LIÈGE université
Sciences Appliquées

MSc. in Mechanical Engineering

MECA0027-1 | Structural & Multidisciplinary Optimization

Computer Work 1
Unconstrained optimization

Professors
Patricia Tossings
Michael Bruyneel

Authors
Lilian Steimetz (s224403)
Tom Beauve (s224331)

Academic Year 2025-2026

Contents

1 Introduction

In this report, we will analyze and assess the performance of different optimization and line search algorithms, applied both to a strictly convex quadratic function f_1 (SCQF) and to a general C_1 function f_2 (f_3 in the statement).
The optimization methods analyzed are the following:

- Steepest descent method;
- Conjugate gradients (CG) method with Fletcher-Reeves update rule;
- BFGS Quasi-Newton method.

The line search algorithms analyzed are the following :

- Dichotomy method;
- Secant method.

When comparing the optimization methods, the performance of these methods will be separated as much as possible from the performance of the line search algorithms.
The studied functions f_1 and f_2 (f_3 in the project statement) are :

$$f_1(x, y) = 4x^2 + 5xy + 3y^2 + 4x + -3y + 5 \tag{1.1a}$$

$$f_2(x, y) = 0.1(x^2 + y^2) - 0.3\cos(x)y - 3\sin(y) - 0.1xy \tag{1.1b}$$

2 Analysis of the different methods

2.1 Methods comparison

In this section, we will compare the different methods based on the number of iterations until convergence, the number of evaluation of the function, its gradient and hessian and examine their behavior. In order to isolate the methods performance from all other variables (tolerances, line search procedures, initial points), we will keep all other parameters fixed[1]. Initial points were chosen defavorable on purpose to show methods characteristics. The results of the analysis are shown in Table A, the associated function values at the iterates are shown in the Figure 1, and their trajectories are in figures 2,3, 4 and 5.

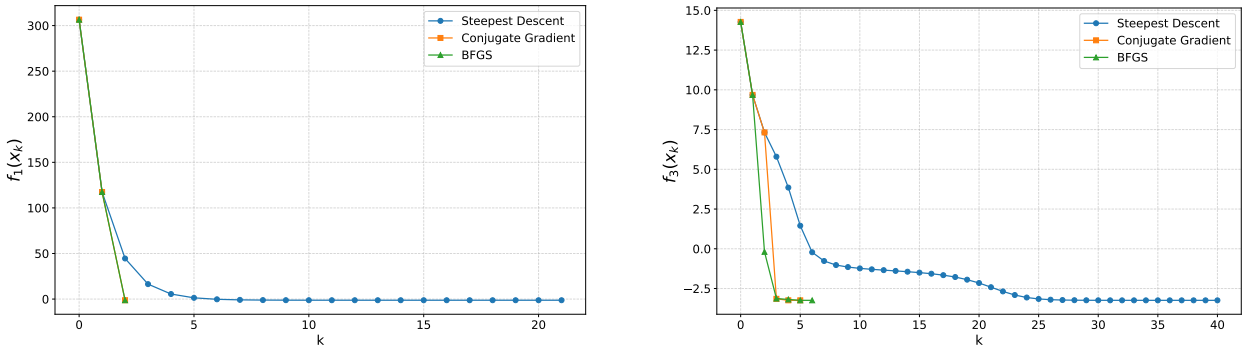
Steepest descent method : This method is the slowest, most computationally intensive among the three.
From the definition of the method itself, the search directions are orthogonal one to another, therefore looping periodically (with a period equal to the dimension n of the problem) over them.
As a result, the optimization path zigzags a lot before convergence – see Figure 2–, with smaller and

[1]Initial points : $x_{0,1} = (10.5, -5.5)$ for the analysis of $f_1(x, y)$ and at $x_{0,2} = (13.5, 1)$ for the analysis of $f_2(x, y)$. Tolerances : $\varepsilon_{\text{grad}} = 10^{-3}$ for gradient-norm $|\nabla f(\mathbf{x})|$, $\varepsilon_{\text{step}} = 10^{-3}$ for step size $|\mathbf{x}^{k+1} - \mathbf{x}^k|$ and $\varepsilon_\alpha = 10^{-9}$ for both gradient norm and step-size norm tolerances in the line search procedure. They are chosen quite loose to highlight the search directions and overall behavior rather than to have results heavily influenced by the stopping criteria.

[2]They are called via a call to a line search method, which takes as an argument the hessian (for generic programming reasons), but could be removed in a search of program efficiency improvement

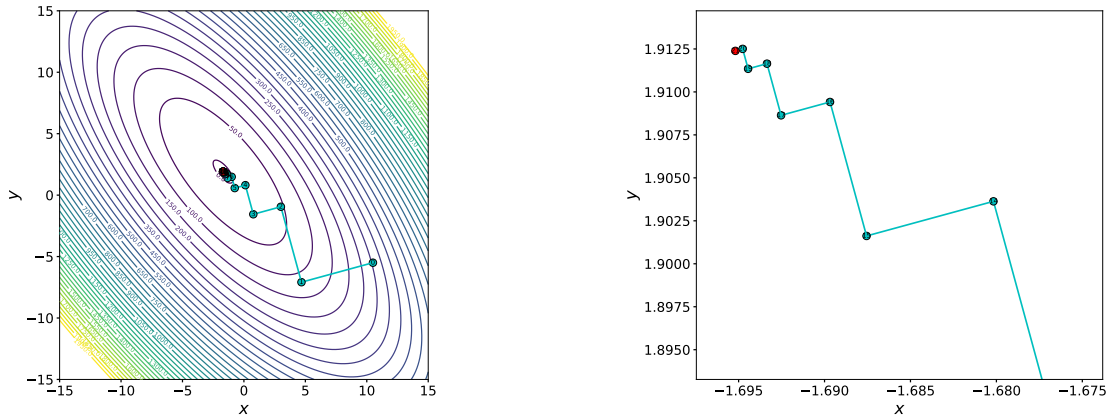
Function	Steepest Descent				CG				BFGS			
	N	f	$\|\nabla f\|$	H	N	f	$\|\nabla f\|$	H	N	f	$\|\nabla f\|$	H
f_1	21	0	43	21	2	0	5	2	2	0	67	0
f_2	40	0	1310	40	5	0	148	5	6	0	184	0

Table A: Number of iterations (N), function evaluations (f), gradient evaluations (∇f) and Hessian evaluations (H) until convergence of the three optimization algorithms for the minimization of $f_1(x, y)$ and $f_2(x, y)$, starting respectively from $x_{0,1} = (10.5, -5.5)$ and $x_{0,2} = (13.5, 1)$, with tolerances $\varepsilon_{\text{grad}} = 10^{-3}$, $\varepsilon_{\text{step}} = 10^{-3}$ and $\varepsilon_\alpha = 10^{-9}$. Note that the Hessian evaluations in the CG method are not used[2].



(a) $f_1(x, y)$, $x_0 = (10.5, -5.5)$ (b) f_2, $x_0 = (13.5, 1)$

Figure 1: Values of the objective function evaluated at each iterate x_k as a function of the iteration number k, for both objective functions and their respective initial points x_0.



(a) Domain $[-15, 15] \times [-15, 15]$ (b) Zoom on the last iterates

Figure 2: Trajectory of the iterates obtained with the Steepest Descent method applied to function f_1, starting from the initial point $x_0 = (10.5, -5.5)$, highlighting the zigzagging phenomenon.

smaller steps, and therefore smaller function value improvement, as seen in Figure 1a) as the number of iterations increase - see Figure 4.

This phenomenon is amplified in regions with eccentric level curves; the steps are even smaller and the method even slower.

The direct consequence of this phenomenon is the very high number of gradient evaluations (and of the hessian in a smaller measure, only one per iteration), requiring high computational time.

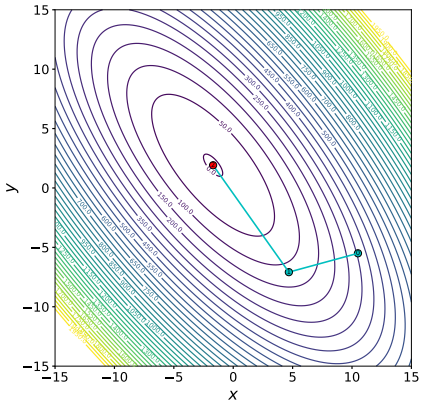
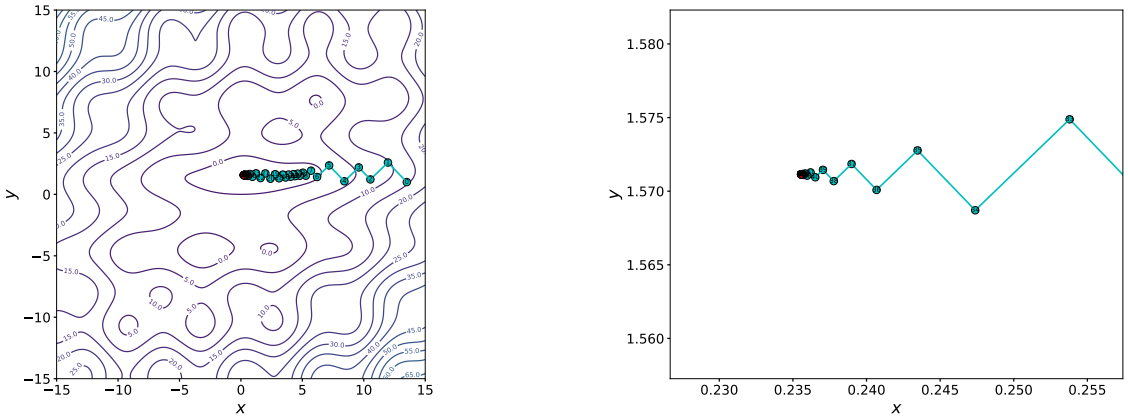


Figure 3: Trajectory of the iterates obtained with the Conjugate Gradients and BFGS methods applied to function f_1, starting from the initial point $x_0 = (10.5, -5.5)$.



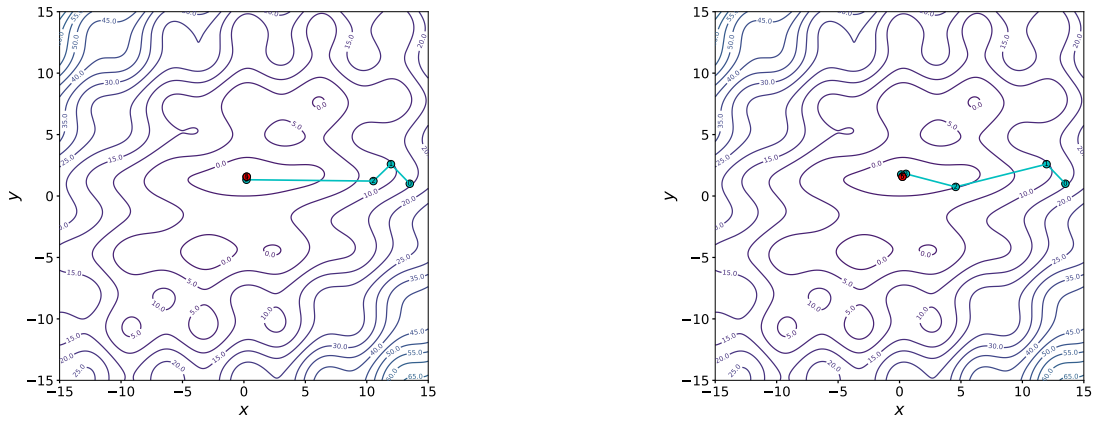
(a) Domain $[-15, 15] \times [-15, 15]$

(b) Zoom on the last iterates

Figure 4: Trajectory of the iterates obtained with the Steepest Descent method applied to function f_2, starting from the initial point $x_0 = (13.5, 1)$.

Conjugate Gradient (CG) method This method appeared to be the most efficient among the three. The first search direction is identical to that of the Steepest Descent method as it is chosen to be equal to the negative gradient. The following directions are computed to be conjugate one to another with regards to the Hessian matrix. For strictly convex quadratic functions (SCQF), this property guarantees convergence in at most n steps, where n is the dimension of the problem. This property is verified for the solution path shown in Figure 3.

However, for general C_1 (non-quadratic) functions, the Hessian is no longer constant — conjugacy then holds only with regards to a local Hessian. As a result, the theoretical n-step convergence guarantee is lost. Nevertheless, by periodically re-initializing the algorithm (resetting the search direction to the negative gradient every n iterations), conjugacy only depends on a most the last $n - 1$ iterates (instead of all the iterates, which makes no sense if $k > n$), which greatly improves robustness and convergence speed, in addition to the guarantee of convergence inherited from the steepest descent method. The trajectory and results for C_1 functions are shown in figures 5 and 1b respectively.



(a) Conjugate Gradient method

(b) BFGS method

Figure 5: Trajectory of the iterates obtained with the CG and BFGS methods applied to function f_2, starting from the initial point $x_0 = (13.5, 1)$.

In practice, the CG method for general C_1 functions only requires gradient evaluations (Fletcher Reeves update rule in this case) — the Hessian is not explicitly computed (unless required by the line search procedure). This makes it an attractive method, combining the simplicity of gradient-based approaches with a much faster convergence rate.

BFGS Algorithm This method showed a performance and behavior very similar to the Conjugate Gradient (CG) method.. The BFGS algorithm works by progressively reconstructing an approximation $\mathbf{H}^k$ of the inverse Hessian matrix. The search directions are then computed to be conjugate with regards to this approximate Hessian.

For strictly convex quadratic functions (SCQF), the sequence $\mathbf{H}^k$ converges exactly to the true inverse Hessian in at most n steps, which guarantees convergence within n iterations—clearly visible in Figure 3, where BFGS and CG follow identical trajectories.: the first direction is the negative gradient (since $\mathbf{H}^0 = \mathbf{I}_n$), and following directions remain conjugate with regards to the Hessian, leading to the same iterates.

For general C_1 functions, the local Hessian varies with position, so $\mathbf{H}^k$ can only approximate its inverse - never (at least it is not guaranteed) exactly reconstructing it. As a result, the n-step convergence property no longer holds–this can be observed in Figure 5.
However, BFGS has a better asymptotic convergence behavior than CG and remains robust even in non-quadratic regions. Its main drawback is the memory requirement: storing and updating the full $n \times n$ matrix $\mathbf{H}^k$ at each iteration can become costly for high-dimensional problems.

Comparison From Table A, several conclusions arise:

- For strictly convex functions, without a doubt, the CG method is the most efficient and accurate method. It delivers the exact minimum (thanks to the analytical step length α_k) in only 2 steps and 5 gradient evaluations;

  The BFGS method also converges in two steps[3], but relies on a line search procedure (here,

[3] Valid for sufficiently loose gradient-norm tolerances; see Figure 6 for sensitivity to tolerance.

dichotomy; the analytical formula yielded wrong results), which limits accuracy to the chosen tolerance and increases the number of gradient evaluations.

The steepest descent is far behind in terms of performance, because of the lack of an n-step convergence guarantee, and accuracy that is only up to the gradient-norm tolerance. The higher computational cost is not a straightforward conclusion from the table, but comes from a rather practical explanation : for real optimization problems, finite differences are used to approximate gradients and hessians, both relying on function evaluations (4 evaluations for centered scheme for gradient and 9 evaluations for a hessian, see 3.2). In that context, the steepest descent would require more function evaluations in total, meaning a higher computational time, due to the high number of hessian evaluations. We can also note that the dichotomy method with the chosen tolerance might not be the best performing one, meaning the BFGS method could be computationally less demanding than the data from the table.

- For general C_1 functions, the more interesting case from an application point of view, where the analytical step length is no longer available, both CG and steepest descent require line search procedures. The CG method converged in the fewest iterations and required the fewest gradient evaluations (Hessian evaluations are not counted, see note in Table A), closely followed by BFGS. Once again, steepest descent was by far the slowest, requiring roughly ten times more computational effort for comparable accuracy.

As a conclusion, the conjugate gradients method appears to be the method to go with when solving optimization problems, whether it is for SCQF or for general C_1 functions, and whether it is with analytical expressions or finite differences for the gradient and hessian.
The BFGS can be a serious alternative in the case of general C_1 functions too, but can require a huge amount of storage for high-dimension problems.

2.2 Initial point

In this section we analyze the influence of the initial point on the performance of the different algorithms.
The effects of the stopping criteria, and the line search are eliminated to keep only the effects of the initial point. Therefore, in this section, the stopping criteria are fixed to $\varepsilon_{grad} = 10^{-3}$ for the gradient-norm tolerance , $\varepsilon_{step} = 10^{-6}$ for the step size tolerance and $\varepsilon_{\alpha} = 10^{-9}$ for both the gradient and step size tolerances in the line search algorithms when used for the general C_1 function.

2.2.1 Strictly Convex Quadratic function

Let's focus on the behavior for a SCQF to see how the three optimization methods compare on this simple case. In order to isolate the line search method from the actual optimization methods performance, we use an optimal step size α_k at each iteration (see formula in the appendix).

Starting from two initial points $x_{0,1} = (10, 10)$ and $x_{0,2} = (10.5, 5.5)$, the three methods yield the results shown in Table B. These points were selected to highlight the **strong sensitivity** of the Steepest Descent method to the choice of initial point — one weakness of this method.

Point	Steepest Descent	CG	BFGS
$x_{0,1}$	4	2	2
$x_{0,2}$	23	2	2

Table B: Number of iterations until convergence of the 3 algorithms for the optimization of $f_1(x, y)$, starting from $x_{0,1} = (10, 10)$ and $x_{0,2} = (10.5, -5.5)$

In contrast, both the Conjugate Gradient (CG) and BFGS methods showed great robustness, converging to the same minimum in n iterations, regardless of initialization. This difference comes from the fact that Steepest Descent only relies on gradient directions, orthogonal one to another, which can cause inefficient zigzagging, while CG and BFGS have more advanced search directions, conjugate with regards to the hessian.

2.2.2 General C_1 function

Point	Steepest Descent				CG				BFGS				End Point & $f(x_{\text{final}})$
	N	f	$\|\nabla f\|$	H	N	f	$\|\nabla f\|$	H	N	f	$\|\nabla f\|$	H	
$x_{0,1}$	3	0	109	3	2	0	68	2	2	0	78	0	$(0.23, 1.57)$, $f = -3.24$
$x_{0,2}$	40	0	1310	40	5	0	155	5	6	0	184	0	$(0.23, 1.57)$, $f = -3.24$
$x_{0,3}$	5	0	169	5	5	0	148	5	5	0	146	0	$(-3, -4.6)$, $f = -2.712$

Table C: Number of iterations (N), function evaluations (f), gradient evaluations (∇f) and hessian evaluations (H) until convergence of the three optimization algorithms for the optimization of $f_3(x, y)$, starting from $x_{0,1} = (13.5, 2)$, $x_{0,2} = (13.5, 1)$ and $x_{0,3} = (0, -7)$, and the end point and value of the objective function at the end point

As before, the dichotomy method is used for the line search in all cases to ensure a fair comparison. The results are presented in Table C.

The initial points were again chosen to emphasize highlight behaviors.

The first observation is that the final solutions differ between runs. This clearly illustrates that the algorithms converge to a stationary point, not necessarily the global minimum. For general nonlinear or non-convex functions, multiple local minima exist, and minimization methods are therefore sensitive to the initial condition.

Although the theoretical n-step convergence property of the Conjugate Gradient (CG) and BFGS methods no longer applies here, both keep strong efficiency and robustness compared to the Steepest Descent method.

The CG algorithm, particularly when periodically reinitialized, keeps fast convergence despite the varying Hessian. The BFGS method also performs consistently well, as its Hessian approximation continues to provide well-oriented search directions leading to efficient convergence. However, it is important to keep in mind that all these observations strongly depend on the choice of the initial point, which ultimately determines which local minimum the algorithm will reach.

2.3 Stopping criteria

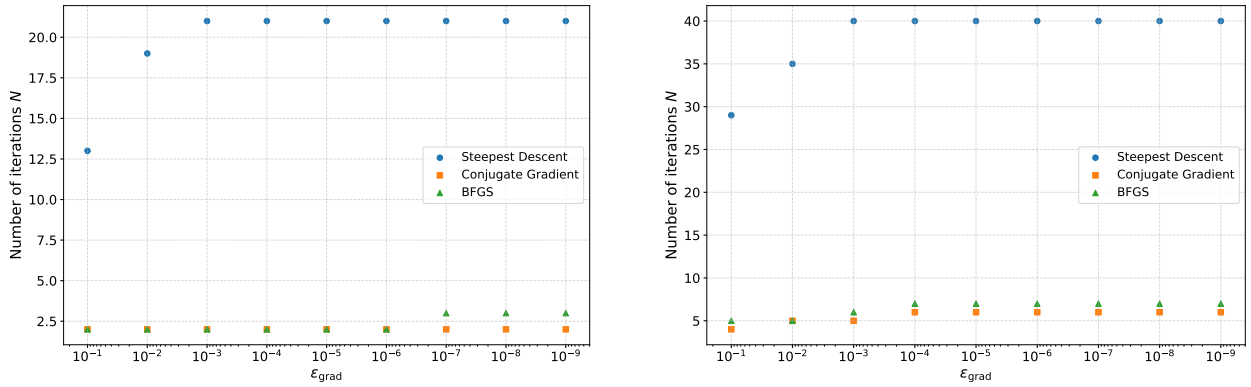
Here we discuss the impact of the stopping criteria on the performance of the optimization algorithms. Performance can be separated into two categories that often act opposite: the speed of convergence and the accuracy of the result.
Two stopping criteria are used in this study. The first one is based on the gradient norm, motivated by theoretical considerations, as it directly measures how close the current iterate is to a stationary point:

$$\|\nabla f(x_k)\| < \varepsilon_{\text{grad}}. \tag{2.1}$$

The second criterion is based on the step-size between two successive iterates:

$$\|x_{k+1} - x_k\| < \varepsilon_{\text{step}}. \tag{2.2}$$

It is used as a practical safeguard to prevent unnecessary iterations when the algorithm stagnates or makes only very small, negligible, incremental improvements. Therefore, the step-size criterion does not directly measure the convergence quality or speed, but ensures that the optimization terminates once further progress becomes negligible. Its tolerance can thus be fixed from empirical measurements, with a typical value $\varepsilon_{\text{step}} = 10^{-6}$.
A sensitivity analysis on the gradient norm threshold $\varepsilon_{\text{grad}}$ is carried out to identify the optimal trade-off between computational efficiency and solution accuracy.



(a) $f_1(x, y)$, $x_0 = (10.5, -5.5)$ (b) f_2, $x_0 = (13.5, 1)$

Figure 6: Number of iterations N until convergence of the three optimization algorithms as a function of the gradient-norm tolerance $\varepsilon_{\text{grad}}$, for both test functions and their initial points x_0.

Results of this analysis can be found in Figure 6.
Conclusions for both SCQF and general C_1 functions are identical. It can once more be seen that the CG and BFGS methods largely outperform the steepest descent method, especially when choosing a "bad" initial point as in this case.
For the steepest descent method, the optimal trade-off, although somewhat subject to the desired accuracy, appears to lie between $\varepsilon_{\text{grad}} = 10^{-3}$ and 10^{-6}. Beyond this range, the gain in precision becomes negligible compared to the extra computational time required, thus degrading the overall computational efficiency. On the other hand, setting the tolerance higher than 10^{-3} can lead to stopping too early, before actually reaching a proper stationary point.

However, for BFGS and CG methods, the gradient norm tolerance $\varepsilon_{\text{grad}}$ has very little influence on the computational costs associated to these two methods. Therefore, for these two methods, a tight tolerance, such as 10^{-7}, could be applied with no huge marginal cost compared to looser, lower accuracy, ones.
However, ultimately, the optimal choice of the tolerance $\varepsilon_{\text{grad}}$ depends on the desired level of accuracy and the specific application context, as well as the computational resources at our disposition.

3 Line search methods

3.1 Convergence

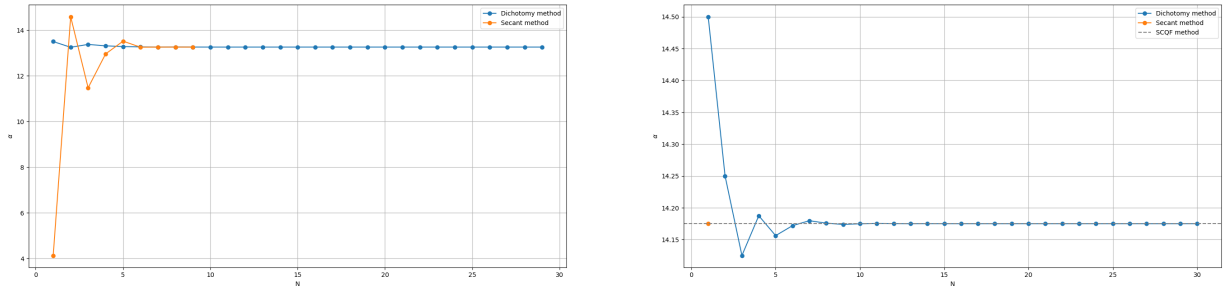
In this section, the consequences of varying the stopping criterion won't be considered. The stopping criterion is fixed to $\varepsilon_\alpha = 10^{-9}$for both gradient norm and step-size norm.

3.1.1 Global convergence

In theory, the secant method isn't globally convergent. However, there are 2 cases in which it can be improved. Since this method is an approximation of the Newton-Raphson method, we can state that if $\phi(\alpha)$ is a quadratic function, then the method converges in one step. Moreover, if we assume that ϕ is C_3 at α^* and $\phi^{''}(\alpha^*) > 0$, for well chosen initial points, the method is *p*-superlineary convergent with *p* equals to the golden number.

On the other hand, the dichotomy method is globally convergent. Furthermore, assuming that $\phi^{'}(0) < 0$ and that there exists $\overline{\alpha}$ such that $\alpha \geq \overline{\alpha} \Rightarrow \phi^{'}(\alpha) > 0$, the method mostly converges to a local minimizer of ϕ.If ϕ is unimodal, however, it converges to the unique global minimizer of ϕ.

In this part of the convergence study, many parameters are fixed. Concerning the dichotomy method, $h = 1$ and the stopping criteria are $|\alpha_{max} - \alpha_{min}| < \varepsilon_\alpha$ and $\phi^{'}(\alpha) < \varepsilon_\alpha$. For the secant method, $\alpha^1 = 1, \alpha^0 = 0$ and the stopping criteria are $|\alpha^k - \alpha^{k-1}| < \varepsilon_\alpha$ and $\phi^{'}(\alpha^k) < \varepsilon_\alpha$. The value of the starting point depends on the function studied, either $x_{0,1} = (10, 10)$ for f_1, or $x_{0,2} = (13.5, 2)$ for f_2. The search direction is defined as the steepest descent direction, *i.e.* $s = -\frac{\nabla f}{\|\nabla f\|}$.



(a) $f_2(x, y)$, $x_{02} = (13.5, 2)$ (b) f_1, $x_{01} = (10, 10)$

Figure 7: Convergence of α as a function of the number of iterations N, for both the dichotomy and secant methods

For the secant method, the Figure 7b confirms the property of global convergence in one step. Indeed, there's only one point on the graph, located exactly at the correct value of α, which can be verified using Equation 0.2 (whose solution is denoted as *SCQF method* in the legend of the graph). In addition, in Figure 7a, convergence towards the final value of α is clearly visible, which occurs because the function is C_3. The convergence of the dichotomy method is also verified in both figures.

To effectively compare these two methods, the number of iterations and gradient evaluations should be considered. It is clear that the dichotomy method requires many more iterations to satisfy the stopping criteria, whereas the secant method converges much faster. However, even though the number of iterations is larger, the dichotomy method still reaches a value very close to the exact solution in a few steps. Thus, comparing only the total number of iterations is not sufficient.

Function	**Dichotomy**			**Secant**		
	N	N_I	$\|\nabla f\|$	N	N_I	$\|\nabla f\|$
$f_1(x_{01})$	30	14	45	1	0	3
$f_2(x_{02})$	29	13	43	9	0	11

Table D: Number of iterations (N), number of iterations for the initialization process (N_I) and gradient evaluations (∇f) until convergence of the two line search methods applied on $f_1(x, y)$ and $f_2(x, y)$, with respectively $x_{0,1} = (10, 10)$ and $x_{0,2} = (13.5, 2)$, with tolerances $\varepsilon_\alpha = 10^{-9}$, $h = 1$ and $\alpha^k = 1, \alpha^{k-1} = 0$.

According to Table D, the computational cost of the two methods is easy to compare. The dichotomy method requires a large number of iterations to converge and, even before that, needs several iterations to evaluate the first interval. In Figure 7, only the iterations related to the search of the final value of α were considered, but the initialization steps aren't negligible. In terms of gradient evaluations, the numbers are similar to those of the iterations. On the other hand, the secant method only takes a few iterations to obtain the final value of α, and has no initialization process.

Finally, we can note that the dichotomy method almost performs the same for both an SCQF function and a general C_1 function, which is not the case for the secant method, relying on a property in the case of quadratic functions. The increase in the number of steps is quite significant, even if the final count remains quite small.

3.1.2 Initialization process

The initialization process refers to determining an initial interval before updating the value of α to converge to a minimizer of $\phi(\alpha)$. Indeed, both methods require two different points (α_{min} and α_{max}, or α^0 and α^1) to update the variable.

The secant method is simpler, as it only requires two initial values, α^0 and α^1. There is no real initialization procedure requiring an iterative process to find optimal starting points, but it remains important to understand how the choice of these two variables affects the convergence of the line search method. On the opposite, the dichotomy method involves an iterative process.

1. Choose a step size h and fix $\alpha_{min} = 0$

2. Compute $\phi^{'}(h)$: if $\phi^{'}(h) < 0$, then $\alpha_{min} = h$ and $h = 2h$. Or, if $\phi^{'}(h) > 0$, $\alpha_{max} = h$ and that's it.

h	$f_1(x_{01})$			$f_2(x_{02})$		
	N	N_I	α	N	N_I	α
0.01	24	1417	14.174	20	1325	13.258
1	30	14	14.174	29	13	13.258
10	34	1	14.174	31	1	13.258

Table E: Dichotomy method : effects of varying the step size (h) on the number of iterations (N), the number of iterations of the initialization process (N_I), and of the final value of α.

α^k	$f_1(x_{01})$			$f_2(x_{02})$		
	N	N_I	α	N	N_I	α
0.01	1	0	14.174	8	0	13.258
1	1	0	14.174	9	0	13.258
10	1	0	14.174	17	0	13.258

Table F: Secant method : effects of varying α^1 on the number of iterations (N), the number of iterations of the initialization process (N_I), and of the final value of α.

In Table E, variations in the step size have significant effects on the total number of iterations. Actually, the number of iterations in the initialization procedure changes considerably, whereas the number of iterations of the main method remains roughly constant. There's only a small decrease of N, but a large increase in N_I. Thus, a compromise should be made to minimize the total number of iterations ($N + N_I$). In Table F, the property of convergence in one step, for an SCQF function, is still valid, regardless of the initial α^1. For a more general function, N varies more significantly, especially when α^1 increases. For both methods, and regardless of the initial values, the final value of α remains the same, at least within a precision of 10^{-3}.

3.1.3 Conclusion

Overall, these analyses highlight the strengths and weaknesses of the two methods. The main strength of the secant method lies in its ability to converge rapidly, requiring just a few iterations and gradient evaluations to reach the value of α that satisfies the stopping criteria. However, if the function isn't strictly convex quadratic, or at least C_3, the asymptotic behavior of the method isn't guaranteed. For the dichotomy method, it ensures the property of global convergence for any function. Despite this advantage, its computational cost is considerably higher, which can result in slower convergence for more complex functions.

3.2 Stopping criteria

The purpose of a line search method is to evaluate a value of α^k that minimizes the function at x^{k+1}, *i.e.*

$$\phi(\alpha^k) = \min_{\alpha \geq 0} \phi(\alpha), \text{where} \quad \phi(\alpha) = f(x^k + \alpha s^k) \tag{3.1}$$

In other words, the minimization of the function ϕ results in vanishing its derivative. As a consequence, the first stopping criterion can be expressed as a gradient-norm tolerance: $|\phi^{'}(\alpha^k)| < \varepsilon_\alpha$. A second criterion comes up from the way these methods work. Both use an interval defined by a lower and a upper bound, and iteratively compute a new value located between them. The interval is successively reduced until it surrounds the desired value of α. It can be written as : $|\alpha_{upper} - \alpha_{lower}| < \varepsilon_\alpha$.

So far, ε_α has been fixed at 10^{-9}. It is interesting to understand how varying this value affects the number of iterations, and the accuracy of α^k. Here, $h = 1$ and $\alpha^1 = 1$ (the first upper bound for the secant method), with the same $x_{0,1}$ and $x_{0,2}$ as in the previous section.

ε_α	$f_1(x_{01})$		$f_2(x_{02})$	
	N_t	α	N_t	α
10^{-1}	18	14.156	16	13.312
10^{-3}	24	14.175	22	13.258
10^{-6}	34	14.174	32	13.258
10^{-9}	44	14.174	42	13.258

Table G: Dichotomy method : effects of varying the tolerance (ε_α) on the total number of iterations (N_t), and the value of α.

ε_α	$f_1(x_{01})$		$f_2(x_{02})$	
	N_t	α	N_t	α
10^{-1}	1	14.174	6	13.258
10^{-3}	1	14.174	7	13.258
10^{-6}	1	14.174	8	13.258
10^{-9}	1	14.174	9	13.258

Table H: Secant method : effects of varying the tolerance (ε_α) on the total number of iterations (N_t), and the value of α.

Once again, the dichotomy method appears to be more sensitive to changes in the tolerance. The total number of iterations ($N + N_I$) increases when ε_α decreases. Regarding the value of α, the first evaluations, corresponding to $\varepsilon_\alpha = 10^{-1}$, aren't very accurate but as soon as the tolerance reaches 10^{-3}, they become correct. For the secant method, as expected, the tolerance variation has no effect for a strictly convex quadratic function and only a minor effect for the other function. Regardless of the precision, the evaluated value of α remains essentially the same.

In conclusion, the effects of varying the tolerance are quite similar to those observed when varying the parameters of the initialization process discussed above : the secant method is not much affected – allowing for a higher accuracy for small marginal cost, and there is a trade-off for the dichotomy method between accuracy and computational time.
Finally note that line search methods are not required to be super accurate, therefore one might be satisfied with an tolerance of even 10^{-1}.

Appendix

Optimal step size formula for SCQF In a 1D optimization problem on $f(\mathbf{x}_k + \alpha \mathbf{s}_k)$ working on a SCQF, the optimal step α is given by

$$\alpha = \frac{\|\nabla f(x_k)\|^2}{[\nabla f(x_k)]^T A \nabla f(x_k)} \tag{0.2}$$

where A is the hessian of the function f.

Centered Finite Difference for the Gradient For a function $f(x, y)$, the components of the gradient are approximated using a centered finite difference scheme as:

$$\frac{\partial f}{\partial x}(x, y) \approx \frac{f(x + h, y) - f(x - h, y)}{2h}$$

$$\frac{\partial f}{\partial y}(x, y) \approx \frac{f(x, y + h) - f(x, y - h)}{2h}$$

where h is a small step, requiring 4 function evaluations .

Centered Finite Difference for the Hessian The Hessian matrix $\mathbf{H}$ is approximated using second-order centered finite differences:

$$H_{xx} \approx \frac{f(x + h, y) - 2f(x, y) + f(x - h, y)}{h^2}$$

$$H_{yy} \approx \frac{f(x, y + y) - 2f(x, y) + f(x, y - h)}{h^2}$$

$$H_{xy} \approx \frac{f(x + h, y + h) - f(x - h, y + h) - f(x + h, y - h) + f(x - h, y - h)}{4h^2},$$

$$H_{yx} = H_{xy}$$

requiring 9 function evaluations.