NUMERICAL OPTIMIZATION FOR LARGE SCALE PROBLEMS AND STOCHASTIC OPTIMIZATION PROBLEM
Uncostrained Optimization

Eleonora Carletti e Gianluca Guzzetta

A.Y. 2021/2022

1 Introduction to the project

The goal of the project is implementing several optimization methods for unconstrained optimization and testing them on the Rosenbrock function and other test problems. The proposed optimization methods were: Nonlinear conjugate gradient method, Newton method, Newton method with finite differences, Inexact Newton method, Nelder-Mead, Steepest descent method. All methods will be applied with **backtracking**.

In this analysis the following methods will be considered and compared:

- Newton method
- Steepest descent method

At the beginning the two methods will be applied to the Rosenbrock function starting from two different points and result will be compared. Finally, code will be tested on three different test problems:

- "Chained Rosenbrock" function (Problem 1)
- "Extended Rosenbrock" function (Problem 25)

- Problem 76

Results will be compared in terms of: **dimensionality, optimization method, number of iterations, different starting points.** For all problems and functions the following parameters will be used:

- $\rho = 0.5$,
- $c = 10^-4$
- $tollgrad = 10^-6$
- $\alpha_0 = 1$

2 Steepest Descent Method

Steepest descent is an iterative method for finding the minimum of a quadratic function J consisting in moving step by step along the steepest descent direction (or residual) $-\nabla J(x_k)$ starting from a given point $x_0 \in \mathbb{R}^n$. The method can be synthesized by the following steps: At each step, until you find the minimum, or the modulus of the residual is smaller than a given tolerance:

1. Compute the residual $-\nabla J(x_k)$
2. Compute α_k (step length)
3. Compute the new step $x_{k+1} = x_k - \alpha_k \nabla J(x_k)$

3 Newton Method

The Newton method is an algorithm that locally approximates a given function (twice differentiable and sufficiently smooth) with a quadratic model. Given a point x_k belonging to a function $f(x)$ the Newton method performs the following approximation:

$$f(x+p) \simeq \nabla m_k(p)$$

$$\nabla m_k(p) := f(x_k) + f'(x_k)p + \frac{1}{2}p^T \nabla^2(x_k)p$$

And if $\nabla^2 f(x_k)$ is positive definite the approximated model $\nabla m_k(p)$ is a convex model for the function f around x_k. Another task of the Newton method consists in minimizing at every step the local model $\nabla m_k(p)$. If the given function is quadratic Newton method coincides with gradient method. As pro's Newton method provides a fast (quadratic) local range convergence as con's instead the method requires the Hessian to be symmetric positive definite to be a descent direction. Moreover, it requires a huge computation cost for computing the Hessian. To deal with the last described problem in this analysis the Hessian has been computed exploiting a sparsity pattern. This also helped in reducing computation time.

4 Backtracking

For both Steepest Descent and Newton method Backtracking strategy has been exploited. Backtracking is a method for iteratively find the best possible step length α_k based on the satisfaction of the Armijo condition.
The Armijo condition has the following expression:

$$f(x_k + \alpha * pk) \leq f(x_k) + c1 * \alpha * \nabla f(x_k)^T * p_k$$

The strategy can be summarized by the following steps:

1. Choose an initial step length $\alpha_k^{(0)}$ (in this case always equal to **1** to best fit Newton method.
2. For $j \geq 0$ if Armijo condition is satisfied accept $\alpha_k^{(j)}$ otherwise set $\alpha_k^{(j+1)} = \rho * \alpha_k^{(j)}$ with a given $\rho \in [\rho_L, \rho_U]$ with $\rho_U < 1$.

In this case also a value to limit the number of backtracking iterations equal to 50 has been introduced.

5 Application to Rosenbrock Function

Rosenbrock function is a standard test function frequently used in optimization tasks. It can be described by the following expression:

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

and in this analysis the chosen optimization methods will be applied to the following starting points:

- $x_0 = (1.2, 1.2)$
- $x_0 = (-1.2, 1)$

Results can be found in [Tab 1] in which it is possible to notice that, as expected, Newton method behaves very well compared to Steepest Descent in minimizing the given function. Better results can be seen in terms of total execution time, number of total iterations and number of backtracking iterations.

x0	iterations	f(k)	gradfk_norm	Exec. Time (s)	Bcktrck iterations range	Steepest Descent	Newton
[1.2, 1.2]	13011	6.1844e-13	9.9876e-07	37	0-10	Yes	No
[-1.2, 1]	13756	6.12e-13	9.8334e-07	39	8-9	Yes	No
[1.2, 1.2]	8	1.0927e-25	1.4393e-11	0.43	0-1	No	Yes
[-1.2, 1]	21	3.7419e-21	4.4742e-10	0.68	0-3	No	Yes

Table 1: Results for Rosenbrock Function after Steepest Descent and Newton method application

6 Problem 1: Chained Rosenbrock Function

The Chained Rosenbrock function is a function defined as follows:

$$F(x) = \sum_{i=2}^{n} [100(x_{i-1}^2 - x_i)^2 + (x_{i-1} - 1)^2]$$

$$\bar{x}_i = -1.2, \quad mod(i,2) = 1, \quad \bar{x}_i = 1.0, \quad mod(i,2) = 0$$

At the beginning Steepest Descent method has been applied and then Newton method has been exploited. For the proposed starting point it is possible to notice that both Steepest Descent and Newton method have problems in convergence. Regarding Steepest Descent the method converges for low dimensionalities (10^2) but for higher one it's application seems to be unfeasible for both high computation time and extremely huge number of iteration. For this reason a different starting point:

$$\bar{x}_i = 0$$

has been chosen and results of the methods application on it has been compared. In particular from [Table 2] it is possible to notice that the method takes a huge number of iterations to reach convergence and that number does not depend on the dimensionality.

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10^3	21585	5.6003e-13	9.9922e-07	43.2
10^4	21585	5.6003e-13	9.9922e-07	231.8

Table 2: Steepest Descent results for Chained Rosenbrock with new starting point $\bar{x}_i = 0$

Moreover the more the dimensionality increase the more the execution time increases. Regarding inner iterations instead from [Fig 1] it is possible to find that at each iteration the model performs 9 or 10 backtracking iterations.

Unfortunately it is unfeasible to increase more the dimensionality due to machine memory constraints.

By applying Newton method instead it is possible to notice that the more the dimensionality increase the more the execution time and the number of iterations increase. The above mentioned results can be found in [Tab 3]

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10^3	1472	2.097e-16	5.0204e-07	2.15
10^4	14662	7.308e-16	7.862e-07	111.07

Table 3: Newton Method results for Chained Rosenbrock with new starting point $\bar{x}_i = 0$

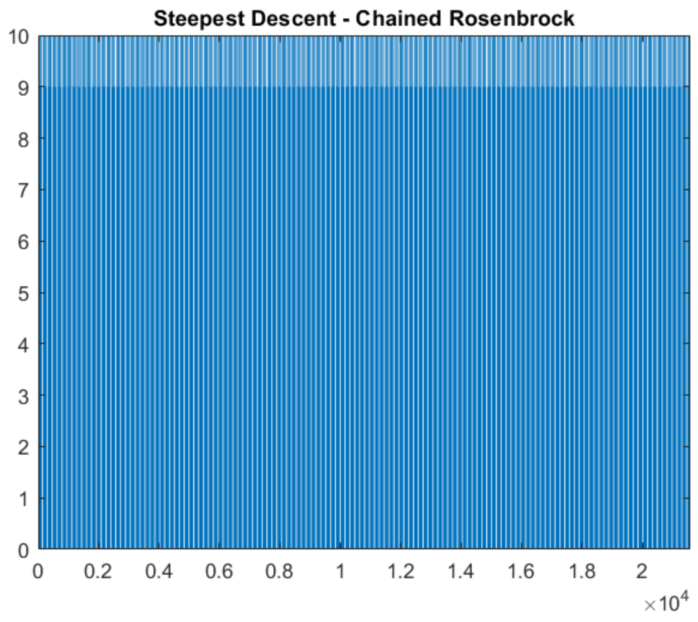

Figure 1: Backtracking iteration of Steepest Descent on Chained Rosenbrock

Moreover it is possible to notice that for the same task the above mentioned method takes on average less iterations and execution time compared to Steepest Descent. An explanation can be due to backtracking iteration number performed by the model. From [Fig 2] it is possible to notice that the algorithm only takes between 0 and 1 inner iteration to find alpha.

Finally, since the real minimum of the function seems to be $f(x) = 0$, Newton method seems to arrive closer to that (of one order of magnitude) than Steepest Descent. For this reason **Newton method seems to be the most suitable optimization method for solving this problem**.

7 Problem 25: Extended Rosenbrock Function

Extended Rosenbrock function can be described instead as:

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

$$f_k(x) = 10(x_k^2 - x_{k+1})$$

$$f_k(x) = x_{k-1} - 1$$

$$\bar{x}_i = -1.2, \quad mod(i,2) = 1, \quad \bar{x}_i = 1.0, \quad mod(i,2) = 0$$

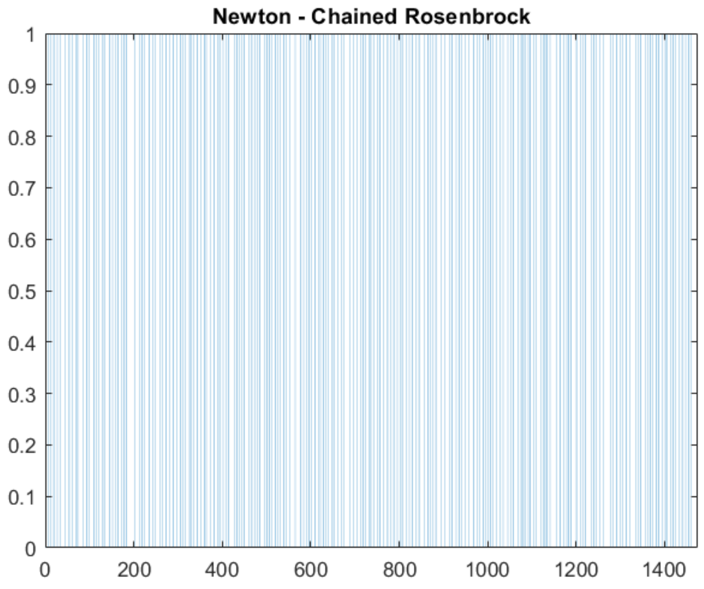

Figure 2: Backtracking iteration of Newton Method on Chained Rosenbrock

By applying Steepest Descent to the function it is possible to notice that it takes a lot of iterations to reach convergence. Moreover from [Table 4] it's clear that computation time increases fast on increasing of the dimensionality. Regarding

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10ˆ3	16747	1.1953e-12	9.9373e-07	34
10ˆ4	18167	1.1942e-12	9.924e-07	409

Table 4: Steepest Descent results for Extended Rosenbrock

inner iterations [Fig 3] it is possible to notice that the method performs around 7/8 backtracking iterations per outer iteration.

On the other hand, when Newton Method is applied it can be seen that iteration number abruptly decrease and that remains constant on dimensionality increase. Regarding execution time instead in [Table 5] it is possible to notice that it decreases with respect to the one needed for Steepest Descent and that moreover it doesn't increase so much while increasing dimensionality.

This result can be seen also by looking at [Fig 4] in which it is possible to notice that backtracking iterations are not that much in number (from 0 to 3) and that are exponentially decreasing at each new outer iteration.

Also in this case it is possible to conclude that **Newton Method seems to be the most suitable algorithm** for solving the given problem.

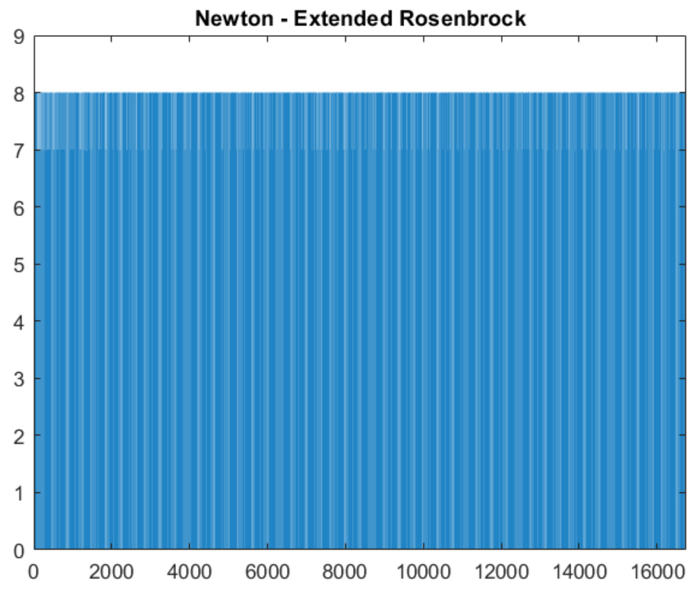

Figure 3: Backtracking iteration of Steepest Descent Extended Ronsenbrock

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10ˆ3	21	9.367e-19	5.0014e-09	0.15
10ˆ4	21	9.367e-18	1.5816e-08	0.25

Table 5: Newton Method results for Extended Rosenbrock

8 Problem 76

Problem 76 is a characterized by the following expression:

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

$$f_k(x) = x_k - \frac{x_{k+1}^2}{10} \quad 1 \leq k < n$$

$$f_k(x) = x_k - \frac{x_1^2}{10} \quad k = n$$

$$\bar{x}_l = 2, \quad l \geq 1$$

By applying Steepest Descent algorithm [Tab 6] it is possible to notice that the method is able to converge in an extremely low number of iterations even for very high values of the dimensionalities.

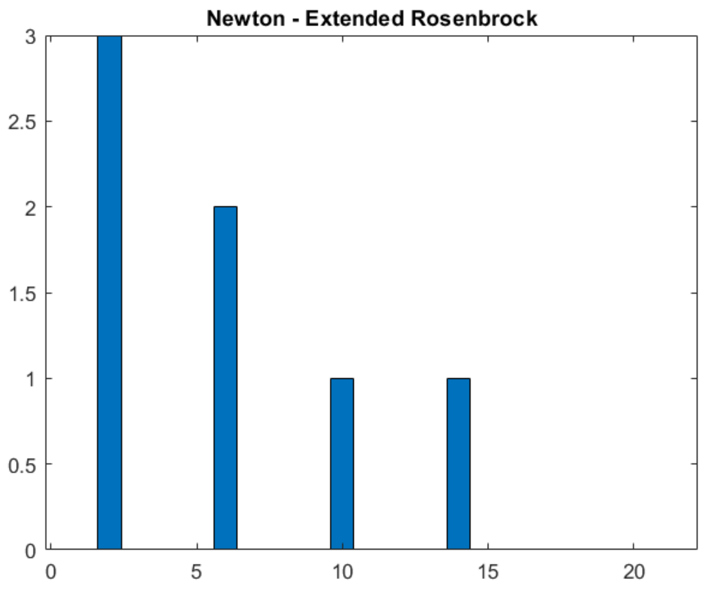

Figure 4: Backtracking iteration of Newton Method Extended Ronsenbrock

Even the execution time is extremely fast even if in general seems to be increasing with problem dimensionality.

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10ˆ3	5	9.6516e-14	4.3936e-07	0.10
10ˆ4	6	1.6768e-29	5.791e-15	0.16
10ˆ5	6	1.6768e-29	5.791e-15	0.15
10ˆ6	6	1.6768e-27	5.791e-14	3.87
10ˆ7	6	1.6768e-26	1.8313e-13	65.5

Table 6: Steepest Descent results for Problem 76

Moreover execution time is fastened by the fact that the model doesn't perform any backtracking iteration at every execution cycle as can be seen in [Fig 5].

After the application of Newton method [Tab 7] instead it is possible to notice that also in this case the iteration number is very low and similar to those of Steepest Descent. Also in this case the method seems to perform very well. Regarding latency time instead it is possible to notice that Newton method is in general slower than Steepest Descent.

This is mainly due to two reasons: the first is that in this case also the Hessian Matrix must be calculated and this operation requires time. The second

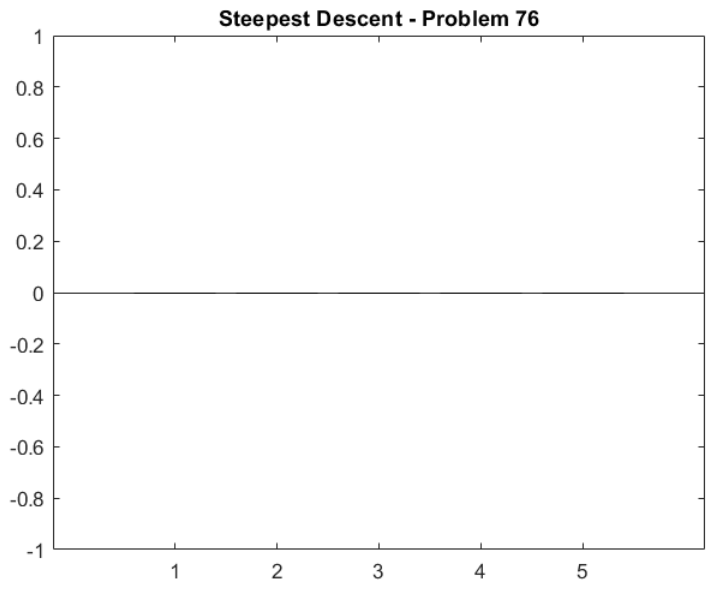

Figure 5: Backtracking iteration of Steepest Descent on problem 76

Dimensionality	Number of iterations	f(xk)	gradfk_norm	Execution time (s)
10ˆ3	6	1.0825e-16	1.4714e-08	0.63
10ˆ4	5	1.0825e-15	4.6529e-08	0.41
10ˆ5	5	1.0825e-14	1.4714e-07	5.1
10ˆ6	5	1.0825e-13	4.6529e-07	41.6
10ˆ7	6	2.1092e-32	2.0539e-16	115.2

Table 7: Newton method results for Problem 76

is that, as it is possible to notice in [Fig 6], at iteration 1 the problem performs 3 backtracking iterations that end up in slowing the process a little bit. In this case, even if both methods are valuable **Steepest Descent is preferred to Newton method** due to faster execution time.

9 Conclusion

In conclusion it is possible to say that for the first two problems Newton method seems to be the most suitable method to perform the optimization task. For the last problems instead, even if both methods perform well Steepest Descent is preferred due to lowest execution latency. Moreover it is possible to say that in general backtracking iterations in any case consistently slow down the

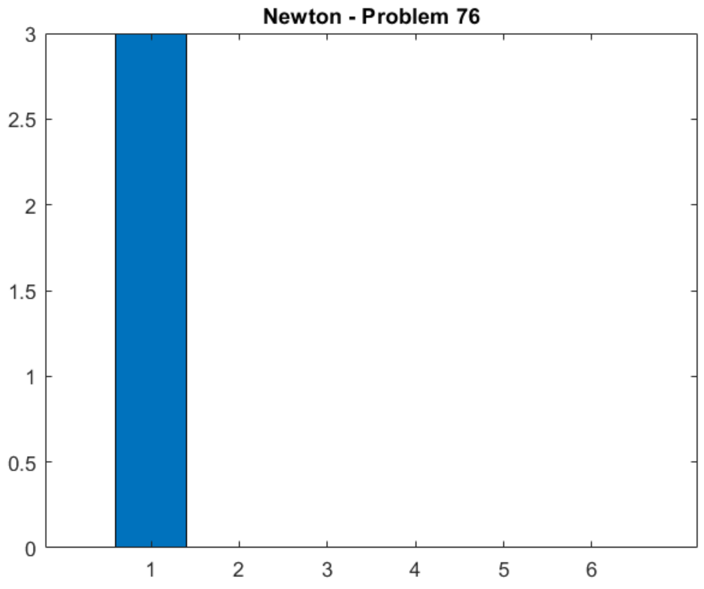

Figure 6: Backtracking iteration of Newton Method on problem 76

computation time and that performing calculation of the Hessian, even if seems to be the most time and memory consuming task becomes slim and fast if handled with the right sparsity pattern.