# Numerical Optimization Assignment 3

Dilnaz-str989, Hamsa-nbp737, Rasmus-kpn134, Parth-tdh903

February 18, 2026

## 1 Introduction

This report answers the given theoretical questions (described in Section 2). We then describe the implementation of the approximate Newton algorithm and compare it to the Newton and Steepest Descent methods from last week. Subsequently, we go over the methodology and results for the requested benchmarking of Steepest Descent and four Newton variants (described in Section 3). This is then followed by a discussion of the results (Section 4), before a final conclusion (Section 5). *For the project, all team-members have contributed equally.*

## 2 Theory

**2.1)** From equation (20) for any $k$ : $p_k = v_k - \sum_{i=0}^{k-1} \frac{v_k^T Q p_i}{p_i^T Q p_i} p_i$, with $v_k = -\nabla f(x_k)$
We rearrange the equation (20) and put $v_k = -\nabla f(x_k)$ in the equation:

$$\nabla f(x_k) = -p_k + \sum_{i=0}^{k-1} -\frac{(-\nabla f(x_k))^T Q p_i}{p_i^T Q p_i} p_i$$

Let $\gamma_k = -1$, since, $(-\nabla f)^T = -(\nabla f^T)$ then we get $\gamma_i = \frac{(\nabla f(x_k))^T Q p_i}{p_i^T Q p_i}$, for $i = 0, \ldots, k-1$.
Then we obtain $\nabla f(x_k) = \sum_{i=0}^{k} \gamma_i p_i$. So the gradient $\nabla f(x_k)$ can be written as linear combination of $p_0, \ldots, p_k$ i.e. $\nabla f(x_k) \in \text{span}\{p_0, \ldots, p_k\}$.
Using our prove we let $\nabla f(x_j) = \sum_{i=0}^{k} \gamma_i p_i$ for all $j < k$ insert that in equation, $\nabla f(x_k)^T \nabla f(x_j) = \nabla f(x_k)^T \left( \sum_{i=0}^{j} \gamma_i p_i \right) = \sum_{i=0}^{j} \gamma_i \underbrace{\nabla f(x_k)^T p_i}_{\text{Theorem 6}} = 0$ for all $j < k$

**2.2)** From part 2.1 we proved: $\nabla f(x_k) = \sum_{i=0}^{k} \gamma_i p_i$, hence, $\nabla f(x_k) \in \text{span}\{p_0, \ldots, p_k\}$
Now suppose: $\sum_{i=0}^{k} \gamma_i p_i = 0$
Multiply both sides by $p_j^T Q$: $p_j^T Q \sum_{i=0}^{k} \gamma_i p_i = 0 \Rightarrow \sum_{i=0}^{k} \gamma_i p_j^T Q p_i = 0$
Since the directions are $Q$-conjugate, $p_j^T Q p_i = 0$ $(i \neq j)$ and $Q$ is symmetric positive definite, i.e. $p_j^T Q p_j > 0$ for $p_j \neq 0$
So $\gamma_j \, p_j^T Q p_j = 0 \Rightarrow \gamma_j = 0$. Hence all coefficients are zero. Therefore, $p_0, \ldots, p_k$ are linearly independent.
From previous result, $\nabla f(x_k)^T \nabla f(x_j) = 0$ for $j < k$, the gradients are mutually orthogonal and therefore linearly independent. Thus both sets $\{p_0, \ldots, p_k\}$ and $\{\nabla f(x_0), \ldots, \nabla f(x_k)\}$ contain $k+1$ linearly independent vectors i.e. $k+1$ dimension.

$$\text{span}\{p_0, \ldots, p_k\} = \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_k)\}$$

From (20) with $v_{k+1} = -\nabla f(x_{k+1})$ we get: $p_{k+1} = -\nabla f(x_{k+1}) + \sum_{i=0}^{k} \frac{(\nabla f(x_{k+1}))^T Q p_i}{p_i^T Q p_i} p_i$
From equality of spans we have, $p_i \in \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_i)\}$ and the gradients are mutually orthogonal, it follows that

$$\nabla f(x_{k+1})^T Q p_i = 0 \qquad i < k$$

Thus only the term $i = k$ remains and we obtain

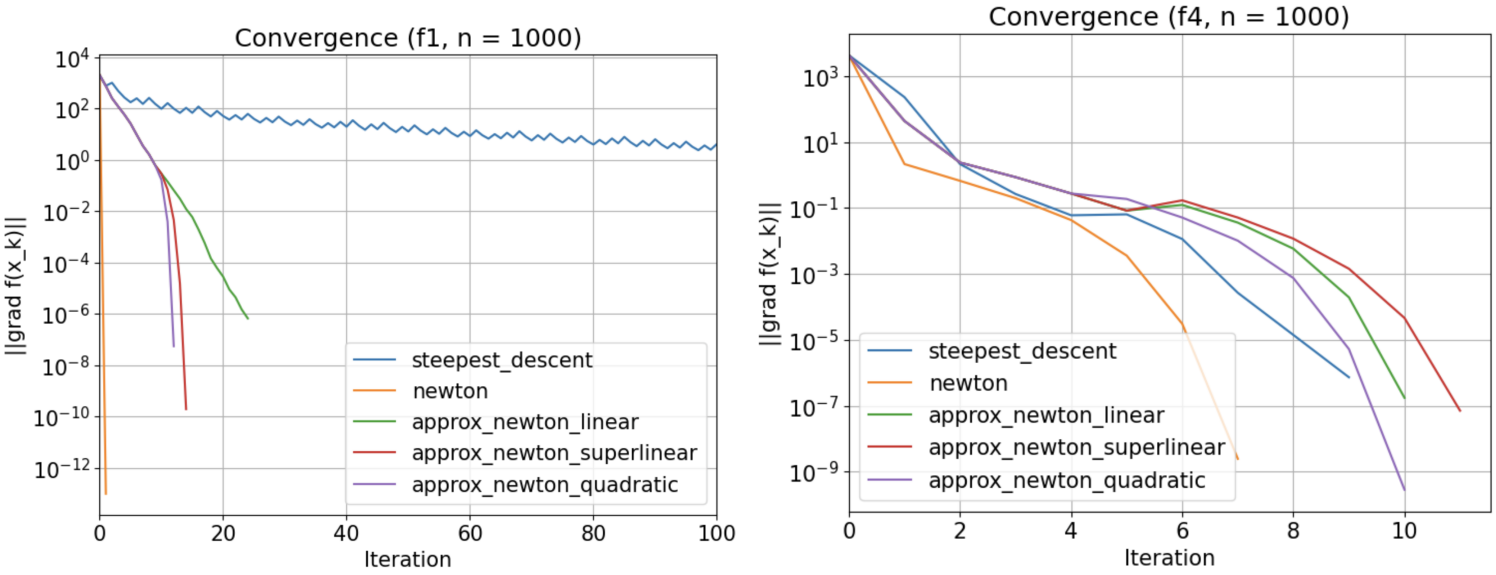
$$p_{k+1} = -\nabla f(x_{k+1}) + \beta_k p_k, \quad \text{where} \quad \beta_k = \frac{\nabla f(x_{k+1})^T Q p_k}{p_k^T Q p_k}$$

# 3 Methodology & Results

To create the Inexact Newton, we implemented a custom version of the CG (Conjugate Gradient) algorithm. To verify its correctness, we tested that $\nabla f(x_{k+1})^T p_k \approx 0$ and $\log(p_k^T Q p_N) \approx 0$ holds asymptotically for $k = 0...N$, as iterations increase and numerical errors consequently decrease. The `backtracking` algorithm was initialized with $c_1 = 10^{-4}$ (set low to relax the decrease requirement), and $\rho = \frac{1}{2}$, which is described to work well in practice.

For each benchmark experiment, we chose one random starting point (to avoid bias) chosen from the standard uniform distribution. As requested, we executed SD (Steepest Descent), Newton, and several Inexact Newton minimizers for three values of dimension $n$, specifically 100, 500, and 1000 (representing a wide range of dimensions with $n > 20$). To also bound the execution time in extreme cases, we limited the maximum number of iterations to 1000. For the Inexact Newton, we chose values of $\eta_k$ as $\frac{1}{2}$, $\min\{\frac{1}{2}, ||\nabla f(x_k)||\}$, and $\min\{\frac{1}{2}, \sqrt{||\nabla f(x_k)||}\}$ respectively, to allow for linear, superlinear, and quadratic convergence, as according to known theory.

As requested, we measured, for each benchmark execution, the length of the gradient, the number of iterations, the practical execution time, and the average iterations of the the CG method for the Newton-based minimizers. The benchmarks were executed for the $f_1$ and $f_4$ case functions, which meet the CG requirement of being strictly convex (with positive definite hessians). The partial results are shown at Figure 1 (1a for $f_1$ and 1b for $f_4$) which show the progressive decrease of the gradient length (in log-scale) as a function of the iteration count with $n = 1000$. The gradient length is here sufficient to indicate convergence, given that the selected functions are strictly convex – all points with zero gradient will therefore be a local minimum. For visual clarity, we lastly upper-bounded the x-axis of Figure 1a to $x = 100$.




(a) Shows $f_1$ convergence.

(b) Shows $f_4$ convergence.

Figure 1: Convergence rates for Steepest Descent, Pure Newton, and Inexact/approximate Newton, with $\eta_k$ values chosen to yield linear, superlinear, and quadratic convergence for the $f_1$ and $f_4$ functions. The x-axes are upper-bounded by $x = 100$ for visual clarity.

# 4 Discussion

## 4.1 Benchmarks

**SD Performance** As seen for $f_1$ on Figure 1a, SD performs significantly worse than the Newton methods, with a total of 708 iterations. This may possibly be due to the occurrence of small gradients around the optimum, which the Newton methods are less sensitive to, compared to SD

(where gradients are the main descent mechanism). For $f_4$ however, the performance of SD is comparable to those of the Newton Methods.

**Convergence Rates** For $f_1$ the convergence rates behave in accordance with known theory: The pure Newton converges fastest, followed by the accelerating curve of the quadratic Inexact, the slightly flatter curve of the superlinear Inexact, and linear curve of the linear Inexact (using 2, 13, 15, and 25 iterations respectively). For $f_4$ we see the same relative pattern in the final number of iterations. The curves however, are now surprisingly uniform – each exhibiting an S-shape as opposed to their theoretical curvatures. This could possibly be explained by the more extreme nature of the $f_4$ function, with asymmetric and rapidly changing Hessians around the optimum. For the Newton methods, this could result in initially "cautious" Taylor approximations, which consequently yield uneven convergence rates (before quickly decreasing as we come sufficiently close to the minimum).

**Execution Time** In terms of the practical execution time for $f_1$, we observe benchmarks of 0.0496, 0.1073, 0.1246, and 0.1537 seconds for respectively the Pure, Inexact quadratic, Inexact superlinear, and Inexact linear method. The Pure Newton is thus holistically the fastest method. It does however exhibit a slow *time per iteration* ratio of $0.0496/2 = 0.0248$, as compared to e.g. 0.0082 for the quadratic Inexact. This result matches the corresponding theory, seeing as the Pure Newton relies on expensive matrix operations in contrast to the Inexact variants. Within our benchmarks, this result also generalized to $f_4$ as well as across dimensions. Besides the Newton methods however, the SD method exhibits the fastest descent ratio, with 0.0015 seconds per iteration. When SD behaves well for the problem, it consequently seems competitive with the Newton methods. It may however be less stable in its iteration performance, as indicated by the $f_1$ plot. Finally, it is worth mentioning that the time benchmarks may be unstable, as these are easily influenced by outside factors such as cache behavior.

**CG Steps** In terms of the average number of CG steps for the Newton methods, we observed that the quadratic Inexact Newton took the fewest steps, followed by the superlinear and then linear Inexact method. These had an average of 2.40, 1.73, and 1.20 GC steps respectively for $f_4$. The same trends could again be seen for $f_1$ as well as across dimensions. Theoretically, the results make sense given the context of $\eta_n$ which is set to "decrease" relatively for the methods in the listed order. Since $\eta_k$ defines the CG tolerance $\epsilon_k$, an decrease in its value will result in more CG iterations, in order to reach below the tolerance.

# 5 Conclusion

## 5.1 Benchmarks

As allways, one may conclude that the "best" method is case-dependent. In general, Pure Newton appears most versatile in terms of iterations for general functions. If the case function is strictly convex, one may consider the Inexact quadratic method (or SD, if well-behaved). Based on our benchmarks however, it appears that Pure Newton manages to compensate for its costly iterations by only limiting its amount of overall iterations. To confirm this in the general case however, one would likely need to conduct more extensive benchmarking on different functions.

## 5.2 Theory

For the theory part, we argued that there exist $\gamma_0, .., \gamma_k$ such that $\nabla f(x_k) = \sum_{i=0}^{k} \gamma_i p_i$. Using Theorem 6 and our previous argument we showed that $\nabla f(x_k)^T \nabla f(x_j) = 0$ and finally we showed the derivation of eq(21) from the notes.