1080 rowNames = {'FD1', 'FD2'};
1081 columnNames = [h_labels, 'Exact'];
1082 data = [fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1083
1084 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1085
1086 disp('Number of converged: F79, n=10^5, uquadratic');
1087 disp(T21);
1088 %save the tables
1089
1090 writetable(T7, 'results_f79_quad.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1091 writetable(T8, 'results_f79_quad.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1092 writetable(T9, 'results_f79_quad.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1093 writetable(T18, 'results_f79_quad.xlsx', 'Sheet', 'viol_5', 'WriteRowNames', true);
1094 writetable(T19, 'results_f79_quad.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1095 writetable(T20, 'results_f79_quad.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1096 writetable(T21, 'results_f79_quad.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1097
1098
1099
1100 %% table with the result of the exact derivatives
1101 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1102         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1103         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1104         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1105         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1106         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1107         sum(vec_converged1_ex), sum(vec_converged2_ex), sum(vec_converged3_ex)];
1108
1109 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
    'Average iter cg', 'N converged'};
1110 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1111
1112 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1113 disp(T_compare)
1114
1115 writetable(T_compare, 'results_f79_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames
    ', true);

```

```

1  %% FUNCTION 79 PRECONDITIOINING (with different initial points)- with exact derivatives
   and finite differences
2
3  sparse=true;
4
5  F = @(x) F79(x); % Defining F79 as function handle
6  JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
   handle
7  HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
   function handle (sparse version)
8
9  load forcing_terms.mat % possible terms for adaptive tolerance
10
11 %% n=10^3 (1e3)
12
13 rng(345989);
14
15 n=1e3;
16
17 kmax=1.5e3; % maximum number of iterations of Newton method
18 tolgrad=5e-7; % tolerance on gradient norm
19
20 cg_maxit=50; % maximum number of iterations of coniugate gradient method (for the linear
   system)
21 z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)
22
23 % Backtracking parameters
24 c1=1e-3;
25 rho=0.50;
26 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
27
28 x0=-1*ones(n,1); % initial point
29 N=10; % number of initial points to be generated
30
31 % Initial points:
32 Mat_points= repmat(x0,1,N+1);
33 rand_mat=2*rand(n, N)-1;
34 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
41 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
42 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
43 mat_conv_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for the
   starting point
44 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
   condition in Newton method
46
47 JF_ex = @(x) JF_gen(x,true,false,0);
48 HF_ex = @(x) HF_gen(x,true,false,0);
49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv_fd1=cell(6, N+1);%matrix with che last 12 values of rate of convergence for the
   starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
   condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
   x_j) as increment)

```

```

65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1
79     disp(['Condizione_iniziale_n.',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83
84     [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);
85
86     vec_times1_ex(j)=toc;
87
88     disp(['Exact_derivatives:',flag1])
89     vec_converged1_ex(j)=converged1;
90
91     vec_val1_ex(j)=f1;
92     vec_grad1_ex(j)=gradf_norm1;
93     vec_iter1_ex(j)=k1;
94     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
95     vec_bt1_ex(j)=sum(btseq1)/k1;
96     vec_violations1_ex(j)=violations1;
97
98     last_vals = conv_ord1_ex(max(end-14,1):end);
99     mat_conv_ex(:, j) = last_vals;
100
101
102     for i=2:2:12
103         h=10^(-i);
104
105         % FINITE DIFFERENCES 1
106         JF=@(x) JF_fd1(x,h);
107         HF=@(x) HF_fd1(x,h);
108         tic;
109
110         [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);
111
112         mat_times1_fd1(i/2,j)=toc;
113
114         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag1])
115         mat_converged1_fd1(i/2,j)=converged1;
116         %
117         mat_val1_fd1(i/2,j)=f1;
118         mat_grad1_fd1(i/2,j)=gradf_norm1;
119         mat_iter1_fd1(i/2,j)=k1;
120         mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
121         mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
122         mat_violations1_fd1(i/2,j)=violations1;
123
124
125         last_vals = conv_ord1_df1(max(end-14,1):end);
126         mat_conv_fd1(i/2, j) = {last_vals};
127
128
129
130     % FINITE DIFFERENCES 2
131     JF=@(x) JF_fd2(x,h);

```



```

132     HF=@(x) HF_fd2(x,h);
133     tic;
134     [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);
135     mat_times1_fd2(i/2,j)=toc;
136
137     disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
138     mat_converged1_fd2(i/2,j)=converged1;
139     mat_val1_fd2(i/2,j)=f1;
140     mat_grad1_fd2(i/2,j)=gradf_norm1;
141     mat_iter1_fd2(i/2,j)=k1;
142     mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
143     mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
144     mat_violations1_fd2(i/2,j)=violations1;
145
146     last_vals = conv_ord1_df2(max(end-14,1):end);
147     mat_conv_fd2(i/2, j) = {last_vals};
148
149
150     end
151 end
152
153
154 %% Plot of the last 12 values of experimentale rate of convergence
155 num_initial_points = N + 1;
156 figure;
157 hold on;
158
159 % Plot for every initial condition
160 for j = 1:num_initial_points
161     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
162     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
163     hold on;
164     for i =1:6
165         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
166         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
167         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
168         hold on;
169         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
170         hold on;
171     end
172 end
173
174 % title and legend
175 title('F79P10^3superlinear');
176 xlabel('Iterazione');
177 ylabel('Ordine di Convergenza');
178 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
179 grid on;
180 hold off;
181
182
183 %% Execution Time
184
185 % Exact Derivative
186 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
187 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
188 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
189
190 % FD1
191 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
192 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
193 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
194
195 % FD2
196 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
197 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
198 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
199
200 % Creation of the labels

```

```

201 h_exponents = [2, 4, 6, 8, 10, 12];
202 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
203
204 fd1_vals = avg_fd1';
205 fd2_vals = avg_fd2';
206
207 % Table construction with exact for both the row
208 rowNames = {'FD1', 'FD2'};
209 columnNames = [ h_labels, 'Exact'];
210 data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
211 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
212
213 % visualization
214 disp('Average computation times table (only for successful runs): F79P, n=10^3,
      superlinear');
215 disp(T1);
216
217
218 %% All the tables has the same structure
219 %% Iteration
220
221 vec_times_ex_clean = vec_iter1_ex;
222 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
223 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
224
225 mat_times_fd1_clean = mat_iter1_fd1;
226 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
227 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
228
229 mat_times_fd2_clean = mat_iter1_fd2;
230 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
231 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
232
233 h_exponents = [2, 4, 6, 8, 10, 12];
234 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
235
236 fd1_vals = avg_fd1';
237 fd2_vals = avg_fd2';
238
239 rowNames = {'FD1', 'FD2'};
240 columnNames = [ h_labels, 'Exact'];
241 data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];
242
243 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
244
245 disp('Average computation iteration table (only for successful runs): F79P, n=10^3,
      suplin');
246 disp(T2);
247
248 %% F value
249
250 vec_times_ex_clean = vec_val1_ex;
251 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
252 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
253
254 mat_times_fd1_clean = mat_val1_fd1;
255 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
256 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
257
258 mat_times_fd2_clean = mat_val1_fd2;
259 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
260 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
261
262 h_exponents = [2, 4, 6, 8, 10, 12];
263 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
264
265 fd1_vals = avg_fd1';
266 fd2_vals = avg_fd2';
267
268 rowNames = {'FD1', 'FD2'};
269 columnNames = [ h_labels, 'Exact'];
270 data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];
271

```

```

272 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
273
274 disp('Average computation fmin value table (only for successful runs): F79P, n=10^3,
suplin');
275 disp(T3);
276
277 %% VIOLATION
278
279 vec_times_ex_clean = vec_violations1_ex;
280 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
281 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
282
283 mat_times_fd1_clean = mat_violations1_fd1;
284 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
285 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
286
287 mat_times_fd2_clean = mat_violations1_fd2;
288 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
289 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
290
291 h_exponents = [2, 4, 6, 8, 10, 12];
292 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
293
294 %
295 fd1_vals = avg_fd1';
296 fd2_vals = avg_fd2';
297
298 rowNames = {'FD1', 'FD2'};
299 columnNames = [h_labels, 'Exact'];
300 data = [fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1];
301
302 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
303
304 disp('Average computation violation table (only for successful runs): F79P, n=10^3,
superlinear');
305 disp(T10);
306
307
308 %% BT-SEQ
309 vec_bt_ex_clean = vec_bt1_ex;
310 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
311 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
312
313 mat_bt_fd1_clean = mat_bt1_fd1;
314 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
315 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
316
317 mat_bt_fd2_clean = mat_bt1_fd2;
318 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
319 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
320
321 h_exponents = [2, 4, 6, 8, 10, 12];
322 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
323
324 fd1_vals = avg_fd1';
325 fd2_vals = avg_fd2';
326
327 rowNames = {'FD1', 'FD2'};
328 columnNames = [h_labels, 'Exact'];
329 data = [fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1];
330
331 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
332
333 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^3,
superlinear');
334 disp(T11);
335
336 %% CG-SEQ
337
338 vec_bt_ex_clean = vec_cg_iter1_ex;
339 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
340 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
341

```

```

342 mat_bt_fd1_clean = mat_cg_iter1_fd1;
343 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
344 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
345
346 mat_bt_fd2_clean = mat_cg_iter1_fd2;
347 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
348 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
349
350 h_exponents = [2, 4, 6, 8, 10, 12];
351 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
352
353 fd1_vals = avg_fd1';
354 fd2_vals = avg_fd2';
355
356 rowNames = {'FD1', 'FD2'};
357 columnNames = [h_labels, 'Exact'];
358 data = [fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
359
360 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
361
362 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^3,
superlinear');
363 disp(T12);
364
365 %% Number of starting point converged
366
367 h_exponents = [2, 4, 6, 8, 10, 12];
368 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
369
370 fd1_vals = sum(mat_converged1_fd1, 2)';
371 fd2_vals = sum(mat_converged1_fd2, 2)';
372
373 rowNames = {'FD1', 'FD2'};
374 columnNames = [h_labels, 'Exact'];
375 data = [fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];
376
377 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
378
379 disp('Number of converged: F79P, n=10^3, superlinear');
380 disp(T13);
381
382 %save the table in a file.xlsx
383 writetable(T1, 'results_f79P_suplin.xlsx', 'Sheet', 'time_3', 'WriteRowNames', true);
384 writetable(T2, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_3', 'WriteRowNames', true);
385 writetable(T3, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_3', 'WriteRowNames', true);
386 writetable(T10, 'results_f79P_suplin.xlsx', 'Sheet', 'v_3', 'WriteRowNames', true);
387 writetable(T11, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_3', 'WriteRowNames', true);
388 writetable(T12, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_3', 'WriteRowNames', true);
389 writetable(T13, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv3', 'WriteRowNames', true);
390
391
392 %% n=10^4 (1e4)
393
394 rng(345989);
395
396 n=1e4;
397
398 kmax=1.5e3; % maximum number of iterations of Newton method
399 tolgrad=5e-7; % tolerance on gradient norm
400
401 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
system)
402 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
403
404 % Backtracking parameters
405 c1=1e-4;
406 rho=0.50;
407 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
408
409 x0=-1*ones(n,1); % initial point
410 N=10; % number of initial points to be generated
411
412 % Initial points:

```

```

413 Mat_points= repmat(x0,1,N+1);
414 rand_mat=2*rand(n, N)-1;
415 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
416
417 % Structure for EXACT derivatives
418 vec_times2_ex=zeros(1,N+1); % vector with execution times
419 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
420 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
421 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
422 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
423 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
424 mat_conv2_ex=zeros(15, N+1);%matrix with che last 15 values of rate of convergence for
    the starting point
425 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
426 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
427
428 JF_ex = @(x) JF_gen(x,true,false,0);
429 HF_ex = @(x) HF_gen(x,true,false,0);
430
431 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
432 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
433 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
434 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
435 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
436 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
437 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
438 mat_conv2_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
439 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
440 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
441
442 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
443 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
444
445 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
446 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
447 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
448 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
449 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
450 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
451 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
452 mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
453 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
454 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
455
456 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
457 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
458
459 for j =1:N+1
460     disp(['Condizione iniziale n. ',num2str(j)])
461
462     % EXACT DERIVATIVES
463     tic;
464
465     [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);
466
467     vec_times2_ex(j)=toc;
468
469     disp(['Exact derivatives:',flag2])
470     vec_converged2_ex(j)=converged2;
471
472     vec_val2_ex(j)=f2;
473     vec_grad2_ex(j)=gradf_norm2;
474     vec_iter2_ex(j)=k2;
475     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
476     vec_bt2_ex(j)=sum(btseq2)/k2;

```

```

477     vec_violations2_ex(j)=violations2;
478
479     last_vals = conv_ord2_ex(max(end-14,1):end);
480     mat_conv2_ex(:, j) = last_vals;
481
482     for i=2:2:12
483         h=10^(-i);
484
485         % FINITE DIFFERENCES 1
486         JF=@(x) JF_fd1(x,h);
487         HF=@(x) HF_fd1(x,h);
488         tic;
489
490         [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);
491
492         mat_times2_fd1(i/2,j)=toc;
493
494         disp(['Finite differences (classical version) with h=1e-', num2str(i), ' : ', flag2])
495         mat_converged2_fd1(i/2,j)=converged2;
496
497         mat_val2_fd1(i/2,j)=f2;
498         mat_grad2_fd1(i/2,j)=gradf_norm2;
499         mat_iter2_fd1(i/2,j)=k2;
500         mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
501         mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
502         mat_violations2_fd1(i/2,j)=violations2;
503
504
505         last_vals = conv_ord2_df1(max(end-14,1):end);
506         mat_conv2_fd1(i/2, j) = {last_vals};
507
508
509
510         % FINITE DIFFERENCES 2
511         JF=@(x) JF_fd2(x,h);
512         HF=@(x) HF_fd2(x,h);
513         tic;
514
515         [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);
516
517         mat_times2_fd2(i/2,j)=toc;
518
519         disp(['Finite differences (new version) with h=1e-', num2str(i), ' : ', flag2])
520         mat_converged2_fd2(i/2,j)=converged2;
521
522         mat_val2_fd2(i/2,j)=f2;
523         mat_grad2_fd2(i/2,j)=gradf_norm2;
524         mat_iter2_fd2(i/2,j)=k2;
525         mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
526         mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
527         mat_violations2_fd2(i/2,j)=violations2;
528
529         last_vals = conv_ord2_df2(max(end-14,1):end);
530         mat_conv2_fd2(i/2, j) = {last_vals};
531
532     end
533 end
534
535 %% The Plot has the same structure
536 num_initial_points = N + 1;
537 figure;
538 hold on;
539
540 for j = 1:num_initial_points
541     conv_ord_ex = mat_conv2_ex(:,j);
542     plot(1:12, conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
543     hold on;
544     for i = 1:6
545         conv_ord_fd1 = mat_conv2_fd1{i, j};
546         conv_ord_fd2 = mat_conv2_fd2{i, j};

```

```

546         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
547         hold on;
548         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
549         hold on;
550     end
551 end
552
553 title('F79P_10^4_superlinear');
554 xlabel('Iterazione');
555 ylabel('Ordine di Convergenza');
556 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
557 grid on;
558 hold off;
559
560 %% Execution time
561
562 % Exact derivative
563 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
564 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
565 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
566
567 % FD1
568 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
569 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
570 converge
571 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
572
573 % FD2
574 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
575 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
576 converge
577 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
578
579 % Creation of the labels
580 h_exponents = [2, 4, 6, 8, 10, 12];
581 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
582
583 fd1_vals = avg_fd1';
584 fd2_vals = avg_fd2';
585
586 % Table creation
587 rowNames = {'FD1', 'FD2'};
588 columnNames = [h_labels, 'Exact'];
589 data = [fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2];
590 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
591 %display the table
592 disp('Average computation times table (only for successful runs): F79P, n=10^4,
593 superlinear');
594 disp(T4);
595
596 %% Iteration
597
598 vec_times_ex_clean = vec_iter2_ex;
599 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
600 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
601
602 mat_times_fd1_clean = mat_iter2_fd1;
603 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
604 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
605
606 mat_times_fd2_clean = mat_iter2_fd2;
607 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
608 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
609
610 h_exponents = [2, 4, 6, 8, 10, 12];
611 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
612
613 fd1_vals = avg_fd1';
614 fd2_vals = avg_fd2';
615
616 rowNames = {'FD1', 'FD2'};
617 columnNames = [h_labels, 'Exact'];

```

```

616 data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];
617
618 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
619
620 disp('Average computation iteration table (only for successful runs): F79P, n=10^4,
superlinear');
621 disp(T5);
622
623 %% Function value
624
625 vec_times_ex_clean = vec_val2_ex;
626 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
627 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
628
629 mat_times_fd1_clean = mat_val2_fd1;
630 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
631 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
632
633 mat_times_fd2_clean = mat_val2_fd2;
634 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
635 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
636
637 h_exponents = [2, 4, 6, 8, 10, 12];
638 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
639
640 fd1_vals = avg_fd1';
641 fd2_vals = avg_fd2';
642
643 rowNames = {'FD1', 'FD2'};
644 columnNames = [ h_labels, 'Exact'];
645 data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];
646
647 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
648
649 disp('Average computation fmin value table (only for successful runs): F79P, n=10^4,
superlinear');
650 disp(T6);
651
652 %% VIOLATION
653
654 vec_times_ex_clean = vec_violations2_ex;
655 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
656 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
657
658 mat_times_fd1_clean = mat_violations2_fd1;
659 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
660 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
661
662 mat_times_fd2_clean = mat_violations2_fd2;
663 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
664 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
665
666 h_exponents = [2, 4, 6, 8, 10, 12];
667 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
668
669 fd1_vals = avg_fd1';
670 fd2_vals = avg_fd2';
671
672 rowNames = {'FD1', 'FD2'};
673 columnNames = [ h_labels, 'Exact'];
674 data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];
675
676 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
677
678 disp('Average computation violation table (only for successful runs): F79P, n=10^4,
suplinear');
679 disp(T14);
680
681 %% BT-SEQ
682
683 vec_bt_ex_clean = vec_bt2_ex;
684 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
685 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');

```



```

686
687 mat_bt_fdl_clean = mat_bt2_fdl;
688 mat_bt_fdl_clean(mat_converged2_fdl == 0) = NaN;
689 avg_fdl = mean(mat_bt_fdl_clean, 2, 'omitnan');
690
691 mat_bt_fd2_clean = mat_bt2_fd2;
692 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
693 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
694
695 h_exponents = [2, 4, 6, 8, 10, 12];
696 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
697
698 fd1_vals = avg_fdl';
699 fd2_vals = avg_fd2';
700
701 rowNames = {'FD1', 'FD2'};
702 columnNames = [ h_labels, 'Exact'];
703 data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];
704
705 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
706
707 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^4,
708      superlinear');
709 disp(T15);
710
711 %% CG-SEQ
712
713 vec_bt_ex_clean = vec_cg_iter2_ex;
714 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
715 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
716
717 mat_bt_fdl_clean = mat_cg_iter2_fdl;
718 mat_bt_fdl_clean(mat_converged2_fdl == 0) = NaN;
719 avg_fdl = mean(mat_bt_fdl_clean, 2, 'omitnan');
720
721 mat_bt_fd2_clean = mat_cg_iter2_fd2;
722 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
723 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
724
725 h_exponents = [2, 4, 6, 8, 10, 12];
726 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
727
728 fd1_vals = avg_fdl';
729 fd2_vals = avg_fd2';
730
731 rowNames = {'FD1', 'FD2'};
732 columnNames = [ h_labels, 'Exact'];
733 data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];
734
735 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
736
737 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^4,
738      superlinear');
739 disp(T16);
740
741 %% Number of initial point converged
742
743 h_exponents = [2, 4, 6, 8, 10, 12];
744 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
745
746 fd1_vals = sum(mat_converged2_fdl, 2)';
747 fd2_vals = sum(mat_converged2_fd2, 2)';
748
749 rowNames = {'FD1', 'FD2'};
750 columnNames = [ h_labels, 'Exact'];
751 data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex);];
752
753 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
754
755 disp('Number of converged: F79P, n=10^4, superlinear');
756 disp(T17);
757 %save the table in a file xlsx
758 writetable(T4, 'results_f79P_suplin.xlsx', 'Sheet', 'time_4', 'WriteRowNames', true);

```

```

757 writetable(T5, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
758 writetable(T6, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
759 writetable(T14, 'results_f79P_suplin.xlsx', 'Sheet', 'v_4','WriteRowNames', true);
760 writetable(T15, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
761 writetable(T16, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
762 writetable(T17, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);
763
764
765 %% n=105 (1e5)
766
767 rng(345989);
768
769 n=1e5;
770
771 kmax=1.5e3; % maximum number of iterations of Newton method
772 tolgrad=5e-7; % tolerance on gradient norm
773
774
775 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
       system)
776 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
777
778 % Backtracking parameters
779 c1=1e-4;
780 rho=0.50;
781 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
782
783 x0=-1*ones(n,1); % initial point
784 N=10; % number of initial points to be generated
785
786 % Initial points:
787 Mat_points= repmat(x0,1,N+1);
788 rand_mat=2*rand(n, N)-1;
789 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
790
791 % Structure for EXACT derivatives
792 vec_times3_ex=zeros(1,N+1); % vector with execution times
793 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
794 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
795 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
796 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
797 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
798 mat_conv3_ex=zeros(15:N+1); %matrix with che last 15 values of rate of convergence for the
       starting point
799 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
800 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
       condition in Newton method
801
802 JF_ex = @(x) JF_gen(x,true,false,0);
803 HF_ex = @(x) HF_gen(x,true,false,0);
804
805 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
806 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
807 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
808 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
809 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
810 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
811 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
812 mat_conv3_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
       starting point
813 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
814 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
       condition in Newton method
815
816 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
817 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
818
819 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
       x_j) as increment)
820 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
821 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
822 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
823 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations

```

```

824 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
825 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
826 mat_conv3_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
      starting point
827 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
828 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method

829
830 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
831 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
832
833 for j =1:N+1
834     disp(['Condizione_iniziale_n.',num2str(j)])
835
836     % EXACT DERIVATIVES
837     tic;
838
839     [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);
840     vec_times3_ex(j)=toc;
841
842     disp(['Exact_derivatives:',num2str(j)])
843     vec_converged3_ex(j)=converged3;
844
845     vec_val3_ex(j)=f3;
846     vec_grad3_ex(j)=gradf_norm3;
847     vec_iter3_ex(j)=k3;
848     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
849     vec_bt3_ex(j)=sum(btseq3)/k3;
850     vec_violations3_ex(j)=violations3;
851
852     last_vals = conv_ord3_ex(max(end-14,1):end);
853     mat_conv3_ex(:, j) = last_vals;
854
855     for i=2:2:12
856         h=10^(-i);
857
858         % FINITE DIFFERENCES 1
859         JF=@(x) JF_fd1(x,h);
860         HF=@(x) HF_fd1(x,h);
861         tic;
862
863         [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);
864         mat_times3_fd1(i/2,j)=toc;
865
866
867         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag3])
868         mat_converged3_fd1(i/2,j)=converged3;
869
870         mat_val3_fd1(i/2,j)=f3;
871         mat_grad3_fd1(i/2,j)=gradf_norm3;
872         mat_iter3_fd1(i/2,j)=k3;
873         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
874         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
875         mat_violations3_fd1(i/2,j)=violations3;
876
877
878         last_vals = conv_ord3_df1(max(end-14,1):end);
879         mat_conv3_fd1(i/2, j) = {last_vals};
880
881
882         % FINITE DIFFERENCES 2
883         JF=@(x) JF_fd2(x,h);
884         HF=@(x) HF_fd2(x,h);
885         tic;
886
887
888         [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);

```

```

889     mat_times3_fd2(i/2,j)=toc;
890
891     disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag3])
892     mat_converged3_fd2(i/2,j)=converged2;
893
894     mat_val3_fd2(i/2,j)=f3;
895     mat_grad3_fd2(i/2,j)=gradf_norm3;
896     mat_iter3_fd2(i/2,j)=k3;
897     mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
898     mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
899     mat_violations3_fd2(i/2,j)=violations3;
900
901     last_vals = conv_ord3_df2(max(end-14,1):end);
902     mat_conv3_fd2(i/2, j) = {last_vals};
903
904
905     end
906 end
907
908 %% The plot has the same structure as n=10^3
909 num_initial_points = N + 1;
910 figure;
911 hold on;
912
913 for j = 1:num_initial_points
914     conv_ord_ex = mat_conv3_ex(:,j);
915     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
916     hold on;
917     for i =1:6
918         conv_ord_fd1 = mat_conv3_fd1{i, j};
919         conv_ord_fd2 = mat_conv3_fd2{i, j};
920         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
921         hold on;
922         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
923         hold on;
924     end
925 end
926
927 title('F79P_10^5_superlinear');
928 xlabel('Iterazione');
929 ylabel('Ordine di Convergenza');
930 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
931 grid on;
932 hold off;
933
934 %% Time
935
936 vec_times_ex_clean = vec_times3_ex;
937 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
938 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
939
940 mat_times_fd1_clean = mat_times3_fd1;
941 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
942 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
943
944 mat_times_fd2_clean = mat_times3_fd2;
945 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
946 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
947
948 h_exponents = [2, 4, 6, 8, 10, 12];
949 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
950
951 fd1_vals = avg_fd1';
952 fd2_vals = avg_fd2';
953
954 rowNames = {'FD1', 'FD2'};
955 columnNames = [h_labels, 'Exact'];
956 data = [fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3];
957
958 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
959
960 disp('Average computation times table (only for successful runs): F79P, n=10^5,
superlinear');

```

```

961 disp(T7);
962
963 %% Iteration
964
965 vec_times_ex_clean = vec_iter3_ex;
966 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
967 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
968
969 mat_times_fd1_clean = mat_iter3_fd1;
970 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
971 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
972
973 mat_times_fd2_clean = mat_iter3_fd2;
974 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
975 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
976
977 h_exponents = [2, 4, 6, 8, 10, 12];
978 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
979
980 fd1_vals = avg_fd1';
981 fd2_vals = avg_fd2';
982
983 rowNames = {'FD1', 'FD2'};
984 columnNames = [h_labels, 'Exact'];
985 data = [fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3];
986
987 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
988
989 disp('Average computation iteration table (only for successful runs): F79P, n=10^5,
    superlinear');
990 disp(T8);
991
992 %% function value
993
994 vec_times_ex_clean = vec_val3_ex;
995 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
996 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
997
998 mat_times_fd1_clean = mat_val3_fd1;
999 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1000 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1001
1002 mat_times_fd2_clean = mat_val3_fd2;
1003 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1004 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1005
1006 h_exponents = [2, 4, 6, 8, 10, 12];
1007 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1008
1009 fd1_vals = avg_fd1';
1010 fd2_vals = avg_fd2';
1011
1012 rowNames = {'FD1', 'FD2'};
1013 columnNames = [h_labels, 'Exact'];
1014 data = [fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3];
1015
1016 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1017
1018 disp('Average computation fmin value table (only for successful runs): F79P, n=10^5,
    superlinear');
1019 disp(T9);
1020
1021 %% VIOLATION
1022
1023 vec_times_ex_clean = vec_violations3_ex;
1024 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
1025 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
1026
1027 mat_times_fd1_clean = mat_violations3_fd1;
1028 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1029 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1030
1031 mat_times_fd2_clean = mat_violations3_fd2;

```

```

1032 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1033 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1034
1035 h_exponents = [2, 4, 6, 8, 10, 12];
1036 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1037
1038 fd1_vals = avg_fd1';
1039 fd2_vals = avg_fd2';
1040
1041 rowNames = {'FD1', 'FD2'};
1042 columnNames = [h_labels, 'Exact'];
1043 data = [fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3];
1044
1045 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1046
1047 disp('Average computation violation table (only for successful runs): F79P, n=10^5,
1048      superlinear');
1049 disp(T18);
1050
1051 %% BT-SEQ
1052
1053 vec_bt_ex_clean = vec_bt3_ex;
1054 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1055 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1056
1057 mat_bt_fd1_clean = mat_bt3_fd1;
1058 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1059 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1060
1061 mat_bt_fd2_clean = mat_bt3_fd2;
1062 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1063 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1064
1065 h_exponents = [2, 4, 6, 8, 10, 12];
1066 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1067
1068 fd1_vals = avg_fd1';
1069 fd2_vals = avg_fd2';
1070
1071 rowNames = {'FD1', 'FD2'};
1072 columnNames = [h_labels, 'Exact'];
1073 data = [fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3];
1074
1075 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1076
1077 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^5,
1078      superlinear');
1079 disp(T19);
1080
1081 %% CG-SEQ
1082
1083 vec_bt_ex_clean = vec_cg_iter3_ex;
1084 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1085 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1086
1087 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1088 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1089 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1090
1091 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1092 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1093 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1094
1095 h_exponents = [2, 4, 6, 8, 10, 12];
1096 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1097
1098 fd1_vals = avg_fd1';
1099 fd2_vals = avg_fd2';
1100
1101 rowNames = {'FD1', 'FD2'};
1102 columnNames = [h_labels, 'Exact'];
1103 data = [fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1104

```

```

1103 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1104
1105 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^5,
      superlinear');
1106 disp(T20);
1107
1108 %% Number of initial condition converged
1109
1110 h_exponents = [2, 4, 6, 8, 10, 12];
1111 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1112
1113 fd1_vals = sum(mat_converged3_fd1,2)';
1114 fd2_vals = sum(mat_converged3_fd2,2)';
1115
1116 rowNames = {'FD1', 'FD2'};
1117 columnNames = [h_labels, 'Exact'];
1118 data = [fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1119
1120 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1121
1122 disp('Number of converged: F79P, n=10^5, superlinear');
1123 disp(T21);
1124
1125 writetable(T7, 'results_f79P_suplin.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1126 writetable(T8, 'results_f79P_suplin.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1127 writetable(T9, 'results_f79P_suplin.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1128 writetable(T18, 'results_f79P_suplin.xlsx', 'Sheet', 'v_5', 'WriteRowNames', true);
1129 writetable(T19, 'results_f79P_suplin.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1130 writetable(T20, 'results_f79P_suplin.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1131 writetable(T21, 'results_f79P_suplin.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1132
1133
1134
1135 %% Creation of the table with the result of exact derivatives
1136 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1137         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1138         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1139         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1140         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1141         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1142         sum(vec_converged1_ex), sum(vec_converged2_ex), sum(vec_converged3_ex)];
1143
1144 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
1145             'Average iter cg', 'N converged'};
1146 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1147
1148 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1149 disp(T_compare)
1150
1151 writetable(T_compare, 'results_f79P_suplin.xlsx', 'Sheet', 'ExactComparison', '
      WriteRowNames', true);

```

```

1 %% FUNCTION 79 PRECONDITIONING QUAD (with different initial points)- with exact
2   derivatives and finite differences
3
4 sparse=true;
5
6 F = @(x) F79(x); % Defining F79 as function handle
7 JF_gen = @(x,exact,fin_dif2,h) JF79(x,exact,fin_dif2,h); % Defining JF79 as function
8   handle
9 HF_gen= @(x,exact,fin_dif2,h) HF79(x,sparse,exact,fin_dif2,h); % Defining HF79 as
10  function handle (sparse version)
11
12 load forcing_terms.mat % possible terms for adaptive tolerance
13
14 %% n=10^3 (1e3)
15
16 rng(345989);
17
18 n=1e3;
19
20 kmax=1.5e3; % maximum number of iterations of Newton method
21 tolgrad=5e-7; % tolerance on gradient norm

```

```

19
20 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
    system)
21 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
22
23 % Backtracking parameters
24 c1=1e-3;
25 rho=0.50;
26 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
27
28 x0=-1*ones(n,1); % initial point
29 N=10; % number of initial points to be generated
30
31 % Initial points:
32 Mat_points=repmat(x0,1,N+1);
33 rand_mat=2*rand(n, N)-1;
34 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
41 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
42 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
43 mat_conv_ex=zeros(15, N+1);%matrix with the last 15 values of rate of convergence for the
    starting point
44 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
46
47 JF_ex = @(x) JF_gen(x,true,false,0);
48 HF_ex = @(x) HF_gen(x,true,false,0);
49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv_fd1=cell(6, N+1);%matrix with the last 15 values of rate of convergence for the
    starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv_fd2=cell(6,N+1); %matrix with the last 15 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1
79     disp(['Condizione iniziale n. ',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83

```



```

84 [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);
85
86 vec_times1_ex(j)=toc;
87
88 disp(['Exact derivatives: ', flag1])
89 vec_converged1_ex(j)=converged1;
90
91 vec_val1_ex(j)=f1;
92 vec_grad1_ex(j)=gradf_norm1;
93 vec_iter1_ex(j)=k1;
94 vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
95 vec_bt1_ex(j)=sum(btseq1)/k1;
96 vec_violations1_ex(j)=violations1;
97
98 last_vals = conv_ord1_ex(max(end-14,1):end);
99 mat_conv_ex(:, j) = last_vals;
100
101
102 for i=2:2:12
103 h=10^(-i);
104
105 % FINITE DIFFERENCES 1
106 JF=@(x) JF_fd1(x,h);
107 HF=@(x) HF_fd1(x,h);
108 tic;
109
110 [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);
111
112 mat_times1_fd1(i/2,j)=toc;
113
114 disp(['Finite differences (classical version) with h=1e-', num2str(i), ': ', flag1])
115 mat_converged1_fd1(i/2,j)=converged1;
116
117 mat_val1_fd1(i/2,j)=f1;
118 mat_grad1_fd1(i/2,j)=gradf_norm1;
119 mat_iter1_fd1(i/2,j)=k1;
120 mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
121 mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
122 mat_violations1_fd1(i/2,j)=violations1;
123
124
125 last_vals = conv_ord1_df1(max(end-14,1):end);
126 mat_conv_fd1(i/2, j) = {last_vals};
127
128
129
130 % FINITE DIFFERENCES 2
131 JF=@(x) JF_fd2(x,h);
132 HF=@(x) HF_fd2(x,h);
133 tic;
134
135
136 mat_times1_fd2(i/2,j)=toc;
137
138 disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag1])
139 mat_converged1_fd2(i/2,j)=converged1;
140 mat_val1_fd2(i/2,j)=f1;
141 mat_grad1_fd2(i/2,j)=gradf_norm1;
142 mat_iter1_fd2(i/2,j)=k1;
143 mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
144 mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
145 mat_violations1_fd2(i/2,j)=violations1;
146
147 last_vals = conv_ord1_df2(max(end-14,1):end);
148 mat_conv_fd2(i/2, j) = {last_vals};
149
150
151 end
152 end

```

```

153
154
155 %% Plot of the last 12 values of experimentale rate of convergence
156 num_initial_points = N + 1;
157 figure;
158 hold on;
159
160 % Plot for every initial condition
161 for j = 1:num_initial_points
162     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
163     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
164     hold on;
165     for i =1:6
166         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
167         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
168         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
169         hold on;
170         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
171         hold on;
172     end
173 end
174
175 % title and legend
176 title('F79P_10^3_quadratic');
177 xlabel('Iterazione');
178 ylabel('Ordine_di_Convergenza');
179 legend({'Exact_Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
180 grid on;
181 hold off;
182
183
184 %% Execution Time
185
186 % Exact Derivative
187 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
188 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
189 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
190
191 % FD1
192 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
193 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
194 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
195
196 % FD2
197 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
198 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
199 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
200
201 % Creation of the labels
202 h_exponents = [2, 4, 6, 8, 10, 12];
203 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
204
205 fd1_vals = avg_fd1';
206 fd2_vals = avg_fd2';
207
208 % Table costruction with exact for both the row
209 rowNames = {'FD1', 'FD2'};
210 columnNames = [ h_labels, 'Exact'];
211 data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
212 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
213
214 % visualization
215 disp('Average_computation_times_table_(only_for_successful_runs):_F79P,_n=10^3,_quadratic
    ');
216 disp(T1);
217
218
219 %% All the tables has the same structure
220 %% Iteration
221
222 vec_times_ex_clean = vec_iter1_ex;

```

```

223 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
224 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
225
226 mat_times_fd1_clean = mat_iter1_fd1;
227 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
228 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
229
230 mat_times_fd2_clean = mat_iter1_fd2;
231 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
232 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
233
234 h_exponents = [2, 4, 6, 8, 10, 12];
235 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
236
237 fd1_vals = avg_fd1';
238 fd2_vals = avg_fd2';
239
240 rowNames = {'FD1', 'FD2'};
241 columnNames = [h_labels, 'Exact'];
242 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1];
243
244 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
245
246 disp('Average computation iteration table (only for successful runs): F79P, n=10^3, quadratic');
247 disp(T2);
248
249 %% F value
250
251 vec_times_ex_clean = vec_val1_ex;
252 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
253 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
254
255 mat_times_fd1_clean = mat_val1_fd1;
256 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
257 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
258
259 mat_times_fd2_clean = mat_val1_fd2;
260 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
261 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
262
263 h_exponents = [2, 4, 6, 8, 10, 12];
264 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
265
266 fd1_vals = avg_fd1';
267 fd2_vals = avg_fd2';
268
269 rowNames = {'FD1', 'FD2'};
270 columnNames = [h_labels, 'Exact'];
271 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1];
272
273 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
274
275 disp('Average computation fmin value table (only for successful runs): F79P, n=10^3, quadratic');
276 disp(T3);
277
278 %% VIOLATION
279
280 vec_times_ex_clean = vec_violations1_ex;
281 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
282 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
283
284 mat_times_fd1_clean = mat_violations1_fd1;
285 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
286 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
287
288 mat_times_fd2_clean = mat_violations1_fd2;
289 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
290 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
291
292 h_exponents = [2, 4, 6, 8, 10, 12];
293 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```

294 %
295 fd1_vals = avg_fd1';
296 fd2_vals = avg_fd2';
297
298
299 rowNames = {'FD1', 'FD2'};
300 columnNames = [ h_labels, 'Exact'];
301 data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];
302
303 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
304
305 disp('Average computation violation table (only for successful runs): F79P, n=10^3, quadratic');
306 disp(T10);
307
308
309 %% BT-SEQ
310 vec_bt_ex_clean = vec_bt1_ex;
311 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
312 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
313
314 mat_bt_fd1_clean = mat_bt1_fd1;
315 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
316 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
317
318 mat_bt_fd2_clean = mat_bt1_fd2;
319 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
320 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
321
322 h_exponents = [2, 4, 6, 8, 10, 12];
323 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
324
325 fd1_vals = avg_fd1';
326 fd2_vals = avg_fd2';
327
328 rowNames = {'FD1', 'FD2'};
329 columnNames = [ h_labels, 'Exact'];
330 data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
331
332 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
333
334 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^3, quadratic');
335 disp(T11);
336
337 %% CG-SEQ
338
339 vec_bt_ex_clean = vec_cg_iter1_ex;
340 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
341 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
342
343 mat_bt_fd1_clean = mat_cg_iter1_fd1;
344 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
345 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
346
347 mat_bt_fd2_clean = mat_cg_iter1_fd2;
348 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
349 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
350
351 h_exponents = [2, 4, 6, 8, 10, 12];
352 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
353
354 fd1_vals = avg_fd1';
355 fd2_vals = avg_fd2';
356
357 rowNames = {'FD1', 'FD2'};
358 columnNames = [ h_labels, 'Exact'];
359 data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
360
361 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
362
363 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^3, quadratic');

```

```

364 disp(T12);
365
366 %% Number of starting point converged
367
368 h_exponents = [2, 4, 6, 8, 10, 12];
369 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
370
371 fd1_vals = sum(mat_converged1_fd1,2)';
372 fd2_vals = sum(mat_converged1_fd2,2)';
373
374 rowNames = {'FD1', 'FD2'};
375 columnNames = [ h_labels, 'Exact'];
376 data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex)];
377
378 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
379
380 disp('Number of converged F79P, n=10^3, quadratic');
381 disp(T13);
382 %save the table in a file.xlsx
383 writetable(T1, 'results_f79P_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
384 writetable(T2, 'results_f79P_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
385 writetable(T3, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
386 writetable(T10, 'results_f79P_quad.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
387 writetable(T11, 'results_f79P_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
388 writetable(T12, 'results_f79P_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
389 writetable(T13, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);
390
391
392
393
394
395 %% n=10^4 (1e4)
396
397 rng(345989);
398
399 n=1e4;
400
401 kmax=1.5e3; % maximum number of iterations of Newton method
402 tolgrad=5e-7; % tolerance on gradient norm
403
404 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
405 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
406
407 % Backtracking parameters
408 c1=1e-4;
409 rho=0.50;
410 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
411
412 x0=-1*ones(n,1); % initial point
413 N=10; % number of initial points to be generated
414
415 % Initial points:
416 Mat_points= repmat(x0,1,N+1);
417 rand_mat=2*rand(n, N)-1;
418 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
419
420 % Structure for EXACT derivatives
421 vec_times2_ex=zeros(1,N+1); % vector with execution times
422 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
423 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
424 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
425 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
426 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
427 mat_conv2_ex=zeros(15, N+1);%matrix with the last 15 values of rate of convergence for
    the starting point
428 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
429 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
430
431 JF_ex = @(x) JF_gen(x,true,false,0);
432 HF_ex = @(x) HF_gen(x,true,false,0);
433

```

```

444 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
445 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
446 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
447 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
448 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
449 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
450 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
451 mat_conv2_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
452 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
453 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
454
455 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
456 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
457
458 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
459 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
460 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
461 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
462 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
463 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
464 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
465 mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
466 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
467 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
468
469 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
470 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
471
472 for j =1:N+1
473     disp(['Condizione_iniziale_',num2str(j)])
474
475     % EXACT DERIVATIVES
476     tic;
477
478     [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);
479
480     vec_times2_ex(j)=toc;
481
482     disp(['Exact_derivatives:',flag2])
483     vec_converged2_ex(j)=converged2;
484
485     vec_val2_ex(j)=f2;
486     vec_grad2_ex(j)=gradf_norm2;
487     vec_iter2_ex(j)=k2;
488     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
489     vec_bt2_ex(j)=sum(btseq2)/k2;
490     vec_violations2_ex(j)=violations2;
491
492     last_vals = conv_ord2_ex(max(end-14,1):end);
493     mat_conv2_ex(:, j) = last_vals;
494
495     for i=2:2:12
496         h=10^(-i);
497
498         % FINITE DIFFERENCES 1
499         JF=@(x) JF_fd1(x,h);
500         HF=@(x) HF_fd1(x,h);
501         tic;
502
503         [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);
504
505         mat_times2_fd1(i/2,j)=toc;
506
507         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag2])

```

```

498     mat_converged2_fd1(i/2,j)=converged2;
499
500     mat_val2_fd1(i/2,j)=f2;
501     mat_grad2_fd1(i/2,j)=gradf_norm2;
502     mat_iter2_fd1(i/2,j)=k2;
503     mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
504     mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
505     mat_violations2_fd1(i/2,j)=violations2;
506
507
508     last_vals = conv_ord2_df1(max(end-14,1):end);
509     mat_conv2_fd1(i/2, j) = {last_vals};
510
511
512
513     % FINITE DIFFERENCES 2
514     JF=@(x) JF_fd2(x,h);
515     HF=@(x) HF_fd2(x,h);
516     tic;
517     [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);
518     mat_times2_fd2(i/2,j)=toc;
519
520     disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag2])
521     mat_converged2_fd2(i/2,j)=converged2;
522
523     mat_val2_fd2(i/2,j)=f2;
524     mat_grad2_fd2(i/2,j)=gradf_norm2;
525     mat_iter2_fd2(i/2,j)=k2;
526     mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
527     mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
528     mat_violations2_fd2(i/2,j)=violations2;
529
530     last_vals = conv_ord2_df2(max(end-14,1):end);
531     mat_conv2_fd2(i/2, j) = {last_vals};
532
533
534     end
535 end
536
537
538 %% The Plot has the same structure
539 num_initial_points = N + 1;
540 figure;
541 hold on;
542
543 for j = 1:num_initial_points
544     conv_ord_ex = mat_conv2_ex(:,j);
545     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
546     hold on;
547     for i =1:6
548         conv_ord_fd1 = mat_conv2_fd1{i, j};
549         conv_ord_fd2 = mat_conv2_fd2{i, j};
550         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
551         hold on;
552         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
553         hold on;
554     end
555 end
556
557 title('F79P_10^4_Quadratic');
558 xlabel('Iterazione');
559 ylabel('Ordine di Convergenza');
560 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
561 grid on;
562 hold off;
563
564
565
566 %% Execution time
567
568 % Exact derivative

```

```

569 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
570 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
571 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
572
573 % FD1
574 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
575 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
576 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
577
578 % FD2
579 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
580 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
581 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
582
583 % Creation of the labels
584 h_exponents = [2, 4, 6, 8, 10, 12];
585 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
586
587 fd1_vals = avg_fd1';
588 fd2_vals = avg_fd2';
589
590 % Table creation
591 rowNames = {'FD1', 'FD2'};
592 columnNames = [ h_labels, 'Exact'];
593 data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
594 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
595 %display the table
596 disp('Average computation times table (only for successful runs): F79P, n=10^4, quadratic
    ');
597 disp(T4);
598
599 %% Iteration
600
601 vec_times_ex_clean = vec_iter2_ex;
602 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
603 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
604
605 mat_times_fd1_clean = mat_iter2_fd1;
606 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
607 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
608
609 mat_times_fd2_clean = mat_iter2_fd2;
610 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
611 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
612
613 h_exponents = [2, 4, 6, 8, 10, 12];
614 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
615
616 fd1_vals = avg_fd1';
617 fd2_vals = avg_fd2';
618
619 rowNames = {'FD1', 'FD2'};
620 columnNames = [ h_labels, 'Exact'];
621 data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];
622
623 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
624
625 disp('Average computation iteration table (only for successful runs): F79P, n=10^4,
    quadratic');
626 disp(T5);
627
628 %% Function value
629
630 vec_times_ex_clean = vec_val2_ex;
631 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
632 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
633
634 mat_times_fd1_clean = mat_val2_fd1;
635 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
636 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
637

```



```

638 mat_times_fd2_clean = mat_val2_fd2;
639 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
640 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
641
642 h_exponents = [2, 4, 6, 8, 10, 12];
643 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
644
645 fd1_vals = avg_fd1';
646 fd2_vals = avg_fd2';
647
648 rowNames = {'FD1', 'FD2'};
649 columnNames = [h_labels, 'Exact'];
650 data = [fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2];
651
652 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
653
654 disp('Average computation fmin value table (only for successful runs): F79P, n=10^4, quadratic');
655 disp(T6);
656
657 %% VIOLATION
658
659 vec_times_ex_clean = vec_violations2_ex;
660 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
661 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
662
663 mat_times_fd1_clean = mat_violations2_fd1;
664 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
665 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
666
667 mat_times_fd2_clean = mat_violations2_fd2;
668 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
669 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
670
671 h_exponents = [2, 4, 6, 8, 10, 12];
672 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
673
674 fd1_vals = avg_fd1';
675 fd2_vals = avg_fd2';
676
677 rowNames = {'FD1', 'FD2'};
678 columnNames = [h_labels, 'Exact'];
679 data = [fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2];
680
681 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
682
683 disp('Average computation violation table (only for successful runs): F79P, n=10^4, quadratic');
684 disp(T14);
685
686 %% BT-SEQ
687
688 vec_bt_ex_clean = vec_bt2_ex;
689 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
690 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
691
692 mat_bt_fd1_clean = mat_bt2_fd1;
693 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
694 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
695
696 mat_bt_fd2_clean = mat_bt2_fd2;
697 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
698 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
699
700 h_exponents = [2, 4, 6, 8, 10, 12];
701 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
702
703 fd1_vals = avg_fd1';
704 fd2_vals = avg_fd2';
705
706 rowNames = {'FD1', 'FD2'};
707 columnNames = [h_labels, 'Exact'];
708 data = [fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2];

```

```

709
710 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
711
712 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^4, quadratic');
713 disp(T15);
714
715 %% CG-SEQ
716
717 vec_bt_ex_clean = vec_cg_iter2_ex;
718 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
719 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
720
721 mat_bt_fd1_clean = mat_cg_iter2_fd1;
722 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
723 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
724
725 mat_bt_fd2_clean = mat_cg_iter2_fd2;
726 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
727 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
728
729 h_exponents = [2, 4, 6, 8, 10, 12];
730 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
731
732 fd1_vals = avg_fd1';
733 fd2_vals = avg_fd2';
734
735 rowNames = {'FD1', 'FD2'};
736 columnNames = [h_labels, 'Exact'];
737 data = [fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2];
738
739 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
740
741 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^4, quadratic');
742 disp(T16);
743
744 %% Number of initial point converged
745
746 h_exponents = [2, 4, 6, 8, 10, 12];
747 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
748
749 fd1_vals = sum(mat_converged2_fd1, 2)';
750 fd2_vals = sum(mat_converged2_fd2, 2)';
751
752 rowNames = {'FD1', 'FD2'};
753 columnNames = [h_labels, 'Exact'];
754 data = [fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex)];
755
756 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
757
758 disp('Number of converged: F79P, n=10^4, quadratic');
759 disp(T17);
760 %save the table if a file.xlsx
761 writetable(T4, 'results_f79P_quad.xlsx', 'Sheet', 'time_4', 'WriteRowNames', true);
762 writetable(T5, 'results_f79P_quad.xlsx', 'Sheet', 'niter_4', 'WriteRowNames', true);
763 writetable(T6, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_4', 'WriteRowNames', true);
764 writetable(T14, 'results_f79P_quad.xlsx', 'Sheet', 'v_4', 'WriteRowNames', true);
765 writetable(T15, 'results_f79P_quad.xlsx', 'Sheet', 'bt_4', 'WriteRowNames', true);
766 writetable(T16, 'results_f79P_quad.xlsx', 'Sheet', 'cg_4', 'WriteRowNames', true);
767 writetable(T17, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv4', 'WriteRowNames', true);
768
769
770
771
772 %% n=10^5 (1e5)
773
774 rng(345989);
775
776 n=1e5;
777
778 kmax=1.5e3; % maximum number of iterations of Newton method
779 tolgrad=5e-7; % tolerance on gradient norm

```

```

780
781 cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
      system)
782 z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)
783
784 % Backtracking parameters
785 c1=1e-4;
786 rho=0.50;
787 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
788
789 x0=-1*ones(n,1); % initial point
790 N=10; % number of initial points to be generated
791
792 % Initial points:
793 Mat_points=repmat(x0,1,N+1);
794 rand_mat=2*rand(n, N)-1;
795 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
796
797 % Structure for EXACT derivatives
798 vec_times3_ex=zeros(1,N+1); % vector with execution times
799 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
800 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
801 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
802 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
803 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
804 mat_conv3_ex=zeros(15:N+1); %matrix with che last 15 values of rate of convergence for the
      starting point
805 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
806 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
      condition in Newton method
807
808 JF_ex = @(x) JF_gen(x,true,false,0);
809 HF_ex = @(x) HF_gen(x,true,false,0);
810
811 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
812 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
813 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
814 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
815 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
816 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
817 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
818 mat_conv3_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
      starting point
819 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
820 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
821
822 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
823 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
824
825 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
      x_j) as increment)
826 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
827 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
828 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
829 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
830 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
831 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
832 mat_conv3_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
      starting point
833 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
834 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
835
836 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
837 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
838
839 for j =1:N+1
840     disp(['Condizione iniziale n. ',num2str(j)])
841
842     % EXACT DERIVATIVES
843     tic;
844

```

```

845 [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);
846
847 vec_times3_ex(j)=toc;
848
849 disp(['Exact derivatives: ', flag3])
850 vec_converged3_ex(j)=converged3;
851 vec_val3_ex(j)=f3;
852 vec_grad3_ex(j)=gradf_norm3;
853 vec_iter3_ex(j)=k3;
854 vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
855 vec_bt3_ex(j)=sum(btseq3)/k3;
856 vec_violations3_ex(j)=violations3;
857
858 last_vals = conv_ord3_ex(max(end-14,1):end);
859 mat_conv3_ex(:, j) = last_vals;
860
861 for i=2:2:12
862 h=10^(-i);
863
864 % FINITE DIFFERENCES 1
865 JF=@(x) JF_fd1(x,h);
866 HF=@(x) HF_fd1(x,h);
867 tic;
868
869 [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);
870 mat_times3_fd1(i/2,j)=toc;
871
872 disp(['Finite differences (classical version) with h=1e-', num2str(i), ': ', flag3])
873 mat_converged3_fd1(i/2,j)=converged3;
874 mat_val3_fd1(i/2,j)=f3;
875 mat_grad3_fd1(i/2,j)=gradf_norm3;
876 mat_iter3_fd1(i/2,j)=k3;
877 mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
878 mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
879 mat_violations3_fd1(i/2,j)=violations3;
880
881
882 last_vals = conv_ord3_df1(max(end-14,1):end);
883 mat_conv3_fd1(i/2, j) = {last_vals};
884
885
886
887 % FINITE DIFFERENCES 2
888 JF=@(x) JF_fd2(x,h);
889 HF=@(x) HF_fd2(x,h);
890 tic;
891
892
893 [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);
894 mat_times3_fd2(i/2,j)=toc;
895
896 disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag3])
897 mat_converged3_fd2(i/2,j)=converged2;
898 mat_val3_fd2(i/2,j)=f3;
899 mat_grad3_fd2(i/2,j)=gradf_norm3;
900 mat_iter3_fd2(i/2,j)=k3;
901 mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
902 mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
903 mat_violations3_fd2(i/2,j)=violations3;
904
905 last_vals = conv_ord3_df2(max(end-14,1):end);
906 mat_conv3_fd2(i/2, j) = {last_vals};
907
908
909 end
910 end
911

```

```

912 %% The plot has the same structure as n=10^3
913 num_initial_points = N + 1;
914 figure;
915 hold on;
916
917 for j = 1:num_initial_points
918     conv_ord_ex = mat_conv3_ex(:,j);
919     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
920     hold on;
921     for i =1:6
922         conv_ord_fd1 = mat_conv3_fd1{i, j};
923         conv_ord_fd2 = mat_conv3_fd2{i, j};
924         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
925         hold on;
926         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
927         hold on;
928     end
929 end
930
931 title('F79P_10^5_quadratic');
932 xlabel('Iterazione');
933 ylabel('Ordine di Convergenza');
934 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
935 grid on;
936 hold off;
937
938 %% Time
939
940 vec_times_ex_clean = vec_times3_ex;
941 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
942 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
943
944 mat_times_fd1_clean = mat_times3_fd1;
945 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
946 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
947
948 mat_times_fd2_clean = mat_times3_fd2;
949 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
950 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
951
952 h_exponents = [2, 4, 6, 8, 10, 12];
953 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
954
955 fd1_vals = avg_fd1';
956 fd2_vals = avg_fd2';
957
958 rowNames = {'FD1', 'FD2'};
959 columnNames = [h_labels, 'Exact'];
960 data = [fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3];
961
962 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
963
964 disp('Average computation times table (only for successful runs): F79P, n=10^5, quadratic');
965 disp(T7);
966
967 %% Iteration
968
969 vec_times_ex_clean = vec_iter3_ex;
970 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
971 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
972
973 mat_times_fd1_clean = mat_iter3_fd1;
974 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
975 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
976
977 mat_times_fd2_clean = mat_iter3_fd2;
978 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
979 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
980
981 h_exponents = [2, 4, 6, 8, 10, 12];
982 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
983

```

```

984 fd1_vals = avg_fd1';
985 fd2_vals = avg_fd2';
986
987 rowNames = {'FD1', 'FD2'};
988 columnNames = [ h_labels, 'Exact'];
989 data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];
990
991 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
992
993 disp('Average computation iteration table (only for successful runs): F79P, n=10^5, quadratic');
994 disp(T8);
995
996 %% function value
997
998 vec_times_ex_clean = vec_val3_ex;
999 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
1000 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
1001
1002 mat_times_fd1_clean = mat_val3_fd1;
1003 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1004 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1005
1006 mat_times_fd2_clean = mat_val3_fd2;
1007 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1008 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1009
1010 h_exponents = [2, 4, 6, 8, 10, 12];
1011 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1012
1013 fd1_vals = avg_fd1';
1014 fd2_vals = avg_fd2';
1015
1016 rowNames = {'FD1', 'FD2'};
1017 columnNames = [ h_labels, 'Exact'];
1018 data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];
1019
1020 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1021
1022 disp('Average computation fmin value table (only for successful runs): F79P, n=10^5, quadratic');
1023 disp(T9);
1024
1025 %% VIOLATION
1026
1027 vec_times_ex_clean = vec_violations3_ex;
1028 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
1029 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
1030
1031 mat_times_fd1_clean = mat_violations3_fd1;
1032 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1033 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1034
1035 mat_times_fd2_clean = mat_violations3_fd2;
1036 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1037 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1038
1039 h_exponents = [2, 4, 6, 8, 10, 12];
1040 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1041
1042 fd1_vals = avg_fd1';
1043 fd2_vals = avg_fd2';
1044
1045 rowNames = {'FD1', 'FD2'};
1046 columnNames = [ h_labels, 'Exact'];
1047 data = [ fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3;];
1048
1049 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1050
1051 disp('Average computation violation table (only for successful runs): F79P, n=10^5, quadratic');
1052 disp(T18);
1053

```

```

1054 %% BT-SEQ
1055
1056 vec_bt_ex_clean = vec_bt3_ex;
1057 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1058 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1059
1060 mat_bt_fd1_clean = mat_bt3_fd1;
1061 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1062 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1063
1064 mat_bt_fd2_clean = mat_bt3_fd2;
1065 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1066 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1067
1068 h_exponents = [2, 4, 6, 8, 10, 12];
1069 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1070
1071 fd1_vals = avg_fd1';
1072 fd2_vals = avg_fd2';
1073
1074 rowNames = {'FD1', 'FD2'};
1075 columnNames = [h_labels, 'Exact'];
1076 data = [fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3];
1077
1078 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1079
1080 disp('Average computation bt iteration table (only for successful runs): F79P, n=10^5, quadratic');
1081 disp(T19);
1082
1083 %% CG-SEQ
1084
1085 vec_cg_iter3_ex = vec_cg_iter3_ex;
1086 vec_cg_iter3_ex(vec_converged3_ex == 0) = NaN;
1087 avg_exact_cg3 = mean(vec_cg_iter3_ex, 'omitnan');
1088
1089 mat_cg_iter3_fd1 = mat_cg_iter3_fd1;
1090 mat_cg_iter3_fd1(mat_converged3_fd1 == 0) = NaN;
1091 avg_fd1 = mean(mat_cg_iter3_fd1, 2, 'omitnan');
1092
1093 mat_cg_iter3_fd2 = mat_cg_iter3_fd2;
1094 mat_cg_iter3_fd2(mat_converged3_fd2 == 0) = NaN;
1095 avg_fd2 = mean(mat_cg_iter3_fd2, 2, 'omitnan');
1096
1097 h_exponents = [2, 4, 6, 8, 10, 12];
1098 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1099
1100 fd1_vals = avg_fd1';
1101 fd2_vals = avg_fd2';
1102
1103 rowNames = {'FD1', 'FD2'};
1104 columnNames = [h_labels, 'Exact'];
1105 data = [fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1106
1107 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1108
1109 disp('Average computation cg iteration table (only for successful runs): F79P, n=10^5, quadratic');
1110 disp(T20);
1111
1112 %% Number of initial condition converged
1113
1114 h_exponents = [2, 4, 6, 8, 10, 12];
1115 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1116
1117 fd1_vals = sum(mat_converged3_fd1, 2)';
1118 fd2_vals = sum(mat_converged3_fd2, 2)';
1119
1120 rowNames = {'FD1', 'FD2'};
1121 columnNames = [h_labels, 'Exact'];
1122 data = [fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1123
1124 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);

```

```

1125 disp('Number of converged: F79P, n=10^5, quadratic');
1126 disp(T21);
1127 %save the tables
1128 writetable(T7, 'results_f79P_quad.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1129 writetable(T8, 'results_f79P_quad.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1130 writetable(T9, 'results_f79P_quad.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1131 writetable(T18, 'results_f79P_quad.xlsx', 'Sheet', 'v_5', 'WriteRowNames', true);
1132 writetable(T19, 'results_f79P_quad.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1133 writetable(T20, 'results_f79P_quad.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1134 writetable(T21, 'results_f79P_quad.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1135
1136
1137
1138 %% table with the result of the exact derivatives
1139 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1140         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1141         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1142         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1143         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1144         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1145         sum(vec_converged1_ex), sum(vec_converged2_ex), sum(vec_converged3_ex)];
1146
1147 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
1148             ', 'Average iter cg', 'N converged'};
1149 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1150
1151 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1152 disp(T_compare)
1153
1154 writetable(T_compare, 'results_f79P_quad.xlsx', 'Sheet', 'ExactComparison', '
    WriteRowNames', true);

```

```

1 %% FUNCTION 27 (with different initial points)- with exact derivatives and finite
2   differences
3
4 F = @(x) F27(x); % Defining F27 as function handle
5 JF_gen = @(x,exact,fin_dif2,h) JF27(x,exact,fin_dif2,h); % Defining JF27 as function
6   handle
7
8 load forcing_terms.mat % possible terms for adaptive tolerance
9
10 %% n=10^3 (1e3)
11
12 rng(345989);
13
14 n=1e3;
15
16 kmax=1e3; % maximum number of iterations of Newton method
17 tolgrad=5e-7; % tolerance on gradient norm
18
19 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
20   system)
21 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
22
23 % Backtracking parameters
24 c1=1e-4;
25 rho=0.50;
26 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
27
28 x0=(1:n)'; % Initial point
29 N=10; % number of initial points to be generated
30
31 % Initial points:
32 Mat_points= repmat(x0,1,N+1);
33 rand_mat=2*rand(n, N)-1;
34 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations

```



```

38 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
39 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
40 mat_conv1_ex=zeros(15,N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
41 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
42 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
43
44 JF_ex = @(x) JF_gen(x,true,false,0);
45
46 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
47 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
48 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
49 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
50 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
51 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
52 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
53 mat_conv1_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
54 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
55 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
56
57 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
58
59 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
60 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
61 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
62 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
63 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
64 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
65 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
66 mat_conv1_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
67 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
68 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
69
70 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
71
72 for j =1:N+1
73     disp(['Condizione_iniziale_n.',num2str(j)])
74
75     % EXACT DERIVATIVES
76     tic;
77     [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);
78
79     vec_times1_ex(j)=toc;
80
81     disp(['Exact_derivatives:',flag1])
82     vec_converged1_ex(j)=converged1;
83     vec_val1_ex(j)=f1;
84     vec_grad1_ex(j)=gradf_norm1;
85     vec_iter1_ex(j)=k1;
86     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
87     vec_bt1_ex(j)=sum(btseq1)/k1;
88     vec_violations1_ex(j)=violations1;
89
90     last_vals = conv_ord1_ex(max(end-14,1):end);
91     mat_conv1_ex(:, j) = last_vals;
92
93     for i=2:2:12
94         h=10^(-i);
95
96         % FINITE DIFFERENCES 1
97         JF=@(x) JF_fd1(x,h);
98
99         tic;
100
101         [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);

102
103 mat_times1_fd1(i/2,j)=toc;
104
105 disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag1])
106 mat_converged1_fd1(i/2,j)=converged1;
107 mat_val1_fd1(i/2,j)=f1;
108 mat_grad1_fd1(i/2,j)=gradf_norm1;
109 mat_iter1_fd1(i/2,j)=k1;
110 mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
111 mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
112 mat_violations1_fd1(i/2,j)=violations1;
113
114 last_vals = conv_ord1_df1(max(end-14,1):end);
115 mat_conv1_fd1(i/2, j) = {last_vals};
116
117
118 % FINITE DIFFERENCES 2
119 JF=@(x) JF_fd2(x,h);
120 tic;
121
122 [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);

123
124 mat_times1_fd2(i/2,j)=toc;
125
126 disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
127 mat_converged1_fd2(i/2,j)=converged1;
128 mat_val1_fd2(i/2,j)=f1;
129 mat_grad1_fd2(i/2,j)=gradf_norm1;
130 mat_iter1_fd2(i/2,j)=k1;
131 mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
132 mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
133 mat_violations1_fd2(i/2,j)=violations1;
134
135 last_vals = conv_ord1_df2(max(end-14,1):end);
136 mat_conv1_fd2(i/2, j) = {last_vals};
137
138
139
140 end
141 end
142
143 %% Plot of the last 12 values of experimentale rate of convergence
144 num_initial_points = N + 1;
145 figure;
146 hold on;
147
148 % Plot for every initial condition
149 for j = 1:num_initial_points
150     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
151     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
152     hold on;
153     for i =1:6
154         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
155         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
156         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
157         hold on;
158         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
159         hold on;
160     end
161 end
162
163 % title and legend
164 title('F27_10^3_superlinear');
165 xlabel('Iterazione');
166 ylabel('Ordine di Convergenza');
167 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
168 grid on;
169 hold off;
170

```

```

171
172 %% Execution Time
173
174 % Exact Derivative
175 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
176 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
177 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
178
179 % FD1
180 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
181 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
182 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
183
184 % FD2
185 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
186 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
187 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
188
189 % Creation of the labels
190 h_exponents = [2, 4, 6, 8, 10, 12];
191 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
192
193 fd1_vals = avg_fd1';
194 fd2_vals = avg_fd2';
195
196 % Table construction with exact for both the row
197 rowNames = {'FD1', 'FD2'};
198 columnNames = [h_labels, 'Exact'];
199 data = [fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1];
200 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
201
202 % visualization
203 disp('Average computation times table (only for successful runs): F27, n=10^3,
    superlinear');
204 disp(T1);
205
206
207 %% All the tables has the same structure
208 %% Iteration
209
210 vec_times_ex_clean = vec_iter1_ex;
211 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
212 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
213
214 mat_times_fd1_clean = mat_iter1_fd1;
215 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
216 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
217
218 mat_times_fd2_clean = mat_iter1_fd2;
219 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
220 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
221
222 h_exponents = [2, 4, 6, 8, 10, 12];
223 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
224
225 fd1_vals = avg_fd1';
226 fd2_vals = avg_fd2';
227
228 rowNames = {'FD1', 'FD2'};
229 columnNames = [h_labels, 'Exact'];
230 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1];
231
232 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
233
234 disp('Average computation iteration table (only for successful runs): F27, n=10^3,
    suplin');
235 disp(T2);
236
237 %% F value
238
239 vec_times_ex_clean = vec_val1_ex;

```

```

240 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
241 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
242
243 mat_times_fd1_clean = mat_val1_fd1;
244 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
245 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
246
247 mat_times_fd2_clean = mat_val1_fd2;
248 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
249 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
250
251 h_exponents = [2, 4, 6, 8, 10, 12];
252 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
253
254 fd1_vals = avg_fd1';
255 fd2_vals = avg_fd2';
256
257 rowNames = {'FD1', 'FD2'};
258 columnNames = [h_labels, 'Exact'];
259 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1];
260
261 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
262
263 disp('Average computation fmin value table (only for successful runs): F27, n=10^3,
suplin');
264 disp(T3);
265
266 %% VIOLATION
267
268 vec_times_ex_clean = vec_violations1_ex;
269 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
270 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
271
272 mat_times_fd1_clean = mat_violations1_fd1;
273 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
274 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
275
276 mat_times_fd2_clean = mat_violations1_fd2;
277 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
278 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
279
280 h_exponents = [2, 4, 6, 8, 10, 12];
281 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
282
283 fd1_vals = avg_fd1';
284 fd2_vals = avg_fd2';
285
286 rowNames = {'FD1', 'FD2'};
287 columnNames = [h_labels, 'Exact'];
288 data = [fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1];
289
290 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
291
292 disp('Average computation violation table (only for successful runs): F27, n=10^3,
superlinear');
293 disp(T10);
294
295
296 %% BT-SEQ
297 vec_bt_ex_clean = vec_bt1_ex;
298 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
299 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
300
301 mat_bt_fd1_clean = mat_bt1_fd1;
302 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
303 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
304
305 mat_bt_fd2_clean = mat_bt1_fd2;
306 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
307 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
308
309 h_exponents = [2, 4, 6, 8, 10, 12];
310 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```

311 fd1_vals = avg_fd1';
312 fd2_vals = avg_fd2';
313
314 rowNames = {'FD1', 'FD2'};
315 columnNames = [ h_labels, 'Exact'];
316 data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
317
318 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
319
320 disp('Average computation bt iteration table (only for successful runs): F27, n=10^3,
321      superlinear');
322 disp(T11);
323
324 %% CG-SEQ
325
326 vec_bt_ex_clean = vec_cg_iter1_ex;
327 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
328 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
329
330 mat_bt_fd1_clean = mat_cg_iter1_fd1;
331 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
332 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
333
334 mat_bt_fd2_clean = mat_cg_iter1_fd2;
335 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
336 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
337
338 h_exponents = [2, 4, 6, 8, 10, 12];
339 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
340
341 fd1_vals = avg_fd1';
342 fd2_vals = avg_fd2';
343
344 rowNames = {'FD1', 'FD2'};
345 columnNames = [ h_labels, 'Exact'];
346 data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
347
348 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
349
350 disp('Average computation cg iteration table (only for successful runs): F27, n=10^3,
351      superlinear');
352 disp(T12);
353
354 %% Number of starting point converged
355
356 h_exponents = [2, 4, 6, 8, 10, 12];
357 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
358
359 fd1_vals = sum(mat_converged1_fd1, 2)';
360 fd2_vals = sum(mat_converged1_fd2, 2)';
361
362 rowNames = {'FD1', 'FD2'};
363 columnNames = [ h_labels, 'Exact'];
364 data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];
365
366 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
367
368 disp('Number of converged: F27, n=10^3, superlinear');
369 disp(T13);
370
371 %save the table in a file.xlsx
372 writetable(T1, 'results_f27_suplin.xlsx', 'Sheet', 'time_3', 'WriteRowNames', true);
373 writetable(T2, 'results_f27_suplin.xlsx', 'Sheet', 'niter_3', 'WriteRowNames', true);
374 writetable(T3, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_3', 'WriteRowNames', true);
375 writetable(T10, 'results_f27_suplin.xlsx', 'Sheet', 'v_3', 'WriteRowNames', true);
376 writetable(T11, 'results_f27_suplin.xlsx', 'Sheet', 'bt_3', 'WriteRowNames', true);
377 writetable(T12, 'results_f27_suplin.xlsx', 'Sheet', 'cg_3', 'WriteRowNames', true);
378 writetable(T13, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv3', 'WriteRowNames', true);
379
380 %% n=10^4 (1e4)
381

```

```

382 rng(345989);
383
384 n=1e4;
385
386 kmax=1.5e3; % maximum number of iterations of Newton method
387 tolgrad=5e-7; % tolerance on gradient norm
388
389 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
390 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
391
392 % Backtracking parameters
393 c1=1e-4;
394 rho=0.50;
395 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
396
397 x0=(1:n)'; % Initial point
398 N=10; % number of initial points to be generated
399
400 % Initial points:
401 Mat_points= repmat(x0,1,N+1);
402 rand_mat=2*rand(n, N)-1;
403 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
404
405 % Structure for EXACT derivatives
406 vec_times2_ex=zeros(1,N+1); % vector with execution times
407 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
408 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
409 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
410 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
411 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
412 mat_conv2_ex=zeros(15,N+1); %matrix with the last 15 values of rate of convergence for
    the starting point
413 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
414 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
415
416 JF_ex = @(x) JF_gen(x,true,false,0);
417
418 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
419 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
420 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
421 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
422 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
423 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
424 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
425 mat_conv2_fd1=cell(6,N+1); %matrix with the last 15 values of rate of convergence for the
    starting point
426 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
427 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
428
429 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
430
431
432 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
433 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
434 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
435 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
436 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
437 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
438 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
439 mat_conv2_fd2=cell(6,N+1); %matrix with the last 15 values of rate of convergence for the
    starting point
440 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
441 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
442
443 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
444
445 for j =1:N+1
446     disp(['Condizione iniziale n. ',num2str(j)])

```

```

447 % EXACT DERIVATIVES
448 tic;
449
450 [x2, f2, gradf_norm2, k2, xseq2, btseq2, cgiterseq2, conv_ord2_ex, flag2, converged2,
451 violations2] = truncated_newton_27(Mat_points(:,j), F, JF_ex, true, false, 0, kmax,
    tolgrad, fterms_suplin, cg_maxit, z0, c1, rho, btmax);
452
453 vec_times2_ex(j)=toc;
454
455 disp(['Exact derivatives: ', flag2])
456 vec_converged2_ex(j)=converged2;
457
458 vec_val2_ex(j)=f2;
459 vec_grad2_ex(j)=gradf_norm2;
460 vec_iter2_ex(j)=k2;
461 vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
462 vec_bt2_ex(j)=sum(btseq2)/k2;
463 vec_violations2_ex(j)=violations2;
464
465 last_vals = conv_ord2_ex(max(end-14,1):end);
466 mat_conv2_ex(:, j) = last_vals;
467
468
469
470 for i=2:2:12
471 h=10^(-i);
472
473 % FINITE DIFFERENCES 1
474 JF=@(x) JF_fd1(x,h);
475
476 tic;
477
478 [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);
479
480 mat_times2_fd1(i/2,j)=toc;
481
482 disp(['Finite differences (classical version) with h=1e-', num2str(i), ': ', flag2])
483 mat_converged2_fd1(i/2,j)=converged2;
484
485 mat_val2_fd1(i/2,j)=f2;
486 mat_grad2_fd1(i/2,j)=gradf_norm2;
487 mat_iter2_fd1(i/2,j)=k2;
488 mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
489 mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
490 mat_violations2_fd1(i/2,j)=violations2;
491
492 last_vals = conv_ord2_df1(max(end-14,1):end);
493 mat_conv2_fd1(i/2, j) = {last_vals};
494
495 % FINITE DIFFERENCES 2
496 JF=@(x) JF_fd2(x,h);
497
498 tic;
499
500 [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);
501 mat_times2_fd2(i/2,j)=toc;
502
503 disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag2])
504 mat_converged2_fd2(i/2,j)=converged2;
505
506 mat_val2_fd2(i/2,j)=f2;
507 mat_grad2_fd2(i/2,j)=gradf_norm2;
508 mat_iter2_fd2(i/2,j)=k2;
509 mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
510 mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
511 mat_violations2_fd2(i/2,j)=violations2;
512 last_vals = conv_ord2_df2(max(end-14,1):end);
513 mat_conv2_fd2(i/2, j) = {last_vals};

```

```

514
515     end
516 end
517
518
519 %% The Plot has the same structure
520 num_initial_points = N + 1;
521 figure;
522 hold on;
523
524 for j = 1:num_initial_points
525     conv_ord_ex = mat_conv2_ex(:,j);
526     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
527     hold on;
528     for i =1:6
529         conv_ord_fd1 = mat_conv2_fd1{i, j};
530         conv_ord_fd2 = mat_conv2_fd2{i, j};
531         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
532         hold on;
533         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
534         hold on;
535     end
536 end
537
538 title('F27_10^4_superlinear');
539 xlabel('Iterazione');
540 ylabel('Ordine di Convergenza');
541 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
542 grid on;
543 hold off;
544
545
546 %% Execution time
547
548 % Exact derivative
549 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
550 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
551 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
552
553 % FD1
554 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
555 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
556 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
557
558 % FD2
559 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
560 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
561 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
562
563 % Creation of the labels
564 h_exponents = [2, 4, 6, 8, 10, 12];
565 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
566
567 fd1_vals = avg_fd1';
568 fd2_vals = avg_fd2';
569
570 % Table creation
571 rowNames = {'FD1', 'FD2'};
572 columnNames = [ h_labels, 'Exact'];
573 data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
574 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
575 %display the table
576 disp('Average computation times table (only for successful runs): F27, n=10^4,
    superlinear');
577 disp(T4);
578
579 %% Iteration
580
581 vec_times_ex_clean = vec_iter2_ex;
582 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
583 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');

```



```

584
585 mat_times_fd1_clean = mat_iter2_fd1;
586 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
587 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
588
589 mat_times_fd2_clean = mat_iter2_fd2;
590 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
591 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
592
593 h_exponents = [2, 4, 6, 8, 10, 12];
594 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
595
596 fd1_vals = avg_fd1';
597 fd2_vals = avg_fd2';
598
599 rowNames = {'FD1', 'FD2'};
600 columnNames = [h_labels, 'Exact'];
601 data = [fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2];
602
603 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
604
605 disp('Average computation iteration table (only for successful runs): F27, n=10^4,
        superlinear');
606 disp(T5);
607
608 %% Function value
609
610 vec_times_ex_clean = vec_val2_ex;
611 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
612 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
613
614 mat_times_fd1_clean = mat_val2_fd1;
615 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
616 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
617
618 mat_times_fd2_clean = mat_val2_fd2;
619 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
620 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
621
622 h_exponents = [2, 4, 6, 8, 10, 12];
623 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
624
625 fd1_vals = avg_fd1';
626 fd2_vals = avg_fd2';
627
628 rowNames = {'FD1', 'FD2'};
629 columnNames = [h_labels, 'Exact'];
630 data = [fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2];
631
632 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
633
634 disp('Average computation fmin value table (only for successful runs): F27, n=10^4,
        superlinear');
635 disp(T6);
636
637 %% VIOLATION
638
639 vec_times_ex_clean = vec_violations2_ex;
640 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
641 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
642
643 mat_times_fd1_clean = mat_violations2_fd1;
644 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
645 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
646
647 mat_times_fd2_clean = mat_violations2_fd2;
648 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
649 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
650
651 h_exponents = [2, 4, 6, 8, 10, 12];
652 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
653
654 fd1_vals = avg_fd1';

```

```

655 fd2_vals = avg_fd2';
656
657 rowNames = {'FD1', 'FD2'};
658 columnNames = [ h_labels, 'Exact'];
659 data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];
660
661 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
662
663 disp('Average computation violation table (only for successful runs): F27, n=10^4,
suplinear');
664 disp(T14);
665
666 %% BT-SEQ
667
668 vec_bt_ex_clean = vec_bt2_ex;
669 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
670 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
671
672 mat_bt_fd1_clean = mat_bt2_fd1;
673 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
674 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
675
676 mat_bt_fd2_clean = mat_bt2_fd2;
677 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
678 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
679
680 h_exponents = [2, 4, 6, 8, 10, 12];
681 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
682
683 fd1_vals = avg_fd1';
684 fd2_vals = avg_fd2';
685
686 rowNames = {'FD1', 'FD2'};
687 columnNames = [ h_labels, 'Exact'];
688 data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];
689
690 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
691
692 disp('Average computation bt iteration table (only for successful runs): F27, n=10^4,
superlinear');
693 disp(T15);
694
695 %% CG-SEQ
696
697 vec_bt_ex_clean = vec_cg_iter2_ex;
698 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
699 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
700
701 mat_bt_fd1_clean = mat_cg_iter2_fd1;
702 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
703 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
704
705 mat_bt_fd2_clean = mat_cg_iter2_fd2;
706 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
707 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
708
709 h_exponents = [2, 4, 6, 8, 10, 12];
710 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
711
712 fd1_vals = avg_fd1';
713 fd2_vals = avg_fd2';
714
715 rowNames = {'FD1', 'FD2'};
716 columnNames = [ h_labels, 'Exact'];
717 data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];
718
719 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
720
721 disp('Average computation cg iteration table (only for successful runs): F27, n=10^4,
superlinear');
722 disp(T16);
723
724 %% Number of initial point converged

```

```

725
726 h_exponents = [2, 4, 6, 8, 10, 12];
727 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
728
729 fd1_vals = sum(mat_converged2_fd1,2)';
730 fd2_vals = sum(mat_converged2_fd2,2)';
731
732 rowNames = {'FD1', 'FD2'};
733 columnNames = [ h_labels, 'Exact'];
734 data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex)];
735
736 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
737
738 disp('Number of converged: F27, n=10^4, superlinear');
739 disp(T17);
740 %save the table in a file.xlsx
741 writetable(T4, 'results_f27_suplin.xlsx', 'Sheet', 'time_4', 'WriteRowNames', true);
742 writetable(T5, 'results_f27_suplin.xlsx', 'Sheet', 'niter_4', 'WriteRowNames', true);
743 writetable(T6, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_4', 'WriteRowNames', true);
744 writetable(T14, 'results_f27_suplin.xlsx', 'Sheet', 'v_4', 'WriteRowNames', true);
745 writetable(T15, 'results_f27_suplin.xlsx', 'Sheet', 'bt_4', 'WriteRowNames', true);
746 writetable(T16, 'results_f27_suplin.xlsx', 'Sheet', 'cg_4', 'WriteRowNames', true);
747 writetable(T17, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv4', 'WriteRowNames', true);
748
749
750
751 %% n=10^5 (1e5)
752
753 rng(345989);
754
755 n=1e5;
756
757 kmax=1.5e3; % maximum number of iterations of Newton method
758 tolgrad=5e-7; % tolerance on gradient norm
759
760 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
761 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
762
763 % Backtracking parameters
764 c1=1e-4;
765 rho=0.50;
766 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
767
768 x0=(1:n)'; % Initial point
769 N=10; % number of initial points to be generated
770
771 % Initial points:
772 Mat_points= repmat(x0,1,N+1);
773 rand_mat=2*rand(n, N)-1;
774 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
775
776 % Structure for EXACT derivatives
777 vec_times3_ex=zeros(1,N+1); % vector with execution times
778 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
779 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
780 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
781 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
782 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
783 mat_conv3_ex=zeros(15:N+1); %matrix with the last 15 values of rate of convergence for the
    starting point
784 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
785 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
786
787 JF_ex = @(x) JF_gen(x,true,false,0);
788
789 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
790 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
791 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
792 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
793 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
794 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations

```

```

795 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
796 mat_conv3_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
797 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
798 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
799
800 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
801
802 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
803 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
804 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
805 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
806 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
807 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
808 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
809 mat_conv3_fd2=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
810 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
811 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
812
813 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
814
815
816 for j =1:N+1
817     disp(['Condizione_iniziale_n. ',num2str(j)])
818
819     % EXACT DERIVATIVES
820     tic;
821
822     [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);
823
824     vec_times3_ex(j)=toc;
825
826     disp(['Exact_derivatives:',flag3])
827     vec_converged3_ex(j)=converged3;
828     vec_val3_ex(j)=f3;
829     vec_grad3_ex(j)=gradf_norm3;
830     vec_iter3_ex(j)=k3;
831     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
832     vec_bt3_ex(j)=sum(btseq3)/k3;
833     vec_violations3_ex(j)=violations3;
834     last_vals = conv_ord3_ex(max(end-14,1):end);
835     mat_conv3_ex(:, j) = last_vals;
836
837
838     for i=2:2:12
839         h=10^(-i);
840
841         % FINITE DIFFERENCES 1
842         JF=@(x) JF_fd1(x,h);
843
844         tic;
845
846         [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);
847         mat_times3_fd1(i/2,j)=toc;
848
849         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag3])
850         mat_converged3_fd1(i/2,j)=converged3;
851
852         mat_val3_fd1(i/2,j)=f3;
853         mat_grad3_fd1(i/2,j)=gradf_norm3;
854         mat_iter3_fd1(i/2,j)=k3;
855         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
856         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
857         mat_violations3_fd1(i/2,j)=violations3;
858         last_vals = conv_ord3_df1(max(end-14,1):end);

```

```

859     mat_conv3_fd1(i/2, j) = {last_vals};
860
861
862     % FINITE DIFFERENCES 2
863     JF=@(x) JF_fd2(x,h);
864
865     tic;
866
867     [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);
868     mat_times3_fd2(i/2, j)=toc;
869
870     disp(['Finite differences (new version) with h=1e-', num2str(i), 'e-', flag3])
871     mat_converged3_fd2(i/2, j)=converged3;
872     mat_val3_fd2(i/2, j)=f3;
873     mat_grad3_fd2(i/2, j)=gradf_norm3;
874     mat_iter3_fd2(i/2, j)=k3;
875     mat_cg_iter3_fd2(i/2, j)=sum(cgiterseq3)/k3;
876     mat_bt3_fd2(i/2, j)=sum(btseq3)/k3;
877     mat_violations3_fd2(i/2, j)=violations3;
878     last_vals = conv_ord3_df2(max(end-14,1):end);
879     mat_conv3_fd2(i/2, j) = {last_vals};
880
881     end
882 end
883
884
885 %% The plot has the same structure as n=10^3
886 num_initial_points = N + 1;
887 figure;
888 hold on;
889
890 for j = 1:num_initial_points
891     conv_ord_ex = mat_conv3_ex(:,j);
892     plot(1:12, conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
893     hold on;
894     for i = 1:6
895         conv_ord_fd1 = mat_conv3_fd1{i, j};
896         conv_ord_fd2 = mat_conv3_fd2{i, j};
897         plot(1:12, conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
898         hold on;
899         plot(1:12, conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
900         hold on;
901     end
902 end
903
904 title('F27_10^5_superlinear');
905 xlabel('Iterazione');
906 ylabel('Ordine di Convergenza');
907 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
908 grid on;
909 hold off;
910
911 %% Time
912
913 vec_times_ex_clean = vec_times3_ex;
914 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
915 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
916
917 mat_times_fd1_clean = mat_times3_fd1;
918 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
919 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
920
921 mat_times_fd2_clean = mat_times3_fd2;
922 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
923 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
924
925 h_exponents = [2, 4, 6, 8, 10, 12];
926 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
927
928 fd1_vals = avg_fd1';
929 fd2_vals = avg_fd2';

```

```

930
931 rowNames = {'FD1', 'FD2'};
932 columnNames = [ h_labels, 'Exact'];
933 data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];
934
935 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
936
937 disp('Average computation times table (only for successful runs): F27, n=10^5,
    superlinear');
938 disp(T7);
939
940 %% Iteration
941
942 vec_times_ex_clean = vec_iter3_ex;
943 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
944 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
945
946 mat_times_fd1_clean = mat_iter3_fd1;
947 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
948 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
949
950 mat_times_fd2_clean = mat_iter3_fd2;
951 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
952 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
953
954 h_exponents = [2, 4, 6, 8, 10, 12];
955 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
956
957 fd1_vals = avg_fd1';
958 fd2_vals = avg_fd2';
959
960 rowNames = {'FD1', 'FD2'};
961 columnNames = [ h_labels, 'Exact'];
962 data = [ fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3;];
963
964 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
965
966 disp('Average computation iteration table (only for successful runs): F27, n=10^5,
    superlinear');
967 disp(T8);
968
969 %% function value
970
971 vec_times_ex_clean = vec_val3_ex;
972 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
973 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
974
975 mat_times_fd1_clean = mat_val3_fd1;
976 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
977 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
978
979 mat_times_fd2_clean = mat_val3_fd2;
980 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
981 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
982
983 h_exponents = [2, 4, 6, 8, 10, 12];
984 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
985
986 fd1_vals = avg_fd1';
987 fd2_vals = avg_fd2';
988
989 rowNames = {'FD1', 'FD2'};
990 columnNames = [ h_labels, 'Exact'];
991 data = [ fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3;];
992
993 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
994
995 disp('Average computation fmin value table (only for successful runs): F27, n=10^5,
    superlinear');
996 disp(T9);
997
998 %% VIOLATION
999

```

```

1000 vec_times_ex_clean = vec_violations3_ex;
1001 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
1002 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
1003
1004 mat_times_fd1_clean = mat_violations3_fd1;
1005 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1006 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1007
1008 mat_times_fd2_clean = mat_violations3_fd2;
1009 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1010 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1011
1012 h_exponents = [2, 4, 6, 8, 10, 12];
1013 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1014
1015 fd1_vals = avg_fd1';
1016 fd2_vals = avg_fd2';
1017
1018 rowNames = {'FD1', 'FD2'};
1019 columnNames = [h_labels, 'Exact'];
1020 data = [fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3];
1021
1022 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1023
1024 disp('Average computation violation table (only for successful runs): F27, n=10^5,
1025      superlinear');
1026 disp(T18);
1027
1028 %% BT-SEQ
1029
1030 vec_bt_ex_clean = vec_bt3_ex;
1031 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1032 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1033
1034 mat_bt_fd1_clean = mat_bt3_fd1;
1035 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1036 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1037
1038 mat_bt_fd2_clean = mat_bt3_fd2;
1039 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1040 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1041
1042 h_exponents = [2, 4, 6, 8, 10, 12];
1043 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1044
1045 fd1_vals = avg_fd1';
1046 fd2_vals = avg_fd2';
1047
1048 rowNames = {'FD1', 'FD2'};
1049 columnNames = [h_labels, 'Exact'];
1050 data = [fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3];
1051
1052 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1053
1054 disp('Average computation bt iteration table (only for successful runs): F27, n=10^5,
1055      superlinear');
1056 disp(T19);
1057
1058 %% CG-SEQ
1059
1060 vec_bt_ex_clean = vec_cg_iter3_ex;
1061 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1062 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1063
1064 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1065 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1066 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1067
1068 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1069 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1070 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1071
1072 h_exponents = [2, 4, 6, 8, 10, 12];

```

```

1071 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1072
1073 fd1_vals = avg_fd1';
1074 fd2_vals = avg_fd2';
1075
1076 rowNames = {'FD1', 'FD2'};
1077 columnNames = [ h_labels, 'Exact'];
1078 data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1079
1080 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1081
1082 disp('Average computation cg iteration table (only for successful runs): F27, n=10^5,
superlinear');
1083 disp(T20);
1084
1085 %% Number of initial condition converged
1086
1087 h_exponents = [2, 4, 6, 8, 10, 12];
1088 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1089
1090 fd1_vals = sum(mat_converged3_fd1,2)';
1091 fd2_vals = sum(mat_converged3_fd2,2)';
1092
1093 rowNames = {'FD1', 'FD2'};
1094 columnNames = [ h_labels, 'Exact'];
1095 data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1096
1097 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1098
1099 disp('Number of converged: F27, n=10^5, superlinear');
1100 disp(T21);
1101 %save the tables
1102 writetable(T7, 'results_f27_suplin.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
1103 writetable(T8, 'results_f27_suplin.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
1104 writetable(T9, 'results_f27_suplin.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
1105 writetable(T18, 'results_f27_suplin.xlsx', 'Sheet', 'v_5','WriteRowNames', true);
1106 writetable(T19, 'results_f27_suplin.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
1107 writetable(T20, 'results_f27_suplin.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
1108 writetable(T21, 'results_f27_suplin.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);
1109
1110
1111
1112 %% table with the results of the exact derivatives
1113
1114 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1115         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1116         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1117         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1118         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1119         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1120         sum(vec_converged1_ex), sum(vec_converged2_ex), sum(vec_converged3_ex)];
1121
1122 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
1123             'Average iter cg', 'N converged'};
1124 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1125
1126 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1127 disp(T_compare)
1128
1129 writetable(T_compare, 'results_f27_suplin.xlsx', 'Sheet', 'ExactComparison', '
WriteRowNames', true);

```

```

1 %% FUNCTION 27 QUADRATIC (with different initial points)- with exact derivatives and
finite differences
2
3
4 F = @(x) F27(x); % Defining F27 as function handle
5 JF_gen = @(x,exact,fin_dif2,h) JF27(x,exact,fin_dif2,h); % Defining JF27 as function
handle
6
7 load forcing_terms.mat % possible terms for adaptive tolerance
8

```



```

9  %% n=10^3 (1e3)
10
11  rng(345989);
12
13  n=1e3;
14
15  kmax=1e3; % maximum number of iterations of Newton method
16  tolgrad=5e-7; % tolerance on gradient norm
17
18  cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
19  system)
20  z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
21
22  % Backtracking parameters
23  c1=1e-4;
24  rho=0.50;
25  btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
26
27  x0=(1:n)'; % Initial point
28  N=10; % number of initial points to be generated
29
30  % Initial points:
31  Mat_points= repmat(x0,1,N+1);
32  rand_mat=2*rand(n, N)-1;
33  Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
34
35  % Structure for EXACT derivatives
36  vec_times1_ex=zeros(1,N+1); % vector with execution times
37  vec_val1_ex=zeros(1,N+1); %vector with minimal values found
38  vec_grad1_ex=zeros(1,N+1); %vector with final gradient
39  vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
40  vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
41  vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
42  mat_conv1_ex=zeros(15,N+1); %matrix with che last 15 values of rate of convergence for
43  the starting point
44  vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45  vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
46  condition in Newton method
47
48  JF_ex = @(x) JF_gen(x,true,false,0);
49
50  % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51  mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52  mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53  mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54  mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55  mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56  mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57  mat_conv1_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
58  starting point
59  mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
60  mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
61  condition in Newton method
62
63  JF_fd1 = @(x,h) JF_gen(x,false,false,h);
64
65  % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
66  x_j) as increment)
67  mat_times1_fd2=zeros(6,N+1); % matrix with execution times
68  mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
69  mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
70  mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
71  mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
72  mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
73  mat_conv1_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
74  starting point
75  mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
76  mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
77  condition in Newton method
78
79  JF_fd2 = @(x,h) JF_gen(x,false,true,h);
80
81  for j =1:N+1

```

```

74 disp(['Condizione iniziale n. ', num2str(j)])
75
76 % EXACT DERIVATIVES
77 tic;
78 [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);
79
80 vec_times1_ex(j)=toc;
81
82 disp(['Exact derivatives: ', flag1])
83 vec_converged1_ex(j)=converged1;
84 %conv_ord1(end-10:end) %aggiustare
85 vec_val1_ex(j)=f1;
86 vec_grad1_ex(j)=gradf_norm1;
87 vec_iter1_ex(j)=k1;
88 vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
89 vec_bt1_ex(j)=sum(btseq1)/k1;
90 vec_violations1_ex(j)=violations1;
91
92 last_vals = conv_ord1_ex(max(end-14,1):end);
93 mat_conv1_ex(:, j) = last_vals;
94
95 for i=2:2:12
96 h=10^(-i);
97
98 % FINITE DIFFERENCES 1
99 JF=@(x) JF_fd1(x,h);
100
101 tic;
102
103 [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);
104
105 mat_times1_fd1(i/2,j)=toc;
106
107 disp(['Finite differences (classical version) with h=1e-', num2str(i), ': ', flag1])
108 mat_converged1_fd1(i/2,j)=converged1;
109 mat_val1_fd1(i/2,j)=f1;
110 mat_grad1_fd1(i/2,j)=gradf_norm1;
111 mat_iter1_fd1(i/2,j)=k1;
112 mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
113 mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
114 mat_violations1_fd1(i/2,j)=violations1;
115 last_vals = conv_ord1_df1(max(end-14,1):end);
116 mat_conv1_fd1(i/2, j) = {last_vals};
117
118
119 % FINITE DIFFERENCES 2
120 JF=@(x) JF_fd2(x,h);
121 tic;
122
123 [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);
124
125 mat_times1_fd2(i/2,j)=toc;
126
127 disp(['Finite differences (new version) with h=1e-', num2str(i), ': ', flag1])
128 mat_converged1_fd2(i/2,j)=converged1;
129 mat_val1_fd2(i/2,j)=f1;
130 mat_grad1_fd2(i/2,j)=gradf_norm1;
131 mat_iter1_fd2(i/2,j)=k1;
132 mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
133 mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
134 mat_violations1_fd2(i/2,j)=violations1;
135 last_vals = conv_ord1_df2(max(end-14,1):end);
136 mat_conv1_fd2(i/2, j) = {last_vals};
137
138 end
139 end
140

```

```

141
142 %% Plot of the last 12 values of experimentale rate of convergence
143 num_initial_points = N + 1;
144 figure;
145 hold on;
146
147 % Plot for every initial condition
148 for j = 1:num_initial_points
149     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
150     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
151     hold on;
152     for i = 1:6
153         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
154         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
155         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
156         hold on;
157         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
158         hold on;
159     end
160 end
161
162 % title and legend
163 title('F27_10^3_quadratic');
164 xlabel('Iterazione');
165 ylabel('Ordine di Convergenza');
166 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
167 grid on;
168 hold off;
169
170
171 %% Execution Time
172
173 % Exact Derivative
174 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
175 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
176 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
177
178 % FD1
179 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
180 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
181 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
182
183 % FD2
184 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
185 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
186 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
187
188 % Creation of the labels
189 h_exponents = [2, 4, 6, 8, 10, 12];
190 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
191
192 fd1_vals = avg_fd1';
193 fd2_vals = avg_fd2';
194
195 % Table costruction with exact for both the row
196 rowNames = {'FD1', 'FD2'};
197 columnNames = [ h_labels, 'Exact'];
198 data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
199 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
200
201 % visualization
202 disp('Average computation times table (only for successful runs): F27, n=10^3, quadratic'
    );
203 disp(T1);
204
205
206 %% All the tables has the same structure
207 %% Iteration
208
209 vec_times_ex_clean = vec_iter1_ex;
210 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;

```

```

211 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
212
213 mat_times_fd1_clean = mat_iter1_fd1;
214 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
215 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
216
217 mat_times_fd2_clean = mat_iter1_fd2;
218 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
219 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
220
221 h_exponents = [2, 4, 6, 8, 10, 12];
222 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
223
224 fd1_vals = avg_fd1';
225 fd2_vals = avg_fd2';
226
227 rowNames = {'FD1', 'FD2'};
228 columnNames = [h_labels, 'Exact'];
229 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1];
230
231 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
232
233 disp('Average computation iteration table (only for successful runs): F27, n=10^3, quadratic');
234 disp(T2);
235
236 %% F value
237
238 vec_times_ex_clean = vec_val1_ex;
239 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
240 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
241
242 mat_times_fd1_clean = mat_val1_fd1;
243 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
244 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
245
246 mat_times_fd2_clean = mat_val1_fd2;
247 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
248 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
249
250 h_exponents = [2, 4, 6, 8, 10, 12];
251 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
252
253 fd1_vals = avg_fd1';
254 fd2_vals = avg_fd2';
255
256 rowNames = {'FD1', 'FD2'};
257 columnNames = [h_labels, 'Exact'];
258 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1];
259
260 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
261
262 disp('Average computation fmin value table (only for successful runs): F27, n=10^3, quadratic');
263 disp(T3);
264
265 %% VIOLATION
266
267 vec_times_ex_clean = vec_violations1_ex;
268 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
269 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
270
271 mat_times_fd1_clean = mat_violations1_fd1;
272 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
273 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
274
275 mat_times_fd2_clean = mat_violations1_fd2;
276 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
277 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
278
279 h_exponents = [2, 4, 6, 8, 10, 12];
280 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
281

```

```

282 %
283 fd1_vals = avg_fd1';
284 fd2_vals = avg_fd2';
285
286 rowNames = {'FD1', 'FD2'};
287 columnNames = [ h_labels, 'Exact'];
288 data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];
289
290 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
291
292 disp('Average computation violation table (only for successful runs): F27, n=10^3,
293      quadratic');
294 disp(T10);
295
296 %% BT-SEQ
297 vec_bt_ex_clean = vec_bt1_ex;
298 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
299 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
300
301 mat_bt_fd1_clean = mat_bt1_fd1;
302 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
303 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
304
305 mat_bt_fd2_clean = mat_bt1_fd2;
306 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
307 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
308
309 h_exponents = [2, 4, 6, 8, 10, 12];
310 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
311
312 fd1_vals = avg_fd1';
313 fd2_vals = avg_fd2';
314
315 rowNames = {'FD1', 'FD2'};
316 columnNames = [ h_labels, 'Exact'];
317 data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
318
319 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
320
321 disp('Average computation bt iteration table (only for successful runs): F27, n=10^3,
322      quadratic');
323 disp(T11);
324
325 %% CG-SEQ
326 vec_bt_ex_clean = vec_cg_iter1_ex;
327 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
328 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
329
330 mat_bt_fd1_clean = mat_cg_iter1_fd1;
331 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
332 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
333
334 mat_bt_fd2_clean = mat_cg_iter1_fd2;
335 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
336 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
337
338 h_exponents = [2, 4, 6, 8, 10, 12];
339 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
340
341 fd1_vals = avg_fd1';
342 fd2_vals = avg_fd2';
343
344 rowNames = {'FD1', 'FD2'};
345 columnNames = [ h_labels, 'Exact'];
346 data = [ fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
347
348 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
349
350 disp('Average computation cg iteration table (only for successful runs): F27, n=10^3,
351      quadratic');
352 disp(T12);

```

```

352
353 %% Number of starting point converged
354
355 h_exponents = [2, 4, 6, 8, 10, 12];
356 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
357
358 fd1_vals = sum(mat_converged1_fd1,2)';
359 fd2_vals = sum(mat_converged1_fd2,2)';
360
361 rowNames = {'FD1', 'FD2'};
362 columnNames = [ h_labels, 'Exact'];
363 data = [ fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex)];
364
365 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
366
367 disp('Number of converged F27, n=10^3, quadratic');
368 disp(T13);
369 %save the table in a file.xlsx
370 writetable(T1, 'results_f27_quad.xlsx', 'Sheet', 'time_3','WriteRowNames', true);
371 writetable(T2, 'results_f27_quad.xlsx', 'Sheet', 'niter_3','WriteRowNames', true);
372 writetable(T3, 'results_f27_quad.xlsx', 'Sheet', 'f_val_3','WriteRowNames', true);
373 writetable(T10, 'results_f27_quad.xlsx', 'Sheet', 'v_3','WriteRowNames', true);
374 writetable(T11, 'results_f27_quad.xlsx', 'Sheet', 'bt_3','WriteRowNames', true);
375 writetable(T12, 'results_f27_quad.xlsx', 'Sheet', 'cg_3','WriteRowNames', true);
376 writetable(T13, 'results_f27_quad.xlsx', 'Sheet', 'n_conv3','WriteRowNames', true);
377
378
379
380 %% n=10^4 (1e4)
381
382 rng(345989);
383
384 n=1e4;
385
386 kmax=1.5e3; % maximum number of iterations of Newton method
387 tolgrad=5e-7; % tolerance on gradient norm
388
389 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
390 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
391
392 % Backtracking parameters
393 c1=1e-4;
394 rho=0.50;
395 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
396
397 x0=(1:n)'; % Initial point
398 N=10; % number of initial points to be generated
399
400 % Initial points:
401 Mat_points=repmat(x0,1,N+1);
402 rand_mat=2*rand(n, N)-1;
403 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
404
405 % Structure for EXACT derivatives
406 vec_times2_ex=zeros(1,N+1); % vector with execution times
407 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
408 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
409 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
410 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
411 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
412 mat_conv2_ex=zeros(15,N+1); %matrix with the last 15 values of rate of convergence for the
    starting point
413 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
414 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
415
416 JF_ex = @(x) JF_gen(x,true,false,0);
417
418 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
419 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
420 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
421 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient

```

```

422 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
423 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
424 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
425 mat_conv2_fd1=cell(6,N+1);%matrix with che last 15 values of rate of convergence for the
    starting point
426 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
427 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
428
429 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
430
431
432 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
433 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
434 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
435 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
436 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
437 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
438 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
439 mat_conv2_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
440 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
441 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
442
443 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
444
445 for j =1:N+1
446     disp(['Condizione_iniziale_',num2str(j)])
447
448     % EXACT DERIVATIVES
449     tic;
450
451     [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);
452
453     vec_times2_ex(j)=toc;
454
455     disp(['Exact_derivatives:',flag2])
456     vec_converged2_ex(j)=converged2;
457     vec_val2_ex(j)=f2;
458     vec_grad2_ex(j)=gradf_norm2;
459     vec_iter2_ex(j)=k2;
460     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
461     vec_bt2_ex(j)=sum(btseq2)/k2;
462     vec_violations2_ex(j)=violations2;
463
464     last_vals = conv_ord2_ex(max(end-14,1):end);
465     mat_conv2_ex(:, j) = last_vals;
466
467
468
469     for i=2:2:12
470         h=10^(-i);
471
472         % FINITE DIFFERENCES 1
473         JF= @(x) JF_fd1(x,h);
474
475         tic;
476
477         [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);
478
479         mat_times2_fd1(i/2,j)=toc;
480
481         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag2])
482         mat_converged2_fd1(i/2,j)=converged2;
483         mat_val2_fd1(i/2,j)=f2;
484         mat_grad2_fd1(i/2,j)=gradf_norm2;
485         mat_iter2_fd1(i/2,j)=k2;

```

```

486     mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
487     mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
488     mat_violations2_fd1(i/2,j)=violations2;
489
490     last_vals = conv_ord2_df1(max(end-14,1):end);
491     mat_conv2_fd1(i/2, j) = {last_vals};
492
493     % FINITE DIFFERENCES 2
494     JF=@(x) JF_fd2(x,h);
495
496     tic;
497
498     [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
499     violations2] = truncated_newton_27(Mat_points(:,j), F, JF, false,true,h, kmax,
500     tolgrad, fterms_quad, cg_maxit,z0, c1, rho, btmax);
501     mat_times2_fd2(i/2,j)=toc;
502
503     disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag2])
504     mat_converged2_fd2(i/2,j)=converged2;
505     mat_val2_fd2(i/2,j)=f2;
506     mat_grad2_fd2(i/2,j)=gradf_norm2;
507     mat_iter2_fd2(i/2,j)=k2;
508     mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
509     mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
510     mat_violations2_fd2(i/2,j)=violations2;
511     last_vals = conv_ord2_df2(max(end-14,1):end);
512     mat_conv2_fd2(i/2, j) = {last_vals};
513
514     end
515 end
516
517 %% The Plot has the same structure
518 num_initial_points = N + 1;
519 figure;
520 hold on;
521
522 for j = 1:num_initial_points
523     conv_ord_ex = mat_conv2_ex(:,j);
524     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
525     hold on;
526     for i =1:6
527         conv_ord_fd1 = mat_conv2_fd1{i, j};
528         conv_ord_fd2 = mat_conv2_fd2{i, j};
529         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
530         hold on;
531         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
532         hold on;
533     end
534 end
535
536 title('F27_10^4_quadratic');
537 xlabel('Iterazione');
538 ylabel('Ordine di Convergenza');
539 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
540 grid on;
541 hold off;
542
543 %% Execution time
544
545 % Exact derivative
546 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
547 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
548 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
549
550 % FD1
551 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
552 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
553     converge
554 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
555
556 % FD2

```



```

556 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
557 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
558 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
559
560 % Creation of the labels
561 h_exponents = [2, 4, 6, 8, 10, 12];
562 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
563
564 fd1_vals = avg_fd1';
565 fd2_vals = avg_fd2';
566
567 % Table creation
568 rowNames = {'FD1', 'FD2'};
569 columnNames = [h_labels, 'Exact'];
570 data = [fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2];
571 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
572 %display the table
573 disp('Average computation times table (only for successful runs): F27, n=10^4, quadratic'
    );
574 disp(T4);
575
576 %% Iteration
577
578 vec_times_ex_clean = vec_iter2_ex;
579 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
580 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
581
582 mat_times_fd1_clean = mat_iter2_fd1;
583 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
584 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
585
586 mat_times_fd2_clean = mat_iter2_fd2;
587 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
588 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
589
590 h_exponents = [2, 4, 6, 8, 10, 12];
591 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
592
593 fd1_vals = avg_fd1';
594 fd2_vals = avg_fd2';
595
596 rowNames = {'FD1', 'FD2'};
597 columnNames = [h_labels, 'Exact'];
598 data = [fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2];
599
600 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
601
602 disp('Average computation iteration table (only for successful runs): F27, n=10^4,
    quadratic');
603 disp(T5);
604
605 %% Function value
606
607 vec_times_ex_clean = vec_val2_ex;
608 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
609 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
610
611 mat_times_fd1_clean = mat_val2_fd1;
612 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
613 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
614
615 mat_times_fd2_clean = mat_val2_fd2;
616 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
617 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
618
619 h_exponents = [2, 4, 6, 8, 10, 12];
620 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
621
622 fd1_vals = avg_fd1';
623 fd2_vals = avg_fd2';
624
625 rowNames = {'FD1', 'FD2'};

```

```

626 columnNames = [ h_labels, 'Exact'];
627 data = [ fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2];
628
629 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
630
631 disp('Average computation fmin value table (only for successful runs): F27, n=10^4,
quadratic');
632 disp(T6);
633
634 %% VIOLATION
635
636 vec_times_ex_clean = vec_violations2_ex;
637 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
638 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
639
640 mat_times_fd1_clean = mat_violations2_fd1;
641 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
642 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
643
644 mat_times_fd2_clean = mat_violations2_fd2;
645 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
646 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
647
648 h_exponents = [2, 4, 6, 8, 10, 12];
649 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
650
651 fd1_vals = avg_fd1';
652 fd2_vals = avg_fd2';
653
654 rowNames = {'FD1', 'FD2'};
655 columnNames = [ h_labels, 'Exact'];
656 data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2];
657
658 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
659
660 disp('Average computation violation table (only for successful runs): F27, n=10^4,
quadratic');
661 disp(T14);
662
663 %% BT-SEQ
664
665 vec_bt_ex_clean = vec_bt2_ex;
666 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
667 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
668
669 mat_bt_fd1_clean = mat_bt2_fd1;
670 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
671 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
672
673 mat_bt_fd2_clean = mat_bt2_fd2;
674 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
675 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
676
677 h_exponents = [2, 4, 6, 8, 10, 12];
678 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
679
680 fd1_vals = avg_fd1';
681 fd2_vals = avg_fd2';
682
683 rowNames = {'FD1', 'FD2'};
684 columnNames = [ h_labels, 'Exact'];
685 data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2];
686
687 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
688
689 disp('Average computation bt iteration table (only for successful runs): F27, n=10^4,
quadratic');
690 disp(T15);
691
692 %% CG-SEQ
693
694 vec_bt_ex_clean = vec_cg_iter2_ex;
695 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;

```

```

696 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
697
698 mat_bt_fd1_clean = mat_cg_iter2_fd1;
699 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
700 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
701
702 mat_bt_fd2_clean = mat_cg_iter2_fd2;
703 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
704 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
705
706 h_exponents = [2, 4, 6, 8, 10, 12];
707 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
708
709 fd1_vals = avg_fd1';
710 fd2_vals = avg_fd2';
711
712 rowNames = {'FD1', 'FD2'};
713 columnNames = [h_labels, 'Exact'];
714 data = [fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2];
715
716 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
717
718 disp('Average computation cg iteration table (only for successful runs): F27, n=10^4, quadratic');
719 disp(T16);
720
721 %% Number of initial point converged
722
723 h_exponents = [2, 4, 6, 8, 10, 12];
724 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
725
726 fd1_vals = sum(mat_converged2_fd1, 2)';
727 fd2_vals = sum(mat_converged2_fd2, 2)';
728
729 rowNames = {'FD1', 'FD2'};
730 columnNames = [h_labels, 'Exact'];
731 data = [fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex)];
732
733 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
734
735 disp('Number of converged: F27, n=10^4, quadratic');
736 disp(T17);
737 %save the table if a file.xlsx
738 writetable(T4, 'results_f27_quad.xlsx', 'Sheet', 'time_4', 'WriteRowNames', true);
739 writetable(T5, 'results_f27_quad.xlsx', 'Sheet', 'niter_4', 'WriteRowNames', true);
740 writetable(T6, 'results_f27_quad.xlsx', 'Sheet', 'f_val_4', 'WriteRowNames', true);
741 writetable(T14, 'results_f27_quad.xlsx', 'Sheet', 'v_4', 'WriteRowNames', true);
742 writetable(T15, 'results_f27_quad.xlsx', 'Sheet', 'bt_4', 'WriteRowNames', true);
743 writetable(T16, 'results_f27_quad.xlsx', 'Sheet', 'cg_4', 'WriteRowNames', true);
744 writetable(T17, 'results_f27_quad.xlsx', 'Sheet', 'n_conv4', 'WriteRowNames', true);
745
746
747
748 %% n=10^5 (1e5)
749
750 rng(345989);
751
752 n=1e5;
753
754 kmax=1.5e3; % maximum number of iterations of Newton method
755 tolgrad=5e-7; % tolerance on gradient norm
756
757 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear system)
758 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
759
760 % Backtracking parameters
761 c1=1e-4;
762 rho=0.50;
763 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
764
765 x0=(1:n)'; % Initial point
766 N=10; % number of initial points to be generated

```

```

767
768 % Initial points:
769 Mat_points= repmat(x0,1,N+1);
770 rand_mat=2*rand(n, N)-1;
771 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
772
773 % Structure for EXACT derivatives
774 vec_times3_ex=zeros(1,N+1); % vector with execution times
775 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
776 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
777 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
778 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
779 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
780 mat_conv3_ex=zeros(15:N+1); %matrix with che last 15 values of rate of convergence for
    the starting point
781 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
782 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
783
784 JF_ex = @(x) JF_gen(x,true,false,0);
785
786
787 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
788 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
789 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
790 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
791 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
792 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
793 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
794 mat_conv3_fd1=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
795 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
796 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
797
798 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
799
800
801 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
802 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
803 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
804 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
805 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
806 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
807 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
808 mat_conv3_fd2=cell(6,N+1); %matrix with che last 15 values of rate of convergence for the
    starting point
809 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
810 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
811
812 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
813
814
815 for j =1:N+1
816     disp(['Condizione_iniziale_n.',num2str(j)])
817
818     % EXACT DERIVATIVES
819     tic;
820
821     [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);
822
823     vec_times3_ex(j)=toc;
824
825     disp(['Exact_derivatives:',flag3])
826     vec_converged3_ex(j)=converged3;
827
828     vec_val3_ex(j)=f3;
829     vec_grad3_ex(j)=gradf_norm3;
830     vec_iter3_ex(j)=k3;

```

```

831     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
832     vec_bt3_ex(j)=sum(btseq3)/k3;
833     vec_violations3_ex(j)=violations3;
834     last_vals = conv_ord3_ex(max(end-14,1):end);
835     mat_conv3_ex(:, j) = last_vals;
836
837
838     for i=2:2:12
839         h=10^(-i);
840
841         % FINITE DIFFERENCES 1
842         JF=@(x) JF_fd1(x,h);
843
844         tic;
845
846         [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);
847         mat_times3_fd1(i/2,j)=toc;
848
849
850         disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag3])
851         mat_converged3_fd1(i/2,j)=converged3;
852         mat_val3_fd1(i/2,j)=f3;
853         mat_grad3_fd1(i/2,j)=gradf_norm3;
854         mat_iter3_fd1(i/2,j)=k3;
855         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
856         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
857         mat_violations3_fd1(i/2,j)=violations3;
858         last_vals = conv_ord3_df1(max(end-14,1):end);
859         mat_conv3_fd1(i/2, j) = {last_vals};
860
861
862         % FINITE DIFFERENCES 2
863         JF=@(x) JF_fd2(x,h);
864
865         tic;
866
867         [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);
868         mat_times3_fd2(i/2,j)=toc;
869
870         disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag3])
871         mat_converged3_fd2(i/2,j)=converged3;
872         mat_val3_fd2(i/2,j)=f3;
873         mat_grad3_fd2(i/2,j)=gradf_norm3;
874         mat_iter3_fd2(i/2,j)=k3;
875         mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
876         mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
877         mat_violations3_fd2(i/2,j)=violations3;
878         last_vals = conv_ord3_df2(max(end-14,1):end);
879         mat_conv3_fd2(i/2, j) = {last_vals};
880
881     end
882 end
883
884 %% The plot has the same structure as n=10^3
885 num_initial_points = N + 1;
886 figure;
887 hold on;
888
889 for j = 1:num_initial_points
890     conv_ord_ex = mat_conv3_ex(:,j);
891     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
892     hold on;
893     for i =1:6
894         conv_ord_fd1 = mat_conv3_fd1{i, j};
895         conv_ord_fd2 = mat_conv3_fd2{i, j};
896         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
897         hold on;
898         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
899         hold on;

```

```

900     end
901 end
902
903 title('F27_10^5_□quadratic');
904 xlabel('Iterazione');
905 ylabel('Ordine_□di_□Convergenza');
906 legend({'Exact_□Derivatives', 'dif_□fin_1', 'dif_□fin_2'}, 'Location', 'Best');
907 grid on;
908 hold off;
909
910 %% Time
911
912 vec_times_ex_clean = vec_times3_ex;
913 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
914 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
915
916 mat_times_fd1_clean = mat_times3_fd1;
917 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
918 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
919
920 mat_times_fd2_clean = mat_times3_fd2;
921 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
922 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
923
924 h_exponents = [2, 4, 6, 8, 10, 12];
925 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
926
927 fd1_vals = avg_fd1';
928 fd2_vals = avg_fd2';
929
930 rowNames = {'FD1', 'FD2'};
931 columnNames = [h_labels, 'Exact'];
932 data = [fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3];
933
934 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
935
936 disp('Average_□computation_□times_□table_□(only_□for_□successful_□runs):_□F27,□n=10^5,□quadratic'
937 );
938 disp(T7);
939
940 %% Iteration
941
942 vec_times_ex_clean = vec_iter3_ex;
943 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
944 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
945
946 mat_times_fd1_clean = mat_iter3_fd1;
947 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
948 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
949
950 mat_times_fd2_clean = mat_iter3_fd2;
951 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
952 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
953
954 h_exponents = [2, 4, 6, 8, 10, 12];
955 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
956
957 fd1_vals = avg_fd1';
958 fd2_vals = avg_fd2';
959
960 rowNames = {'FD1', 'FD2'};
961 columnNames = [h_labels, 'Exact'];
962 data = [fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3];
963
964 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
965
966 disp('Average_□computation_□iteration_□table_□(only_□for_□successful_□runs):_□F27,□n=10^5,□
967 quadratic');
968 disp(T8);
969
970 %% function value
971
972 vec_times_ex_clean = vec_val3_ex;

```

```

971 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
972 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
973
974 mat_times_fd1_clean = mat_val3_fd1;
975 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
976 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
977
978 mat_times_fd2_clean = mat_val3_fd2;
979 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
980 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
981
982 h_exponents = [2, 4, 6, 8, 10, 12];
983 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
984
985 fd1_vals = avg_fd1';
986 fd2_vals = avg_fd2';
987
988 rowNames = {'FD1', 'FD2'};
989 columnNames = [h_labels, 'Exact'];
990 data = [fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3];
991
992 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
993
994 disp('Average computation fmin value table (only for successful runs): F27, n=10^5, quadratic');
995 disp(T9);
996
997 %% VIOLATION
998
999 vec_times_ex_clean = vec_violations3_ex;
1000 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
1001 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
1002
1003 mat_times_fd1_clean = mat_violations3_fd1;
1004 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1005 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
1006
1007 mat_times_fd2_clean = mat_violations3_fd2;
1008 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1009 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
1010
1011 h_exponents = [2, 4, 6, 8, 10, 12];
1012 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1013
1014 fd1_vals = avg_fd1';
1015 fd2_vals = avg_fd2';
1016
1017 rowNames = {'FD1', 'FD2'};
1018 columnNames = [h_labels, 'Exact'];
1019 data = [fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3];
1020
1021 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1022
1023 disp('Average computation violation table (only for successful runs): F27, n=10^5, quadratic');
1024 disp(T18);
1025
1026 %% BT-SEQ
1027
1028 vec_bt_ex_clean = vec_bt3_ex;
1029 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1030 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1031
1032 mat_bt_fd1_clean = mat_bt3_fd1;
1033 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1034 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1035
1036 mat_bt_fd2_clean = mat_bt3_fd2;
1037 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1038 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1039
1040 h_exponents = [2, 4, 6, 8, 10, 12];
1041 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);

```

```

1042 fd1_vals = avg_fd1';
1043 fd2_vals = avg_fd2';
1044
1045 rowNames = {'FD1', 'FD2'};
1046 columnNames = [ h_labels, 'Exact'];
1047 data = [ fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3;];
1048
1049 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1050
1051 disp('Average computation bt iteration table (only for successful runs): F27, n=10^5, quadratic');
1052 disp(T19);
1053
1054 %% CG-SEQ
1055
1056 vec_bt_ex_clean = vec_cg_iter3_ex;
1057 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1058 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1059
1060 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1061 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1062 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1063
1064 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1065 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1066 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1067
1068 h_exponents = [2, 4, 6, 8, 10, 12];
1069 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1070
1071 fd1_vals = avg_fd1';
1072 fd2_vals = avg_fd2';
1073
1074 rowNames = {'FD1', 'FD2'};
1075 columnNames = [ h_labels, 'Exact'];
1076 data = [ fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3;];
1077
1078 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1079
1080 disp('Average computation cg iteration table (only for successful runs): F27, n=10^5, quadratic');
1081 disp(T20);
1082
1083 %% Number of initial condition converged
1084
1085 h_exponents = [2, 4, 6, 8, 10, 12];
1086 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1087
1088 fd1_vals = sum(mat_converged3_fd1, 2)';
1089 fd2_vals = sum(mat_converged3_fd2, 2)';
1090
1091 rowNames = {'FD1', 'FD2'};
1092 columnNames = [ h_labels, 'Exact'];
1093 data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex);];
1094
1095 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1096
1097 disp('Number of converged: F27, n=10^5, quadratic');
1098 disp(T21);
1099 %save the tables
1100
1101 writetable(T7, 'results_f27_quad.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1102 writetable(T8, 'results_f27_quad.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1103 writetable(T9, 'results_f27_quad.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1104 writetable(T18, 'results_f27_quad.xlsx', 'Sheet', 'v_5', 'WriteRowNames', true);
1105 writetable(T19, 'results_f27_quad.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1106 writetable(T20, 'results_f27_quad.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1107 writetable(T21, 'results_f27_quad.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1108
1109
1110
1111 %% table with the result of the exact derivatives
1112

```



```

1113 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1114         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1115         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1116         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1117         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1118         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1119         sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];
1120
1121 rowNames = {'Average_Time', 'Average_Iter', 'Average_fval', 'Violation', 'Average_iter_Bt',
1122            'Average_iter_cg', 'N_converged'};
1123 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1124
1125 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1126 disp(T_compare)
1127
1128 writetable(T_compare, 'results_f27_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames', true);

```

```

1  %% FUNCTION 16 (with different initial points)- with exact derivatives and finite
2  differences - QUADRATIC TERM OF CONVERGENCE
3
4  sparse=true;
5
6  F = @(x) F16(x); % Defining F16 as function handle
7  JF_gen = @(x,exact,fin_dif2,h) JF16(x,exact,fin_dif2,h); % Defining JF16 as function
8  handle
9  HF_gen= @(x,exact,fin_dif2,h) HF16(x,sparse,exact,fin_dif2,h); % Defining HF16 as
10 function handle (sparse version)
11
12 load forcing_terms.mat % possible terms for adaptive tolerance
13
14 %% n=10^3 (1e3)
15
16 rng(345989);
17
18 n=1e3;
19
20 kmax=1.5e3; % maximum number of iterations of Newton method
21 tolgrad=5e-7; % tolerance on gradient norm
22
23 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
24 system)
25 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
26
27 % Backtracking parameters
28 c1=1e-4;
29 rho=0.50;
30 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
31
32 x0 = ones(n, 1); % initial point
33 N=10; % number of initial points to be generated
34
35 % Initial points:
36 Mat_points= repmat(x0,1,N+1);
37 rand_mat=2*rand(n, N)-1;
38 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
39
40 % Structure for EXACT derivatives
41 vec_times1_ex=zeros(1,N+1); % vector with execution times
42 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
43 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
44 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
45 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
46 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
47 mat_conv1_ex=zeros(12, N+1); %matrix with the last 12 values of rate of convergence for
48 the starting point
49 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
50 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
51 condition in Newton method
52
53 JF_ex = @(x) JF_gen(x,true,false,0);
54 HF_ex = @(x) HF_gen(x,true,false,0);

```

```

49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv1_fd1=cell(6, N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv1_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1
79     disp(['Condizione_iniziale_n.',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83     [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);
84     vec_times1_ex(j)=toc;
85
86     disp(['Exact_derivatives:',flag1])
87     vec_converged1_ex(j)=converged1;
88     vec_val1_ex(j)=f1;
89     vec_grad1_ex(j)=gradf_norm1;
90     vec_iter1_ex(j)=k1;
91     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
92     vec_bt1_ex(j)=sum(btseq1)/k1;
93     vec_violations1_ex(j)=violations1;
94     last_vals = conv_ord1_ex(max(end-11,1):end);
95     mat_conv1_ex(:, j) = last_vals;
96
97
98     for i=2:2:12
99         h=10^(-i);
100
101         % FINITE DIFFERENCES 1
102         JF=@(x)JF_fd1(x,h);
103         HF=@(x)HF_fd1(x,h);
104         tic;
105         [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);
106         mat_times1_fd1(i/2,j)=toc;
107
108         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag1])
109         mat_converged1_fd1(i/2,j)=converged1;
110         mat_val1_fd1(i/2,j)=f1;
111         mat_grad1_fd1(i/2,j)=gradf_norm1;
112         mat_iter1_fd1(i/2,j)=k1;

```

```

113     mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
114     mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
115     mat_violations1_fd1(i/2,j)=violations1;
116     last_vals = conv_ord1_df1(max(end-11,1):end);
117     mat_conv1_fd1(i/2, j) = {last_vals};
118
119
120     % FINITE DIFFERENCES 2
121     JF=@(x) JF_fd2(x,h);
122     HF=@(x) HF_fd2(x,h);
123     tic;
124     [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);
125     mat_times1_fd2(i/2,j)=toc;
126
127     disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag1])
128     mat_converged1_fd2(i/2,j)=converged1;
129     mat_val1_fd2(i/2,j)=f1;
130     mat_grad1_fd2(i/2,j)=gradf_norm1;
131     mat_iter1_fd2(i/2,j)=k1;
132     mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
133     mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
134     mat_violations1_fd2(i/2,j)=violations1;
135     last_vals = conv_ord1_df2(max(end-11,1):end);
136     mat_conv1_fd2(i/2, j) = {last_vals};
137
138     end
139 end
140
141
142 %% Plot of the last 12 values of experimentale rate of convergence
143 num_initial_points = N + 1;
144 figure;
145 hold on;
146
147 % Plot for every initial condition
148 for j = 1:num_initial_points
149     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
150     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
151     hold on;
152     for i =1:6
153         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
154         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
155         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
156         hold on;
157         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
158         hold on;
159     end
160 end
161
162 % title and legend
163 title('F16_10^3_quadratic');
164 xlabel('Iterazione');
165 ylabel('Ordine di Convergenza');
166 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
167 grid on;
168 hold off;
169
170
171 %% Execution Time
172
173 % Exact Derivative
174 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
175 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
176 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
177
178 % FD1
179 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
180 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
181 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
182

```

```

183 % FD2
184 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
185 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
186 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
187
188 % Creation of the labels
189 h_exponents = [2, 4, 6, 8, 10, 12];
190 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
191
192 fd1_vals = avg_fd1';
193 fd2_vals = avg_fd2';
194
195 % Table construction with exact for both the row
196 rowNames = {'FD1', 'FD2'};
197 columnNames = [ h_labels, 'Exact'];
198 data = [ fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
199 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
200
201 % visualization
202 disp('Average computation times table (only for successful runs): F16, n=10^3, quadratic'
    );
203 disp(T1);
204
205
206 %% All the tables has the same structure
207 %% Iteration
208
209 vec_times_ex_clean = vec_iter1_ex;
210 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
211 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
212
213 mat_times_fd1_clean = mat_iter1_fd1;
214 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
215 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
216
217 mat_times_fd2_clean = mat_iter1_fd2;
218 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
219 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
220
221 h_exponents = [2, 4, 6, 8, 10, 12];
222 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
223
224 fd1_vals = avg_fd1';
225 fd2_vals = avg_fd2';
226
227 rowNames = {'FD1', 'FD2'};
228 columnNames = [ h_labels, 'Exact'];
229 data = [ fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];
230
231 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
232
233 disp('Average computation iteration table (only for successful runs): F16, n=10^3,
    quadratic');
234 disp(T2);
235
236 %% F value
237
238 vec_times_ex_clean = vec_val1_ex;
239 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
240 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
241
242 mat_times_fd1_clean = mat_val1_fd1;
243 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
244 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
245
246 mat_times_fd2_clean = mat_val1_fd2;
247 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
248 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
249
250 h_exponents = [2, 4, 6, 8, 10, 12];
251 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
252

```

```

253 fd1_vals = avg_fd1';
254 fd2_vals = avg_fd2';
255
256 rowNames = {'FD1', 'FD2'};
257 columnNames = [ h_labels, 'Exact'];
258 data = [ fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1;];
259
260 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
261
262 disp('Average computation fmin value table (only for successful runs): F16, n=10^3,
quadratic');
263 disp(T3);
264
265 %% VIOLATION
266
267 vec_times_ex_clean = vec_violations1_ex;
268 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
269 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
270
271 mat_times_fd1_clean = mat_violations1_fd1;
272 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
273 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
274
275 mat_times_fd2_clean = mat_violations1_fd2;
276 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
277 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
278
279 h_exponents = [2, 4, 6, 8, 10, 12];
280 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
281
282 %
283 fd1_vals = avg_fd1';
284 fd2_vals = avg_fd2';
285
286 rowNames = {'FD1', 'FD2'};
287 columnNames = [ h_labels, 'Exact'];
288 data = [ fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1;];
289
290 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
291
292 disp('Average computation violation table (only for successful runs): F16, n=10^3,
quadratic');
293 disp(T10);
294
295
296 %% BT-SEQ
297 vec_bt_ex_clean = vec_bt1_ex;
298 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
299 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
300
301 mat_bt_fd1_clean = mat_bt1_fd1;
302 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
303 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
304
305 mat_bt_fd2_clean = mat_bt1_fd2;
306 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
307 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
308
309 h_exponents = [2, 4, 6, 8, 10, 12];
310 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
311
312 fd1_vals = avg_fd1';
313 fd2_vals = avg_fd2';
314
315 rowNames = {'FD1', 'FD2'};
316 columnNames = [ h_labels, 'Exact'];
317 data = [ fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
318
319 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
320
321 disp('Average computation bt iteration table (only for successful runs): F16, n=10^3,
quadratic');
322 disp(T11);

```

```

323
324 %% CG-SEQ
325
326 vec_bt_ex_clean = vec_cg_iter1_ex;
327 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
328 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
329
330 mat_bt_fd1_clean = mat_cg_iter1_fd1;
331 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
332 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
333
334 mat_bt_fd2_clean = mat_cg_iter1_fd2;
335 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
336 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
337
338 h_exponents = [2, 4, 6, 8, 10, 12];
339 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
340
341 fd1_vals = avg_fd1';
342 fd2_vals = avg_fd2';
343
344 rowNames = {'FD1', 'FD2'};
345 columnNames = [h_labels, 'Exact'];
346 data = [fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1];
347
348 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
349
350 disp('Average computation cg iteration table (only for successful runs): F16, n=10^3, quadratic');
351 disp(T12);
352
353 %% Number of starting point converged
354
355 h_exponents = [2, 4, 6, 8, 10, 12];
356 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
357
358 fd1_vals = sum(mat_converged1_fd1, 2)';
359 fd2_vals = sum(mat_converged1_fd2, 2)';
360
361 rowNames = {'FD1', 'FD2'};
362 columnNames = [h_labels, 'Exact'];
363 data = [fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex)];
364
365 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
366
367 disp('Number of converged: F16, n=10^3, quadratic');
368 disp(T13);
369 %save the table in a file.xlsx
370 writetable(T1, 'results_f16_quad.xlsx', 'Sheet', 'time_3', 'WriteRowNames', true);
371 writetable(T2, 'results_f16_quad.xlsx', 'Sheet', 'niter_3', 'WriteRowNames', true);
372 writetable(T3, 'results_f16_quad.xlsx', 'Sheet', 'f_val_3', 'WriteRowNames', true);
373 writetable(T10, 'results_f16_quad.xlsx', 'Sheet', 'viol_3', 'WriteRowNames', true);
374 writetable(T11, 'results_f16_quad.xlsx', 'Sheet', 'bt_3', 'WriteRowNames', true);
375 writetable(T12, 'results_f16_quad.xlsx', 'Sheet', 'cg_3', 'WriteRowNames', true);
376 writetable(T13, 'results_f16_quad.xlsx', 'Sheet', 'n_conv3', 'WriteRowNames', true);
377
378
379
380 %% n=10^4 (1e4)
381
382 rng(345989);
383
384 n=1e4;
385
386 kmax=1.5e3; % maximum number of iterations of Newton method
387 tolgrad=1e-5; % tolerance on gradient norm
388
389 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear system)
390 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
391
392 % Backtracking parameters
393 c1=1e-4;

```

```

394 rho=0.50;
395 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
396
397 x0 = ones(n, 1); % initial point
398 N=10; % number of initial points to be generated
399
400 % Initial points:
401 Mat_points= repmat(x0,1,N+1);
402 rand_mat=2*rand(n, N)-1;
403 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
404
405 % Structure for EXACT derivatives
406 vec_times2_ex=zeros(1,N+1); % vector with execution times
407 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
408 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
409 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
410 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
411 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
412 mat_conv2_ex=zeros(12,N+1); %matrix with the last 12 values of rate of convergence for the
    starting point
413 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
414 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
415
416 JF_ex = @(x) JF_gen(x,true,false,0);
417 HF_ex = @(x) HF_gen(x,true,false,0);
418
419 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
420 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
421 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
422 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
423 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
424 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
425 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
426 mat_conv2_fd1=cell(6,N+1); %matrix with the last 12 values of rate of convergence for the
    starting point
427 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
428 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
429
430 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
431 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
432
433 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
434 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
435 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
436 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
437 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
438 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
439 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
440 mat_conv2_fd2=cell(6,N+1); %matrix with the last 12 values of rate of convergence for the
    starting point
441 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
442 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
443
444 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
445 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
446
447 for j =1:N+1
448     disp(['Condizione_iniziale_n. ',num2str(j)])
449
450     % EXACT DERIVATIVES
451     tic;
452     [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);
453     vec_times2_ex(j)=toc;
454
455     disp(['Exact_derivatives: ',flag2])
456     vec_converged2_ex(j)=converged2;
457     vec_val2_ex(j)=f2;

```

```

458     vec_grad2_ex(j)=gradf_norm2;
459     vec_iter2_ex(j)=k2;
460     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
461     vec_bt2_ex(j)=sum(btseq2)/k2;
462     vec_violations2_ex(j)=violations2;
463     last_vals = conv_ord2_ex(max(end-11,1):end);
464     mat_conv2_ex(:, j) = last_vals;
465
466
467     for i=2:2:12
468         h=10^(-i);
469
470         % FINITE DIFFERENCES 1
471         JF=@(x) JF_fd1(x,h);
472         HF=@(x) HF_fd1(x,h);
473         tic;
474         [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df1,flag2, converged2,
475             violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
476             fterms_quad, cg_maxit,z0, c1, rho, btmax);
477         mat_times2_fd1(i/2,j)=toc;
478
479         disp(['Finite differences (classical version) with h=1e-',num2str(i),' : ',flag2])
480         mat_converged2_fd1(i/2,j)=converged2;
481         mat_val2_fd1(i/2,j)=f2;
482         mat_grad2_fd1(i/2,j)=gradf_norm2;
483         mat_iter2_fd1(i/2,j)=k2;
484         mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
485         mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
486         mat_violations2_fd1(i/2,j)=violations2;
487         last_vals = conv_ord2_df1(max(end-11,1):end);
488         mat_conv2_fd1(i/2, j) = {last_vals};
489
490
491         % FINITE DIFFERENCES 2
492         JF=@(x) JF_fd2(x,h);
493         HF=@(x) HF_fd2(x,h);
494         tic;
495         [x2, f2, gradf_norm2, k2, xseq2, btseq2,cgiterseq2,conv_ord2_df2,flag2, converged2,
496             violations2] = truncated_newton(Mat_points(:,j), F, JF, HF, kmax, tolgrad,
497             fterms_quad, cg_maxit,z0, c1, rho, btmax);
498         mat_times2_fd2(i/2,j)=toc;
499
500         disp(['Finite differences (new version) with h=1e-',num2str(i),' : ',flag2])
501         mat_converged2_fd2(i/2,j)=converged2;
502         mat_val2_fd2(i/2,j)=f2;
503         mat_grad2_fd2(i/2,j)=gradf_norm2;
504         mat_iter2_fd2(i/2,j)=k2;
505         mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
506         mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
507         mat_violations2_fd2(i/2,j)=violations2;
508         last_vals = conv_ord2_df2(max(end-11,1):end);
509         mat_conv2_fd2(i/2, j) = {last_vals};
510
511     end
512 end
513
514 %% The Plot has the same structure
515 num_initial_points = N + 1;
516 figure;
517 hold on;
518
519 for j = 1:num_initial_points
520     conv_ord_ex = mat_conv2_ex(:,j);
521     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
522     hold on;
523     for i =1:6
524         conv_ord_fd1 = mat_conv2_fd1{i, j};
525         conv_ord_fd2 = mat_conv2_fd2{i, j};
526         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
527         hold on;
528         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);

```



```

527         hold on;
528     end
529 end
530
531 title('F16_10^4_quadratic');
532 xlabel('Iterazione');
533 ylabel('Ordine di Convergenza');
534 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
535 grid on;
536 hold off;
537
538
539
540 %% Execution time
541
542 % Exact derivative
543 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
544 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
545 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
546
547 % FD1
548 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
549 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
550 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
551
552 % FD2
553 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
554 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
555 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
556
557 % Creation of the labels
558 h_exponents = [2, 4, 6, 8, 10, 12];
559 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
560
561 fd1_vals = avg_fd1';
562 fd2_vals = avg_fd2';
563
564 % Table creation
565 rowNames = {'FD1', 'FD2'};
566 columnNames = [ h_labels, 'Exact'];
567 data = [ fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
568 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
569 %display the table
570 disp('Average computation times table (only for successful runs): F16, n=10^4, quadratic'
    );
571 disp(T4);
572
573 %% Iteration
574
575 vec_times_ex_clean = vec_iter2_ex;
576 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
577 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
578
579 mat_times_fd1_clean = mat_iter2_fd1;
580 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
581 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
582
583 mat_times_fd2_clean = mat_iter2_fd2;
584 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
585 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
586
587 h_exponents = [2, 4, 6, 8, 10, 12];
588 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
589
590 fd1_vals = avg_fd1';
591 fd2_vals = avg_fd2';
592
593 rowNames = {'FD1', 'FD2'};
594 columnNames = [ h_labels, 'Exact'];
595 data = [ fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];
596

```

```

597 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
598
599 disp('Average computation iteration table (only for successful runs): F16, n=10^4, quadratic');
600 disp(T5);
601
602 %% Function value
603
604 vec_times_ex_clean = vec_val2_ex;
605 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
606 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
607
608 mat_times_fd1_clean = mat_val2_fd1;
609 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
610 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
611
612 mat_times_fd2_clean = mat_val2_fd2;
613 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
614 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
615
616 h_exponents = [2, 4, 6, 8, 10, 12];
617 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
618
619 fd1_vals = avg_fd1';
620 fd2_vals = avg_fd2';
621
622 rowNames = {'FD1', 'FD2'};
623 columnNames = [h_labels, 'Exact'];
624 data = [fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];
625
626 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
627
628 disp('Average computation fmin value table (only for successful runs): F16, n=10^4, quadratic');
629 disp(T6);
630
631 %% VIOLATION
632
633 vec_times_ex_clean = vec_violations2_ex;
634 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
635 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
636
637 mat_times_fd1_clean = mat_violations2_fd1;
638 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
639 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
640
641 mat_times_fd2_clean = mat_violations2_fd2;
642 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
643 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
644
645 h_exponents = [2, 4, 6, 8, 10, 12];
646 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
647
648 fd1_vals = avg_fd1';
649 fd2_vals = avg_fd2';
650
651 rowNames = {'FD1', 'FD2'};
652 columnNames = [h_labels, 'Exact'];
653 data = [fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];
654
655 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
656
657 disp('Average computation violation table (only for successful runs): F16, n=10^4, quadratic');
658 disp(T14);
659
660 %% BT-SEQ
661
662 vec_bt_ex_clean = vec_bt2_ex;
663 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
664 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
665
666 mat_bt_fd1_clean = mat_bt2_fd1;

```

```

667 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
668 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
669
670 mat_bt_fd2_clean = mat_bt2_fd2;
671 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
672 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
673
674 h_exponents = [2, 4, 6, 8, 10, 12];
675 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
676
677 fd1_vals = avg_fd1';
678 fd2_vals = avg_fd2';
679
680 rowNames = {'FD1', 'FD2'};
681 columnNames = [h_labels, 'Exact'];
682 data = [fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2];
683
684 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
685
686 disp('Average computation bt iteration table (only for successful runs): F16, n=10^4, quadratic');
687 disp(T15);
688
689 %% CG-SEQ
690
691 vec_bt_ex_clean = vec_cg_iter2_ex;
692 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
693 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
694
695 mat_bt_fd1_clean = mat_cg_iter2_fd1;
696 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
697 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
698
699 mat_bt_fd2_clean = mat_cg_iter2_fd2;
700 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
701 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
702
703 h_exponents = [2, 4, 6, 8, 10, 12];
704 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
705
706 fd1_vals = avg_fd1';
707 fd2_vals = avg_fd2';
708
709 rowNames = {'FD1', 'FD2'};
710 columnNames = [h_labels, 'Exact'];
711 data = [fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2];
712
713 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
714
715 disp('Average computation cg iteration table (only for successful runs): F16, n=10^4, quadratic');
716 disp(T16);
717
718 %% Number of initial point converged
719
720 h_exponents = [2, 4, 6, 8, 10, 12];
721 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
722
723 fd1_vals = sum(mat_converged2_fd1, 2)';
724 fd2_vals = sum(mat_converged2_fd2, 2)';
725
726 rowNames = {'FD1', 'FD2'};
727 columnNames = [h_labels, 'Exact'];
728 data = [fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex)];
729
730 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
731
732 disp('Number of converged: F16, n=10^4, quadratic');
733 disp(T17);
734 %save the table if a file.xlsx
735 writetable(T4, 'results_f16_quad.xlsx', 'Sheet', 'time_4', 'WriteRowNames', true);
736 writetable(T5, 'results_f16_quad.xlsx', 'Sheet', 'niter_4', 'WriteRowNames', true);
737 writetable(T6, 'results_f16_quad.xlsx', 'Sheet', 'f_val_4', 'WriteRowNames', true);

```

```

738 writetable(T14, 'results_f16_quad.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
739 writetable(T15, 'results_f16_quad.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
740 writetable(T16, 'results_f16_quad.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
741 writetable(T17, 'results_f16_quad.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);
742
743
744
745 %% n=105 (1e5)
746
747 rng(345989);
748
749 n=1e5;
750
751 kmax=1.5e3; % maximum number of iterations of Newton method
752 tolgrad=5e-4; % tolerance on gradient norm
753
754 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
      system)
755 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
756
757 % Backtracking parameters
758 c1=1e-4;
759 rho=0.50;
760 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
761
762 x0 = ones(n, 1); % initial point
763 N=10; % number of initial points to be generated
764
765 % Initial points:
766 Mat_points=repmat(x0,1,N+1);
767 rand_mat=2*rand(n, N)-1;
768 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
769
770 % Structure for EXACT derivatives
771 vec_times3_ex=zeros(1,N+1); % vector with execution times
772 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
773 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
774 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
775 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
776 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
777 mat_conv3_ex=zeros(12:N+1);%matrix with che last 12 values of rate of convergence for the
      starting point
778 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
779 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
      condition in Newton method
780
781 JF_ex = @(x) JF_gen(x,true,false,0);
782 HF_ex = @(x) HF_gen(x,true,false,0);
783
784 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
785 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
786 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
787 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
788 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
789 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
790 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
791 mat_conv3_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
      starting point
792 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
793 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
794
795 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
796 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
797
798 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
      x_j) as increment)
799 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
800 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
801 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
802 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
803 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
804 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations

```

```

805 mat_conv3_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
      starting point
806 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
807 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
808
809 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
810 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
811
812 for j =1:N+1
813     disp(['Condizione_iniziale_n.',num2str(j)])
814
815     % EXACT DERIVATIVES
816     tic;
817     [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);
818     vec_times3_ex(j)=toc;
819
820     disp(['Exact_derivatives:',flag3])
821     vec_converged3_ex(j)=converged3;
822     vec_val3_ex(j)=f3;
823     vec_grad3_ex(j)=gradf_norm3;
824     vec_iter3_ex(j)=k3;
825     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
826     vec_bt3_ex(j)=sum(btseq3)/k3;
827     vec_violations3_ex(j)=violations3;
828     last_vals = conv_ord3_ex(max(end-11,1):end);
829     mat_conv3_ex(:, j) = last_vals;
830
831     for i=2:2:12
832         h=10^(-i);
833
834         % FINITE DIFFERENCES 1
835         JF=@(x) JF_fd1(x,h);
836         HF=@(x) HF_fd1(x,h);
837         tic;
838         [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);
839         mat_times3_fd1(i/2,j)=toc;
840
841         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag3])
842         mat_converged3_fd1(i/2,j)=converged3;
843         mat_val3_fd1(i/2,j)=f3;
844         mat_grad3_fd1(i/2,j)=gradf_norm3;
845         mat_iter3_fd1(i/2,j)=k3;
846         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
847         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
848         mat_violations3_fd1(i/2,j)=violations3;
849         last_vals = conv_ord3_df1(max(end-11,1):end);
850         mat_conv3_fd1(i/2, j) = {last_vals};
851
852
853         % FINITE DIFFERENCES 2
854         JF=@(x) JF_fd2(x,h);
855         HF=@(x) HF_fd2(x,h);
856         tic;
857         [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);
858         mat_times3_fd2(i/2,j)=toc;
859
860         disp(['Finite_differences_(new_version)_with_h=1e-',num2str(i),'_',flag3])
861         mat_converged3_fd2(i/2,j)=converged3;
862         mat_val3_fd2(i/2,j)=f3;
863         mat_grad3_fd2(i/2,j)=gradf_norm3;
864         mat_iter3_fd2(i/2,j)=k3;
865         mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
866         mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
867         mat_violations3_fd2(i/2,j)=violations3;
868         last_vals = conv_ord3_df2(max(end-11,1):end);
869         mat_conv3_fd2(i/2, j) = {last_vals};

```

```

870
871     end
872 end
873 %% The plot has the same structure as n=10^3
874 num_initial_points = N + 1;
875 figure;
876 hold on;
877
878 for j = 1:num_initial_points
879     conv_ord_ex = mat_conv3_ex(:,j);
880     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
881     hold on;
882     for i =1:6
883         conv_ord_fd1 = mat_conv3_fd1{i, j};
884         conv_ord_fd2 = mat_conv3_fd2{i, j};
885         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
886         hold on;
887         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
888         hold on;
889     end
890 end
891
892 title('F16_10^5_Quadratic');
893 xlabel('Iterazione');
894 ylabel('Ordine di Convergenza');
895 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
896 grid on;
897 hold off;
898
899 %% Time
900
901 vec_times_ex_clean = vec_times3_ex;
902 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
903 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
904
905 mat_times_fd1_clean = mat_times3_fd1;
906 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
907 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
908
909 mat_times_fd2_clean = mat_times3_fd2;
910 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
911 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
912
913 h_exponents = [2, 4, 6, 8, 10, 12];
914 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
915
916 fd1_vals = avg_fd1';
917 fd2_vals = avg_fd2';
918
919 rowNames = {'FD1', 'FD2'};
920 columnNames = [ h_labels, 'Exact'];
921 data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3;];
922
923 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
924
925 disp('Average computation times table (only for successful runs): F16, n=10^5, Quadratic');
926 disp(T7);
927
928 %% Iteration
929
930 vec_times_ex_clean = vec_iter3_ex;
931 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
932 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
933
934 mat_times_fd1_clean = mat_iter3_fd1;
935 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
936 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
937
938 mat_times_fd2_clean = mat_iter3_fd2;
939 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
940 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
941

```

```

942 h_exponents = [2, 4, 6, 8, 10, 12];
943 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
944
945 fd1_vals = avg_fd1';
946 fd2_vals = avg_fd2';
947
948 rowNames = {'FD1', 'FD2'};
949 columnNames = [h_labels, 'Exact'];
950 data = [fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3];
951
952 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
953
954 disp('Average computation iteration table (only for successful runs): F16, n=10^5, quadratic');
955 disp(T8);
956
957 %% function value
958
959 vec_times_ex_clean = vec_val3_ex;
960 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
961 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
962
963 mat_times_fd1_clean = mat_val3_fd1;
964 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
965 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
966
967 mat_times_fd2_clean = mat_val3_fd2;
968 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
969 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
970
971 h_exponents = [2, 4, 6, 8, 10, 12];
972 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
973
974 fd1_vals = avg_fd1';
975 fd2_vals = avg_fd2';
976
977 rowNames = {'FD1', 'FD2'};
978 columnNames = [h_labels, 'Exact'];
979 data = [fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3];
980
981 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
982
983 disp('Average computation fmin value table (only for successful runs): F16, n=10^5, quadratic');
984 disp(T9);
985
986 %% VIOLATION
987
988 vec_times_ex_clean = vec_violations3_ex;
989 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
990 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
991
992 mat_times_fd1_clean = mat_violations3_fd1;
993 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
994 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
995
996 mat_times_fd2_clean = mat_violations3_fd2;
997 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
998 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
999
1000 h_exponents = [2, 4, 6, 8, 10, 12];
1001 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1002
1003 fd1_vals = avg_fd1';
1004 fd2_vals = avg_fd2';
1005
1006 rowNames = {'FD1', 'FD2'};
1007 columnNames = [h_labels, 'Exact'];
1008 data = [fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3];
1009
1010 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1011
1012 disp('Average computation violation table (only for successful runs): F16, n=10^5, quadratic');

```

```

    quadratic');
1013 disp(T18);
1014
1015 %% BT-SEQ
1016
1017 vec_bt_ex_clean = vec_bt3_ex;
1018 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1019 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1020
1021 mat_bt_fd1_clean = mat_bt3_fd1;
1022 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1023 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1024
1025 mat_bt_fd2_clean = mat_bt3_fd2;
1026 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1027 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1028
1029 h_exponents = [2, 4, 6, 8, 10, 12];
1030 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1031
1032 fd1_vals = avg_fd1';
1033 fd2_vals = avg_fd2';
1034
1035 rowNames = {'FD1', 'FD2'};
1036 columnNames = [h_labels, 'Exact'];
1037 data = [fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3];
1038
1039 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1040
1041 disp('Average computation bt iteration table (only for successful runs): F16, n=10^5, quadratic');
1042 disp(T19);
1043
1044 %% CG-SEQ
1045
1046 vec_bt_ex_clean = vec_cg_iter3_ex;
1047 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1048 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1049
1050 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1051 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1052 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1053
1054 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1055 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1056 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1057
1058 h_exponents = [2, 4, 6, 8, 10, 12];
1059 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1060
1061 fd1_vals = avg_fd1';
1062 fd2_vals = avg_fd2';
1063
1064 rowNames = {'FD1', 'FD2'};
1065 columnNames = [h_labels, 'Exact'];
1066 data = [fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];
1067
1068 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1069
1070 disp('Average computation cg iteration table (only for successful runs): F16, n=10^5, quadratic');
1071 disp(T20);
1072
1073 %% Number of initial condition converged
1074
1075 h_exponents = [2, 4, 6, 8, 10, 12];
1076 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1077
1078 fd1_vals = sum(mat_converged3_fd1, 2)';
1079 fd2_vals = sum(mat_converged3_fd2, 2)';
1080
1081 rowNames = {'FD1', 'FD2'};
1082 columnNames = [h_labels, 'Exact'];

```



```

1083 data = [ fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1084
1085 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1086
1087 disp('Number of converged F16, n=10^5, quadratic');
1088 disp(T21);
1089 %save the tables
1090
1091 writetable(T7, 'results_f16_quad.xlsx', 'Sheet', 'time_5','WriteRowNames', true);
1092 writetable(T8, 'results_f16_quad.xlsx', 'Sheet', 'niter_5','WriteRowNames', true);
1093 writetable(T9, 'results_f16_quad.xlsx', 'Sheet', 'f_val_5','WriteRowNames', true);
1094 writetable(T18, 'results_f16_quad.xlsx', 'Sheet', 'viol_5','WriteRowNames', true);
1095 writetable(T19, 'results_f16_quad.xlsx', 'Sheet', 'bt_5','WriteRowNames', true);
1096 writetable(T20, 'results_f16_quad.xlsx', 'Sheet', 'cg_5','WriteRowNames', true);
1097 writetable(T21, 'results_f16_quad.xlsx', 'Sheet', 'n_conv5','WriteRowNames', true);
1098
1099
1100 %% table with the result of the exact derivatives
1101 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1102         avg_exact_i1, avg_exact_i2, avg_exact_i3;
1103         avg_exact_f1, avg_exact_f2, avg_exact_f3;
1104         avg_exact_v1, avg_exact_v2, avg_exact_v3;
1105         avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1106         avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1107         sum(vec_converged1_ex),sum(vec_converged2_ex),sum(vec_converged3_ex)];
1108
1109 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
1110             'Average iter cg', 'N converged'};
1111 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1112
1113 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1114 disp(T_compare)
1115
1116 writetable(T_compare, 'results_f16_quad.xlsx', 'Sheet', 'ExactComparison', 'WriteRowNames', true);

```

```

1  %% FUNCTION 16 (with different initial points)- with exact derivatives and finite
2  differences
3
4  sparse=true;
5
6  F = @(x) F16(x); % Defining F16 as function handle
7  JF_gen = @(x,exact,fin_dif2,h) JF16(x,exact,fin_dif2,h); % Defining JF16 as function
8  handle
9  HF_gen = @(x,exact,fin_dif2,h) HF16(x,sparse,exact,fin_dif2,h); % Defining HF16 as
10 function handle (sparse version)
11
12 load forcing_terms.mat % possible terms for adaptive tolerance
13
14 %% n=10^3 (1e3)
15
16 rng(345989);
17
18 n=1e3;
19
20 kmax=1.5e3; % maximum number of iterations of Newton method
21 tolgrad=5e-7; % tolerance on gradient norm
22
23 cg_maxit=50; % maximum number of iterations of conjugate gradient method (for the linear
24 system)
25 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
26
27 % Backtracking parameters
28 c1=1e-4;
29 rho=0.50;
30 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
31
32 x0 = ones(n, 1); % initial point
33 N=10; % number of initial points to be generated
34
35 % Initial points:
36 Mat_points= repmat(x0,1,N+1);
37 rand_mat=2*rand(n, N)-1;

```

```

34 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
35
36 % Structure for EXACT derivatives
37 vec_times1_ex=zeros(1,N+1); % vector with execution times
38 vec_val1_ex=zeros(1,N+1); %vector with minimal values found
39 vec_grad1_ex=zeros(1,N+1); %vector with final gradient
40 vec_iter1_ex=zeros(1,N+1); %vector with number of iterations
41 vec_cg_iter1_ex=zeros(1,N+1); %vector with mean number of inner iterations
42 vec_bt1_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
43 mat_conv1_ex=zeros(12, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
44 vec_converged1_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
45 vec_violations1_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
46
47 JF_ex = @(x) JF_gen(x,true,false,0);
48 HF_ex = @(x) HF_gen(x,true,false,0);
49
50 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
51 mat_times1_fd1=zeros(6,N+1); % matrix with execution times
52 mat_val1_fd1=zeros(6,N+1); %matrix with minimal values found
53 mat_grad1_fd1=zeros(6,N+1); %matrix with final gradient
54 mat_iter1_fd1=zeros(6,N+1); %matrix with number of iterations
55 mat_cg_iter1_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
56 mat_bt1_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
57 mat_conv1_fd1=cell(6, N+1); %matrix with che last 12 values of rate of convergence for
    the starting point
58 mat_converged1_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
59 mat_violations1_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
60
61 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
62 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
63
64 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
65 mat_times1_fd2=zeros(6,N+1); % matrix with execution times
66 mat_val1_fd2=zeros(6,N+1); %matrix with minimal values found
67 mat_grad1_fd2=zeros(6,N+1); %matrix with final gradient
68 mat_iter1_fd2=zeros(6,N+1); %matrix with number of iterations
69 mat_cg_iter1_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
70 mat_bt1_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
71 mat_conv1_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
    starting point
72 mat_converged1_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
73 mat_violations1_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
74
75 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
76 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
77
78 for j =1:N+1
79     disp(['Condizione_iniziale_n.',num2str(j)])
80
81     % EXACT DERIVATIVES
82     tic;
83     [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);
84     vec_times1_ex(j)=toc;
85
86     disp(['Exact_derivatives:',num2str(j)])
87     vec_converged1_ex(j)=converged1;
88     vec_val1_ex(j)=f1;
89     vec_grad1_ex(j)=gradf_norm1;
90     vec_iter1_ex(j)=k1;
91     vec_cg_iter1_ex(j)=sum(cgiterseq1)/k1;
92     vec_bt1_ex(j)=sum(btseq1)/k1;
93     vec_violations1_ex(j)=violations1;
94     last_vals = conv_ord1_ex(max(end-11,1):end);
95     mat_conv1_ex(:, j) = last_vals;
96
97

```

```

98     for i=2:2:12
99         h=10^(-i);
100
101         % FINITE DIFFERENCES 1
102         JF=@(x) JF_fd1(x,h);
103         HF=@(x) HF_fd1(x,h);
104         tic;
105         [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);
106         mat_times1_fd1(i/2,j)=toc;
107
108         disp(['Finite differences (classical version) with h=1e-', num2str(i), ' : ', flag1])
109         mat_converged1_fd1(i/2,j)=converged1;
110         mat_val1_fd1(i/2,j)=f1;
111         mat_grad1_fd1(i/2,j)=gradf_norm1;
112         mat_iter1_fd1(i/2,j)=k1;
113         mat_cg_iter1_fd1(i/2,j)=sum(cgiterseq1)/k1;
114         mat_bt1_fd1(i/2,j)=sum(btseq1)/k1;
115         mat_violations1_fd1(i/2,j)=violations1;
116         last_vals = conv_ord1_df1(max(end-11,1):end);
117         mat_conv1_fd1(i/2, j) = {last_vals};
118
119
120         % FINITE DIFFERENCES 2
121         JF=@(x) JF_fd2(x,h);
122         HF=@(x) HF_fd2(x,h);
123         tic;
124         [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);
125         mat_times1_fd2(i/2,j)=toc;
126
127         disp(['Finite differences (new version) with h=1e-', num2str(i), ' : ', flag1])
128         mat_converged1_fd2(i/2,j)=converged1;
129         mat_val1_fd2(i/2,j)=f1;
130         mat_grad1_fd2(i/2,j)=gradf_norm1;
131         mat_iter1_fd2(i/2,j)=k1;
132         mat_cg_iter1_fd2(i/2,j)=sum(cgiterseq1)/k1;
133         mat_bt1_fd2(i/2,j)=sum(btseq1)/k1;
134         mat_violations1_fd2(i/2,j)=violations1;
135         last_vals = conv_ord1_df2(max(end-11,1):end);
136         mat_conv1_fd2(i/2, j) = {last_vals};
137
138     end
139 end
140
141
142
143 %% Plot of the last 12 values of experimentale rate of convergence
144 num_initial_points = N + 1;
145 figure;
146 hold on;
147
148 % Plot for every initial condition
149 for j = 1:num_initial_points
150     conv_ord_ex = mat_conv1_ex(:,j); %exact derivarives
151     plot(1:12, conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
152     hold on;
153     for i =1:6
154         conv_ord_fd1 = mat_conv1_fd1{i, j}; % FD1
155         conv_ord_fd2 = mat_conv1_fd2{i, j}; % FD2
156         plot(1:12, conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
157         hold on;
158         plot(1:12, conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
159         hold on;
160     end
161 end
162
163 % title and legend
164 title('F16 103 superlinear');
165 xlabel('Iterazione');
166 ylabel('Ordine di Convergenza');

```

```

167 legend({'Exact_Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
168 grid on;
169 hold off;
170
171
172 %% Execution Time
173
174 % Exact Derivative
175 vec_times_ex_clean = vec_times1_ex; %a copy of the vector
176 vec_times_ex_clean(vec_converged1_ex == 0) = NaN; %Set NaN for those that do not converge
177 avg_exact_t1 = mean(vec_times_ex_clean, 'omitnan'); %calculate the mean
178
179 % FD1
180 mat_times_fd1_clean = mat_times1_fd1; %a copy of the matrix
181 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN; %Set NaN for those that do not
    converge.
182 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); %calculate the mean
183
184 % FD2
185 mat_times_fd2_clean = mat_times1_fd2; %a copy of the matrix
186 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN; %Set NaN for those that do not
    converge.
187 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); %calculate the mean
188
189 % Creation of the labels
190 h_exponents = [2, 4, 6, 8, 10, 12];
191 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
192
193 fd1_vals = avg_fd1';
194 fd2_vals = avg_fd2';
195
196 % Table construction with exact for both the row
197 rowNames = {'FD1', 'FD2'};
198 columnNames = [h_labels, 'Exact'];
199 data = [fd1_vals, avg_exact_t1; fd2_vals, avg_exact_t1;];
200 T1 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
201
202 % visualization
203 disp('Average_computation_times_table_(only_for_successful_runs):_F16,_n=10^3,_
    superlinear');
204 disp(T1);
205
206
207 %% All the tables has the same structure
208 %% Iteration
209
210 vec_times_ex_clean = vec_iter1_ex;
211 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
212 avg_exact_i1 = mean(vec_times_ex_clean, 'omitnan');
213
214 mat_times_fd1_clean = mat_iter1_fd1;
215 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
216 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
217
218 mat_times_fd2_clean = mat_iter1_fd2;
219 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
220 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
221
222 h_exponents = [2, 4, 6, 8, 10, 12];
223 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
224
225 fd1_vals = avg_fd1';
226 fd2_vals = avg_fd2';
227
228 rowNames = {'FD1', 'FD2'};
229 columnNames = [h_labels, 'Exact'];
230 data = [fd1_vals, avg_exact_i1; fd2_vals, avg_exact_i1;];
231
232 T2 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
233
234 disp('Average_computation_iteration_table_(only_for_successful_runs):_F16,_n=10^3,_suplin
    ');
235 disp(T2);

```

```

236
237 %% F value
238
239 vec_times_ex_clean = vec_val1_ex;
240 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
241 avg_exact_f1 = mean(vec_times_ex_clean, 'omitnan');
242
243 mat_times_fd1_clean = mat_val1_fd1;
244 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
245 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
246
247 mat_times_fd2_clean = mat_val1_fd2;
248 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
249 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
250
251 h_exponents = [2, 4, 6, 8, 10, 12];
252 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
253
254 fd1_vals = avg_fd1';
255 fd2_vals = avg_fd2';
256
257 rowNames = {'FD1', 'FD2'};
258 columnNames = [h_labels, 'Exact'];
259 data = [fd1_vals, avg_exact_f1; fd2_vals, avg_exact_f1];
260
261 T3 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
262
263 disp('Average computation fmin value table (only for successful runs): F16, n=10^3,
suplin');
264 disp(T3);
265
266 %% VIOLATION
267
268 vec_times_ex_clean = vec_violations1_ex;
269 vec_times_ex_clean(vec_converged1_ex == 0) = NaN;
270 avg_exact_v1 = mean(vec_times_ex_clean, 'omitnan');
271
272 mat_times_fd1_clean = mat_violations1_fd1;
273 mat_times_fd1_clean(mat_converged1_fd1 == 0) = NaN;
274 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
275
276 mat_times_fd2_clean = mat_violations1_fd2;
277 mat_times_fd2_clean(mat_converged1_fd2 == 0) = NaN;
278 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
279
280 h_exponents = [2, 4, 6, 8, 10, 12];
281 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
282
283 %
284 fd1_vals = avg_fd1';
285 fd2_vals = avg_fd2';
286
287 rowNames = {'FD1', 'FD2'};
288 columnNames = [h_labels, 'Exact'];
289 data = [fd1_vals, avg_exact_v1; fd2_vals, avg_exact_v1];
290
291 T10 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
292
293 disp('Average computation violation table (only for successful runs): F16, n=10^3,
superlinear');
294 disp(T10);
295
296
297 %% BT-SEQ
298 vec_bt_ex_clean = vec_bt1_ex;
299 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
300 avg_exact_bt1 = mean(vec_bt_ex_clean, 'omitnan');
301
302 mat_bt_fd1_clean = mat_bt1_fd1;
303 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
304 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
305
306 mat_bt_fd2_clean = mat_bt1_fd2;

```

```

307 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
308 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
309
310 h_exponents = [2, 4, 6, 8, 10, 12];
311 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
312
313 fd1_vals = avg_fd1';
314 fd2_vals = avg_fd2';
315
316 rowNames = {'FD1', 'FD2'};
317 columnNames = [h_labels, 'Exact'];
318 data = [fd1_vals, avg_exact_bt1; fd2_vals, avg_exact_bt1;];
319
320 T11 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
321
322 disp('Average computation bt iteration table (only for successful runs): F16, n=10^3,
323      superlinear');
324 disp(T11);
325
326 %% CG-SEQ
327
328 vec_bt_ex_clean = vec_cg_iter1_ex;
329 vec_bt_ex_clean(vec_converged1_ex == 0) = NaN;
330 avg_exact_cg1 = mean(vec_bt_ex_clean, 'omitnan');
331
332 mat_bt_fd1_clean = mat_cg_iter1_fd1;
333 mat_bt_fd1_clean(mat_converged1_fd1 == 0) = NaN;
334 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
335
336 mat_bt_fd2_clean = mat_cg_iter1_fd2;
337 mat_bt_fd2_clean(mat_converged1_fd2 == 0) = NaN;
338 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
339
340 h_exponents = [2, 4, 6, 8, 10, 12];
341 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
342
343 fd1_vals = avg_fd1';
344 fd2_vals = avg_fd2';
345
346 rowNames = {'FD1', 'FD2'};
347 columnNames = [h_labels, 'Exact'];
348 data = [fd1_vals, avg_exact_cg1; fd2_vals, avg_exact_cg1;];
349
350 T12 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
351
352 disp('Average computation cg iteration table (only for successful runs): F16, n=10^3,
353      superlinear');
354 disp(T12);
355
356 %% Number of starting point converged
357
358 h_exponents = [2, 4, 6, 8, 10, 12];
359 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
360
361 fd1_vals = sum(mat_converged1_fd1, 2)';
362 fd2_vals = sum(mat_converged1_fd2, 2)';
363
364 rowNames = {'FD1', 'FD2'};
365 columnNames = [h_labels, 'Exact'];
366 data = [fd1_vals, sum(vec_converged1_ex); fd2_vals, sum(vec_converged1_ex);];
367
368 T13 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
369
370 disp('Number of converged: F16, n=10^3, superlinear');
371 disp(T13);
372
373 %save the table in a file.xlsx
374 writetable(T1, 'results_f16_suplin.xlsx', 'Sheet', 'time_3', 'WriteRowNames', true);
375 writetable(T2, 'results_f16_suplin.xlsx', 'Sheet', 'niter_3', 'WriteRowNames', true);
376 writetable(T3, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_3', 'WriteRowNames', true);
377 writetable(T10, 'results_f16_suplin.xlsx', 'Sheet', 'viol_3', 'WriteRowNames', true);
378 writetable(T11, 'results_f16_suplin.xlsx', 'Sheet', 'bt_3', 'WriteRowNames', true);
379 writetable(T12, 'results_f16_suplin.xlsx', 'Sheet', 'cg_3', 'WriteRowNames', true);
380 writetable(T13, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv3', 'WriteRowNames', true);

```

```

378
379
380
381 %% n=10^4 (1e4)
382
383 rng(345989);
384
385 n=1e4;
386
387 kmax=1.5e3; % maximum number of iterations of Newton method
388 tolgrad=1e-5; % tolerance on gradient norm %%%%%%%%%%% decide if we want to keep
    the tolerance 5e-7
389
390 cg_maxit=100; % maximum number of iterations of coniugate gradient method (for the linear
    system)
391 z0=zeros(n,1); % initial point of coniugate gradient method (for the linear system)
392
393 % Backtracking parameters
394 c1=1e-4;
395 rho=0.50;
396 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
397
398 x0 = ones(n, 1); % initial point
399 N=10; % number of initial points to be generated
400
401 % Initial points:
402 Mat_points= repmat(x0,1,N+1);
403 rand_mat=2*rand(n, N)-1;
404 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
405
406 % Structure for EXACT derivatives
407 vec_times2_ex=zeros(1,N+1); % vector with execution times
408 vec_val2_ex=zeros(1,N+1); %vector with minimal values found
409 vec_grad2_ex=zeros(1,N+1); %vector with final gradient
410 vec_iter2_ex=zeros(1,N+1); %vector with number of iterations
411 vec_cg_iter2_ex=zeros(1,N+1); %vector with mean number of inner iterations
412 vec_bt2_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
413 mat_conv2_ex=zeros(12,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
414 vec_converged2_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
415 vec_violations2_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
416
417 JF_ex = @(x) JF_gen(x,true,false,0);
418 HF_ex = @(x) HF_gen(x,true,false,0);
419
420 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
421 mat_times2_fd1=zeros(6,N+1); % matrix with execution times
422 mat_val2_fd1=zeros(6,N+1); %matrix with minimal values found
423 mat_grad2_fd1=zeros(6,N+1); %matrix with final gradient
424 mat_iter2_fd1=zeros(6,N+1); %matrix with number of iterations
425 mat_cg_iter2_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
426 mat_bt2_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
427 mat_conv2_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
428 mat_converged2_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)
429 mat_violations2_fd1=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
430
431 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
432 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
433
434 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
    x_j) as increment)
435 mat_times2_fd2=zeros(6,N+1); % matrix with execution times
436 mat_val2_fd2=zeros(6,N+1); %matrix with minimal values found
437 mat_grad2_fd2=zeros(6,N+1); %matrix with final gradient
438 mat_iter2_fd2=zeros(6,N+1); %matrix with number of iterations
439 mat_cg_iter2_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
440 mat_bt2_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
441 mat_conv2_fd2=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
442 mat_converged2_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)

```

```

443 mat_violations2_fd2=zeros(6,N+1); % matrix with number of violations of curvature
    condition in Newton method
444
445 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
446 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
447
448 for j =1:N+1
449     disp(['Condizione iniziale n.',num2str(j)])
450
451     % EXACT DERIVATIVES
452     tic;
453     [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);
454     vec_times2_ex(j)=toc;
455
456     disp(['Exact derivatives:',flag2])
457     vec_converged2_ex(j)=converged2;
458     vec_val2_ex(j)=f2;
459     vec_grad2_ex(j)=gradf_norm2;
460     vec_iter2_ex(j)=k2;
461     vec_cg_iter2_ex(j)=sum(cgiterseq2)/k2;
462     vec_bt2_ex(j)=sum(btseq2)/k2;
463     vec_violations2_ex(j)=violations2;
464     last_vals = conv_ord2_ex(max(end-11,1):end);
465     mat_conv2_ex(:, j) = last_vals;
466
467     for i=2:2:12
468         h=10^(-i);
469
470         % FINITE DIFFERENCES 1
471         JF=@(x) JF_fd1(x,h);
472         HF=@(x) HF_fd1(x,h);
473         tic;
474         [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);
475         mat_times2_fd1(i/2,j)=toc;
476
477         disp(['Finite differences (classical version) with h=1e-',num2str(i),' :',flag2])
478         mat_converged2_fd1(i/2,j)=converged2;
479         mat_val2_fd1(i/2,j)=f2;
480         mat_grad2_fd1(i/2,j)=gradf_norm2;
481         mat_iter2_fd1(i/2,j)=k2;
482         mat_cg_iter2_fd1(i/2,j)=sum(cgiterseq2)/k2;
483         mat_bt2_fd1(i/2,j)=sum(btseq2)/k2;
484         mat_violations2_fd1(i/2,j)=violations2;
485         last_vals = conv_ord2_df1(max(end-11,1):end);
486         mat_conv2_fd1(i/2, j) = {last_vals};
487
488         % FINITE DIFFERENCES 2
489         JF=@(x) JF_fd2(x,h);
490         HF=@(x) HF_fd2(x,h);
491         tic;
492         [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);
493         mat_times2_fd2(i/2,j)=toc;
494
495         disp(['Finite differences (new version) with h=1e-',num2str(i),' :',flag2])
496         mat_converged2_fd2(i/2,j)=converged2;
497         mat_val2_fd2(i/2,j)=f2;
498         mat_grad2_fd2(i/2,j)=gradf_norm2;
499         mat_iter2_fd2(i/2,j)=k2;
500         mat_cg_iter2_fd2(i/2,j)=sum(cgiterseq2)/k2;
501         mat_bt2_fd2(i/2,j)=sum(btseq2)/k2;
502         mat_violations2_fd2(i/2,j)=violations2;
503         last_vals = conv_ord2_df2(max(end-11,1):end);
504         mat_conv2_fd2(i/2, j) = {last_vals};
505
506     end
507 end
508

```



```

509
510 %% The Plot has the same structure
511 num_initial_points = N + 1;
512 figure;
513 hold on;
514
515 for j = 1:num_initial_points
516     conv_ord_ex = mat_conv2_ex(:,j);
517     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
518     hold on;
519     for i =1:6
520         conv_ord_fd1 = mat_conv2_fd1{i, j};
521         conv_ord_fd2 = mat_conv2_fd2{i, j};
522         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
523         hold on;
524         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
525         hold on;
526     end
527 end
528
529 title('F16_10^4_superlinear');
530 xlabel('Iterazione');
531 ylabel('Ordine di Convergenza');
532 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
533 grid on;
534 hold off;
535
536
537 %% Execution time
538
539 % Exact derivative
540 vec_times_ex_clean = vec_times2_ex; %a copy of the vector
541 vec_times_ex_clean(vec_converged2_ex == 0) = NaN; %Set NaN for those that do not converge
542 avg_exact_t2 = mean(vec_times_ex_clean, 'omitnan'); % computation of the mean
543
544 % FD1
545 mat_times_fd1_clean = mat_times2_fd1; % a copy of the vector
546 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN; %Set NaN for those that do not
    converge
547 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan'); % computation of the mean
548
549 % FD2
550 mat_times_fd2_clean = mat_times2_fd2; %a copy of the vector
551 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN; %Set NaN for those that do not
    converge
552 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan'); % computation of the mean
553
554 % Creation of the labels
555 h_exponents = [2, 4, 6, 8, 10, 12];
556 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
557
558 fd1_vals = avg_fd1';
559 fd2_vals = avg_fd2';
560
561 % Table creation
562 rowNames = {'FD1', 'FD2'};
563 columnNames = [h_labels, 'Exact'];
564 data = [fd1_vals, avg_exact_t2; fd2_vals, avg_exact_t2;];
565 T4 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
566 %display the table
567 disp('Average computation times table (only for successful runs): F16, n=10^4,
    superlinear');
568 disp(T4);
569
570 %% Iteration
571
572 vec_times_ex_clean = vec_iter2_ex;
573 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
574 avg_exact_i2 = mean(vec_times_ex_clean, 'omitnan');
575
576 mat_times_fd1_clean = mat_iter2_fd1;
577 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
578 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');

```

```

579 mat_times_fd2_clean = mat_iter2_fd2;
580 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
581 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
582
583
584 h_exponents = [2, 4, 6, 8, 10, 12];
585 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
586
587 fd1_vals = avg_fd1';
588 fd2_vals = avg_fd2';
589
590 rowNames = {'FD1', 'FD2'};
591 columnNames = [h_labels, 'Exact'];
592 data = [fd1_vals, avg_exact_i2; fd2_vals, avg_exact_i2;];
593
594 T5 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
595
596 disp('Average computation iteration table (only for successful runs): F16, n=10^4,
597      superlinear');
598 disp(T5);
599
600 %% Function value
601
602 vec_times_ex_clean = vec_val2_ex;
603 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
604 avg_exact_f2 = mean(vec_times_ex_clean, 'omitnan');
605
606 mat_times_fd1_clean = mat_val2_fd1;
607 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
608 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
609
610 mat_times_fd2_clean = mat_val2_fd2;
611 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
612 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
613
614 h_exponents = [2, 4, 6, 8, 10, 12];
615 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
616
617 fd1_vals = avg_fd1';
618 fd2_vals = avg_fd2';
619
620 rowNames = {'FD1', 'FD2'};
621 columnNames = [h_labels, 'Exact'];
622 data = [fd1_vals, avg_exact_f2; fd2_vals, avg_exact_f2;];
623
624 T6 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
625
626 disp('Average computation fmin value table (only for successful runs): F16, n=10^4,
627      superlinear');
628 disp(T6);
629
630 %% VIOLATION
631
632 vec_times_ex_clean = vec_violations2_ex;
633 vec_times_ex_clean(vec_converged2_ex == 0) = NaN;
634 avg_exact_v2 = mean(vec_times_ex_clean, 'omitnan');
635
636 mat_times_fd1_clean = mat_violations2_fd1;
637 mat_times_fd1_clean(mat_converged2_fd1 == 0) = NaN;
638 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
639
640 mat_times_fd2_clean = mat_violations2_fd2;
641 mat_times_fd2_clean(mat_converged2_fd2 == 0) = NaN;
642 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
643
644 h_exponents = [2, 4, 6, 8, 10, 12];
645 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
646
647 fd1_vals = avg_fd1';
648 fd2_vals = avg_fd2';
649
650 rowNames = {'FD1', 'FD2'};
651 columnNames = [h_labels, 'Exact'];

```

```

650 data = [ fd1_vals, avg_exact_v2; fd2_vals, avg_exact_v2;];
651
652 T14 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
653
654 disp('Average computation violation table (only for successful runs): F16, n=10^4,
suplinear');
655 disp(T14);
656
657 %% BT-SEQ
658
659 vec_bt_ex_clean = vec_bt2_ex;
660 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
661 avg_exact_bt2 = mean(vec_bt_ex_clean, 'omitnan');
662
663 mat_bt_fd1_clean = mat_bt2_fd1;
664 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
665 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
666
667 mat_bt_fd2_clean = mat_bt2_fd2;
668 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
669 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
670
671 h_exponents = [2, 4, 6, 8, 10, 12];
672 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
673
674 fd1_vals = avg_fd1';
675 fd2_vals = avg_fd2';
676
677 rowNames = {'FD1', 'FD2'};
678 columnNames = [ h_labels, 'Exact'];
679 data = [ fd1_vals, avg_exact_bt2; fd2_vals, avg_exact_bt2;];
680
681 T15 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
682
683 disp('Average computation bt iteration table (only for successful runs): F16, n=10^4,
superlinear');
684 disp(T15);
685
686 %% CG-SEQ
687
688 vec_bt_ex_clean = vec_cg_iter2_ex;
689 vec_bt_ex_clean(vec_converged2_ex == 0) = NaN;
690 avg_exact_cg2 = mean(vec_bt_ex_clean, 'omitnan');
691
692 mat_bt_fd1_clean = mat_cg_iter2_fd1;
693 mat_bt_fd1_clean(mat_converged2_fd1 == 0) = NaN;
694 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
695
696 mat_bt_fd2_clean = mat_cg_iter2_fd2;
697 mat_bt_fd2_clean(mat_converged2_fd2 == 0) = NaN;
698 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
699
700 h_exponents = [2, 4, 6, 8, 10, 12];
701 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
702
703 fd1_vals = avg_fd1';
704 fd2_vals = avg_fd2';
705
706 rowNames = {'FD1', 'FD2'};
707 columnNames = [ h_labels, 'Exact'];
708 data = [ fd1_vals, avg_exact_cg2; fd2_vals, avg_exact_cg2;];
709
710 T16 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
711
712 disp('Average computation cg iteration table (only for successful runs): F16, n=10^4,
superlinear');
713 disp(T16);
714
715 %% Number of initial point converged
716
717 h_exponents = [2, 4, 6, 8, 10, 12];
718 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
719

```

```

720 fd1_vals = sum(mat_converged2_fd1,2)';
721 fd2_vals = sum(mat_converged2_fd2,2)';
722
723 rowNames = {'FD1', 'FD2'};
724 columnNames = [ h_labels, 'Exact'];
725 data = [ fd1_vals, sum(vec_converged2_ex); fd2_vals, sum(vec_converged2_ex)];
726
727 T17 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
728
729 disp('Number of converged F16, n=10^4, superlinear');
730 disp(T17);
731 %save the table in a file.xlsx
732 writetable(T4, 'results_f16_suplin.xlsx', 'Sheet', 'time_4','WriteRowNames', true);
733 writetable(T5, 'results_f16_suplin.xlsx', 'Sheet', 'niter_4','WriteRowNames', true);
734 writetable(T6, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_4','WriteRowNames', true);
735 writetable(T14, 'results_f16_suplin.xlsx', 'Sheet', 'viol_4','WriteRowNames', true);
736 writetable(T15, 'results_f16_suplin.xlsx', 'Sheet', 'bt_4','WriteRowNames', true);
737 writetable(T16, 'results_f16_suplin.xlsx', 'Sheet', 'cg_4','WriteRowNames', true);
738 writetable(T17, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv4','WriteRowNames', true);
739
740
741 %% n=10^5 (1e5)
742
743 rng(345989);
744
745 n=1e5;
746
747 kmax=1.5e3; % maximum number of iterations of Newton method
748 tolgrad=5e-4; % tolerance on gradient norm
749
750 cg_maxit=100; % maximum number of iterations of conjugate gradient method (for the linear
    system)
751 z0=zeros(n,1); % initial point of conjugate gradient method (for the linear system)
752
753 % Backtracking parameters
754 c1=1e-4;
755 rho=0.50;
756 btmax=50; % compatible with rho (with alpha0=1 you get min_step 8.8e-16)
757
758 x0 = ones(n, 1); % initial point
759 N=10; % number of initial points to be generated
760
761 % Initial points:
762 Mat_points=repmat(x0,1,N+1);
763 rand_mat=2*rand(n, N)-1;
764 Mat_points(:,2:end)=Mat_points(:,2:end) + rand_mat; % matrix with columns=initial points
765
766 % Structure for EXACT derivatives
767 vec_times3_ex=zeros(1,N+1); % vector with execution times
768 vec_val3_ex=zeros(1,N+1); %vector with minimal values found
769 vec_grad3_ex=zeros(1,N+1); %vector with final gradient
770 vec_iter3_ex=zeros(1,N+1); %vector with number of iterations
771 vec_cg_iter3_ex=zeros(1,N+1); %vector with mean number of inner iterations
772 vec_bt3_ex=zeros(1,N+1); %vector with mean number of backtracking iterations
773 mat_conv3_ex=zeros(12:N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
774 vec_converged3_ex=zeros(1,N+1); % vector of booleans (true if it has converged)
775 vec_violations3_ex=zeros(1,N+1); % vector with number of violations of curvature
    condition in Newton method
776
777 JF_ex = @(x) JF_gen(x,true,false,0);
778 HF_ex = @(x) HF_gen(x,true,false,0);
779
780 % Structure for derivatives approximated with FINITE DIFFERENCES (classical version)
781 mat_times3_fd1=zeros(6,N+1); % matrix with execution times
782 mat_val3_fd1=zeros(6,N+1); %matrix with minimal values found
783 mat_grad3_fd1=zeros(6,N+1); %matrix with final gradient
784 mat_iter3_fd1=zeros(6,N+1); %matrix with number of iterations
785 mat_cg_iter3_fd1=zeros(6,N+1); %matrix with mean number of inner iterations
786 mat_bt3_fd1=zeros(6,N+1); %matrix with mean number of backtracking iterations
787 mat_conv3_fd1=cell(6,N+1);%matrix with che last 12 values of rate of convergence for the
    starting point
788 mat_converged3_fd1=zeros(6,N+1); % matrix of booleans (true if it has converged)

```

```

789 mat_violations3_fd1=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
790
791 JF_fd1 = @(x,h) JF_gen(x,false,false,h);
792 HF_fd1 = @(x,h) HF_gen(x,false,false,h);
793
794 % Structure for derivatives approximated with FINITE DIFFERENCES (version with h=h*abs(
      x_j) as increment)
795 mat_times3_fd2=zeros(6,N+1); % matrix with execution times
796 mat_val3_fd2=zeros(6,N+1); %matrix with minimal values found
797 mat_grad3_fd2=zeros(6,N+1); %matrix with final gradient
798 mat_iter3_fd2=zeros(6,N+1); %matrix with number of iterations
799 mat_cg_iter3_fd2=zeros(6,N+1); %matrix with mean number of inner iterations
800 mat_bt3_fd2=zeros(6,N+1); %matrix with mean number of backtracking iterations
801 mat_conv3_fd2=cell(6,N+1); %matrix with che last 12 values of rate of convergence for the
      starting point
802 mat_converged3_fd2=zeros(6,N+1); % matrix of booleans (true if it has converged)
803 mat_violations3_fd2=zeros(6,N+1); % matrix with number of violations of curvature
      condition in Newton method
804
805 JF_fd2 = @(x,h) JF_gen(x,false,true,h);
806 HF_fd2 = @(x,h) HF_gen(x,false,true,h);
807
808 for j =1:N+1
809     disp(['Condizione_iniziale_n.',num2str(j)])
810
811     % EXACT DERIVATIVES
812     tic;
813     [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);
814     vec_times3_ex(j)=toc;
815
816     disp(['Exact_derivatives:',flag3])
817     vec_converged3_ex(j)=converged3;
818     vec_val3_ex(j)=f3;
819     vec_grad3_ex(j)=gradf_norm3;
820     vec_iter3_ex(j)=k3;
821     vec_cg_iter3_ex(j)=sum(cgiterseq3)/k3;
822     vec_bt3_ex(j)=sum(btseq3)/k3;
823     vec_violations3_ex(j)=violations3;
824     last_vals = conv_ord3_ex(max(end-11,1):end);
825     mat_conv3_ex(:, j) = last_vals;
826
827     for i=2:2:12
828         h=10^(-i);
829
830         % FINITE DIFFERENCES 1
831         JF=@(x) JF_fd1(x,h);
832         HF=@(x) HF_fd1(x,h);
833         tic;
834         [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);
835         mat_times3_fd1(i/2,j)=toc;
836
837         disp(['Finite_differences_(classical_version)_with_h=1e-',num2str(i),'_',flag3])
838         mat_converged3_fd1(i/2,j)=converged3;
839         mat_val3_fd1(i/2,j)=f3;
840         mat_grad3_fd1(i/2,j)=gradf_norm3;
841         mat_iter3_fd1(i/2,j)=k3;
842         mat_cg_iter3_fd1(i/2,j)=sum(cgiterseq3)/k3;
843         mat_bt3_fd1(i/2,j)=sum(btseq3)/k3;
844         mat_violations3_fd1(i/2,j)=violations3;
845         last_vals = conv_ord3_df1(max(end-11,1):end);
846         mat_conv3_fd1(i/2, j) = {last_vals};
847
848
849         % FINITE DIFFERENCES 2
850         JF=@(x) JF_fd2(x,h);
851         HF=@(x) HF_fd2(x,h);
852         tic;
853         [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);
854 mat_times3_fd2(i/2,j)=toc;
855
856 disp(['Finite differences (new version) with h=1e-', num2str(i), ' : ', flag3])
857 mat_converged3_fd2(i/2,j)=converged3;
858 mat_val3_fd2(i/2,j)=f3;
859 mat_grad3_fd2(i/2,j)=gradf_norm3;
860 mat_iter3_fd2(i/2,j)=k3;
861 mat_cg_iter3_fd2(i/2,j)=sum(cgiterseq3)/k3;
862 mat_bt3_fd2(i/2,j)=sum(btseq3)/k3;
863 mat_violations3_fd2(i/2,j)=violations3;
864 last_vals = conv_ord3_df2(max(end-11,1):end);
865 mat_conv3_fd2(i/2, j) = {last_vals};
866
867
868 end
869 end
870
871
872 %% The plot has the same structure as n=10^3
873 num_initial_points = N + 1;
874 figure;
875 hold on;
876
877 for j = 1:num_initial_points
878     conv_ord_ex = mat_conv3_ex(:,j);
879     plot(1:12,conv_ord_ex, 'Color', 'b', 'LineWidth', 1.5);
880     hold on;
881     for i =1:6
882         conv_ord_fd1 = mat_conv3_fd1{i, j};
883         conv_ord_fd2 = mat_conv3_fd2{i, j};
884         plot(1:12,conv_ord_fd1, '-', 'Color', 'r', 'LineWidth', 1.5);
885         hold on;
886         plot(1:12,conv_ord_fd2, '-o', 'Color', 'g', 'LineWidth', 1.5);
887         hold on;
888     end
889 end
890
891 title('F79_10^5_superlinear');
892 xlabel('Iterazione');
893 ylabel('Ordine di Convergenza');
894 legend({'Exact Derivatives', 'dif_fin_1', 'dif_fin_2'}, 'Location', 'Best');
895 grid on;
896 hold off;
897
898 %% Time
899
900 vec_times_ex_clean = vec_times3_ex;
901 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
902 avg_exact_t3 = mean(vec_times_ex_clean, 'omitnan');
903
904 mat_times_fd1_clean = mat_times3_fd1;
905 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
906 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
907
908 mat_times_fd2_clean = mat_times3_fd2;
909 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
910 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
911
912 h_exponents = [2, 4, 6, 8, 10, 12];
913 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
914
915 fd1_vals = avg_fd1';
916 fd2_vals = avg_fd2';
917
918 rowNames = {'FD1', 'FD2'};
919 columnNames = [ h_labels, 'Exact'];
920 data = [ fd1_vals, avg_exact_t3; fd2_vals, avg_exact_t3];
921
922 T7 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
923
924 disp('Average computation times table (only for successful runs): F79, n=10^5,

```

```

    superlinear');
925 disp(T7);
926
927 %% Iteration
928
929 vec_times_ex_clean = vec_iter3_ex;
930 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
931 avg_exact_i3 = mean(vec_times_ex_clean, 'omitnan');
932
933 mat_times_fd1_clean = mat_iter3_fd1;
934 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
935 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
936
937 mat_times_fd2_clean = mat_iter3_fd2;
938 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
939 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
940
941 h_exponents = [2, 4, 6, 8, 10, 12];
942 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
943
944 fd1_vals = avg_fd1';
945 fd2_vals = avg_fd2';
946
947 rowNames = {'FD1', 'FD2'};
948 columnNames = [h_labels, 'Exact'];
949 data = [fd1_vals, avg_exact_i3; fd2_vals, avg_exact_i3];
950
951 T8 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
952
953 disp('Average computation iteration table (only for successful runs): F79, n=10^5,
    superlinear');
954 disp(T8);
955
956 %% function value
957
958 vec_times_ex_clean = vec_val3_ex;
959 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
960 avg_exact_f3 = mean(vec_times_ex_clean, 'omitnan');
961
962 mat_times_fd1_clean = mat_val3_fd1;
963 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
964 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
965
966 mat_times_fd2_clean = mat_val3_fd2;
967 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
968 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
969
970 h_exponents = [2, 4, 6, 8, 10, 12];
971 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
972
973 fd1_vals = avg_fd1';
974 fd2_vals = avg_fd2';
975
976 rowNames = {'FD1', 'FD2'};
977 columnNames = [h_labels, 'Exact'];
978 data = [fd1_vals, avg_exact_f3; fd2_vals, avg_exact_f3];
979
980 T9 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
981
982 disp('Average computation fmin value table (only for successful runs): F79, n=10^5,
    superlinear');
983 disp(T9);
984
985 %% VIOLATION
986
987 vec_times_ex_clean = vec_violations3_ex;
988 vec_times_ex_clean(vec_converged3_ex == 0) = NaN;
989 avg_exact_v3 = mean(vec_times_ex_clean, 'omitnan');
990
991 mat_times_fd1_clean = mat_violations3_fd1;
992 mat_times_fd1_clean(mat_converged3_fd1 == 0) = NaN;
993 avg_fd1 = mean(mat_times_fd1_clean, 2, 'omitnan');
994

```

```

995 mat_times_fd2_clean = mat_violations3_fd2;
996 mat_times_fd2_clean(mat_converged3_fd2 == 0) = NaN;
997 avg_fd2 = mean(mat_times_fd2_clean, 2, 'omitnan');
998
999 h_exponents = [2, 4, 6, 8, 10, 12];
1000 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1001
1002 fd1_vals = avg_fd1';
1003 fd2_vals = avg_fd2';
1004
1005 rowNames = {'FD1', 'FD2'};
1006 columnNames = [h_labels, 'Exact'];
1007 data = [fd1_vals, avg_exact_v3; fd2_vals, avg_exact_v3];
1008
1009 T18 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1010
1011 disp('Average computation violation table (only for successful runs): F79, n=10^5,
1012      superlinear');
1012 disp(T18);
1013
1014 %% BT-SEQ
1015
1016 vec_bt_ex_clean = vec_bt3_ex;
1017 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1018 avg_exact_bt3 = mean(vec_bt_ex_clean, 'omitnan');
1019
1020 mat_bt_fd1_clean = mat_bt3_fd1;
1021 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1022 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1023
1024 mat_bt_fd2_clean = mat_bt3_fd2;
1025 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1026 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1027
1028 h_exponents = [2, 4, 6, 8, 10, 12];
1029 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1030
1031 fd1_vals = avg_fd1';
1032 fd2_vals = avg_fd2';
1033
1034 rowNames = {'FD1', 'FD2'};
1035 columnNames = [h_labels, 'Exact'];
1036 data = [fd1_vals, avg_exact_bt3; fd2_vals, avg_exact_bt3];
1037
1038 T19 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1039
1040 disp('Average computation bt iteration table (only for successful runs): F79, n=10^5,
1041      superlinear');
1041 disp(T19);
1042
1043 %% CG-SEQ
1044
1045 vec_bt_ex_clean = vec_cg_iter3_ex;
1046 vec_bt_ex_clean(vec_converged3_ex == 0) = NaN;
1047 avg_exact_cg3 = mean(vec_bt_ex_clean, 'omitnan');
1048
1049 mat_bt_fd1_clean = mat_cg_iter3_fd1;
1050 mat_bt_fd1_clean(mat_converged3_fd1 == 0) = NaN;
1051 avg_fd1 = mean(mat_bt_fd1_clean, 2, 'omitnan');
1052
1053 mat_bt_fd2_clean = mat_cg_iter3_fd2;
1054 mat_bt_fd2_clean(mat_converged3_fd2 == 0) = NaN;
1055 avg_fd2 = mean(mat_bt_fd2_clean, 2, 'omitnan');
1056
1057 h_exponents = [2, 4, 6, 8, 10, 12];
1058 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1059
1060 fd1_vals = avg_fd1';
1061 fd2_vals = avg_fd2';
1062
1063 rowNames = {'FD1', 'FD2'};
1064 columnNames = [h_labels, 'Exact'];
1065 data = [fd1_vals, avg_exact_cg3; fd2_vals, avg_exact_cg3];

```



```

1066 T20 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1067
1068 disp('Average computation cg iteration table (only for successful runs): F79, n=10^5,
1069      superlinear');
1070 disp(T20);
1071
1072 %% Number of initial condition converged
1073
1074 h_exponents = [2, 4, 6, 8, 10, 12];
1075 h_labels = arrayfun(@(e) sprintf('h=1e-%d', e), h_exponents, 'UniformOutput', false);
1076
1077 fd1_vals = sum(mat_converged3_fd1,2)';
1078 fd2_vals = sum(mat_converged3_fd2,2)';
1079
1080 rowNames = {'FD1', 'FD2'};
1081 columnNames = [h_labels, 'Exact'];
1082 data = [fd1_vals, sum(vec_converged3_ex); fd2_vals, sum(vec_converged3_ex)];
1083
1084 T21 = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1085
1086 disp('Number of converged: F79, n=10^5, superlinear');
1087 disp(T21);
1088 %save the tables
1089 writetable(T7, 'results_f16_suplin.xlsx', 'Sheet', 'time_5', 'WriteRowNames', true);
1090 writetable(T8, 'results_f16_suplin.xlsx', 'Sheet', 'niter_5', 'WriteRowNames', true);
1091 writetable(T9, 'results_f16_suplin.xlsx', 'Sheet', 'f_val_5', 'WriteRowNames', true);
1092 writetable(T18, 'results_f16_suplin.xlsx', 'Sheet', 'viol_5', 'WriteRowNames', true);
1093 writetable(T19, 'results_f16_suplin.xlsx', 'Sheet', 'bt_5', 'WriteRowNames', true);
1094 writetable(T20, 'results_f16_suplin.xlsx', 'Sheet', 'cg_5', 'WriteRowNames', true);
1095 writetable(T21, 'results_f16_suplin.xlsx', 'Sheet', 'n_conv5', 'WriteRowNames', true);
1096
1097
1098
1099 %% table with the results of the exact derivatives
1100 data = [avg_exact_t1, avg_exact_t2, avg_exact_t3;
1101        avg_exact_i1, avg_exact_i2, avg_exact_i3;
1102        avg_exact_f1, avg_exact_f2, avg_exact_f3;
1103        avg_exact_v1, avg_exact_v2, avg_exact_v3;
1104        avg_exact_bt1, avg_exact_bt2, avg_exact_bt3;
1105        avg_exact_cg1, avg_exact_cg2, avg_exact_cg3;
1106        sum(vec_converged1_ex), sum(vec_converged2_ex), sum(vec_converged3_ex)];
1107
1108 rowNames = {'Average Time', 'Average Iter', 'Average fval', 'Violation', 'Average iter Bt',
1109            'Average iter cg', 'N converged'};
1110 columnNames = {'n=10^3', 'n=10^4', 'n=10^5'};
1111
1112 T_compare = array2table(data, 'VariableNames', columnNames, 'RowNames', rowNames);
1113 disp(T_compare)
1114
1115 writetable(T_compare, 'results_f16_suplin.xlsx', 'Sheet', 'ExactComparison', '
1116      WriteRowNames', true);

```