A Comparative Study of Modified and Truncated Newton Methods on Large-Scale Test Functions

Salvatore Nocita
Politecnico di Torino
Turin, Italy
s346378@studenti.polito.it

Luciano Scarpino
Politecnico di Torino
Turin, Italy
s346205@studenti.polito.it

September 2025
Code: https://github.com/LucianoScarpino/Newton_Methods_Optimization.git

1 Introduction

The study of numerical optimization is fundamental in applied mathematics, computer science, and engineering, since many real-world problems can be formulated as the minimization of nonlinear functions. In this project we focus on two variants of second-order optimization methods: the *Truncated Newton method* and the *Modified Newton method*. Our work consists of implementing both algorithms from scratch in Python, testing them under different scenarios as described in Appendix A, and analyzing their performance. The main objective is to highlight the strengths and weaknesses of these approaches and to compare them with respect to efficiency, accuracy, and robustness.

2 Newton Methods

The central idea behind Newton-type methods is the use of second-order information to accelerate convergence. While first-order methods such as Gradient Descent rely solely on the gradient of the objective function, Newton's method incorporates curvature information through the Hessian matrix. If the objective function f is at least twice continuously differentiable, then by Taylor's theorem we can construct the quadratic model around the current iterate $x^{(k)}$ as

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

Minimizing this local approximation yields the Newton step, which is obtained by solving the linear system

$$\nabla^2 f(x^{(k)})\, p^{(k)} = -\nabla f(x^{(k)}).$$

If the Hessian $\nabla^2 f(x^{(k)})$ is positive definite, the resulting direction $p^{(k)}$ is guaranteed to be a descent direction, and the method exhibits *quadratic convergence* in a neighborhood of the solution. This feature makes Newton's method one of the most powerful techniques in unconstrained optimization.

Nevertheless, several drawbacks limit its direct applicability. First, the method is sensitive to the choice of the initial guess: a poor starting point can lead to divergence. Second, when the Hessian is not positive definite, the quadratic model is not convex, and the Newton step may not correspond to a descent direction. A common remedy is to modify the Hessian to enforce positive definiteness, though this may reduce accuracy. Third, the computational cost is significant, as each iteration requires forming the Hessian and solving a linear system, which can be prohibitive for large-scale problems.

These challenges have motivated the development of more practical Newton-type strategies. In particular, the truncated and modified Newton methods aim to mitigate the computational burden and stability issues, while retaining the desirable fast local convergence of Newton's approach. In this report, we explore these two methods in detail, discussing their theoretical foundations, practical implementations, and experimental results.

2.1 Modified Newton Method

In Newton-type methods, the update direction is obtained by using second-order curvature information through the Hessian. This choice can accelerate convergence, but it also risks producing directions that do not decrease the objective. For this reason, imposing the *positive definiteness* (PD) property on the Hessian matrix is essential in Newton-type methods. In fact, ensuring that $\nabla^2 f(x^{(k)})$ is PD guarantees that the resulting $p^{(k)}$ is indeed a descent direction, since

$$\nabla f(x^{(k)})^T p^{(k)} < 0.$$

Moreover, a PD matrix is always invertible, which is required for solving the linear system, and it ensures that any stationary point ($\nabla f(x^\star) = 0$) is also a *local minimizer* of f.

The *Modified Newton Method* aims to guarantee a PD Hessian at each iteration of the algorithm. The idea is that if $H(x^{(k)})$ is not PD, it is modified by adding a small correction:

$$B_k = H(x^{(k)}) + E_k,$$

where E_k should be as small as possible (in order to preserve second-order information) but large enough to guarantee that B_k is PD. The resulting quadratic model is

$$m_k(p) = f(x^{(k)}) + \nabla f(x^{(k)})^T p + \tfrac{1}{2} p^T B_k p,$$

where B_k is the modified Hessian at the k-th iteration.

This approach finds mathematical justification in the condition

$$\|B_k\| \, \|B_k^{-1}\| \leq C,$$

which, if satisfied for every k, ensures that

$$\lim_{k \to \infty} \nabla f(x^{(k)}) = 0,$$

i.e., the sequence $\{x^{(k)}\}_{k \geq 0}$ is attracted to the stationary points of f. Since B_k is enforced to be PD, such stationary points correspond to *local minimizers*.

Finally, several strategies can be adopted to modify the Hessian at each iteration; in section 3.2 , we discuss the one employed in our work.

2.2 Truncated Newton Method

A central difficulty of the Newton method is the repeated solution of the linear system

$$\nabla^2 f(x^{(k)}) \, p^{(k)} = -\nabla f(x^{(k)}),$$

which can be prohibitively expensive in high dimensions. Solving this system exactly at every iteration often leads to an unnecessary computational burden, especially when the current iterate $x^{(k)}$ is still far from the minimizer. In such cases, demanding high accuracy produces little benefit in terms of the outer iterations, a phenomenon known as *oversolving*.

The *Inexact Newton method* mitigates this cost by computing the Newton step only approximately, typically using an iterative Krylov subspace solver such as the Preconditioned Conjugate Gradient (PCG) method. Instead of solving the system to full precision, the iterations are terminated once the residual satisfies

$$\|\nabla^2 f(x^{(k)}) p^{(k)} + \nabla f(x^{(k)})\| \;\leq\; \eta_k \, \|\nabla f(x^{(k)})\|,$$

where η_k is the *forcing term*. The forcing term is chosen adaptively: when $\|\nabla f(x^{(k)})\|$ is large (far from the solution), a loose tolerance is allowed, reducing inner computational cost; when the iterates approach the minimizer, the tolerance is tightened to recover the fast local convergence of Newton's method. This adaptive strategy strikes a balance between efficiency and accuracy, preventing oversolving while preserving desirable convergence properties.

The *Truncated Newton method* extends this idea by addressing another critical issue: the Hessian $\nabla^2 f(x^{(k)})$ may not be positive definite. Since iterative solvers such as CG rely on positive definiteness, indefiniteness of the Hessian can cause breakdowns or non-descent directions. To avoid this, the truncated Newton method incorporates a *negative curvature check*: during the inner iterations, if a search direction $p_{\text{in}}^{(i)}$ satisfies

$$p_{\text{in}}^{(i)^T} \nabla^2 f(x^{(k)}) \, p_{\text{in}}^{(i)} \leq 0,$$

the procedure is terminated early, and the last reliable direction is taken as $p^{(k)}$. In this way, the algorithm guarantees that the resulting step is always a descent direction, even when the Hessian is indefinite.

By combining inexact solves with adaptive tolerances and negative curvature safeguards, the truncated Newton method achieves scalability to large-scale problems while retaining the attractive local convergence behavior of Newton's method. This makes it a powerful and practical alternative to the exact formulation. The Truncated Newton method used in our experiments is reported in detail in Appendix B.1.

3 Implementation

3.1 General Tools

Inexact Line Search.

A crucial tool implemented for both Truncated and Modified Newton methods is the *inexact line search*. In the standard Newton method, the full Newton step is taken at each iteration, i.e., the solution of the linear system directly defines the step length. However, this approach can be problematic:

- If the current iterate is very close to the solution, the full step leads to fast convergence.
- If the iterate is far from the minimizer, taking the entire step may produce overly large updates, potentially causing divergence or instability.

To overcome this issue, we implement an inexact line search to dynamically select a suitable step length α_k at each iteration. In particular, we adopt a *backtracking strategy*, which starts from an initial step length $\alpha_0 = 1$ and iteratively reduces it until the *Armijo condition* is satisfied:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\,\alpha_k\,\nabla f(x_k)^\top p_k,$$

where p_k is the search direction, $c \in (0, 1)$ is a fixed parameter, and f is the objective function. If the Armijo condition is not met, the step length is scaled by a contraction factor $\rho \in (0, 1)$ until the sufficient decrease criterion is satisfied.

This simple backtracking approach ensures that the step length is either the initially chosen value α_0 or a suitably small value that guarantees sufficient decrease along the search direction, avoiding steps that are too large or unnecessarily small.

The pseudocode of this backtracking line search are reported in Appendix C.1.

Finite Difference Approximations.

When the objective function is too complex or expensive to differentiate analytically, we resort to numerical differentiation via *finite differences*. We

implemented forward, backward, and central difference schemes, both with fixed perturbation parameter h and with an *adaptive perturbation*, defined as

$$h_i = |x_i| \cdot h,$$

where h is a global scalar step size. This strategy allows the perturbation to scale with the magnitude of the variable, improving numerical stability.

The forward difference approximation of the derivative of a univariate function $f : \mathbb{R} \to \mathbb{R}$ is

$$f'(\bar{x}) \simeq \frac{f(\bar{x} + h) - f(\bar{x})}{h},$$

with an error of $\mathcal{O}(h)$. Similarly, the backward difference is

$$f'(\bar{x}) \simeq \frac{f(\bar{x}) - f(\bar{x} - h)}{h},$$

while the central difference, which leverages function evaluations on both sides, yields

$$f'(\bar{x}) \simeq \frac{f(\bar{x} + h) - f(\bar{x} - h)}{2h},$$

with a truncation error of $\mathcal{O}(h^2)$.

To make the approach computationally feasible for large-scale problems, the gradient is computed in parallel. Each partial derivative is evaluated independently across coordinates, enabling efficient use of multicore architectures.

The implementation provides a general class `SparseApproximativeDerivatives`, which allows the user to select the finite difference scheme (`forward`, `backward`, or `central`), the step size rule (fixed or adaptive), and the parallelism degree.

3.2 Modified Newton Method

Positive definiteness check.

A central feature of Newton-type methods is that the search direction is a descent direction if and only if the Hessian is positive definite. To verify this property, we attempt a Cholesky factorization of H_k. Recall that a symmetric matrix $A \in \mathbb{R}^{n \times n}$ admits a Cholesky factorization $A = LL^\top$ with L lower triangular if and only if A is symmetric and positive definite. Thus, the success of the factorization provides a practical numerical test for positive definiteness. If the factorization fails, H_k is not positive definite and must be modified before proceeding.

Hessian adjustments.

In practice, the Hessian approximation may fail to be positive definite or perfectly symmetric. To guarantee a valid descent direction, we first apply a diagonal correction

$$B_k = H_k + \delta I,$$

with $\delta > 0$ chosen adaptively. The correction is increased until B_k admits a successful Cholesky factorization, thereby ensuring positive definiteness. This mechanism preserves curvature information as much as possible while guaranteeing that the search direction

$$p_k = -B_k^{-1} \nabla f(x_k)$$

is always a descent direction. Moreover, convergence theory ensures that as long as δ remains bounded and vanishes asymptotically, global convergence is preserved.

Even after this modification, small asymmetries may still be present due to numerical errors. Hence, at each iteration we enforce symmetry by projection:

$$B_k \;\mapsto\; \tfrac{1}{2}(B_k + B_k^\top).$$

This adjustment perturbs the eigenstructure only minimally while ensuring that factorization routines operate on a valid symmetric matrix. Together, these safeguards make the Modified Newton method both numerically stable and theoretically sound.

Linear System Solver

In the Modified Newton method, the Cholesky factorization of B_k plays a dual role: it serves both as a test of positive definiteness and as the solver for the Newton system

$$B_k p_k = -\nabla f(x_k).$$

If the factorization succeeds, B_k is guaranteed to be SPD, and the resulting triangular factors can be reused directly to obtain p_k at negligible extra cost. This strategy is efficient for low and medium dimensions, but becomes infeasible at larger scales: the Hessian cannot be stored in memory, and even sparse variants of Cholesky suffer from fill–in effects that overwhelm computational resources.

In such regimes, the Modified Newton method cannot be applied, whereas the Truncated Newton method remains viable. By avoiding explicit factorization and relying instead on Hessian–vector products with Conjugate Gradient and negative curvature detection, the Truncated Newton provides a complementary alternative for large–scale problems.

3.3 Truncated Newton Method

Adaptive Tolerance.

An important aspect of the Truncated Newton method is the choice of the forcing term η_k, which determines the tolerance required when solving the inner linear system. As established in the theory of inexact Newton methods, the sequence $\{\eta_k\}$ plays a crucial role in the global behavior of the algorithm:

- If η_k is bounded away from zero ($0 < \eta_k \leq \hat{\eta} < 1$), the method converges only at a *linear rate*;
- If $\eta_k \to 0$ as $k \to \infty$, the method achieves *superlinear convergence*;
- If $\eta_k = \mathcal{O}(\|\nabla f(x^{(k)})\|)$, the method recovers the *quadratic convergence* typical of exact Newton.

The balance between computational cost and convergence speed is thus encoded in the choice of η_k. A more demanding tolerance leads to faster asymptotic convergence but requires more iterations of the inner solver, while a looser tolerance reduces inner costs but slows down the outer convergence. This trade-off is at the heart of practical implementations of the truncated Newton method.

In our implementation, this mechanism is explicitly built into the solver architecture. The class `TruncatedNewtonMethod` allows the user to specify a desired rate of convergence, which automatically determines the form of η_k through the internal routine `_set_eta_k`. Two different settings have been tested:

$$\eta_k = \min\{\hat{\eta}, \sqrt{(\|\nabla f(x^{(k)})\|)}\}, \qquad 0 < \hat{\eta} < 1 \quad \Longrightarrow \quad \text{superlinear convergence},$$

$$\eta_k = \min\{\hat{\eta}, (\|\nabla f(x^{(k)})\|)\}, \qquad 0 < \hat{\eta} < 1 \quad \Longrightarrow \quad \text{quadratic convergence}.$$

where $\hat{\eta}$ is a user-chosen cap.
By designing the solver in this way, we can directly compare the practical performance of the truncated Newton method under different forcing terms, evaluating both the asymptotic rate of convergence and the computational effort required at each iteration. The results of these tests will highlight the trade-offs between accuracy and efficiency that motivate the use of adaptive tolerances in large-scale optimization.

Matrix-free nature of Truncated Newton.

A crucial advantage of the Truncated Newton method is that it does not require the Hessian $\nabla^2 f(x)$ to be built, stored, or factorized. The only operation involving second-order information is the computation of a Hessian–vector product

$$\nabla^2 f(x^{(k)})\, v.$$

This feature avoids the main bottleneck of classical Newton-type methods: the cost of forming and storing the Hessian matrix, which in large dimension can be prohibitive both in terms of memory and operations. Instead, it suffices to approximate the *directional derivative* of the gradient along v:

$$\nabla^2 f(x)\, v = \lim_{h \to 0} \frac{\nabla f(x + hv) - \nabla f(x)}{h}.$$

Our implementation provides a general-purpose method `hessian_vector_product`, which can approximate gradients and Hessian–vector products by finite differences. The advantage is that one only evaluates the gradient at shifted points;

no Hessian matrix is ever assembled. This matrix-free approach is particularly attractive in high dimensions (e.g., $n \sim 10^5$), where storing the full Hessian would be infeasible.

For benchmark problems, we have also implemented efficient routines to compute *exact* Hessian–vector products. These exploit the specific algebraic structure of the test function, leading to faster and more accurate products than finite differences. Such exact products are expected to yield better numerical performance, since they avoid the discretization error inherent in finite-difference schemes.

Conjugate Gradient Method.

In our implementation, the linear systems arising in the Truncated Newton method are solved through a *Conjugate Gradient (CG)* scheme implemented from scratch. The algorithm differs from the standard CG method because, at each inner iteration, it explicitly checks for *negative curvature*, i.e., it verifies

$$p_{\text{in}}^{(i)^T} \nabla^2 f(x^{(k)})\, p_{\text{in}}^{(i)} \leq 0,$$

where $p_{\text{in}}^{(i)}$ is the current conjugate direction. This guarantees that the solver always exits with a valid descent direction:

- if the negative curvature condition is detected at the first iteration ($i = 0$), the method returns the steepest descent direction, i.e. $-\nabla f(x)$;
- if the condition is encountered at a later iteration ($i > 0$), the method returns the last iterate corresponding to a direction with positive curvature.

The stopping tolerance of the inner CG loop is not fixed, but *adaptive*, and depends on the distance from the solution, measured by the norm of the gradient, as described in the previous paragraph. This mechanism ensures efficiency in early iterations, while improving accuracy when approaching the minimizer.

Moreover, the CG solver has been designed to preserve the *matrix-free* nature of the Truncated Newton method: it never builds or stores the Hessian explicitly, but only requires Hessian–vector products. No other linear operators are invoked during the computation.

Finally, we did not incorporate any preconditioning strategy. Indeed, standard preconditioners cannot be applied in a matrix-free framework, while ad hoc preconditioners are problem-dependent and fall outside the scope of this work.

4 Experiments

4.1 Rosenbrock Test

The first experiment tests the implemented optimization methods on the Rosenbrock function

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

which is known for its narrow, curved valley and flat regions that make optimization challenging. $x_0 = (1.2, 1.2)$ and $x_0 = (-1.2, 1)$, using exact derivatives and recording their behavior in tables and figures. Following the assignment instructions (Appendix A),the methods were applied from two starting points. A sufficient decrease condition for the backtracking line search was imposed with $\rho = 0.5$ and $c = 10^{-4}$.

Modified Newton on Rosenbrock.

The preliminary test in Table 1 confirms that the Modified Newton method converges rapidly from both starting points, reaching near–machine precision in a limited number of iterations. Different initial conditions, however, affect the convergence trajectory.

Table 1: Rosenbrock, Modified Newton.

Starting Point	Derivative	Time [s]	Iterations	$\|\nabla f(x)\|$	EOC
$[-1.2;\ 1]$	exact	0.0024	21	4.47e-10	0.775
$[1.2;\ 1.2]$	exact	0.0014	7	1.45e-06	0.407

As shown in Figure 1, starting from $x_0 = [-1.2,\ 1]$ (blue) the gradient norm spikes at the beginning and exhibits mild oscillations before stabilizing, whereas starting from $x_0 = [1.2,\ 1.2]$ (red) produces a smoother and more direct descent. This indicates that some initial points require handling larger and irregular gradients, while others naturally align with the descent direction. The estimated orders of convergence remain modest (0.4–0.8), reflecting a damped behaviour rather than full quadratic acceleration, but still sufficient to ensure reliable convergence in these low–dimensional tests.

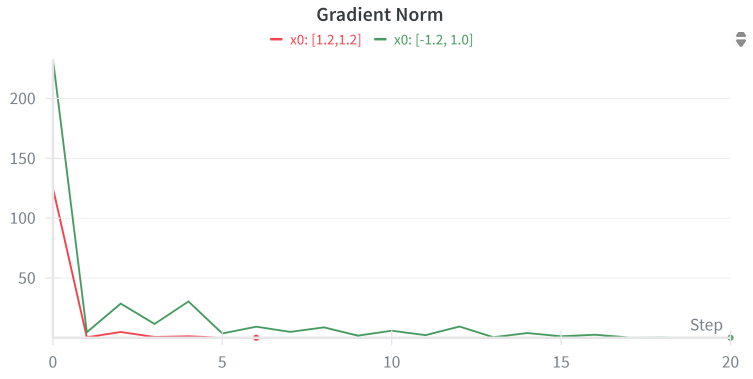


Figure 1: Gradient Norm vs. iteration

For $x_0 = [1.2, 1.2]$, convergence is reached in only 7 iterations, producing a shorter curve that terminates earlier. The cumulative execution time plot (Figure 2) highlights the influence of the starting point. In contrast, for $x_0 = [-1.2, 1]$, the run requires 21 iterations, leading to a steadily increasing execution time despite similar per-iteration costs. Thus, the difference is not due to efficiency per step, but to the number of iterations dictated by the initial condition.

Figure 2: Gradient Norm vs. iteration

Truncated Newton on Rosenbrock.

The preliminary assessment in Table 2 shows that the Truncated Newton method successfully converges from both starting points, albeit with some notable differences.

Table 2: Truncated Newton: outcomes by starting point and expected ROC.

Starting Point	Expected ROC	Time [s]	Iter	$f(x)$	$\|\nabla f\|$	EOC
$[-1.2; 1]$	superlinear	0.004	63	2.84e-15	2.37e-06	-0.154
$[-1.2; 1]$	quadratic	0.004	63	2.84e-15	2.37e-06	-0.154
$[1.2; 1.2]$	superlinear	0.001	9	5.55e-18	1.05e-07	-0.054
$[1.2; 1.2]$	quadratic	0.000	9	5.55e-18	1.05e-07	-0.054

For $x_0 = [-1.2, 1]$, convergence requires about 63 iterations and a runtime of 4 ms, reaching function values close to 10^{-15} and gradient norms on the order of 10^{-6}. For $x_0 = [1.2, 1.2]$, the method converges much faster, in just 9 iterations and 1 ms, while achieving even smaller function values (10^{-18}) and gradient norms (10^{-7}). The estimated orders of convergence (EOC) are negative in all cases, indicating that the truncated inner solves introduce inaccuracies that mask superlinear or quadratic acceleration, despite the solver reaching acceptable solutions.

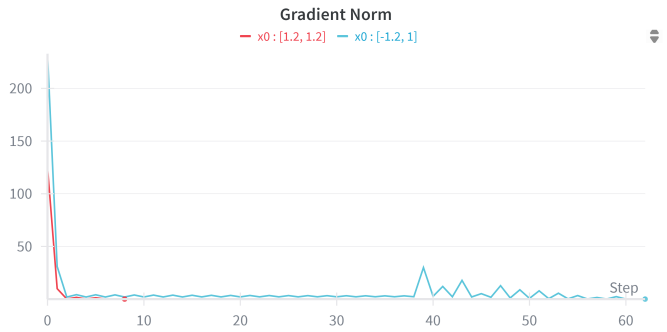
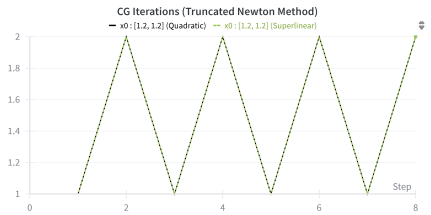
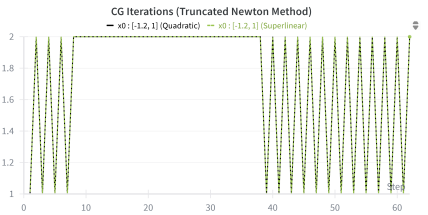


Figure 3: Gradient norm over iterations for the Truncated Newton method, starting from $[-1.2, 1]$ (blue) and $[1.2, 1.2]$ (red).

Figure 3 illustrates these trends. From $x_0 = [-1.2, 1]$, the gradient norm exhibits a sharp initial drop but then stabilizes with small oscillations before eventually decaying, reflecting the slower progress and higher iteration count. In contrast, starting from $[1.2, 1.2]$ yields a smoother and more direct descent, consistent with the faster convergence observed in the tabular results. This comparison highlights how the choice of initial conditions can heavily affect the performance of the Truncated Newton method, with some requiring many truncated solves to refine the solution, while others lead quickly to a stationary point. The behaviour of the inner Conjugate Gradient iterations under the Truncated Newton method is reported in Figures 4a and 4b. Interestingly, varying the starting point has no significant impact on the number of inner iterations. For $x_0 = [1.2, 1.2]$ the solver alternates regularly between one and two iterations per step, with no distinction between the quadratic and superlinear tolerance regimes. A similar pattern is observed for $x_0 = [-1.2, 1]$, where the CG iterations remain confined within the same narrow range. This suggests that, at least in these low–dimensional test cases, the truncated linear solves are insensitive to the choice of initial condition, producing nearly identical behaviour regardless of the expected rate of convergence.



(a) $x_0 = [1.2, 1.2]$ (Quadratic vs. Superlinear).



(b) $x_0 = [-1.2, 1]$ (Quadratic vs. Superlinear).

Figure 4: CG iterations per step for the Truncated Newton method, varying the starting point. Both cases show nearly identical patterns across ROC settings.

4.2 Setting

The numerical experiments follow the guidelines in Appendix A. We applied the Truncated Newton and Modified Newton methods to three benchmark problems from [0]: the Extended Rosenbrock, Extended Powell singular, and Broyden tridiagonal functions (formulations and suggested starting points are reported in Appendix D). A random seed equal to the minimum student ID ensures reproducibility.

Each problem was tested for dimensions $n = 10^d$, $d = 3, 4, 5$, using the suggested starting point $\bar{x}$ and 10 additional points sampled uniformly in the hypercube $[\bar{x}_i - 1, \bar{x}_i + 1]$. The goal of each test was to reach a gradient norm tolerance of 10^{-5}. Both methods employ a backtracking line search with parameters $\rho = 0.5$ and $c = 10^{-4}$. Derivatives were tested in two settings: exact and finite-difference approximations, using either uniform steps $h = 10^{-k}$ or adaptive steps $h_i = |x_i|10^{-k}$, with $k \in \{2, 4, 6, 8, 10, 12\}$.

Altogether, this resulted in 74 different combinations of experiment, starting point, derivative type and parameter settings for each dimensionality. Since Truncated Newton method was tested using two different forcing terms, chosen to ensure superlinear and quadratic rates of convergence, the combinations are doubled reaching 148.

The case $n = 10^5$ was not attempted for the Modified Newton method because forming and factorizing the Hessian explicitly requires $O(n^2)$ memory and $O(n^3)$ operations, which exceeds practical limits even with sparse matrix techniques. Attempts to run the solver were terminated by the operating system due to excessive memory usage.

Performance was evaluated in terms of successful runs, iteration counts, convergence rate, and execution time. To control runtime, a maximum convergence time was imposed via the heuristic

$$T_{\max} = \text{clip}\left(20 \cdot \left(\frac{n}{1000}\right)^{0.6}, 10, 300\right) \text{ [s]},$$

resulting in thresholds of 20s for $n = 10^3$, 79.6s for $n = 10^4$, and 300s for

$n = 10^5$. These values define the runtime limit for each individual run: if the last iteration starts immediately before $T_{\max}$, the run may slightly exceed the limit. Success ratios should therefore be interpreted relative to these thresholds.

5 Results

5.1 Extended Rosenbrock Function

The extended Rosenbrock function is used as defined in Appendix D, following [0]. Its structure exhibits strong nonlinearity in the odd residuals, forming a narrow, curved valley, while the even residuals couple consecutive variables toward the target value 1, making Newton-type methods sensitive to Hessian conditioning.

Sparse Hessians (Modified Newton).

In the Modified Newton method, two complementary constructions are employed to obtain a sparse second–order model:

1. **Exact sparse Hessian.** Writing $F(x) = \frac{1}{2}\|f(x)\|_2^2$ with Jacobian $J(x)$ of the residual vector $f(x)$, the Hessian decomposes as

$$\nabla^2 F(x) \;=\; J(x)^\top J(x) \;+\; \sum_{k=1}^{n} f_k(x)\,\nabla^2 f_k(x).$$

   For the extended Rosenbrock residuals, $J(x)$ is extremely sparse (each f_k depends on at most two variables), hence $J(x)^\top J(x)$ is banded and sparse. Moreover, $\nabla^2 f_k(x)$ vanishes for even k, while for odd k it contributes only to the diagonal. As a result, $\nabla^2 F(x)$ is assembled directly in compressed sparse row (CSR) format, with bandwidth 1 and a diagonal correction from the odd residuals.

2. **Finite–difference sparse Hessian.** When exact second–order information is not used, the Hessian is approximated by directional finite differences of the gradient. To preserve sparsity and avoid interference among perturbed coordinates, a simple two–coloring of the index set is adopted:

$$\mathcal{I}_{\text{even}} = \{0, 2, 4, \dots\}, \qquad \mathcal{I}_{\text{odd}} = \{1, 3, 5, \dots\}.$$

   Coordinates within the same color are perturbed simultaneously, the gradient is re-evaluated, and incremental quotients are used to populate only the known nonzeros (main diagonal and first sub/super-diagonals). Forward, backward, or central schemes are employed, with either a scalar step h or an adaptive per-coordinate step $h_i = |x_i|\,h$. The resulting nonzeros are then stored directly in CSR format.

Hessian–Vector Products (Truncated Newton).

In the Truncated Newton method, the solver requires only Hessian–vector products. Two complementary strategies are employed:

1. **Structured products exploiting sparsity.** For the extended Rosenbrock function, the residual structure allows Hv to be computed without forming the Hessian explicitly:

$$Hv \;=\; J(x)^\top \big(J(x)\, v\big) \;+\; \sum_{k \text{ odd}} 20\, f_k(x)\, v_k\, e_k,$$

   where e_k is the k-th coordinate vector. Both $J(x)\, v$ and $J(x)^\top(\cdot)$ are computed with sparse operations, yielding a structured and efficient implementation.

2. **Finite–difference products (matrix–free).** When exact second–order information is unavailable, Hv is approximated via directional finite differences of the gradient, following the strategy described in paragraph 3.1. This matrix-free approach integrates naturally with conjugate-gradient iterations and negative-curvature detection.

5.1.1 Modified Newton Method

Global summary.

Table 3 provides an overview of the results. At $n = 10^3$ the solver converged in 26/74 runs (35.1%), with a median of 22 iterations and 3.5 s wall-clock time. At $n = 10^4$ the success rate remains 26/74 (35.1%), with a median of 22 iterations and 40.28 s. The experimental order of convergence (EOC) was consistently subquadratic for $n = 10^3$ (median 0.77), reflecting the globalization effects of backtracking line search, whereas higher dimension manifests a quadratic one (median 1.00).

Table 3: Extended Rosenbrock, Modified Newton: global outcomes.[1]

Dimension	Total runs	Successes	Fails	Median iters	Median time [s]	Median EOC
$n = 10^3$	74	26	48	22.0	3.50	0.77
$n = 10^4$	74	26	48	22.0	40.28	1.00
$n = 10^5$	–	–	–	–	–	–

We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme. The experiments are also categorized into

[1] Median iterations, time, and EOC are computed only over successful runs. This convention is consistently applied to all global summaries, while detailed breakdowns also include unsuccessful cases.

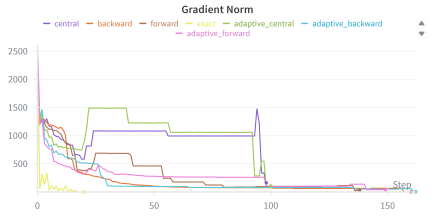
"fixed", using the assignment's prescribed starting point, and "sampled," reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Tables 4 and 5.

Table 4: Extended Rosenbrock ($n = 10^3$), Modified Newton: by derivative and experiment type.

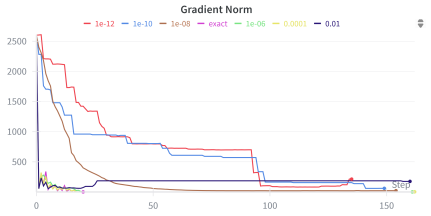
Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	0	6	158.0	20.07	4.00e+01
adaptive_backward	sampled	6	0	6	148.0	20.08	4.00e+01
adaptive_central	fixed	6	3	3	57.5	12.28	1.01e+01
adaptive_central	sampled	6	3	3	54.5	12.33	1.01e+01
adaptive_forward	fixed	6	3	3	80.0	11.71	9.98e+00
adaptive_forward	sampled	6	3	3	77.5	11.80	9.99e+00
backward	fixed	6	0	6	150.0	20.06	3.83e+01
backward	sampled	6	0	6	149.5	20.07	3.83e+01
central	fixed	6	3	3	59.0	12.48	9.88e+00
central	sampled	6	3	3	59.0	12.53	9.88e+00
exact	fixed	1	1	0	21.0	0.30	5.00e-09
exact	sampled	1	1	0	21.0	0.31	5.00e-09
forward	fixed	6	3	3	77.0	11.70	1.37e+01
forward	sampled	6	3	3	77.5	11.76	1.27e+01

Table 5: Extended Rosenbrock ($n = 10^4$), Modified Newton: by derivative and experiment type.

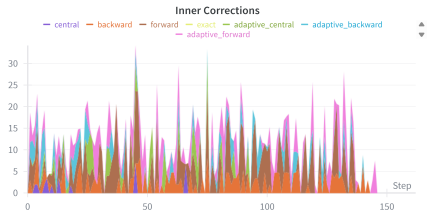
Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	0	6	34.0	80.64	3.37e+02
adaptive_backward	sampled	6	0	6	34.0	81.16	3.37e+02
adaptive_central	fixed	6	3	3	21.0	63.98	3.27e+02
adaptive_central	sampled	6	3	3	21.0	63.90	3.27e+02
adaptive_forward	fixed	6	3	3	19.5	75.87	1.55e+02
adaptive_forward	sampled	6	3	3	19.5	75.92	1.55e+02
backward	fixed	6	0	6	27.0	81.12	6.19e+02
backward	sampled	6	0	6	27.5	80.88	6.19e+02
central	fixed	6	3	3	21.0	68.55	3.28e+02
central	sampled	6	3	3	21.0	68.46	3.28e+02
exact	fixed	1	1	0	21.0	31.75	1.58e-08
exact	sampled	1	1	0	21.0	30.96	1.58e-08
forward	fixed	6	3	3	18.0	75.13	6.71e+02
forward	sampled	6	3	3	18.0	74.78	6.71e+02



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Corrections (Averaged by Derivative Method).

(d) Execution Time (One/Two Eval).[2]

Figure 5: Modified Newton Method Performance on the Extended Rosenbrock Function (n = 1000).

Analysis and Observations.

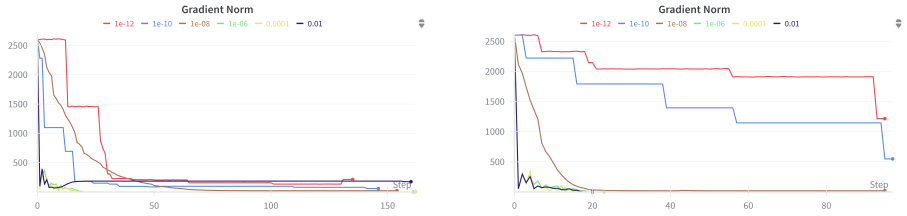
For the Extended Rosenbrock function, the Modified Newton method shows a clear gap between exact and finite–difference (FD) gradients. Figure 5a for $n = 10^3$ illustrates that exact derivatives ensure a stable, monotone decrease of the gradient norm, reaching $\|\nabla f\| \approx 5 \times 10^{-9}$ in about 22 iterations (Table 4), several orders of magnitude better than FD schemes.

Forward and adaptive forward also succeed in half the runs but stabilize at higher norms ($\sim 1.0 \times 10^1$). Similarly, according to results in Table 4, central and adaptive central perform well at $n = 10^3$, with about half of the runs converging and residuals near 10. Backward and adaptive backward never succeed, with residuals between 40.

Figure 5b shows the sensitivity to perturbation size. Nevertheless $h = 10^{-2}$ achieves one the fastest initial descent, smaller values (10^{-12}–10^{-8}) eventually yield better long-term performance. The best compromise is found when implying $h = 10^{-6}$ -10^{-4}, combining a strong early reduction with stable progress.

[2]Panel (d) includes: one–evaluation schemes (forward) and their adaptive variants; two–evaluation schemes (central) and their adaptive variants; their preconditioned (or quadratic, in the case of Truncated Newton) counterparts; and the exact derivative. Backward and adaptive backward, together with their variants, are omitted since they are already represented by the forward case, in order to improve readability. This convention is followed in all subsequent execution–time figures.

Notably, central and adaptive central appear superior in median outcomes, but Figures 6b reveal that they are far more sensitive to perturbation choice: very small steps lead to poor performance, while backward differences (Figure 6a), though less accurate overall, display more consistent behaviour across runs.



(a) Backward differences with varying perturbations ($n = 10^3$).

(b) Central differences with varying perturbations ($n = 10^3$).

Figure 6: Comparison between backward and central finite-difference schemes on the Extended Rosenbrock function ($n = 10^3$).

The inner corrections (Figure 5c) highlight distinct behaviours: adaptive methods produce frequent peaks and irregular patterns, central schemes show more regular corrections in line with their lower residuals, while backward and adaptive backward accumulate higher values with comparatively less correction activity.

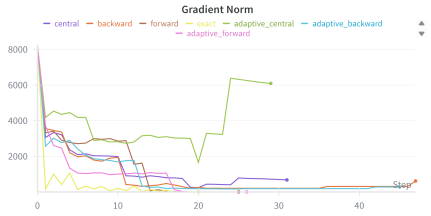
Execution times (Figure 5d) show the expected efficiency gap: exact derivatives converge in 0.3 s, whereas all FD variants saturate (at least once) the 20 s budget with single evaluation schemes being 5 seconds faster on average.

For $n = 10^4$, the same trends hold (Figure 7). Exact derivatives converge in 22 iterations, reducing $\|\nabla f\|$ to 10^{-8} in almost 30 s. FD methods remain inconsistent: some runs (central, adaptive central, forward) succeed, but median residuals arrives even above 10^2. Thus, although more FD runs approach low values than at $n = 10^3$, the overall success rate is still limited.

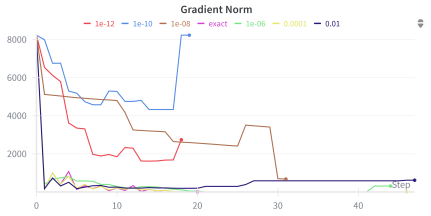
Figure 7b again highlights perturbation effects: $h = 10^{-2}$ descends fastest initially, but $h = 10^{-6}$ achieves the best long-term results. By contrast, the other perturbations display alternating behaviours, with $h = 10^{-8}$ closely resembling the performance of the best settings, though stagnating earlier.

Execution times (Figure 7d) increase as expected, though in this case the gap with exact derivatives is less pronounced. FD methods for single and two evaluations behave similarly, demonstrating no preference in selection.

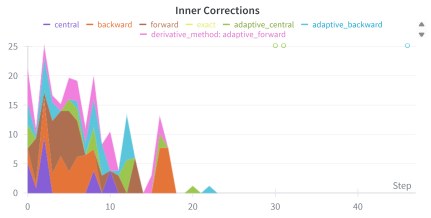
In successful runs, the experimental order of convergence is about 0.77 for $n = 10^3$ and 1.0 for $n = 10^4$, showing that the Modified Newton method remains sublinear in the first scenario but reach the linear rate as dimension grows.



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Corrections (Averaged by Derivative Method).

(d) Execution Time (One/Two Eval).

Figure 7: Modified Newton Method Performance on the Extended Rosenbrock Function (n = 10000)

5.1.2 Truncated Newthon Method

Global summary.

Table 6 provides an overview of the results. At $n = 10^3$ the solver converged in 25/148 runs (16.9%), with a median of 64 iterations and 12.7 s wall-clock time. At $n = 10^4$ the success rate increased slightly to 32/148 (21.6%), with a median of 47 iterations and 14.3 s. At $n = 10^5$ the method succeeded in only 17/148 runs (11.5%), with a median of 15 iterations and 195.5 s. The experimental order of convergence (EOC) was systematically below expectations: median values ranged between 0.33 and 0.56 at lower scales, and even turned negative in the superlinear regime at $n = 10^5$, highlighting the loss of Newton–like acceleration when inner solves are truncated.

Table 6: Extended Rosenbrock, Truncated Newton: global outcomes.

Dimension	Total runs	Successes	Fails	Median iters	Median time [s]	Median EOC (superlinear)	Median EOC (quadratic)
$n = 10^3$	148	25	123	64.0	12.70	0.389	0.366
$n = 10^4$	148	32	116	47.0	14.32	0.334	0.559
$n = 10^5$	148	17	131	15.0	195.50	-0.309	0.578

We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme and expected rate of convergence. The experiments are also categorized into “fixed”, using the assignment’s prescribed

starting point, and "sampled", reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Tables 7, 8 and 9.

Table 7: Extended Rosenbrock ($n = 10^3$), Truncated Newton: by derivative, expected rate of convergence, and experiment type.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	90.5	20.07	1.47e+01
adaptive_backward	fixed	superlinear	6	0	6	90.5	20.09	1.47e+01
adaptive_backward	sampled	quadratic	6	0	6	90.5	20.08	1.47e+01
adaptive_backward	sampled	superlinear	6	0	6	90.5	20.07	1.47e+01
adaptive_central	fixed	quadratic	6	2	4	59.0	20.05	1.45e+01
adaptive_central	fixed	superlinear	6	1	5	59.0	20.09	1.60e+01
adaptive_central	sampled	quadratic	6	2	4	59.0	20.05	1.45e+01
adaptive_central	sampled	superlinear	6	2	4	59.0	20.08	1.45e+01
adaptive_forward	fixed	quadratic	6	1	5	73.5	20.06	5.73e+00
adaptive_forward	fixed	superlinear	6	0	6	102.5	20.07	5.75e+00
adaptive_forward	sampled	quadratic	6	1	5	74.0	20.10	5.73e+00
adaptive_forward	sampled	superlinear	6	0	6	99.0	20.08	5.73e+00
backward	fixed	quadratic	6	0	6	63.0	20.08	3.02e+01
backward	fixed	superlinear	6	0	6	68.5	20.07	3.91e+01
backward	sampled	quadratic	6	0	6	63.0	20.08	3.40e+01
backward	sampled	superlinear	6	0	6	69.0	20.07	3.40e+01
central	fixed	quadratic	6	1	5	49.5	20.10	4.28e+01
central	fixed	superlinear	6	1	5	51.0	20.09	3.53e+01
central	sampled	quadratic	6	1	5	51.5	20.09	3.53e+01
central	sampled	superlinear	6	1	5	51.5	20.10	5.02e+01
exact	fixed	quadratic	1	1	0	64.0	0.33	1.02e-13
exact	fixed	superlinear	1	1	0	64.0	0.36	6.37e-08
exact	sampled	quadratic	1	1	0	64.0	0.33	1.02e-13
exact	sampled	superlinear	1	1	0	64.0	0.32	6.37e-08
forward	fixed	quadratic	6	2	4	78.5	20.01	2.60e-01
forward	fixed	superlinear	6	2	4	78.5	20.07	2.60e-01
forward	sampled	quadratic	6	2	4	79.0	20.04	2.60e-01
forward	sampled	superlinear	6	2	4	79.0	20.03	2.60e-01

Table 8: Extended Rosenbrock ($n = 10^4$), Truncated Newton: by derivative, experiment type, and expected rate of convergence.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	1	5	104.5	79.84	2.94e+00
adaptive_backward	fixed	superlinear	6	1	5	104.0	79.78	2.94e+00
adaptive_backward	sampled	quadratic	6	1	5	108.5	79.78	4.52e+00
adaptive_backward	sampled	superlinear	6	1	5	107.5	79.79	4.51e+00
adaptive_central	fixed	quadratic	6	2	4	59.0	79.81	2.41e+01
adaptive_central	fixed	superlinear	6	1	5	84.0	79.83	1.96e+01
adaptive_central	sampled	quadratic	6	2	4	59.0	79.83	1.58e+01
adaptive_central	sampled	superlinear	6	1	5	86.0	79.78	2.41e+01
adaptive_forward	fixed	quadratic	6	1	5	59.5	79.81	1.69e+00
adaptive_forward	fixed	superlinear	6	1	5	126.0	79.74	1.48e+00
adaptive_forward	sampled	quadratic	6	1	5	59.0	79.79	2.42e+00
adaptive_forward	sampled	superlinear	6	1	5	127.0	79.75	1.48e+00
backward	fixed	quadratic	6	1	5	65.0	79.79	1.80e+00
backward	fixed	superlinear	6	1	5	88.5	79.78	1.33e+00
backward	sampled	quadratic	6	1	5	65.5	79.79	2.08e+00
backward	sampled	superlinear	6	1	5	91.5	79.79	1.88e+00
central	fixed	quadratic	6	2	4	59.5	79.84	7.30e+01
central	fixed	superlinear	6	1	5	92.0	79.80	7.47e+01
central	sampled	quadratic	6	2	4	59.5	79.86	7.70e+01
central	sampled	superlinear	6	1	5	96.5	79.81	5.01e+01
exact	fixed	quadratic	1	1	0	64.0	4.60	3.22e-13
exact	fixed	superlinear	1	1	0	64.0	3.98	2.01e-07
exact	sampled	quadratic	1	1	0	64.0	4.93	3.22e-13
exact	sampled	superlinear	1	1	0	64.0	5.01	2.01e-07
forward	fixed	quadratic	6	1	5	63.0	79.79	1.33e+00
forward	fixed	superlinear	6	1	5	133.0	79.76	1.33e+00
forward	sampled	quadratic	6	1	5	63.5	79.73	1.33e+00
forward	sampled	superlinear	6	1	5	135.5	79.71	1.33e+00

Table 9: Extended Rosenbrock ($n = 10^5$), Truncated Newton: by derivative, experiment type, and expected rate of convergence.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	1	5	14.5	307.26	5.63e+02
adaptive_backward	fixed	superlinear	6	1	5	14.5	307.11	6.67e+02
adaptive_backward	sampled	quadratic	6	1	5	14.5	307.24	4.70e+02
adaptive_backward	sampled	superlinear	6	1	5	14.5	307.90	3.60e+02
adaptive_central	fixed	quadratic	6	1	5	10.5	312.55	4.17e+02
adaptive_central	fixed	superlinear	6	1	5	9.0	308.08	4.19e+02
adaptive_central	sampled	quadratic	6	1	5	10.5	308.09	8.62e+02
adaptive_central	sampled	superlinear	6	1	5	10.5	309.78	8.42e+02
adaptive_forward	fixed	quadratic	6	0	6	10.0	307.47	5.95e+02
adaptive_forward	fixed	superlinear	6	0	6	10.0	308.95	6.26e+02
adaptive_forward	sampled	quadratic	6	1	5	10.0	306.78	7.17e+02
adaptive_forward	sampled	superlinear	6	0	6	10.0	307.41	6.94e+02
backward	fixed	quadratic	6	1	5	9.5	307.36	1.20e+03
backward	fixed	superlinear	6	1	5	9.5	307.62	1.20e+03
backward	sampled	quadratic	6	1	5	9.5	306.87	1.27e+03
backward	sampled	superlinear	6	1	5	9.5	305.98	1.24e+03
central	fixed	quadratic	6	0	6	11.0	309.06	6.29e+02
central	fixed	superlinear	6	0	6	10.0	310.93	4.08e+02
central	sampled	quadratic	6	0	6	12.5	307.57	8.62e+02
central	sampled	superlinear	6	0	6	11.5	308.22	5.83e+02
exact	fixed	quadratic	1	1	0	64.0	55.07	1.02e-12
exact	fixed	superlinear	1	1	0	64.0	54.02	6.37e-07
exact	sampled	quadratic	1	1	0	64.0	56.64	1.02e-12
exact	sampled	superlinear	1	1	0	64.0	56.23	6.37e-07
forward	fixed	quadratic	6	0	6	9.5	308.28	4.25e+02
forward	fixed	superlinear	6	0	6	9.5	308.22	4.25e+02
forward	sampled	quadratic	6	0	6	9.5	308.13	3.86e+02
forward	sampled	superlinear	6	0	6	9.5	307.85	4.25e+02

Analysis and Observations.

The Truncated Newton method applied to the Extended Rosenbrock function shows limited robustness, with performance strongly dependent on dimensionality.

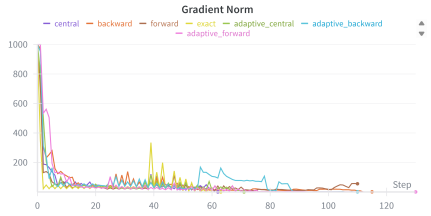
At $n = 10^3$ (Table 7) only 25 runs succeed out of 148, with median iteration counts around 64 and runtimes near 13 s. The estimated orders of convergence are modest (0.39 superlinear, 0.37 quadratic), as the truncation of inner solves prevents the method from fully exploiting Newton–like acceleration. Exact derivatives behave consistently, converging in about 64 iterations with gradient norms reduced to machine precision, but finite–difference (FD) schemes stagnate at residuals between 10^{-1} and 10^2.

Forward differences achieve the best results, while central and adaptive central record more nominal successes but remain stuck at higher values. Adaptive backward is systematically unsuccessful. Figure 8a confirms these trends: most methods reduce the gradient sharply at the beginning but flatten prematurely, and even the exact derivative requires many iterations before convergence.

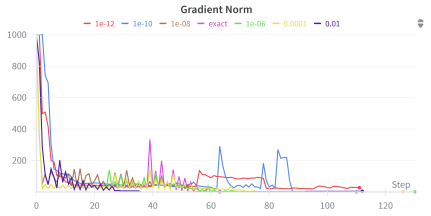
Figure 8b shows how perturbations behave similarly, with $h = 10^{-2}$ descending fastest initially but smaller values eventually more effective.

The analysis of inner iterations (Figure 8c) shows that the curves for superlinear and quadratic tolerances almost always coincide, with only sporadic deviations in which the quadratic variant requires a slightly higher number of iterations.

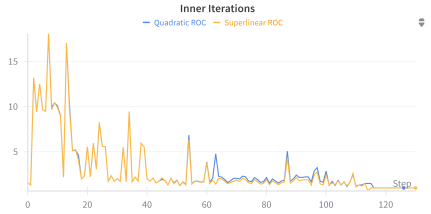
Execution times (Figure 8d) show forward and adaptive forward as the most efficient FD schemes, but still far from exact derivatives.



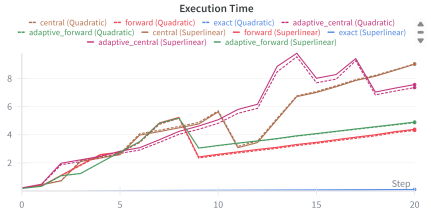
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).



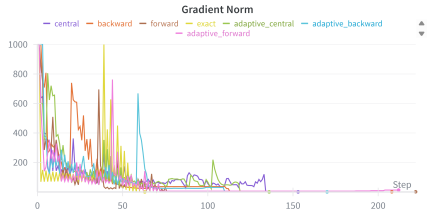
(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 8: Truncated Newton Method Performance on the Extended Rosenbrock Function (n = 1000)

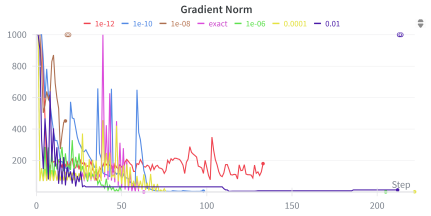
At $n = 10^4$ (Table 8) the method shows slightly higher robustness, with 32 successful runs. Exact derivatives converge reliably in 64 iterations and within 5 s, while FD methods saturate the 80 s budget. Forward and backward reach residuals near one, whereas adaptive forward settles slightly higher and adaptive backward remains weaker. Central and adaptive central look better in terms of median values but stagnate at residuals one or two orders of magnitude larger, explaining their inconsistency.

Figure 9a highlights greater instability than at $n = 10^3$, with oscillations across FD schemes and even a peak in the exact curve. Perturbation analysis (Figure 9b) shows that $h = 10^{-6}$, 10^{-10}, and 10^{-4} achieve the best balance, while 10^{-2} stagnates after a steep initial descent and 10^{-8}–10^{-12} fail earlier.

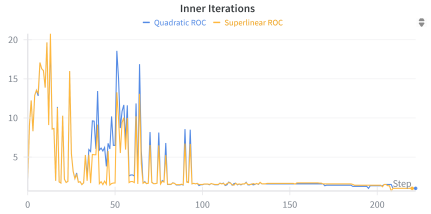
Inner iterations (Figure 9c) reveal that quadratic tolerances often surpass superlinear ones in the mid phases. Execution times (Figure **??**d) highlight the efficiency gap between exact and FD schemes: the former remains nearly constant across iterations, while FD methods accumulate steadily higher costs. Among them, forward and adaptive forward are comparatively faster, though still an order of magnitude slower than exact derivatives.



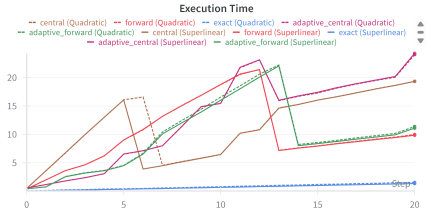
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).



(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

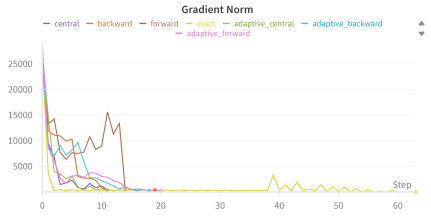
Figure 9: Truncated Newton Method Performance on the Extended Rosenbrock Function (n = 10000)

At $n = 10^5$ (Table 9) the method becomes impractical. Only 17 runs succeed out of 148, with runtimes close to the 300 s budget and median iteration counts dropping to 15. Exact derivatives still converge in 64 iterations, but require nearly one minute, while FD methods stagnate at residuals from several hundred to above 10^3.

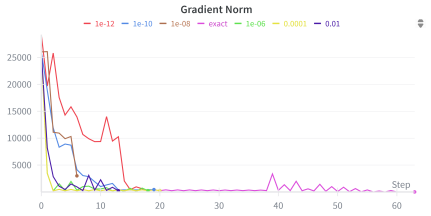
Perturbation (Figure 10b) $h = 10^{-8}$ reproduces the poor behaviour seen at $n = 10^4$, stagnating early, and $h = 10^{-10}$ is slower initially but eventually aligns with the others.

Inner iterations (Figure 10c) show almost no difference between quadratic and superlinear tolerances, while the quadratic setting occasionally lowers per–iteration time, likely because better early solves reduce the cost of subsequent updates.

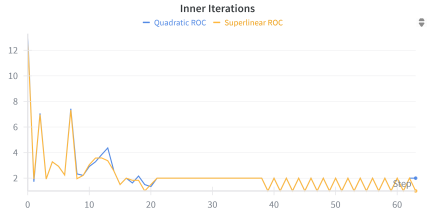
Execution times (Figure 10d)underline the enormous gap between exact and FD methods, as the former exploits accurate gradients to minimize conjugate–gradient iterations, while the latter expend the full budget without meaningful progress.



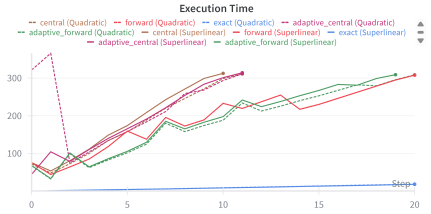
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).



(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 10: Truncated Newton Method Performance on the Extended Rosenbrock Function (n = 100000)

5.2 Extended Powell Function

The extended Powell singular function is used as defined in Appendix D, following [0]. The residuals are grouped in blocks of four variables: two residuals are linear, while the other two introduce quadratic nonlinearities within the same block.

Sparse Hessians (Modified Newton).

Two complementary constructions are employed to obtain a sparse second–order model:

1. **Exact sparse Hessian.** With the standard split

$$\nabla^2 F(x) \;=\; J(x)^\top J(x) \;+\; \sum_{k=1}^{n} f_k(x)\,\nabla^2 f_k(x),$$

the Jacobian $J(x)$ is block-sparse: each residual depends only on the variables of its own block, hence $J(x)^\top J(x)$ is block-diagonal with independent 4×4 blocks. Among the four residuals, the first two are linear (their second derivatives vanish), while the quadratic ones contribute only local corrections inside each block. Thus $\nabla^2 F(x)$ is assembled in sparse CSR format as a block-diagonal matrix with 4×4 blocks.

2. **Finite–difference (approximated) sparse Hessian.** When exact second-order information is not used, the Hessian is approximated from directional finite differences of the gradient. A 4-coloring of the index set, corresponding to positions within each block, ensures that only one coordinate per block is perturbed at a time, avoiding contamination across variables in the same block. Forward, backward, or central schemes are applied, with either a scalar step h or an adaptive per-coordinate step $h_i = |x_i|\,h$. The resulting derivatives are assembled directly into sparse 4×4 blocks.

Hessian–Vector Products (Truncated Newton).

In the Truncated Newton method, the solver requires only Hessian–vector products. Two complementary strategies are employed:

1. **Structured products exploiting sparsity.** For the extended Powell function, the residual structure allows Hv to be computed without forming the Hessian explicitly:

$$Hv \;=\; J(x)^\top J(x)\,v \;+\; \sum_{\text{blocks}} \sum_{k \in \text{block}} f_k(x)\,\nabla^2 f_k(x)\,v_{\text{block}}.$$

   so that each Hv involves only sparse operations within its 4×4 block. This yields Hv without forming $\nabla^2 F(x)$ explicitly and preserves the block sparsity.

2. **Finite–difference products (matrix–free).** When exact second–order information is unavailable, Hv is approximated via directional finite differences of the gradient, following the strategy described in paragraph 3.1. This matrix-free approach integrates naturally with conjugate-gradient iterations and negative-curvature detection.

5.2.1 Modified Newton Method.

Global summary.

In table 11 overview of the results is provided. At $n = 10^3$ the solver converged in 28/74 runs (37.8%), with a median of 12 iterations and 2.63 s wall-clock time. At $n = 10^4$ the success rate increased to 34/74 (45.9%), with a median of 11 iterations and 26.3 s. The experimental order of convergence (EOC) was consistently close to 1.0, indicating that the Modified Newton method behaved more like a globally convergent method with linear rate on this problem.

Table 10: Extended Powell, Modified Newton: global outcomes.

Dimension	Total runs	Successes	Failures	Median iters	Median time [s]	Median EOC
$n = 10^3$	74	28	46	12.0	2.63	1.0
$n = 10^4$	74	34	40	11.0	26.30	1.0
$n = 10^5$	–	–	–	–	–	–

We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme. The experiments are also categorized into "fixed," using the assignment's prescribed starting point, and "sampled," reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Table 11 and 12.

Table 11: Extended Powell ($n = 10^3$), Modified Newton: by derivative and experiment type.

Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	1	5	96.5	20.08	3.12e+00
adaptive_backward	sampled	6	1	5	95.5	20.07	3.12e+00
adaptive_central	fixed	6	3	3	33.5	12.17	2.84e+00
adaptive_central	sampled	6	3	3	34.0	12.32	2.83e+00
adaptive_forward	fixed	6	3	3	53.0	11.34	3.16e+00
adaptive_forward	sampled	6	3	3	53.0	11.32	3.15e+00
backward	fixed	6	1	5	97.0	20.12	3.17e+00
backward	sampled	6	1	5	100.0	20.07	3.17e+00
central	fixed	6	3	3	35.0	12.08	2.83e+00
central	sampled	6	3	3	35.0	12.06	2.83e+00
exact	fixed	1	1	0	13.0	0.20	9.64e-06
exact	sampled	1	1	0	13.0	0.23	9.64e-06
forward	fixed	6	2	4	94.5	20.09	3.14e+00
forward	sampled	6	2	4	96.5	20.05	3.15e+00

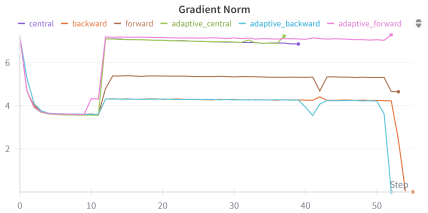
Table 12: Extended Powell ($n = 10^4$), Modified Newton: by derivative and experiment type.

Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	2	4	25.5	80.27	1.11e+00
adaptive_backward	sampled	6	2	4	25.5	80.89	1.11e+00
adaptive_central	fixed	6	3	3	12.5	55.83	1.14e+00
adaptive_central	sampled	6	3	3	12.5	55.24	1.14e+00
adaptive_forward	fixed	6	3	3	13.0	59.89	1.14e+00
adaptive_forward	sampled	6	3	3	13.0	59.57	1.14e+00
backward	fixed	6	2	4	17.5	80.42	1.14e+00
backward	sampled	6	2	4	17.5	81.02	1.14e+00
central	fixed	6	3	3	15.0	56.20	1.10e+00
central	sampled	6	3	3	14.5	55.17	1.10e+00
exact	fixed	1	1	0	14.0	21.40	9.00e-06
exact	sampled	1	1	0	14.0	21.44	9.00e-06
forward	fixed	6	3	3	14.5	58.20	1.05e+00
forward	sampled	6	3	3	14.5	57.54	1.05e+00

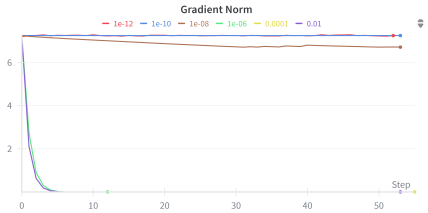
Analysis and Observations.

The results confirm the strong dependence of the Modified Newton method on gradient accuracy.

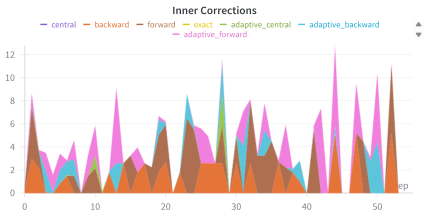
A striking discrepancy appears in the initial gradient norm: exact derivatives start above 3000, while finite–difference estimates begin near 7.2. This gap arises from cancellation in $[F(x+h) - F(x)]/h$, where both terms are large but their difference is small, causing round-off to suppress the dominant first-order term. To avoid compressing the finite–difference curves, the exact derivative trend is shown separately in Figure 12, while Figures 11 and 13 compare approximate methods on a common scale. As a result of this underestimation, finite-difference gradients make the landscape appear flatter from the very first step.



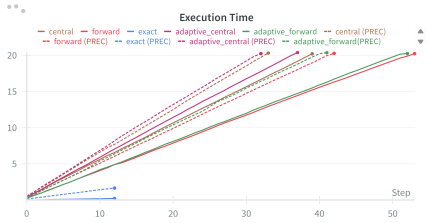
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



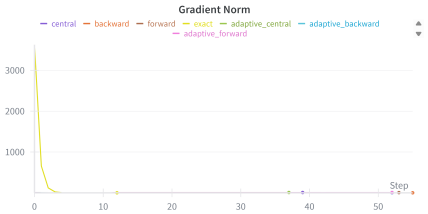
(c) Inner Corrections (Averaged by Derivative Method).



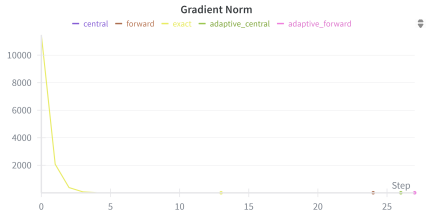
(d) Execution Time (One/Two Eval, No Precond/Precond).

Figure 11: Modified Newton Method Performance on the Extended Powell Function (n = 1000)

Table 11 show this distortion clearly. Exact derivatives converge steeply to 10^{-5} in 13 iterations, while finite-difference schemes plateau near 3.0. The inner corrections (Figure 11c) reflect instability: adaptive schemes oscillate, central differences are steadier but still limited by underestimated gradients, and none approach true stationarity.



(a) All Derivative Methods (1000).



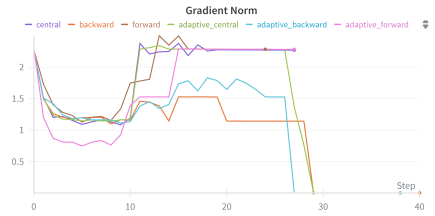
(b) All Derivative Methods (10000).

Figure 12: Gradient Norm Trends for all the Derivative Methods (including the exact).

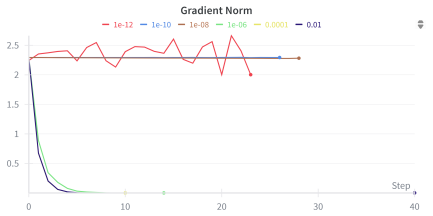
Perturbation size also plays a key role. Figure 5b shows that $h = 10^{-6}$ and $h = 10^{-2}$ achieve the lowest residuals, while very small (10^{-12}–10^{-8}) or intermedate (10^{-4}) steps stagnate.

Execution-time curves (Figure 11d) underline the efficiency gap: exact derivatives converge in 13 iterations with sub-second runtimes for $n = 10^3$ and 30–40 s for $n = 10^4$, while finite-difference schemes require steeper, costlier trajectories.

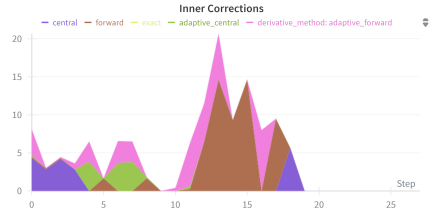
Comparing dimensions shows consistent patterns. For $n = 10^3$, exact derivatives converge in $\sim$0.3 s, while finite-difference schemes saturate the 20 s budget with residuals near 3.0. At $n = 10^4$, exact derivatives converge in 14 iterations and 30–40 s, whereas finite-difference methods need more iterations, take 60–80 s, and stagnate near 1.0–1.1.



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Corrections (Averaged by Derivative Method).

(d) Execution Time (One/Two Eval, No Precond/Precond).

Figure 13: Modified Newton Method Performance on the Extended Powell Function (n = 10000)

Figures 13a–13b confirm this picture. At higher dimension, more finite-difference runs reduce the gradient norm below 1, but dispersion increases: forward and central variants tend to stagnate on average, while adaptive backward performs best together with adaptive central and backward itself, all showing a similar pattern around the 30th iteration. Perturbation size again proves decisive: $h = 10^{-6}$ and $h = 10^{-4}$ yield the steepest descent, while very small steps remain stuck.

Finally, the $n = 10^4$ case shows a slightly higher success rate (34/74 vs. 28/74 for $n = 10^3$), suggesting that despite the increased cost, the Modified Newton method exhibits more robust median performance in larger settings.

5.2.2 Truncated Newton Method

Global summary.

Table 13 provides an overview of the results. At $n = 10^3$ the solver converged in 28/148 runs (18.9%), with a median of 14 iterations and 6.2 s wall-clock time. At $n = 10^4$ the success rate increased to 44/148 (29.7%), with a median of 12 iterations and 17.1 s. At $n = 10^5$ the method essentially broke down, succeeding in only 4/148 runs (2.7%), with a median of 2 iterations and 45.0 s. The experimental order of convergence (EOC) remained close to the ideal quadratic regime at smaller scales, but turned strongly negative at $n = 10^5$, highlighting the instability and loss of Newton–like acceleration in very high dimensions.

Table 13: Extended Powell, Truncated Newton: global outcomes.

Dimension	Total runs	Successes	Fails	Median iters	Median time [s]	Median EOC (superlinear)	Median EOC (quadratic)
$n = 10^3$	148	28	120	14	6.16	0.982	0.993
$n = 10^4$	148	44	104	12	17.08	0.988	1.000
$n = 10^5$	148	4	144	2	44.95	-182.125	-182.125

We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme and expected rate of convergence. The experiments are also categorized into "fixed", using the assignment's prescribed starting point, and "sampled," reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Tables 14, 16 and 16.

Table 14: Extended Powell ($n = 10^3$), Truncated Newton: by derivative, preconditioning, and experiment type.

Derivative	Expected ROC	Exp	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	40.5	20.87	1.20e-01
adaptive_backward	fixed	superlinear	6	0	6	45.5	20.25	1.20e-01
adaptive_backward	sampled	quadratic	6	0	6	39.5	20.26	1.22e-01
adaptive_backward	sampled	superlinear	6	0	6	44.0	20.96	1.22e-01
adaptive_central	fixed	quadratic	6	2	4	14.0	20.21	8.28e-02
adaptive_central	fixed	superlinear	6	2	4	14.5	20.16	7.68e-02
adaptive_central	sampled	quadratic	6	2	4	14.0	20.16	8.53e-02
adaptive_central	sampled	superlinear	6	2	4	14.5	20.04	8.01e-02
adaptive_forward	fixed	quadratic	6	1	5	47.5	20.18	1.11e-01
adaptive_forward	fixed	superlinear	6	1	5	48.0	20.17	1.11e-01
adaptive_forward	sampled	quadratic	6	1	5	44.0	20.14	1.17e-01
adaptive_forward	sampled	superlinear	6	1	5	45.0	20.13	1.17e-01
backward	fixed	quadratic	6	0	6	33.5	20.67	1.25e-01
backward	fixed	superlinear	6	0	6	40.0	20.59	8.70e-02
backward	sampled	quadratic	6	0	6	34.0	20.59	1.22e-01
backward	sampled	superlinear	6	0	6	40.5	20.57	8.62e-02
central	fixed	quadratic	6	2	4	14.0	20.23	6.46e-02
central	fixed	superlinear	6	2	4	14.5	20.09	6.24e-02
central	sampled	quadratic	6	2	4	14.0	20.17	6.46e-02
central	sampled	superlinear	6	2	4	14.5	20.17	6.24e-02
exact	fixed	quadratic	1	1	0	24.0	0.08	2.84e-06
exact	fixed	superlinear	1	1	0	24.0	0.07	3.64e-06
exact	sampled	quadratic	1	1	0	24.0	0.08	2.84e-06
exact	sampled	superlinear	1	1	0	24.0	0.08	3.64e-06
forward	fixed	quadratic	6	1	5	21.5	21.15	1.14e-01
forward	fixed	superlinear	6	1	5	40.5	20.20	6.86e-02
forward	sampled	quadratic	6	1	5	21.5	20.92	1.14e-01
forward	sampled	superlinear	6	1	5	40.5	20.17	6.86e-02

Table 15: Extended Powell ($n = 10^4$), Truncated Newton: by derivative, preconditioning, and experiment type.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	1	5	76.5	79.81	7.34e-02
adaptive_backward	fixed	superlinear	6	1	5	107.5	79.90	6.25e-02
adaptive_backward	sampled	quadratic	6	1	5	76.5	79.84	6.85e-02
adaptive_backward	sampled	superlinear	6	1	5	109.5	79.92	6.77e-02
adaptive_central	fixed	quadratic	6	2	4	19.5	80.08	2.23e-01
adaptive_central	fixed	superlinear	6	2	4	19.5	79.88	2.23e-01
adaptive_central	sampled	quadratic	6	2	4	19.5	79.88	2.26e-01
adaptive_central	sampled	superlinear	6	2	4	19.5	80.08	2.23e-01
adaptive_forward	fixed	quadratic	6	1	5	97.0	79.79	1.01e-01
adaptive_forward	fixed	superlinear	6	1	5	80.5	79.71	1.01e-01
adaptive_forward	sampled	quadratic	6	1	5	96.0	79.70	1.01e-01
adaptive_forward	sampled	superlinear	6	1	5	82.0	79.95	9.98e-02
backward	fixed	quadratic	6	1	5	60.0	81.55	5.64e-02
backward	fixed	superlinear	6	2	4	57.5	79.82	6.35e-02
backward	sampled	quadratic	6	1	5	59.5	79.97	5.64e-02
backward	sampled	superlinear	6	2	4	57.5	79.84	6.03e-02
central	fixed	quadratic	6	2	4	21.0	79.78	2.01e-01
central	fixed	superlinear	6	2	4	21.0	80.06	1.99e-01
central	sampled	quadratic	6	2	4	21.0	79.83	1.90e-01
central	sampled	superlinear	6	2	4	21.0	80.27	1.90e-01
exact	fixed	quadratic	1	1	0	21.0	0.75	8.68e-06
exact	fixed	superlinear	1	1	0	22.0	0.71	1.86e-06
exact	sampled	quadratic	1	1	0	21.0	0.70	8.68e-06
exact	sampled	superlinear	1	1	0	22.0	0.85	1.86e-06
forward	fixed	quadratic	6	2	4	36.0	79.69	3.00e-02
forward	fixed	superlinear	6	2	4	36.5	79.72	3.13e-02
forward	sampled	quadratic	6	2	4	35.5	79.78	3.16e-02
forward	sampled	superlinear	6	2	4	36.5	79.65	3.08e-02

Table 16: Extended Powell ($n = 10^5$), Truncated Newton: by derivative, preconditioning, and experiment type.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	9.5	315.69	4.64e-01
adaptive_backward	fixed	superlinear	6	0	6	10.5	314.91	4.41e-01
adaptive_backward	sampled	quadratic	6	0	6	9.5	314.25	4.65e-01
adaptive_backward	sampled	superlinear	6	0	6	10.5	312.06	4.41e-01
adaptive_central	fixed	quadratic	6	0	6	3.5	319.25	2.15e-01
adaptive_central	fixed	superlinear	6	0	6	5.0	315.15	2.42e-01
adaptive_central	sampled	quadratic	6	0	6	4.0	315.19	2.42e-01
adaptive_central	sampled	superlinear	6	0	6	5.0	325.33	2.99e-01
adaptive_forward	fixed	quadratic	6	0	6	6.0	312.44	5.53e-01
adaptive_forward	fixed	superlinear	6	0	6	7.5	312.70	5.33e-01
adaptive_forward	sampled	quadratic	6	0	6	6.5	313.92	6.05e-01
adaptive_forward	sampled	superlinear	6	0	6	7.5	308.43	5.33e-01
backward	fixed	quadratic	6	1	5	5.0	314.40	8.28e-02
backward	fixed	superlinear	6	1	5	6.5	308.23	7.68e-02
backward	sampled	quadratic	6	1	5	5.0	312.29	8.11e-02
backward	sampled	superlinear	6	1	5	6.5	308.12	7.60e-02
central	fixed	quadratic	6	0	6	4.5	325.48	4.82e-01
central	fixed	superlinear	6	0	6	5.0	316.03	4.38e-01
central	sampled	quadratic	6	0	6	4.5	315.11	4.38e-01
central	sampled	superlinear	6	0	6	5.0	327.11	4.82e-01
exact	fixed	quadratic	1	0	1	695.0	300.08	3.63e+04
exact	fixed	superlinear	1	0	1	718.0	299.57	3.63e+04
exact	sampled	quadratic	1	0	1	703.0	300.04	3.63e+04
exact	sampled	superlinear	1	0	1	704.0	300.18	3.63e+04
forward	fixed	quadratic	6	0	6	9.0	313.70	9.46e-02
forward	fixed	superlinear	6	0	6	10.5	315.33	6.39e-02
forward	sampled	quadratic	6	0	6	9.5	315.35	9.47e-02
forward	sampled	superlinear	6	0	6	10.5	314.73	6.39e-02

Analysis and Observations.

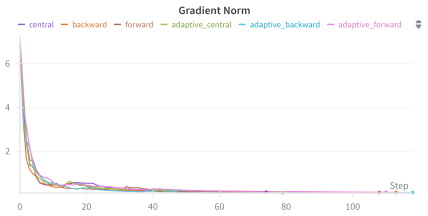
For the Extended Powell function, the Truncated Newton method shows a generally favourable behaviour, though robustness declines at larger scales.

At $n = 10^3$, 28 runs succeed out of 148, with median iteration counts around 14 and runtimes close to 6 s (Table 14). Exact derivatives converge consis-

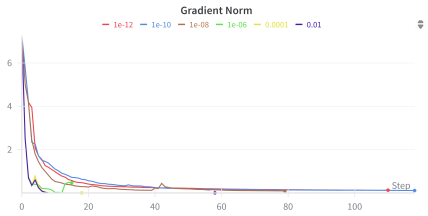
tently in about 24 iterations, reducing $\|\nabla f\|$ to 10^{-6} within one second. Finite–difference methods also perform well: central and adaptive central achieve residuals near 10^{-1}, forward and adaptive forward succeed less often but remain competitive, while backward and adaptive backward stagnate.

Figure 14 confirms these patterns: all schemes show a steep initial drop followed by gradual decay, with perturbations behaving similarly and $h = 10^{-2}, 10^{-4}, 10^{-6}$ producing the sharpest early reductions.

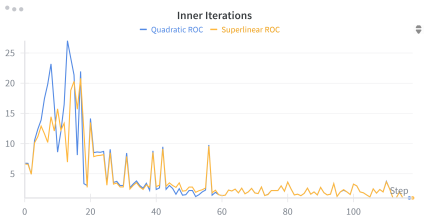
Quadratic ROC tolerances induce slightly higher CG iterations than superlinear in the early stages, but both converge to similar levels. Execution times appear irregular, with two peaks between 7 and 11 s, likely reflecting fluctuations in CG convergence thresholds.



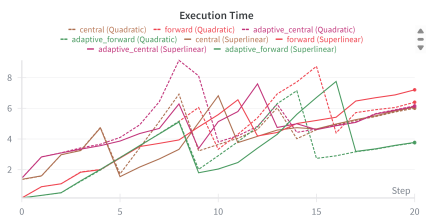
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).



(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 14: Truncated Newton Method Performance on the Extended Powell Function ($n = 1000$).

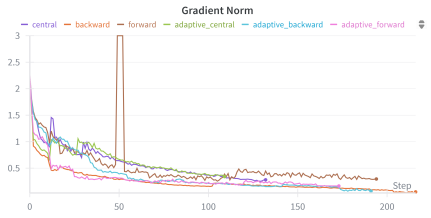
At $n = 10^4$, the number of successes rises to 44 (Table 15). Exact derivatives again converge within one second, while FD schemes diverge in accuracy: forward and backward reach the lowest residuals ($\sim 3 \cdot 10^{-2}$–$6 \cdot 10^{-2}$), central and adaptive central stall higher ($\sim 2 \cdot 10^{-1}$), while adaptive forward/backward remain less reliable.

Figure 15 highlights an anomalous peak of the forward scheme exceeding 3.5×10^4 (due to $h = 10^{-10}$, also visible in panel 15b), inflating the average trajectory and giving the impression of stagnation. Adaptive central also interrupts earlier at higher levels, whereas other schemes descend more smoothly.

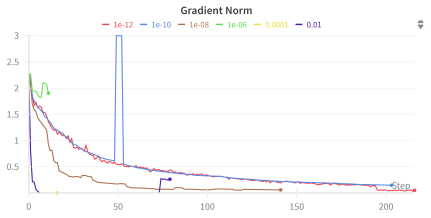
Among perturbations, $h = 10^{-10}$ yields an impressively smooth curve except

for the isolated spike, $h = 10^{-2}$ performs surprisingly well despite a final rise, while $h = 10^{-6}$ stalls after few steps, likely due to round-off issues.

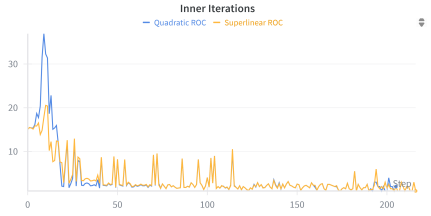
CG iterations (panel 15c) again show quadratic tolerances initially above superlinear but converging later, while execution times (panel 15d) penalize two–evaluation schemes.



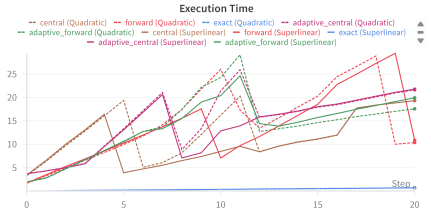
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



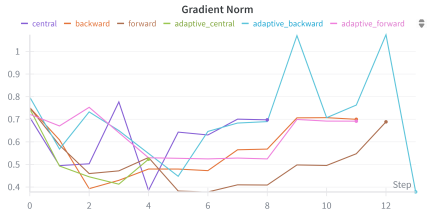
(c) Inner Iterations (Averaged by Rate of Convergence).



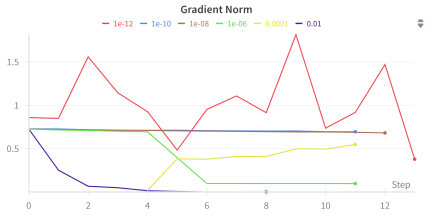
(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 15: Truncated Newton Method Performance on the Extended Powell Function ($n = 10000$). *Note:* Figures a and b truncated at 3 along the y-axis for clarity.

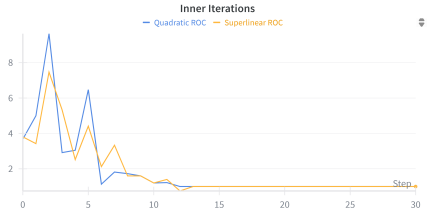
At $n = 10^5$, performance collapses, with only 4 successful runs out of 148 (Table 16). Iterations drop to 2 but runtimes saturate the 300 s budget. FD schemes stagnate between 10^{-1} and 10^0, with forward and backward following nearly identical paths.



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).

(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 16: Truncated Newton Method Performance on the Extended Powell Function (n = 100000). *Note*: graph on Figure d has been truncated at 20 along x-axis, exact reaches 300s after 700 iterations.

Even the exact derivative fails: despite hundreds of iterations, the gradient norm decreases by only 0.01 (Figure 17), showing the severe ill-conditioning of the starting point.

Figure 16 illustrates the confused trajectories of derivative methods, while panel 16b confirms that perturbations play little role, except for the poor behaviour of $h = 10^{-12}$.

Execution times scale nearly linearly for the exact derivative, while quadratic ROC runs are often as fast or faster than superlinear ones.

CG behaviour (Figure 16c) shows quadratic tolerances higher at the start, then alternating with superlinear as iterations progress, with neither clearly dominant.

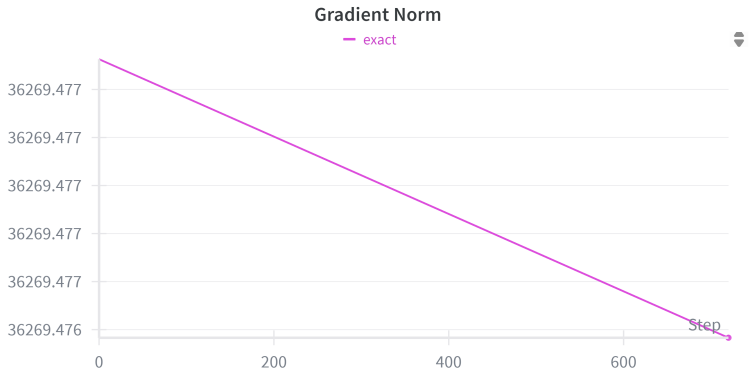


Figure 17: Derivative behaviour of the Truncated Newton Method on the Extended Powell function with 10^5 variables.

5.3 Broyden Tridiagonal Function

The Broyden tridiagonal function is used as defined in Appendix D, following [0]. Each residual couples one variable with its immediate neighbors, leading to a narrow tridiagonal interaction structure.

Sparse Hessian (Modified Newton).

Two complementary constructions are employed to obtain a sparse second–order model:

1. **Exact sparse Hessian.** Writing $F(x) = \frac{1}{2}\|f(x)\|_2^2$ with Jacobian $J(x)$ of the residual vector $f(x)$, the Hessian splits as

$$\nabla^2 F(x) \;=\; J(x)^\top J(x) \;+\; \sum_{k=1}^{n} f_k(x)\,\nabla^2 f_k(x).$$

   For the Broyden tridiagonal residuals, each f_k depends only on (x_{k-1}, x_k, x_{k+1}), so $J(x)$ is tridiagonal and $J(x)^\top J(x)$ is penta-diagonal. The nonlinear correction $\nabla^2 f_k(x)$ contributes only to the diagonal, so the resulting $\nabla^2 F(x)$ is assembled in sparse CSR format as a symmetric penta-diagonal matrix.

2. **Finite–difference (approximated) sparse Hessian.** When exact second-order information is not used, the Hessian is approximated from finite differences of the gradient. To exploit the known bandwidth 2, a distance-2 coloring of the index set (5-coloring) is adopted so that no two variables closer than two indices are perturbed simultaneously. This ensures that only diagonal, first, and second off-diagonal entries are filled. Forward, backward, or central schemes are applied, with either a scalar step h or an adaptive per-coordinate step $h_i = |x_i|\,h$. The nonzeros are then directly inserted into a sparse penta-diagonal matrix (CSR).

Hessian–Vector Products (Truncated Newton).

The truncated Newton solver requires only Hessian–vector products Hv:

1. **Structured Hessian–vector products.** Exploiting the tridiagonal residual structure, the product is computed as

$$Hv \;=\; J(x)^\top (J(x)\, v) \;+\; \sum_{k=1}^{n} f_k(x)\, (\nabla^2 f_k(x)\, v),$$

   where $J(x)$ is tridiagonal and $\nabla^2 f_k(x)$ contributes only a diagonal correction. This computation involves only sparse operations within the pentadiagonal structure, and Hv is obtained without explicitly forming $\nabla^2 F(x)$.

2. **Finite–difference products (matrix–free).** When exact second–order information is unavailable, Hv is approximated via directional finite differences of the gradient, following the strategy described in paragraph 3.1. This matrix-free approach integrates naturally with conjugate-gradient iterations and negative-curvature detection.

5.3.1 Modified Newton Method

Global summary.

Table 17 reports success rates, iteration counts, execution times, and convergence rates. At $n = 10^3$, the method succeeded in only 28/148 runs (18.9%), with a median of 11 iterations and 6.5 s wall-clock time. At $n = 10^4$, the success rate was comparable (14/74, i.e. 18.9%), with a median of 12 iterations and 49.09 s. The experimental order of convergence (EOC) was close to quadratic (≈ 0.99) in all successful runs.

Table 17: Broyden tridiagonal, Modified Newton: global outcomes.

Dimension	Total runs	Successes	Failures	Median iters	Median time [s]	Median EOC
$n = 10^3$	74	14	60	11.0	4.71	0.99
$n = 10^4$	74	14	60	12.0	49.09	0.99
$n = 10^5$	–	–	–	–	–	–

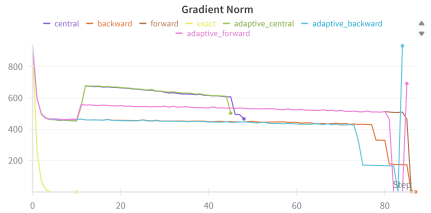
We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme. The experiments are also categorized into "fixed", using the assignment's prescribed starting point, and "sampled," reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Tables 18 and 19.

Table 18: Broyden tridiagonal ($n = 10^3$), Modified Newton: by derivative and experiment type.

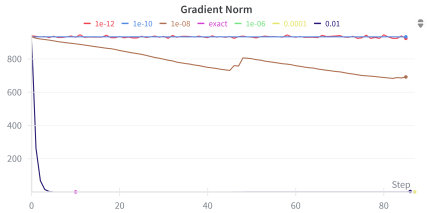
Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	0	6	83.0	20.10	3.31e+02
adaptive_backward	sampled	6	0	6	78.0	20.09	3.39e+02
adaptive_central	fixed	6	2	4	45.5	20.11	2.83e+02
adaptive_central	sampled	6	2	4	43.5	20.09	2.92e+02
adaptive_forward	fixed	6	1	5	82.0	20.11	3.43e+02
adaptive_forward	sampled	6	1	5	78.0	20.05	3.48e+02
backward	fixed	6	0	6	86.5	20.13	3.45e+02
backward	sampled	6	0	6	80.0	20.11	3.57e+02
central	fixed	6	2	4	47.0	20.09	2.71e+02
central	sampled	6	2	4	45.0	20.16	2.79e+02
exact	fixed	1	1	0	11.0	0.16	7.06e-06
exact	sampled	1	1	0	11.0	0.17	7.06e-06
forward	fixed	6	1	5	86.0	20.07	3.37e+02
forward	sampled	6	1	5	80.0	20.18	3.44e+02

Table 19: Broyden tridiagonal ($n = 10^4$), Modified Newton: by derivative and experiment type.

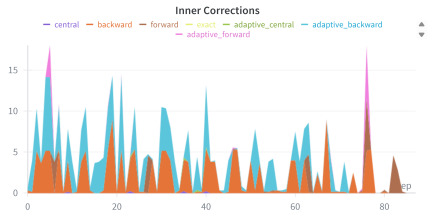
Derivative	Exp.	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	6	0	6	25.5	81.62	1.46e+03
adaptive_backward	sampled	6	0	6	25.5	81.05	1.46e+03
adaptive_central	fixed	6	2	4	17.0	80.25	1.45e+03
adaptive_central	sampled	6	2	4	17.0	79.85	1.45e+03
adaptive_forward	fixed	6	1	5	21.0	81.01	1.46e+03
adaptive_forward	sampled	6	1	5	20.5	80.16	1.46e+03
backward	fixed	6	0	6	25.5	81.29	1.46e+03
backward	sampled	6	0	6	25.5	79.86	1.46e+03
central	fixed	6	2	4	17.5	80.94	1.45e+03
central	sampled	6	2	4	18.0	81.82	1.45e+03
exact	fixed	1	1	0	12.0	18.14	3.51e-06
exact	sampled	1	1	0	12.0	18.17	3.51e-06
forward	fixed	6	1	5	20.0	80.17	1.46e+03
forward	sampled	6	1	5	21.0	80.95	1.46e+03



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Corrections (Averaged by Derivative Method).

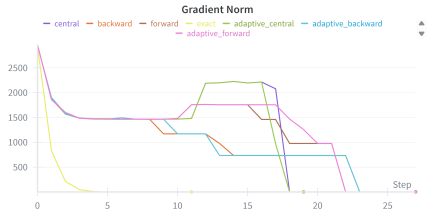
(d) Execution Time (One/Two Eval, No Precond/Precond).

Figure 18: Modified Newton Method Performance on the Broyden Tridiagonal Function (n = 1000)

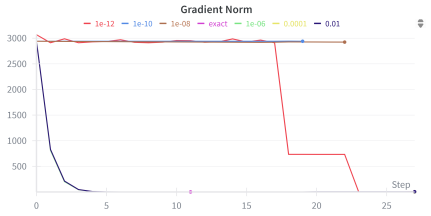
Analysis of Results.

For the Broyden tridiagonal function, the Modified Newton method reveals a clear gap between exact and finite–difference (FD) gradients. Exact derivatives converge rapidly and stably, reducing the gradient norm to 10^{-6} within 11 iterations (Table 18), while FD schemes stagnate around 10^2–10^3 and never approach true stationarity. This several–orders–of–magnitude difference highlights the decisive role of accurate derivatives.

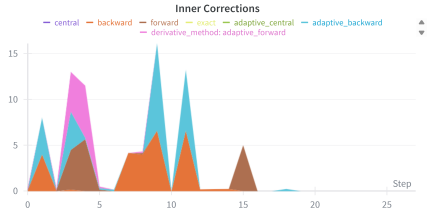
Figure 18b shows the effect of perturbation size. Only $h = 10^{-2}$ and $h = 10^{-6}$ produce trajectories close to the exact case, while other values (10^{-12}, 10^{-10}, 10^{-8}, 10^{-4}) stagnate almost immediately. The inner corrections (Figure 18c) reflect this instability: central differences are somewhat steadier, adaptive variants fluctuate irregularly, and all remain confined to high residual norms.



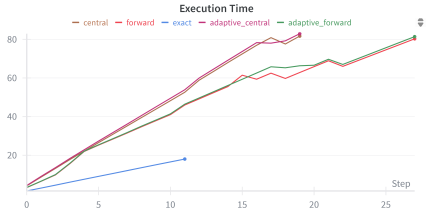
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Corrections (Averaged by Derivative Method).



(d) Execution Time (One/Two Eval, No Precond/Precond).

Figure 19: Modified Newton Method Performance on the Broyden Tridiagonal Function (n = 10000).

In terms of efficiency, exact derivatives converge in under one second, while FD methods require 30–50 iterations and saturate around 20 s.

At $n = 10^4$, the same behaviour persists (Figure 19, Table 19): exact derivatives converge in 12 iterations, reaching $\|\nabla f\| \approx 10^{-6}$ in 18 s, whereas FD methods remain near 1.4×10^3. Again, only $h = 10^{-2}$ and $h = 10^{-6}$ provide descent curves close to the exact case, while others stagnate. A peculiar behaviour is observed for the perturbation $h = 10^{-12}$: the averaged curve initially stagnates, in line with most other perturbations, but later exhibits a sudden steep descent toward the smallest gradient norms. This irregular shape is likely the result of an outlier run, whose contribution disproportionately affects the averaged trend.

From Figure 18a, it can also be seen that at $n = 10^4$ a larger set of FD runs reach lower gradient norms compared with $n = 10^3$, though the overall success rate remains low and most runs fail to converge.

Across both problem sizes, when convergence occurs the experimental order of convergence is about 0.99, consistent with the quasi–quadratic rate of the Modified Newton method. Backward and adaptive backward FD schemes never succeed in either dimension, confirming their unsuitability for this problem.

5.3.2 Truncated Newton Method

Global summary.

Table 20 provides an overview of the results. At $n = 10^3$ the solver converged in 45/148 runs (30.4%), with a median of 11 iterations and 3.1 s wall-clock time. At $n = 10^4$ the success rate was nearly identical, 44/148 runs (29.7%), with a median of 14 iterations and 19.4 s. At $n = 10^5$ robustness dropped sharply, with only 4/148 runs (2.7%) succeeding, median iterations remaining at 14, and runtimes increasing to 24.5 s. The experimental order of convergence (EOC) was above unity at lower scales (1.27–1.37), indicating accelerated progress, but collapsed to 1.000 in both regimes at $n = 10^5$, suggesting essentially linear convergence when success was achieved.

Table 20: Broyden tridiagonal, Truncated Newton: global outcomes.

Dimension	Total runs	Successes	Fails	Median iters	Median time [s]	Median EOC (superlinear)	Median EOC (quadratic)
$n = 10^3$	148	45	103	11	3.09	1.365	1.277
$n = 10^4$	148	44	104	14	19.41	1.150	1.253
$n = 10^5$	148	4	144	14	24.48	1.000	1.000

We also report success/failure, median iterations, time, and final gradient norm grouped by derivative scheme and expected rate of convergence. The experiments are also categorized into "fixed", using the assignment's prescribed starting point, and "sampled," reporting aggregate performance over 10 randomly chosen initial points. These finer details are shown in Tables 21, 22 and 23.

Table 21: Broyden tridiagonal (n = 1000), Truncated Newton: by derivative, expected ROC and experiment type.

Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	38.0	20.09	2.99e-01
adaptive_backward	fixed	superlinear	6	0	6	65.0	20.09	2.72e-01
adaptive_backward	sampled	quadratic	6	0	6	38.5	20.09	2.23e-01
adaptive_backward	sampled	superlinear	6	0	6	64.5	20.08	2.57e-01
adaptive_central	fixed	quadratic	6	2	4	12.0	20.06	1.36e-03
adaptive_central	fixed	superlinear	6	2	4	13.0	20.06	6.98e-03
adaptive_central	sampled	quadratic	6	2	4	12.0	20.09	1.91e-03
adaptive_central	sampled	superlinear	6	2	4	13.0	20.01	6.98e-03
adaptive_forward	fixed	quadratic	6	3	3	11.5	13.46	1.59e-02
adaptive_forward	fixed	superlinear	6	3	3	11.5	11.35	1.59e-02
adaptive_forward	sampled	quadratic	6	3	3	11.5	13.47	1.59e-02
adaptive_forward	sampled	superlinear	6	3	3	11.5	11.40	1.59e-02
backward	fixed	quadratic	6	0	6	38.5	20.07	9.90e-01
backward	fixed	superlinear	6	0	6	74.0	20.10	9.90e-01
backward	sampled	quadratic	6	0	6	39.0	20.08	9.93e-01
backward	sampled	superlinear	6	0	6	79.0	20.08	9.90e-01
central	fixed	quadratic	6	2	4	12.0	20.08	2.54e-03
central	fixed	superlinear	6	3	3	13.0	19.88	6.97e-03
central	sampled	quadratic	6	2	4	12.0	20.05	2.54e-03
central	sampled	superlinear	6	2	4	13.0	20.08	3.75e-03
exact	fixed	quadratic	1	1	0	12.0	0.19	1.92e-06
exact	fixed	superlinear	1	1	0	12.0	0.16	2.56e-06
exact	sampled	quadratic	1	1	0	12.0	0.19	1.92e-06
exact	sampled	superlinear	1	1	0	12.0	0.13	2.56e-06
forward	fixed	quadratic	6	3	3	12.0	13.23	3.28e-02
forward	fixed	superlinear	6	3	3	12.5	11.30	3.28e-02
forward	sampled	quadratic	6	3	3	12.0	13.34	4.37e-02
forward	sampled	superlinear	6	3	3	12.5	11.34	4.03e-02

Table 22: Broyden tridiagonal (n = 10000), Truncated Newton: by derivative, expected ROC and experiment type.

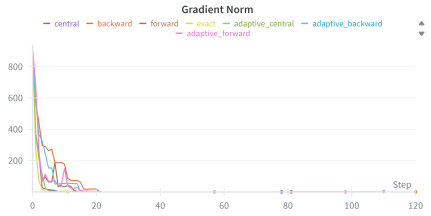
Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	42.0	80.21	9.81e-01
adaptive_backward	fixed	superlinear	6	0	6	47.0	80.04	7.53e-01
adaptive_backward	sampled	quadratic	6	0	6	41.5	80.18	6.12e-01
adaptive_backward	sampled	superlinear	6	0	6	47.5	80.10	8.85e-01
adaptive_central	fixed	quadratic	6	2	4	13.5	80.07	1.29e+00
adaptive_central	fixed	superlinear	6	2	4	13.5	80.19	1.29e+00
adaptive_central	sampled	quadratic	6	2	4	13.5	80.17	1.28e+00
adaptive_central	sampled	superlinear	6	2	4	13.5	80.08	1.30e+00
adaptive_forward	fixed	quadratic	6	3	3	14.5	59.68	4.57e-02
adaptive_forward	fixed	superlinear	6	3	3	14.5	48.67	1.82e-01
adaptive_forward	sampled	quadratic	6	3	3	14.0	60.05	4.57e-02
adaptive_forward	sampled	superlinear	6	3	3	14.0	49.21	4.57e-02
backward	fixed	quadratic	6	0	6	35.0	80.17	1.28e+00
backward	fixed	superlinear	6	0	6	49.5	80.17	1.23e+00
backward	sampled	quadratic	6	0	6	35.0	80.15	1.15e+00
backward	sampled	superlinear	6	0	6	49.5	80.14	1.19e+00
central	fixed	quadratic	6	2	4	13.5	80.15	1.02e-01
central	fixed	superlinear	6	2	4	13.5	80.37	1.00e-01
central	sampled	quadratic	6	2	4	13.5	80.19	1.00e-01
central	sampled	superlinear	6	2	4	13.5	80.11	1.00e-01
exact	fixed	quadratic	1	1	0	14.0	1.82	3.92e-06
exact	fixed	superlinear	1	1	0	14.0	1.59	4.19e-06
exact	sampled	quadratic	1	1	0	14.0	1.90	3.92e-06
exact	sampled	superlinear	1	1	0	14.0	1.68	4.19e-06
forward	fixed	quadratic	6	3	3	14.5	58.87	3.80e-02
forward	fixed	superlinear	6	3	3	15.0	48.94	3.80e-02
forward	sampled	quadratic	6	3	3	14.5	59.55	3.80e-02
forward	sampled	superlinear	6	3	3	14.5	48.96	3.81e-02

Table 23: Broyden tridiagonal (n = 100000), Truncated Newton: by derivative, expected ROC and experiment type.

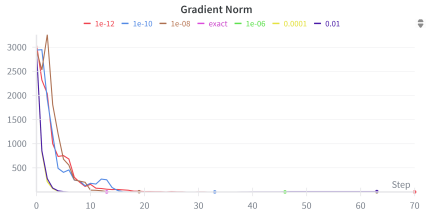
Derivative	Exp.	Expected ROC	Runs	Succ	Fail	Iter_{50}	Time_{50} [s]	$\|\nabla f\|_{50}$
adaptive_backward	fixed	quadratic	6	0	6	4.0	351.19	1.91e+02
adaptive_backward	fixed	superlinear	6	0	6	4.0	362.68	1.91e+02
adaptive_backward	sampled	quadratic	6	0	6	4.0	331.45	1.91e+02
adaptive_backward	sampled	superlinear	6	0	6	4.0	343.16	1.91e+02
adaptive_central	fixed	quadratic	6	0	6	3.0	363.48	3.48e+02
adaptive_central	fixed	superlinear	6	0	6	3.0	352.41	6.66e+02
adaptive_central	sampled	quadratic	6	0	6	3.0	352.63	3.48e+02
adaptive_central	sampled	superlinear	6	0	6	3.0	368.31	3.48e+02
adaptive_forward	fixed	quadratic	6	0	6	3.5	351.77	3.34e+02
adaptive_forward	fixed	superlinear	6	0	6	3.5	350.21	3.37e+02
adaptive_forward	sampled	quadratic	6	0	6	4.0	330.06	2.76e+02
adaptive_forward	sampled	superlinear	6	0	6	4.0	336.58	2.76e+02
backward	fixed	quadratic	6	0	6	4.0	327.73	1.38e+02
backward	fixed	superlinear	6	0	6	4.0	354.26	1.61e+02
backward	sampled	quadratic	6	0	6	4.5	329.70	1.38e+02
backward	sampled	superlinear	6	0	6	4.0	323.65	1.38e+02
central	fixed	quadratic	6	0	6	3.0	382.32	2.13e+02
central	fixed	superlinear	6	0	6	3.0	363.40	6.66e+02
central	sampled	quadratic	6	0	6	3.0	352.16	2.13e+02
central	sampled	superlinear	6	0	6	3.0	355.91	2.13e+02
exact	fixed	quadratic	1	1	0	14.0	26.49	4.29e-06
exact	fixed	superlinear	1	1	0	14.0	22.16	9.85e-06
exact	sampled	quadratic	1	1	0	14.0	26.24	4.29e-06
exact	sampled	superlinear	1	1	0	14.0	22.72	9.85e-06
forward	fixed	quadratic	6	0	6	4.0	352.29	2.80e+02
forward	fixed	superlinear	6	0	6	3.5	362.89	4.25e+02
forward	sampled	quadratic	6	0	6	4.0	337.00	2.76e+02
forward	sampled	superlinear	6	0	6	4.0	349.67	2.80e+02

Analysis and Observations.

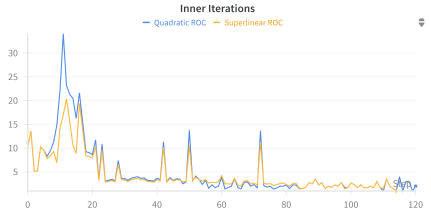
At $n = 10^3$, the detailed breakdown (Table 20) confirms the strong performance of exact derivatives, converging in 12 iterations within 0.2 s, and reducing $\|\nabla f\|$ to 10^{-6}.



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).

(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 20: Truncated Newton Method Performance on the Broyden Tridiagonal Function (n = 1000).

Forward and adaptive forward differences succeed in about half of the runs, stabilising at residuals near 10^{-2}. Central and adaptive central also perform well, with similar iteration counts and residuals around 10^{-3} to 10^{-2}. By contrast, backward and adaptive backward consistently stagnate close to 10^{0}, failing to converge.

Figure 20 supports these findings: panel 20a shows that all methods reduce the gradient norm steeply in the first 20 iterations but then stagnate, while panel 20b illustrates that perturbation size has little effect, except for $h = 10^{-2}$ which closely matches the exact trajectory and reaches low norms almost simultaneously.

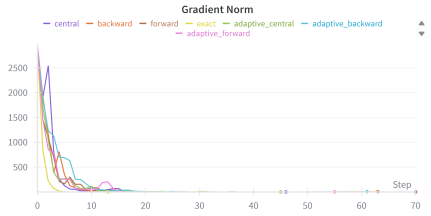
Panel 20c highlights that quadratic tolerances consistently require more inner CG iterations than superlinear ones, and panel 20d reveals the counter–intuitive result that two–evaluation schemes are faster than single–evaluation ones, though exact derivatives remain by far the most efficient.

At $n = 10^4$, the behaviour remains very similar. Exact derivatives converge in 14 iterations within 1.6–1.9 s, reducing the gradient norm to 10^{-6}. Among finite–difference schemes, forward and adaptive forward prove the most effective, with residuals of the order of 10^{-2}, while central and adaptive central succeed less frequently and typically stabilise near 10^{-1}. Backward and adaptive backward again fail consistently, stagnating around unity.

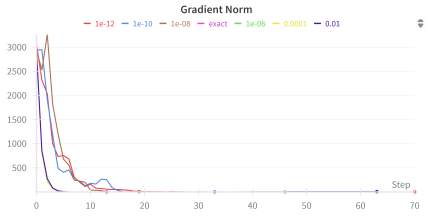
The graphical evidence in Figure 19 mirrors the $n = 10^3$ case: none of the FD methods clearly outperform the others, and perturbation trends are almost

identical to the smaller dimension, with $h = 10^{-2}$ still the best compromise. Quadratic and superlinear tolerances produce almost the same number of inner iterations (panel 21c), while execution times (panel 21d) show that forward and adaptive forward achieve the most efficient timings after the exact, both peaking around 15 s.

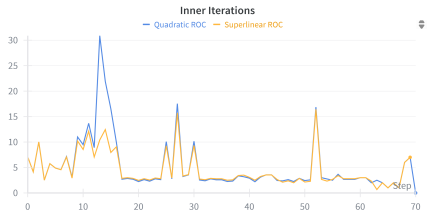
At $n = 10^5$, the situation deteriorates sharply. The tabular breakdown (Table 23) shows that only exact derivatives succeed, converging in 14 iterations within 20–26 s and reducing $\|\nabla f\|$ to 10^{-6}.



(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).

(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

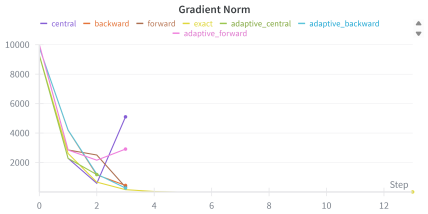
Figure 21: Truncated Newton Method Performance on the Broyden Tridiagonal Function (n = 10000).

All finite–difference methods terminate after just 3–4 iterations, saturating the 300 s budget and stagnating at residuals of 10^2–10^3. As shown in Figure 22, panel 22a highlights the enormous gap in iterations between exact and FD schemes, with the latter limited by poor derivative quality and the resulting slowdown of the inner CG solver. Nevertheless, FD methods still manage a steep initial decrease, suggesting that with more allowed steps they might have converged.

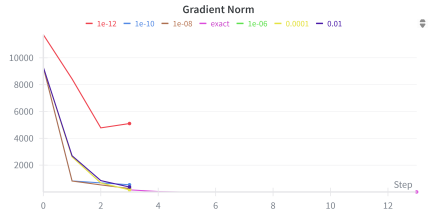
Perturbation analysis in panel 22b reveals that $h = 10^{-8}$ nearly replicates the exact derivative curve, while $h = 10^{-12}$ fails to reduce the norm significantly from the outset.

Panel 22c shows quadratic inner iterations exceeding superlinear beyond the 7th step — a behaviour driven by the stricter tolerance implied by $\|\nabla f(x)\|$ when the gradient norm is already below one, a situation reached only by exact

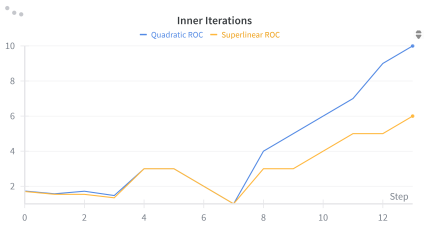
derivatives. Finally, execution times in panel 22d demonstrate the inefficiency of two–evaluation methods (central and adaptive central), which are dramatically slower than single–evaluation schemes. Quadratic and superlinear tolerances yield essentially the same timings, with the exact derivative scaling almost linearly with iteration count.



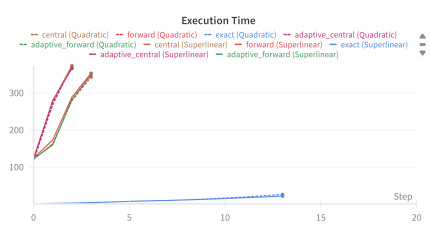
(a) Gradient norm vs. iteration (Averaged by Derivative Method).



(b) Gradient norm vs. iteration (Averaged by Perturbation).



(c) Inner Iterations (Averaged by Rate of Convergence).



(d) Execution Time (One/Two Eval, Superlinear/Quadratic).

Figure 22: Truncated Newton Method Performance on the Broyden Tridiagonal Function (n = 100000).

6 Conclusion

The comparative evidence points to a clear division of labour between the two solvers. *Modified Newton* leverages a (stabilised) Hessian and, when derivatives are computed accurately and perturbations are well tuned, delivers higher fidelity steps and smoother, linear local behaviour. *Truncated Newton* (Newton–CG) relies on Hessian–vector products and truncates the inner linear solves, which reduces per–iteration cost and memory pressure at the expense of step accuracy. This algorithmic contrast explains the performance envelope observed across problems and scales.

A hard scalability boundary emerges for Modified Newton. At $n = 10^5$ the method does not start because the approximate Hessian cannot be stored within RAM, even when exploiting sparsity with graph coloring; the memory footprint and factorisation overhead dominate. In this regime, Truncated Newton is viable precisely because it avoids assembling or storing the Hessian and

only requires Hessian–vector products, making it far more economical in memory and compute per iteration under the same resource constraints.

At small and moderate dimensions, Modified Newton is typically more precise and often more iteration–efficient. On the Extended Rosenbrock, for instance, it reaches low residuals in $\sim$22 steps with exact derivatives (Table 4), whereas Truncated Newton needs many more outer steps to compensate for the truncated inner solves (Table 6). This trade–off is clearly visible in our results: at $n = 10^3$ the Modified Newton method is systematically faster thanks to its much lower iteration counts, while at $n = 10^4$ the Truncated Newton method consistently overtakes it, as its cheaper per–iteration cost outweighs the higher number of steps.

However, Modified Newton is also markedly more sensitive to these hyperparameters: central vs. forward/backward schemes and the perturbation h can produce highly variable trajectories. By contrast, Truncated Newton tends to display more homogeneous aggregate curves across derivative schemes and h, reflecting greater robustness to such choices even when absolute accuracy is lower.

As the dimension grows, the balance shifts decisively toward Truncated Newton's lower per–iteration cost. A representative case is the Extended Rosenbrock at $n = 10^4$: the Modified Newton converges in about 21 iterations with a median runtime of 40.28 s, while the Truncated Newton requires 47 iterations but only 14.322 s, highlighting the trade–off between iteration count and per–iteration cost. The truncated solver needs more than twice as many outer steps (weaker steps due to inexact inner solves), but each step is much cheaper, so the total time is substantially lower. This pattern generalises: where Modified Newton remains iteration–efficient but memory– and factorisation–limited, Truncated Newton is iteration–hungry but time–efficient, and thereby competitive or superior in wall–clock performance at medium–to–large scale.

Convergence indicators corroborate these roles. The experimental order of convergence (EOC) for Modified Newton is relatively stable and often closer to the ideal linear regime at small/medium sizes, while Truncated Newton seldom approaches such rates and may exhibit very poor asymptotics when derivative noise and truncation interact unfavourably. An extreme case appears on the Extended Powell at $n = 10^5$, where the truncated variant's median EOC collapses to -182.125 in both expected regimes (Table 13), a symptom of steps dominated by inner–solve inaccuracy under tight resource budgets. Notwithstanding this, Truncated Newton can still be the only feasible solver at very large scales, whereas Modified Newton becomes inoperable due to memory constraints.

Overall, the methods are best viewed as complementary. When memory permits and accurate derivatives (or well–tuned finite differences) are available, Modified Newton is the method of choice, offering higher–quality steps, fewer outer iterations, and trajectories that can approach quadratic convergence. When scale or memory is the limiting factor, Truncated Newton becomes preferable: although its steps are less accurate and its EOC weaker, its matrix–free inner solves enable far shorter per–iteration times and competitive (often superior) wall–clock performance. In practice, a problem–driven selection emerges:

prioritise Modified Newton for precision at small/medium n; prefer Truncated Newton for scalability at large n, or whenever Hessian storage/factorisation is prohibitive.

A Assignment Instructions

For completeness, the original assignment instructions followed in this work are reported below.

> **Assignment on Unconstrained Optimization**
>
> 1. Implement exactly two out of the following numerical methods for unconstrained optimization:
>
> 1.1. Nelder-Mead [2 points]
>
> 1.2. Steepest descent method [0.5 points]
>
> 1.3. Nonlinear conjugate gradient method (Fletcher-Reeves or Polak-Ribière) [1 point]
>
> 1.4. Modified Newton method [2 points]
>
> 1.5. Inexact Newton method [0.5 points]
>
> 1.6. Truncated Newton method [2 points]
>
> In all cases (except Nelder-Mead), embed the method with a backtracking line search strategy.
>
> 2. Test your implementations on the Rosenbrock function
>
> $$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$
>
> with starting points $x_0 = (1.2, 1.2)$ and $x_0 = (-1.2, 1)$, reporting the behavior in both tables and figures. Impose a sufficient decrease condition for the backtracking strategy; e.g., using $\rho = 0.5$ and $c = 10^{-4}$, tuning if necessary.
>
> 3. Apply the codes to exactly three test problems taken from [1] following these instructions:
>
> - Set a random seed equal to the minimum student ID of the team members.
> - Study the problems for dimensions $n = 10d$, $d = 3, 4, 5$ (for Nelder-Mead also $n = 10, 25, 50$).
> - Use the starting point suggested in [1] and 10 additional randomly generated starting points within a hypercube around it.
> - Impose a sufficient decrease condition for the backtracking strategy, tuning parameters if needed.

- Apply the methods using both exact derivatives and finite differences with increments

$$h = 10^{-k}, \quad k = 2, 4, 6, 8, 10, 12,$$

or variable-specific increments

$$h_i = 10^{-k}|x_i|, \quad i = 1, \ldots, n.$$

- Perform a thorough comparison including: number of successful/failed runs, iterations to meet stopping criteria, experimental rate of convergence, execution time, etc.

Recommendations:

1. In case of inexact Newton method: use at least two different forcing terms and report inner/outer iterations.
2. When using an iterative solver for linear systems, report results with and without preconditioning.
3. Implement finite differences carefully to exploit problem structure for large dimensions (e.g., $n = 10^5$).

B Algorithms

B.1 Truncated Newton Method

Algorithm 1 Truncated Newton Method

Require: Initial point x_0

for $k = 0, 1, 2, \ldots$ **do**

Define tolerance

$$\eta_k = \min\left(\hat{\eta}, \|\nabla f(x_k)\|\right) \quad \text{or} \quad \eta_k = \min\left(0.5, \sqrt{\|\nabla f(x_k)\|}\right)$$

Set $z_0 \leftarrow 0$, $r_0 \leftarrow \nabla f(x_k)$, $d_0 \leftarrow -r_0$

for $j = 0, 1, 2, \ldots$ **do** ▷ Conjugate Gradient loop

if $d_j^T B_k d_j \leq 0$ **then** ▷ Negative curvature

if $j = 0$ **then**

return $p_k \leftarrow -\nabla f(x_k)$

else

return $p_k \leftarrow z_j$

end if

end if

$\alpha_j \leftarrow \frac{r_j^T r_j}{d_j^T B_k d_j}$

$z_{j+1} \leftarrow z_j + \alpha_j d_j$

$r_{j+1} \leftarrow r_j + \alpha_j B_k d_j$

if $\|r_{j+1}\| < \eta_k$ **then** ▷ Adaptive tolerance

return $p_k \leftarrow z_{j+1}$

end if

$\beta_{j+1} \leftarrow \frac{r_{j+1}^T r_{j+1}}{r_j^T r_j}$

$d_{j+1} \leftarrow -r_{j+1} + \beta_{j+1} d_j$

end for

Update

$$x_{k+1} \leftarrow x_k + \alpha_k p_k$$

where α_k satisfies Wolfe, Goldstein, or Armijo backtracking conditions (with $\alpha_k = 1$ if possible).

end for

B.2 Modified Newton Method

Algorithm 2 Modified Newton Method

Require: Initial point x_0, tolerance $\varepsilon > 0$

for $k = 0, 1, 2, \ldots$ **do**
  Compute gradient $g_k = \nabla f(x_k)$
  if $\|g_k\| < \varepsilon$ **then**
    return x_k ▷ Stopping criterion met
  end if
  Compute Hessian $H_k = \nabla^2 f(x_k)$ (exact or finite differences)
  Symmetrize $H_k \leftarrow \frac{1}{2}(H_k + H_k^\top)$
  Try Cholesky factorization $H_k = LL^\top$
  if factorization fails **then**
    Modify Hessian: $B_k \leftarrow H_k + \delta I$ with $\delta > 0$ adaptively increased
  else
    $B_k \leftarrow H_k$
  end if
  Solve linear system $B_k p_k = -g_k$
  if dimension small **then**
    Use direct Cholesky factorization
  end if
  Perform backtracking line search: find α_k such that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k g_k^\top p_k$$

  Update $x_{k+1} \leftarrow x_k + \alpha_k p_k$
end for

B.2.1 Diagonal Modification of the Hessian: Construction of B_k

To guarantee a positive definite (PD) system at each outer iteration, we modify the Hessian by an adaptive diagonal shift, often called *diagonal loading* or *Levenberg–Marquardt* regularization. The idea is to find the smallest shift $\delta \geq 0$ such that $B_k := H_k + \delta I \succ 0$, preserving as much curvature information as possible while ensuring a valid Newton direction.

Algorithm 3 Build_Bk: Cumulative-Shift Diagonal Modification (matches implementation)

Require: Hessian $hessf \in \mathbb{R}^{n \times n}$ (sparse or dense); growth factor $\gamma =$ `corr_fact` > 0; max trials $k_{\max}$
Ensure: Cholesky factor L, modified matrix B_k, number of attempts i

$\beta \leftarrow 10^{-3}$
if $hessf$ is sparse **then**
    $diag\_elements \leftarrow hessf$.diagonal()
    $H \leftarrow$ toarray($hessf$) ▷ convert to dense for Cholesky
else
    $diag\_elements \leftarrow \text{diag}(hessf)$
    $H \leftarrow hessf$
end if
if $\min(diag\_elements) > 0$ **then**
    $\tau_0 \leftarrow 0$
else
    $\tau_0 \leftarrow -\min(diag\_elements) + \beta$
end if
$\tau \leftarrow \tau_0$
$B_k \leftarrow H + \tau I$
$flag \leftarrow$ **false**
$i \leftarrow 0$
while $\neg flag$ **and** $i < k_{\max}$ **do**
    $\tau_{\text{new}} \leftarrow \max(\gamma\tau, \beta)$
    $B_k \leftarrow B_k + \tau_{\text{new}} I$ ▷ *cumulative* diagonal shift
    $\tau \leftarrow \tau_{\text{new}}$
    try $L \leftarrow \text{Cholesky}(B_k)$
        $flag \leftarrow$ **true and return** (L, B_k, i)
    catch LinAlgError $\Rightarrow$ $flag \leftarrow$ **false**
    $i \leftarrow i + 1$
end while
raise LinAlgError "Hessian can't be modified with $k_{\max}$"

C General Tools

C.1 Backtracking

Algorithm 4 Backtracking Line Search

Require: Search direction p_k, current point x_k, initial step $\bar{\alpha} > 0$, parameters $\rho \in (0,1)$, $c \in (0,1)$

Set $\alpha \leftarrow \bar{\alpha}$
while $f(x_k + \alpha p_k) > f(x_k) + c\alpha \nabla f_k^T p_k$ **do**
    $\alpha \leftarrow \rho\alpha$
end while
return $\alpha_k \leftarrow \alpha$

D Benchmark Problems

We report here the explicit definitions of the benchmark problems used in our experiments, following [0].

D.1 Extended Rosenbrock Function

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

with

$$f_k(x) = \begin{cases} 10\big(x_k^2 - x_{k+1}\big), & \text{if } \mathrm{mod}(k,2) = 1, \\ x_{k-1} - 1, & \text{if } \mathrm{mod}(k,2) = 0, \end{cases}$$

and suggested starting point

$$\bar{x}_l = \begin{cases} -1.2, & \mathrm{mod}(l,2) = 1, \\ 1.0, & \mathrm{mod}(l,2) = 0. \end{cases}$$

D.2 Extended Powell Singular Function

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

with

$$f_k(x) = \begin{cases} x_k + 10x_{k+1}, & \mathrm{mod}(k,4) = 1, \\ \sqrt{5}\,\big(x_{k+2} - x_{k+3}\big), & \mathrm{mod}(k,4) = 2, \\ (x_{k+1} - 2x_{k+2})^2, & \mathrm{mod}(k,4) = 3, \\ \sqrt{10}\,(x_k - x_{k+3})^2, & \mathrm{mod}(k,4) = 0, \end{cases}$$

and suggested starting point

$$\bar{x}_l = \begin{cases} 3, & \mathrm{mod}(l,4) = 1, \\ -1, & \mathrm{mod}(l,4) = 2, \\ 0, & \mathrm{mod}(l,4) = 3, \\ 1, & \mathrm{mod}(l,4) = 0. \end{cases}$$

D.3 Broyden Tridiagonal Function

$$F(x) = \tfrac{1}{2} \sum_{k=1}^{m} f_k(x)^2, \quad m = n,$$

with

$$f_k(x) = (3 - 2x_k)x_k - x_{k-1} - 2x_{k+1} + 1,$$

under the convention

$$x_0 = x_{n+1} = 0,$$

and suggested starting point

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

9

Ladislav Lukšan and Jan Vlček, *Test Problems for Unconstrained Optimization*, 2003. `https://www.researchgate.net/publication/325314497_Test_Problems_for_Unconstrained_Optimization`