Assignment 1

Numerical Optimisation

Jacobo Ostos Bollmann

University College London

Exercise 1

(a) A robust computational implementation of $f(x, y)$ handling single coordinate pair inputs possibly in $\mathbb{R}^2$ might involve the following:

```
def f(x, y):
    if x > 0:
          return (y + log(x)) ** 2 + (y - x) ** 2
    else:
          return inf
```

Briefly, such implementation asserts that the function be executed only for values of $x \in \mathbb{R}_+$, hence excluding the limiting case, else to return $\ll \varepsilon$ sufficiently large (e.g., ∞) to correct optimisation algorithms toward the positive x-semiaxis.

(b) Analytical gradient $\nabla f(x, y)$ and Hessian $\nabla^2 f(x, y)$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} \end{bmatrix}^T = \begin{bmatrix} 2\left(\dfrac{1}{x}(y + \ln x) - (y - x)\right) \\ 2\,(2y + \ln x - x) \end{bmatrix}$$

$$\nabla^2 f(x,y) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} \dfrac{2}{x^2}\left(x^2 - \ln x - y + 1\right) & 2\left(\dfrac{1}{x} - 1\right) \\ 2\left(\dfrac{1}{x} - 1\right) & 4 \end{bmatrix}$$

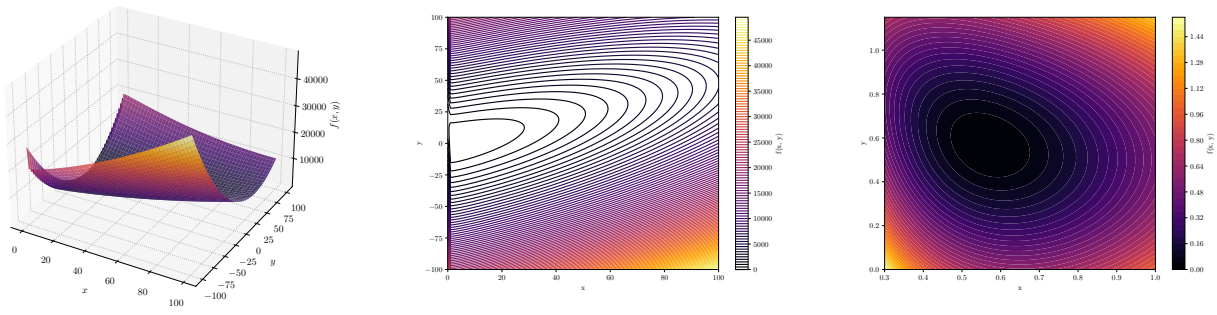


Figure 1: Plots and Contours $f(x, y) = (y + \ln x)^2 + (y - x)^2$

(c) The function plot in *Figure 1* (left) indicates that $f(x, y)$ approximates quadratic behaviour w.r.t the x-axis, while contours (centre) hint the existence of some minima for which the objective function grows unbounded left of it as $x \to 0^+$ toward the limiting case, and where $f(x, y)$ grows gradually as $x \to \infty$. If contours or level sets are sets $\{\boldsymbol{x}_c \in \text{dom} f : f(\boldsymbol{x}) = c\}$, $c \in \mathbb{R}$ constant, we then define sublevel sets $\{\boldsymbol{x}_s \in \text{dom} f : f(\boldsymbol{x}) \leq c\}$, $c \in \mathbb{R}$ constant. Further, we know that for a strictly convex function f, sublevel sets are as well convex for $\forall c \in \mathbb{R}$. Zoomed-in contours in *Figure 1* (right) indicate such is the case for $f(x, y)$, where each subsequent sublevel set is lower indicating convexity, i.e., $\forall\ c_i < c_j$, with $c_i \neq 0, f(\boldsymbol{x}_{s_i}) < f(\boldsymbol{x}_{s_j} \cap (\boldsymbol{x}_{s_i})^C)$, each $\boldsymbol{x}_c$ being a sublevel set.

To determine the amount of stationary points we give without proof the following first order characterisation of critical points of strictly convex functions.

Lemma 1.1 *If $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ is strictly convex, any local minimiser x^* is also a global minimiser of f. Further, if f is continuously differentiable, then any stationary point x^* is a global minimiser.*

Given strict convexity, there must exist at least one stationary point, corresponding to a local and thus global minimiser.

(d) Let $\mathcal{S}_d$ denote the linear subspace of all $d \times d$ symmetric matrices with entries in $\mathbb{R}$ and denote by $\mathcal{S}_d^+$ the set of all symmetric positive definite (s.p.d) matrices.

Numerically, through Descent Line Search method, with Backtracking algorithm, the minimiser was found at the point $\boldsymbol{x}^* \approx (0.56714329, 0.56714329)$; Such minimiser satisfies first and second order sufficient conditions:

$$\nabla f(x^*) = \begin{bmatrix} 2\left(\frac{1}{0.56714329}(0.56714329 + \ln 0.56714329) - (0.56714329 - 0.56714329)\right) \\ 2\left(2(0.56714329) + \ln 0.56714329 - 0.56714329\right) \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\nabla^2 f(x^*) \approx \begin{bmatrix} 2(1 + 0.56714329^{-2}) & 2(0.56714329^{-1} - 1) \\ 2(0.56714329^{-1} - 1) & 4 \end{bmatrix},$$

where $\nabla^2 f(x^*) \in \mathcal{S}_d^+$, since it is orthogonally diagonalisable with positive eigenvalues, i.e., we can find an orthogonal matrix $A \in \mathcal{M}_d(\mathbb{R})$, $AA^\top = I_d$ such that $\nabla^2 f(x^*) = ADA^\top$, $D = diag(\lambda_1, \lambda_2) \approx diag(3.5, 8.1)$. Hence $\forall \lambda_i > 0$, $i = 1, 2$, and we can prove $\boldsymbol{x}^\top \nabla^2 f(\boldsymbol{x}^*)\boldsymbol{x} > 0 \ \forall x \in \mathbb{R}^2 : \boldsymbol{x} \neq 0$ and the Hessian is s.p.d. at the minimiser $\boldsymbol{x}^*$.

To prove uniqueness of $\boldsymbol{x}^*$, it suffices to prove f is strictly convex due to **Lemma 1.1**. In addition to the properties mentioned in *1.b)* we can further make a case for f strictly convex. We immediately notice that $f(x, y)$ is by definition given by the sum of two squared terms, and is a second order polynomial where convexity implications are twofold; quadratic functions with a positive coefficient are convex everywhere and, for any two functions, e.g., $f(x), g(x)$ convex, their sum $h(x) = f(x) + g(x)$ is again convex. We also point out that the convex domain of $f(x, y)$, $\mathbb{R}_+ \times \mathbb{R}$, as well satisfies the formal definition of convex functions, as spaces and half-spaces are well known convex sets and we know that for two sets, e.g., C, S convex, the product set $C \times S$ is also convex.

(d) A gradient ∇f is Local Lipschitz continuous if

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{N}(\boldsymbol{x}) : \exists L > 0 : ||\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)|| \leq L||\boldsymbol{x}_1 - \boldsymbol{x}_2||, \forall \boldsymbol{x} \in dom f.$$

i.e., the gradient of f is Lipschitz continuous in a bounded neighbourhood of every point in its domain. In this case $dom f = \mathbb{R}_+ \times \mathbb{R}$. Then we start from Taylor's Thm. integral form of the gradient, and setting $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \boldsymbol{p} \Rightarrow \boldsymbol{p} = \boldsymbol{x}_2 - \boldsymbol{x}_1$

$$\nabla f(\boldsymbol{x}_2) = \nabla f(\boldsymbol{x}_1) + \int_0^1 \nabla^2 f(\boldsymbol{x}_1 + \tau(\boldsymbol{x}_2 - \boldsymbol{x}_1))(\boldsymbol{x}_2 - \boldsymbol{x}_1) d\tau$$

Then we can rearrange and take L^2-norms to yield

$$\begin{aligned}||\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)|| &= || \int_0^1 \nabla^2 f(\boldsymbol{x}_1 + \tau(\boldsymbol{x}_2 - \boldsymbol{x}_1))(\boldsymbol{x}_2 - \boldsymbol{x}_1) d\tau || \\ &\leq \int_0^1 ||\nabla^2 f(\boldsymbol{x}_1 + \tau(\boldsymbol{x}_2 - \boldsymbol{x}_1))|| ||(\boldsymbol{x}_2 - \boldsymbol{x}_1)|| d\tau\end{aligned}$$

Then we can introduce the Frobenious norm, as hinted in the prompt, as an upper bound to the L^2-norm, and we have

$$||\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)|| \leq \int_0^1 ||\nabla^2 f(\boldsymbol{x}_1 + \tau(\boldsymbol{x}_2 - \boldsymbol{x}_1))||_F ||(\boldsymbol{x}_2 - \boldsymbol{x}_1)|| d\tau$$

And since by definition the Frobenious norm is given by $||A||_F = \sqrt{\sum_{i,j} a_{i,j}^2}$, it will remain finite for a finite dimensional problem and entries $a_{i,j} < \infty$, true for any open neighbourhood of any point in the domain of f, since they would not include the limiting case or infinty. Hence we have concluded that the gradient, ∇f is locally Lipschitz continuous.

Exercise 2

Definition 2.1 (Descent direction) *A descent direction is a vector* $\boldsymbol{p} \in \mathbb{R}^n$ *that points toward a local minimiser* $\boldsymbol{x}^*$ *of the objective function* $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$.

Proposition 2.2 (Wolfe Conditions: Existence) *Let* $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ *continuously differentiable and* $\boldsymbol{p}$ *be a descent direction at* $\boldsymbol{x}_k$. *If* f *is bounded below along the ray* $\{\boldsymbol{x}_k + \alpha\boldsymbol{p} \mid \alpha > 0\}$, *then there exists an interval of step lengths satisfying both the Wolfe conditions and strong Wolfe conditions.*

We begin by showing $f(x, y)$ satisfies existence of Wolfe conditions. Firstly, $\exists\ \nabla f, \nabla^2 f$ thus the objective function is twice continuously differentiable and sufficiently smooth by local Lipschitz continuity of both. Further, $f(x, y) = (y + \ln x)^2 + (y - x)^2$ is constituted by two squared terms implying convexity and $\forall \alpha > 0, \forall \boldsymbol{x}_k \in \mathbb{R}_+ \times \mathbb{R}, \forall \boldsymbol{p} \in \mathbb{R}^2$ s.t. $f(\boldsymbol{x}_k + \alpha\boldsymbol{p}) < f(\boldsymbol{x}_k)$, $f \geq 0$ surely, hence there must exist an interval of step lenghts satisfying both Wolfe and strong Wolfe conditions, and we can apply a line search algorithm with strong Wolfe conditions to minimise $f(x, y)$.

We as well refer to the literature to justify parameter choice. Wolfe conditions combine sufficient decrease conditions (1) and curvature conditions (2):

$$\phi(\alpha) := f(\boldsymbol{x}_k + \alpha\boldsymbol{p}) \leq f(\boldsymbol{x}_k) + c_1\alpha\boldsymbol{p}^\top \nabla f(\boldsymbol{x}_k) =: l(\alpha) \tag{1}$$

$$\phi'(\alpha) = \boldsymbol{p}^\top \nabla f(\boldsymbol{x}_k + \alpha\boldsymbol{p}) \leq c_2\boldsymbol{p}^\top \nabla f(\boldsymbol{x}_k) = \phi'(0) \tag{2}$$

with $0 < c_1 < c_2 < 1$. Taking the absolute value on both sides of (2) gives strong Wolfe conditions. For our sufficient conditions good practice indicates that c_1 be chosen to be small $\approx 1e^{-4}$. However, notice that while $l(\alpha)$ is a function with negative slope, $c_1\alpha\boldsymbol{p}^\top \nabla f(\boldsymbol{x}_k)$, since $c_1 \in (0, 1)$ it lies above ϕ for $\forall\alpha$ sufficiently small. Hence, curvature condition ensures unacceptably small steps are rejected, requiring the gradient at the step to be greater than that of the current step scaled by c_2. Recommended choice of c_2 is ≈ 0.9 in the context of Newton-type methods though it will be generally used as a point of further comparison. Again in this context of Newton / quasi-Newton methods, $\alpha_0 = 1$ is standard practice. While, in general, steepest descent produces poorly scaled search directions the same initial step was valid.

Finally, let us introduce from this point on convergence sequences; to investigate convergence, unless otherwise stated, we build sequences $r_k = \{||\boldsymbol{x}_k - \boldsymbol{x}^*||_2\} \in \mathbb{R}$ indexed by $k = 1, ..., n$ for n iterations, for the normed k^{th} errors of sequences converging to $\boldsymbol{x}^*$.

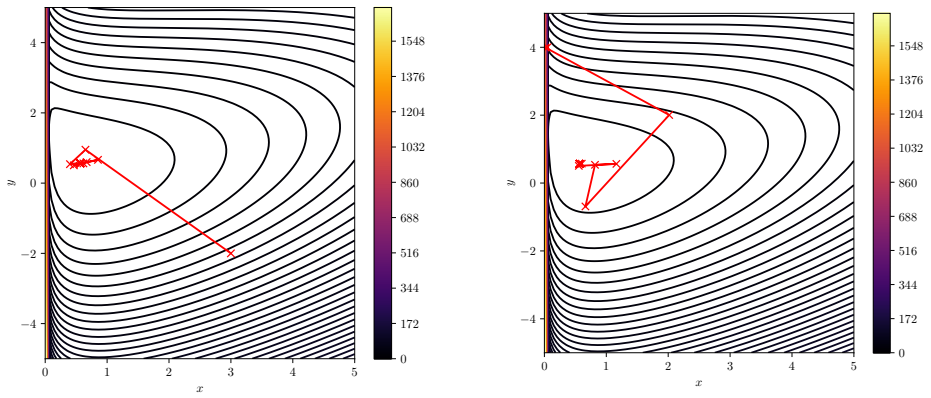


Figure 2: Steepest Descent Over Contours of $f(x, y) = (y + \ln x)^2 + (y - x)^2$

Algorithm	Line Search with Strong Wolfe Conditions
Tolerance	1e-4
α_0	1
c_1	0.1
c_2	0.9
Max Iterations	100

Table 1: Chosen Parameters for Steepest Descent

(a) Method shows convergence from both $x_0 = (3, -2)^\top$ and $x_0 = (0.05, 4)^\top$, in agreement with existence of strong Wolfe conditions for $f(x, y)$, and the characteristic zigzag of the method is apparent for elongated contours. Choice of parameters is again supported in the literature.

(b)

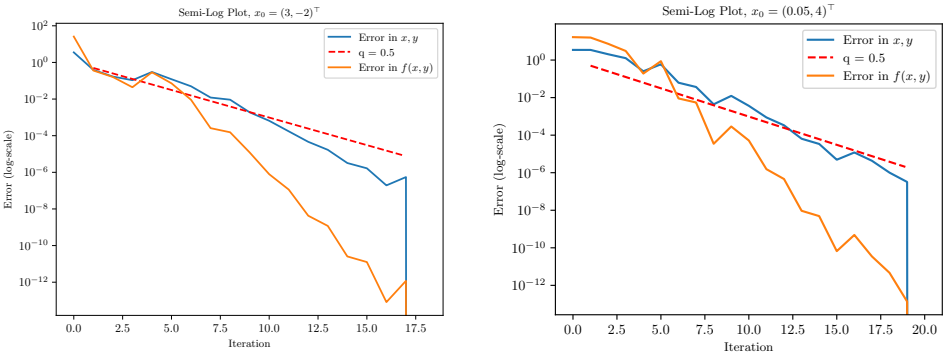


Figure 3: Semi-Log Error Plots Exponential Convergence - Strong Wolfe Conditions Steepest Descent

Let us recall our k^{th} step error definition for convergent sequences $r_k = \{||\boldsymbol{x}_k - \boldsymbol{x}^*||_2\}$. We find that exponentially convergent sequences follow $r_k = Cq^k$ for some $C \in \mathbb{R}, q \in (0, 1)$. Q-linear convergence is then given by $\frac{||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||_2}{||\boldsymbol{x}_k - \boldsymbol{x}^*||_2} \leq r$ for some $r \in (0, 1)$, and sufficiently large k. We immediately notice that taking $r \geq \frac{r_{k+1}}{r_k} \leq \frac{Cq^{k+1}}{Cq^k} = q \Rightarrow r = q$, and exponential convergence implies Q-linear convergence for our error sequence. Then, we know that linear relationship in semi-log plots evidences error Exponential convergence rate; hence in agreement with theoretical results Steepest Descent is Q-linear convergent. We include in the above plots an exponentially convergent series $r_k = 0.5^k$, i.e., $C = 1, q = 0.5$ with the approximate strong bound q-value to further validate hypothesis. Strong bound given by $\frac{\lambda_2 - \lambda_1}{\lambda_1 + \lambda_2} \approx 0.4261$ (both cases), where $\lambda_1 < \lambda_2$ are the eigenvalues of $\nabla^2 f(\boldsymbol{x}^*)$, and $\boldsymbol{x}^*$ is the minimiser of $f(x, y)$.

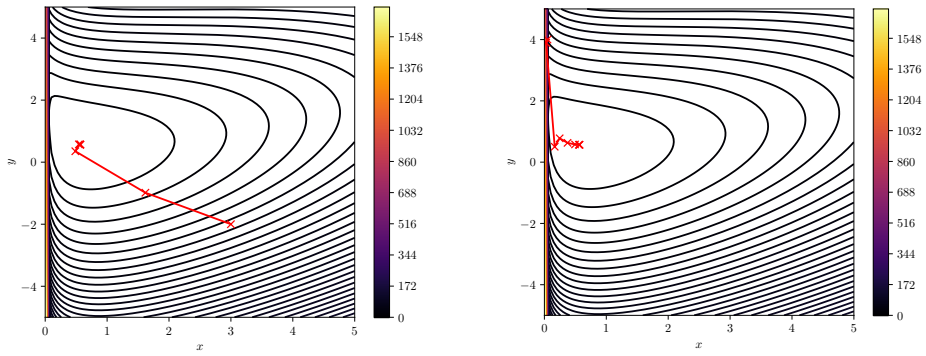


Figure 4: Newton Direction Over Contours of $f(x, y) = (y + \ln x)^2 + (y - x)^2$

Algorithm	Line Search with Strong Wolfe Conditions
Tolerance	1e-4
α_0	1
c_1	0.1
c_2	0.9
Max Iterations	100

Table 2: Chosen Parameters for Newton Direction

(c) Method shows convergence from both $x_0 = (3, -2)^\top$ and $x_0 = (0.05, 4)^\top$, in agreement with existence of strong Wolfe conditions for $f(x, y)$. Choice of parameters is again supported in the literature.

(d)

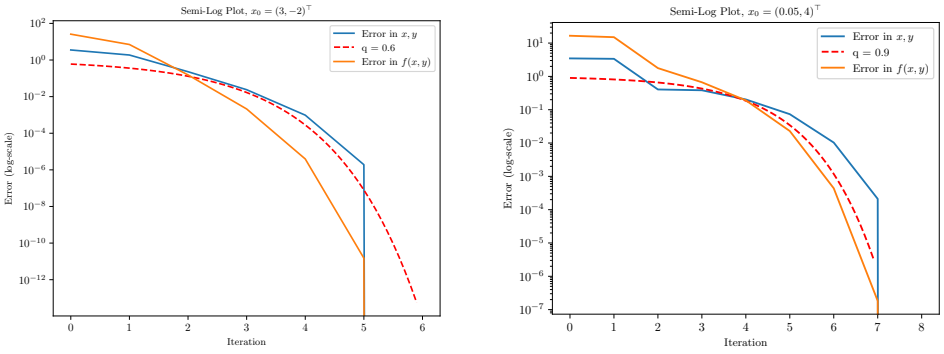


Figure 5: Semi-Log Error Plots Exponential Convergence - Strong Wolfe Conditions Newton Direction

Recall for our f strictly convex $\exists \nabla^2 f$ locally Lipschitz continuous over $\text{dom} f$, and $\exists \boldsymbol{x}^*$ s.t. $f(\boldsymbol{x}^*) < f(\boldsymbol{x}) \forall \boldsymbol{x} \neq \boldsymbol{x}^*$ satisfying sufficient conditions and thus $\nabla^2 f(\boldsymbol{x}^*) \in \mathcal{S}_d^+$. Then we know for the sequence of iterates $\{x_k\}$, and subsequently for the error convergence sequence $\{r_k\}$, that for step sizes satisfying Wolfe Conditions with $c_1 \leq 0.5, \alpha_0 = 1$ convergence will be Q-quadratic. Q-quadratically convergent sequences are given by $Cq^{2^k} \geq ||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||_2 \leq C||\boldsymbol{x}_k - \boldsymbol{x}^*||_2^2$. We analyse *a posteriori* the error convergence sequences in a semi-log plot to show it is neither sublinear or linear, as it descends faster. Then with the approximate q-values found, we plot quadratically convergent series for each starting point to further validate hypothesis. It is recorded in the literature that for appropriate parameters, Newton methods will converge quadratically while quasi-Newton methods will do so superlinearly.

(e) To proof guaranteed global convergence of both methods, recalling $f(x, y)$ satisfies existence of (strong) Wolfe conditions, we refer to Zoutendijk's lemma.
Zoutendijk's lemma states that for a sequence of iterates generated by descent directions scaled by a sequence of step lengths satisfying Wolfe conditions, if $f \geq C, C \in \mathbb{R}$ is bounded from below and continuously differentiable with a Lipschitz continuous gradient we can proof global convergence for a Line Search method. Moreover, we know that strong Wolfe conditions imply the Zoutendijk condition

$$\sum_{k \leq 0} \cos \theta_k{}^2 ||\nabla f(\boldsymbol{x}_k)||^2 < \infty \quad \text{which implies}$$

$$\cos \theta_k{}^2 ||f(\boldsymbol{x}_k)||^2 \to 0$$

and we can derive global convergence results for line search methods as well. All the properties we have pointed thus far for $f(x, y)$ indicate that for any method employing (strong) Wolfe conditions algorithm we can guarantee global convergence. Finally, we outline that despite convergence of both methods is certain, Newton method will do so faster since it is as well informed with curvature of f through the Hessian values.

Exercise 3

(a)

```python
import numpy as np


def solverCMdogleg(F: Type[absObjective], x_k: np.array, Delta: float):
    """quadratic constrained model solver implementing dogleg method
    Args:
    ------
        :param F: function handler; objective function, gradient, Hessian
        :param x_k: current point
        :param Delta: trust-region radius
    Output:
    --------
        :return p_k: new step"""

    g: np.array = F.df(x_k)                                          # gradient
    B: np.array = F.d2f(x_k)                                         # hessian

    d, Q = np.linalg.eig(B)
    pB: np.array = -np.linalg.solve(B, g)

    if np.min(d) > 0 and np.linalg.norm(pB) < Delta:                 # check spd hessian and step within trust region
        return pB                                                    # return step

    else:                                                            # dog leg step
        gTBg = g.T @ B @ g
        Q, _ = np.linalg.qr(np.array([g, pB]))

        if Q.shape == 1:                                             # if g and pB are colinear
            tau = 1 if gTBg <= 0 else min(np.linalg.norm(g) ** 3 / (Delta * gTBg), 1)      # compute tau
            p = -tau * Delta / np.linalg.norm(g) * g
            return p                                                 # return cauchy point

        else:                                                        # unconstrained minimiser of quadratic m(tau) with steepest dir.
            pU: np.array = -(np.dot(g, g) / gTBg) * g

            if np.linalg.norm(pU) >= Delta:                          # if pU is outside trust region
                tau = Delta / np.linalg.norm(pU)                     # compute tau
                return pU * tau                                      #return dogleg step

            else:                                                    # else compute intersection of trust region and dogleg path
                a = np.linalg.norm(pB - pU) ** 2
                b = 2 * np.dot(pB - pU, pU)
                c = np.linalg.norm(pU) ** 2 - Delta**2
                D = b**2 - 4 * a * c

                if D >= 0:
                    tau1 = (-b + np.sqrt(D)) / (2 * a)
                    tau2 = (-b - np.sqrt(D)) / (2 * a)

                tau = np.max([[tau1 + 1, tau2 + 1]])

        if 0 <= tau <= 1:
            p = pU * tau
        if 1 < tau <= 2:
            p = pU + (pB - pU) * (tau - 1)

    return p                                                         # return dogleg step
```

Figure 6: Dogleg Trust Region Implementation

In computing the intersection between Dogleg path and the Trust Region, we first check for the case where the norm of the solution to our unconstraint minimisation problem lies outside the trust region; i.e., $\boldsymbol{x}(\Delta) := \arg\min_{\{\boldsymbol{x}\in\mathbb{R}^n : ||\boldsymbol{x}-\boldsymbol{x}_k||\}} m_k(\boldsymbol{x}) \geq \Delta$, where we realise it intersects exactly once with the Trust Region, and we must compute such a point as the solution to the constrained problem $\boldsymbol{x}(\Delta) := \arg\min_{\{x\in\mathbb{R}^n : ||\boldsymbol{x}-\boldsymbol{x}_k||\leq\Delta\}} m_k(\boldsymbol{x})$. If it lies in the first segment, which we check comparing the norm of the direction from the origin to the Cauchy point with the region radius, then $\tau = ||p^U||^{-1}\Delta_k$, and $p = \tau p^U$. If it lies, however, in the second segment, then we solve for the roots of the quadratic equation $||p^U + (\tau - 1)(p^B - p^U)||^2 = \Delta_k^2$. For this second segment we additionally check for the values of τ, where now

$$\tilde{p}(\tau) = \begin{cases} \tau p^U & 0 \leq \tau < 1 \\ p^U + (\tau - 1)(p^B - pU) & 1 \leq \tau \leq 2 \end{cases}$$

and we know $\exists\ 0 \leq \tau \leq 2$ among the roots satisfying $\tilde{p}(\tau)$.

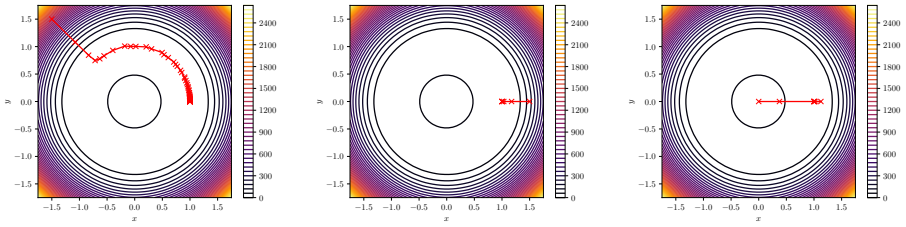


Figure 7: Dogleg Iterates Over Contours of $f(x,y) = 100(y^2 + x^2 - 1)^2 + (1 - x)^2$ from $\boldsymbol{x}_0 = (-1.5,1.5)^\top, (-1.5,0)^\top, (0,0)^\top$.

Dogleg Trust Region method converged well to the objective function minimiser $\boldsymbol{x}^* = (1,0)$ from all starting points, while the former $(-1.5,1.5)$, does so slower and less accurately, taking some 78 iterations, compared to 4 and 5 respectively for the remaining initial points. We immediately notice that by definition $f(x,y)$ is an heterogeneous fourth order polynomial, implying f non-convex. Now, consider the level sets $\{\boldsymbol{x}_c \in \text{dom} f : f(\boldsymbol{x}) = c\}$, $c \in \mathbb{R}$ given by the contours, then we observe a basin surrounding a stationary point corresponding to a global maximiser at $\boldsymbol{x}_1^* \approx (-5.02525 \times 10^{-3}, 0)$ since $\nabla^2 f(\boldsymbol{x}_1^*) \in \mathcal{S}_d^-$, is symmetric negative definite. Since the method descends following the negative gradient values, we can infer that through this basin gradient is small. Contextualising the actual results within the theoretical frame the conclusion is as follows: even though at this basin $\nabla f, \nabla^2 f \ll$, the relative progress, ρ, given by the ratio between decrease in the objective function and model function, is even smaller causing the region to shrink yet stay for the most part active. Minimisation progress is then mostly a sequence of intersections between unconstrained minimisers and the trust region at each iterate. The other two points converge well since the gradients are aligned with the minimiser on the y-axis, and inside trust region boundary. We notice for $\boldsymbol{x}_0 = (0,0)^\top$, the method first goes past the minimiser, likely caused by a better model minimisation by the Cauchy point, and hence a 'poorly' scaled step by the steepest descent, which is picked since both Newton and Cauchy points are inside the TR. Choice of parameters was generally $\hat{\Delta} = 3, \eta = 0.1$, maxIter $= 100$, tol $= 1e^{-6}$. Commonly encapsulated in the question *'have we ran out of money, time, or patience?'*, stopping criteria is an extensive field of discussion for optimisation methods. The core interest lies in the implementation of stopping criterion able to identify, through finite precision algorithms, how dependent and independent variables evolve w.r.t iterations. We discard employing first and second order sufficient conditions exclusively, given the sensibility of crude approximations of the type $\nabla f(\boldsymbol{x}_k) \leq tol$ to the values of $f, \boldsymbol{x}_k$; i.e., for a very flat yet convex function and for a large enough tolerance, we will not see any progress without scaling either $f, \boldsymbol{x}$. To remedy this, we choose to scale through the relative gradient.

$$relgrad(\boldsymbol{x}_k) = \lim_{\delta \to 0} \frac{\frac{f(\boldsymbol{x}_k + \delta) - f(\boldsymbol{x}_k)}{f(\boldsymbol{x}_k)}}{\frac{\delta}{\boldsymbol{x}_k}} = \frac{\nabla f \boldsymbol{x}_k}{f(\boldsymbol{x}_k)}$$

which translates to our test $||\frac{\nabla f(\boldsymbol{x}_k)}{f(\boldsymbol{x}_k)}||_\infty < tol$. In practice, there are issues in measuring relative change for the L-∞ norm argument when f near 0, hence we choose $f(\boldsymbol{x}_k) + 1$.

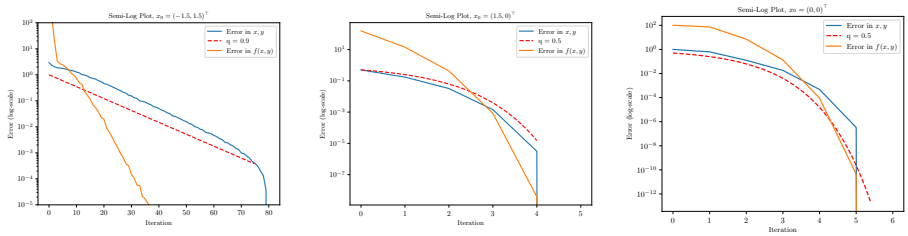


Figure 8: Dogleg k^{th} Error Sequences from $\boldsymbol{x}_0 = (-1.5,1.5)^\top, (-1.5,0)^\top, (0,0)^\top$.

(c) Above we display semi-Log normed error convergence plots for our iterates toward the minimiser of f. For now, (assume) let f be twice continuously differentiable in a neighbourhood of the minimiser $\mathcal{N}(\boldsymbol{x}^*), \boldsymbol{x}^*$ satisfying first and second order sufficient conditions. Then, since iterate sequences $\boldsymbol{x}_k$ are converging to the minimiser for k sufficiently large, we know building our constrained model with exact Hessian at every step and choosing steps satisfying sufficient reduction criteria with Cauchy points we shall obtain a behaviour analogous to exact Newton Line Search for all inactive TR steps. We observe this precisely for the latter two initial points $(-1.5, 0)^\top, (0, 0)^\top$, where we propose Q-quadratic sequences $Cq^{2^k} \geq ||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||_2$ with the approximate q-values to further support hypothesis. From $(-1.5, 1.5)^\top$ we find a linear relationship, which as mentioned in *2.c* implies exponential order and Q-linear convergence. We observe that the trajectory violates the conditions for higher-order convergence, lacking an inactive TR further making a case for the conclusions in *3.b*. From this origin, progress is effectively given by the mentioned intersections of the dogleg path with the TR; Now, we know Cauchy points are given by the steepest direction and hence, depending on the dogleg segment intersecting with TR boundary, convergence rate might be as slow as Q-sublinear, i.e., in sequences $Ck^q \geq ||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||_2$ where $C \in \mathbb{R}, q < 0$, e.g., $\frac{1}{\sqrt{k}}$. But since convergence averages over the sequence, when combined with some Newton point second segment intersections, and perhaps more favourable perfomance of the steepest descent, the resulting convergence order is Q-linear.

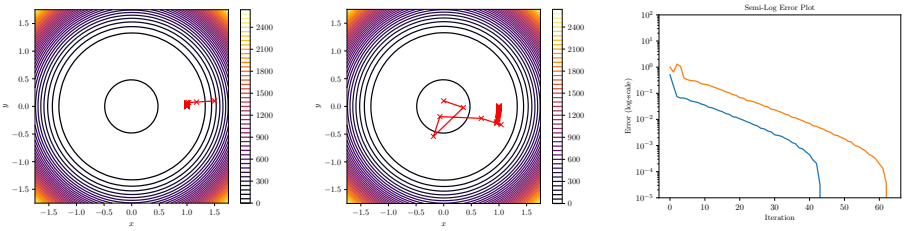


Figure 9: Dogleg Iterates and k^{th} Error Sequences from $\boldsymbol{x}_0 = (1.5{,}0.1)^\top, (0, 01)^\top$.

(d) Disturbing the points returns us to cases with an active TR. From the new origins $\boldsymbol{x}_0 = (1.5, 0.1)^\top, (0, 01)^\top$, our gradient is no longer aligned with the minimiser along the y-axis, forcing the minimisation path onto the basin. We recall in this basin we have gradient values are very small, with an almost indefinite hessian; subsequently we also observe very poor relative progress of the model with respect to the objective function which is again shrinking the TR beyond the unconstrained minimiser. Progress is given by the intersections of dogleg paths with the TR boundaries at each iteration. For the semi-log error convergence plots, we observe irregularities prior to finding the basin; then again convergence approximates Q-linear convergence.

(e) We can guarantee global convergence of the TR method toward a stationary point but only for functions and parameters conforming to the following conditions: we must have f bounded below on some level set and Lipschitz continuously differentiable in the neighbourhood of such level set. Futher we need the hessian $||B_k|| \leq \beta$ bounded as well for some constant β, and the solutions p_k of the constrained model satisfy relaxed sufficient reduction conditions (we allow $||p_k|| \leq \gamma\Delta_k, \gamma > 1$). Our Rosenbrock function has a polynomial form gradient and Hessian, which we know are Lipschitz continuous for any bounded domain; Similarly, despite non-convex, f consists of two squared terms with positive coefficients hence is again bounded below at 0, also true in any neighbourhood of its minimiser, thus meeting conditions for guaranteed convergence. Additionally, and refering back to our implementation, at every step the method includes a check for $\nabla^2 f \in \mathcal{S}_d^+$, which guarantees convergence is to a stationary point (provided appropriate parameters) and that such will be a minimiser.

Exercise 4

(a)

```
while not stopCond and k < maxIter:
    alpha = lineSearchWC(F, x, p, alpha=1, c1=10**-4, c2=0.1)          # WC step length
    info["alphas"].append(alpha)                                        # store step length
    x = info["xs"][-1] + info["alphas"][-1] * info["ps"][-1]            # update x
    info["xs"].append(x)                                                # store x

    beta = (F.df(x).T @ (F.df(x) - F.df(info["xs"][-2]))) / (           # compute Beta
        F.df(info["xs"][-2]).T @ F.df(info["xs"][-2])
    )

    info["betas"].append(beta)                                          # store Beta
    p = -F.df(x) + info["betas"][-1] * info["ps"][-1]                   # compute new direction
    info["ps"].append(p)                                                # store new direction
    k += 1                                                              # increment counter
    stopCond = np.linalg.norm(F["g"](x), ord=np.inf) < tol * (1 + abs(F["f"](x)))   # check stopping criterion
```

Figure 10: Polak-Ribiere Update for Non-Linear Conjugate Gradient Optimisation Method .

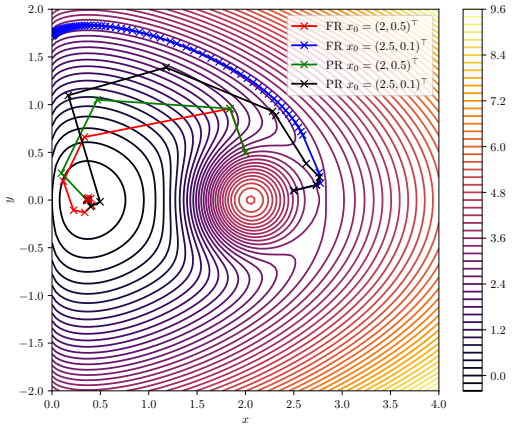


Figure 11: Conjugate Gradient Fletcher-Reeves and Polak-Ribiere methods from $\boldsymbol{x}_0 = (2, 0.5)^\top, (2.5, 01)^\top$ for $f(x, y) = x \ln(x) + y^2 + c \exp -\frac{(x-a)^2+(y-b)^2}{\sigma^2}$.

(b) We are employing descent Line Search with strong Wolfe conditions algorithm to compute each iterate's step length, hence relevant parameters include $c_1 = 1e^{-4}, c_2 = 0.1$, as per the recommendations made in *2.*; however, through a theorem by Al-Baali regarding expected global convergence for unrestarted FR method, we know that we may employ a Line Search satisfying Wolfe conditions with $0 < c_1 < c_2 < 0.5$, so long as we have a gradient scaling bounded above by the FR scaling factor β_k^{FR}, for $k \leq 2$ iterations. Tolerance is given at $1e^{-4}$, and maxIter was set at 150.

(c) We obviate the better results of FR, PR from $\boldsymbol{x}_0 = (2, 0.5)^\top$, and PR from $\boldsymbol{x}_0 = (2.5, 0.1)^\top$, all of which converge well to the minimiser at $\boldsymbol{x}^* \approx (0.3674, 0)^\top$ within the precision dictated by the tolerance. Most prominently, we observe non-convergence of FR method from $\boldsymbol{x}_0 = (2.5, 0.1)^\top$. Now, let us characterise a 'poor search direction' within the context of FR. Consider, our k^{th} search direction is close to orthogonal to the negative gradient at that same iterate state s.t. $\cos\theta_k \approx 0, \theta_k = \angle(\boldsymbol{p}_k, -\nabla f(\boldsymbol{x}_k)) \approx \pi/2$, then we know the norm of the gradient will be much smaller than that of the descent direction causing our $\beta_{k+1}^{FR} \approx 1$ meaning that our next updated descent direction will be very close to the current one. We find strong implications here resulting from our inexact line search. Notice that these nonlinear CG methods have the characteristic of combining weighted information about the current and last iterate's gradient to compute the new descent direction, then find the minimiser along it, in this case through strong Wolfe conditions. By FR's

specific β^{FR}, whenever we encounter a poor descent direction, we begin a cycle of repeatedly small steps; particularly caused by the new direction being almost the same as its previous. Strong Wolfe conditions algorithm will try to find a better minimiser along that new slightly different direction but without significant progress. Plotting the iterates over the contours hints that in this case FR 'rides' the contour level sets until reaches the negative x-semiaxis, which of course and similarly to *Exercise 1*, is beyond the support of the logarithmic function. The relevant quantities we may use to support such hypothesis are α's generated by the strong Wolfe conditions, the β^{FR}'s and the sequence of angles θ_k's over iterates.

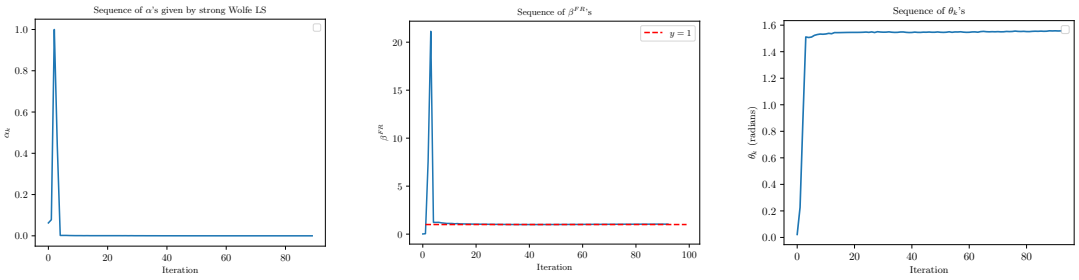


Figure 12: Conjugate Gradient Fletcher-Reeves α's, β^{FR}'s, θ_k's in iterates from $(2.5, 0.1)^\top$ for $f(x, y) = x \ln(x) + y^2 + c \exp -\frac{(x-a)^2+(y-b)^2}{\sigma^2}$.

As concluded above, we observe the sequence of steps scalings $\{\alpha_k\} \to 0$, the sequence of FR betas $\{\beta_k^{FR}\} \to 1$, and the sequence of angles of successive gradients and search directions $\{\theta_k\} \to \pi/2$, indicating that the steps are becoming small and the new descent directions are orthogonal to the negative gradient and thus poor directions. Finally, we remark that such a conclusion is in addition aligned with theoretical evidence of FR's erred performance.

(d) Perhaps it is worth mentioning at this point that PR is faster converging than FR (this will be further developed later). To better understand why this is the case, let us introduce the concept of 'restarts' in non-linear CG methods. This is a common modification that is made in order to avoid computing poor descent directions. Essentially, the mechanism involves periodically setting $\beta_k = 0$, thereby vanishing the process' memory of the previous search direction. An immediate consequence is that the new direction will be exactly the steepest descent. We avoid unravelling the proofs of such process and rather focus on the intuition behind it. Consider a function f strongly convex quadratic in the neighbourhood of its minimiser, and non quadratic elsewhere, then at some point iterates shall arrive at said neighbourhood and eventually prompt the restart. From restart, our nonlinear CG would behave as the linear CG and termination will occur at n steps of the restarts, for an n dimensional problem. This lies at the core of our theoretical guarantee for an n-step quadratic convergence. For our f, we find this is the case when starting in a close neighbourhood of the maximiser at $\boldsymbol{x}^* \approx (2.05439, 0)^\top$. From that point FR experiences a function that is non-quadratic for a couple iterations then arrives at the more quadratic region as it gets further from the hump. Despite we stated strongly convex quadratic f is required, in practice and through Taylor's Theorem we know it can still be approximated closely by a standard quadratic, provided f smooth (although convergence might no longer be achieved in $< n$ iterations). We wrap this theoretical frame by stating that PR does include some form of built-in restart, whereby any poor search direction satisfying orthogonality w.r.t gradient it employs $\nabla f_{k+1} \approx \nabla f_k$ s.t. $\beta_{k+1}^{PR} \approx 0$, then we get close to the steepest direction. In terms of the solution implementation, there are multiple ways to impose restarts, of which we choose and argue for the most

controversial. The most sensible way to implement it is by computing the orthogonality of two consecutive gradients and w.r.t some parameter. If they are far from orthogonal, a restart will be performed. On the other hand, a less political implementation involves restarts every n^{th} iteration, again n is the problem dimensionality. The issue with this paradigm is that for large dimensionality problems we cannot guarantee a restart will take place prior to finding an approximate solution, particularly because CG methods are better suited toward those cases. For our f, however, we find that with a low dimensionality, we can avoid the repeated computation of expensive linear algebra operations to check orthogonality at each point. Rather, we provide an arbitrary interval of iterates that will trigger each time a restart achieving similar results at lesser cost. Still, the first method is generally acknowledged as the better option.

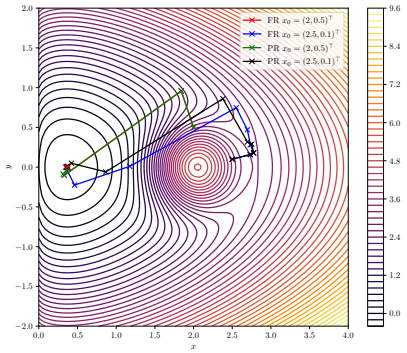
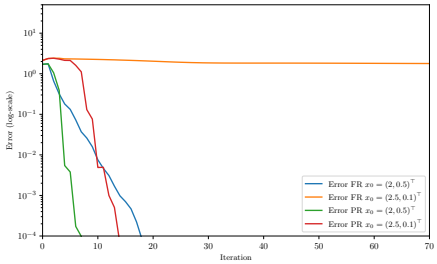
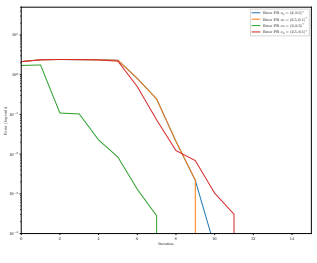


Figure 13: Conjugate Gradient FR, PR with Restarts every n iterations; $n = 2$; from $\boldsymbol{x}_0 = (2, 0.5)^\top, (2.5, 0.1)^\top$ for $f(x, y) = x \ln(x) + y^2 + c \exp -\frac{(x-a)^2+(y-b)^2}{\sigma^2}$.

Briefly, we now see convergence for both PR and FR from both starting points. We now are able to see FR escape from the non-convex region successfully, while all methods are generally converging faster to the minimiser. Below we will further investigate whether approximate the quadratic convergence of our strong theoretical result, or expand on it.



PR, FR, no restarts.



PR, FR restarts.

Figure 14: Convergence Plots of FR, PR With and Without Restarts from $\boldsymbol{x}_0 = (2, 0.5)^\top, (2.5, 01)^\top$ for $f(x, y) = x \ln(x) + y^2 + c \exp -\frac{(x-a)^2+(y-b)^2}{\sigma^2}$.

(e) We explain empirical results by referring to the relevant theoretical evidence. Generally, for our f and

choice of parameters, if CG with either Polak-Ribiere or Fetcher-Reeves methods converges, it will do so Q-linearly — again, linear relationship in normed error semi-log plots implies exponential convergence order, which in turn implies Q-linear convergence. Now, it is evident that for poor descent steps FR will not provide a practical optimisation solution, as per the reasons stated in *(c,d)*. A more thorough interpretation of the plots, however, allows us to read past the irregularities. Firstly, we are able to correspond the initial plateau of convergence plots to the methods' transition phase from the non-convex neighbourhood of initial points $\boldsymbol{x}_0 = (2, 0.5)^\top, (2.5, 0.1)^\top$, which we can observe in the level sets described by the contours to the convex region. Such non-convexity opens in itself new fronts of discussion; f starts reflecting the conditions for theoretical n-steps quadratic convergence — we have, as far as the method is informed, a function which becomes convex in a neighbourhood of the minimiser and is otherwise non-convex. Yet plots are Q-linear. We find by these same contours f not strongly convex quadratic (f-values), and we must set the case of approximating it by some standard quadratic thorugh Taylor's Thm. Then this violation of ideal conditions for quadratic convergence justifies the dichotomy between theoretical and empirical Q-linear behaviour. We find as well, slight deviations in the plots particularly for pure PR, and FR and PR with restarts. These deviations are exactly that, changes in convergece rate (not order) at points where resets take place. We note, at these we compute approximately the steepest descent, which is Q-linearly convergent as seen in 2.. Finally, and in addition to the above convergence result, the global convergence results are discussed. Notice f, despite non-convex, is bounded from below for $\forall c \in \mathbb{R}$ for the sublevel sets $\{\boldsymbol{x}_s \in \text{dom} f : f(\boldsymbol{x}) \leq c\}$, then again for those sublevel sets w.r.t the initial points, and f Lipschitz continuously differentiable in some open neighbourhood in them since they are bounded. Further we build iterative sequences of descent directions scaled by steps satisfying strong Wolfe conditions. Then for unrestarted nonlinear CG method we know by Al-Baali Theorem we can only guarantee a weak convergence for parameters conforming $1 < c_1 < c_2 < 0.5$ and FR algorithm. More generally, for PR we need strongly convex function for a similar result. For restarted methods we recall Zoutendijk lemma in addition to Wolfe conditions, and are able to achieve weak global convergence as well.

Exercise 5

(a)

```
y: np.array = F.df(x_k1) - F.df(x_k)                                    # gradient change
s: np.array = x_k1 - x_k                                                # iterate state change
rho = 1 / np.dot(y, s)                                                  # BFGS update parameter
H: np.array = np.eye(len(x_k)) - rho * (np.outer(s, y)) @ info["Hs"][-1] @ (
    np.eye(len(x_k)) - rho * (np.outer(y, s)) + rho * (np.outer(s, s))
)                                                                       # BFGS inverse Hessian update
info["Hs"].append(H)                                                    # store value of H
```

Figure 15: Code for BFGS updates.

The above implementation is particularly efficient compared to the strict method since it avoids directly computing Hessian matrix inverse. Instead we compute at most vector-vector or matrix-vector operations resulting in an overall lower computational complexity. Additionally we choose the recommended initial approximate Hessian guess of the Identity (or diagonal) matrix, which if computed as a sparse matrix (not in this case, since only numpy was employed overall to reduce overhead, which does not support sparse matrices) further reduces the toll on computational complexity. Overall complexity in Big-O notation for

this implementation is $\mathcal{O}(n^2)$, as opposed to $\mathcal{O}(n^3)$ for standard inverse matrix operation.

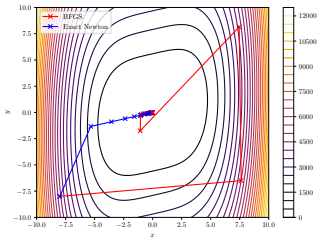


Figure 16: BFGS and Exact Newton plots from $\boldsymbol{x}_0 = (-8, -8)^\top$ for $f(x, y) = (x - 4y)^2 + x^4$.

(b) BFGS method converges well to the minimiser at $\boldsymbol{x}^* = (0, 0)^\top$ to a precision according to the chosen tolerance at about $1e^{-4}$, i.e., 4 significant figures. Our choice of a line search is again rooted in the literature. We know that BFGS has self correcting properties, in the sense that it will overcome, unlike unrestarted FR mehtod before, poor steps due to equally poor approximations to the Hessian. However, this behaviour follows from an adequate line search, where Wolfe conditions stand out, since they ensure that the gradients are sampled at points which do convey curvature information (this is by checking for Curvature Conditions, see *2*) for the quadratic model, m_k. In this context, relevant parameters are the same as for WC exact Newton in *2*, and the parameters remain unchanged; $\alpha_0 = 1, c_1 = 1e^{-4}, c_2 = 0.9, maxIter = 100$

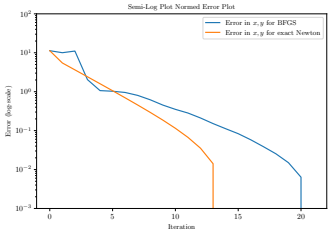


Figure 17: BFGS and Exact Newton Normed Error Sequence Plots from $\boldsymbol{x}_0 = (-8, -8)^\top$ for $f(x, y) = (x - 4y)^2 + x^4$.

(c) We compare two methods above, exact Newton and BFGS, to better portray the convergence behaviour of the latter. Now, let f twice continuously differentiable, and the sequence generated by the iterates of BFGS algorithm converge to some minimiser $\boldsymbol{x}^*$, with $\nabla^2 f$ Lipschitz continuous and s.p.d in a neighbourhood of such minimiser, $\mathcal{N}(\boldsymbol{x}^*)$. If it holds that $\sum_{k=1}^{\infty} ||\boldsymbol{x}_k - \boldsymbol{x}^*|| < \infty$, then $\boldsymbol{x}_k$ converges to $\boldsymbol{x}^*$ at a superlinear rate. For our f, we know that $\exists \nabla^2 f$, implying f twice continuously differentiable, and that since it is a polynomial, bounding it to a region of the domain will also bound it, implying local Lipschitz continuity as well around our minimiser. Yet we observe for both exact Newton and BFGS a linear relationship in a semi-log plot, implying Q-linear convergence, and conflicting with overall theoretical results. The main reason for plotting the exact Newton method as well was to highlight the basin that is formed for our f in the neighbourhood of the minimser. Clearly, the level sets hint at the existence of a very flat region, since f has almost no curvature along the y direction; once in this neighbourhood of the minimiser, we get $\gg \nabla f, \nabla^2 f$, with the Hessian becoming close to indefinite, in fact, $\nabla^2 f(x^*) \succeq 0$ is positive semidefinite only, and conditions for exact Newton Q-quadratic and quasi-Newton Q-superlinear convergence are violated. Further, BFGS has a lower convergence rate than exact Newton still, which is explained by the approximate Hessian of BFGS not capturing all of the curvature information. We highlight the irregularities in the first iterations for BFGS,

and refer to its self-correction properties for our choice of line search with WC. The conclusion is as follows: our initial Hessian guess is taken to be the identity matrix, implying steepest descent is computed as descent direction for the first iterate. At the origin, we find a negative gradient value that is not aligned with the minimiser, which we can also conclude from the level sets given by the contours; after some updates the erratic behaviour ceases due to the gathered information about the function curvature.

(d) Regarding global convergence of quasi-Newton methods, there is no general result; Focusing on BFGS, we can outline some conclusions. Due to the BFGS Global Convergence Theorem, we know for f twice continuously differentiable and $\boldsymbol{x}_0$ be a starting point for which the sublevel set is convex, and there exist two constants $m, M > 0$ s.t.

$$m||z||^2 \leq z^\top \nabla^2 f(x) z \leq M||z||^2, \quad \forall z \in \mathbb{R}^n, x \in \mathcal{L}.$$

Then for $\forall B_0 \in \mathcal{S}_d^+$, s.p.d., the sequence of iterates generated by BFGS algorithm $\{\boldsymbol{x}_k\}$ with $\varepsilon = 0$, converges to a unique minimiser (unique since f convex in the sublevel set) $\boldsymbol{x}^* \in \mathcal{L}$, ($\nabla^2 f(\boldsymbol{x}) \in \mathcal{S}_d^+$ in the sublevel set). Still, by Zoutendijk, we can achive weak global convergence for BFGS method, i.e.,

$$\liminf ||\nabla f_k|| \to 0$$

For our convex f, even though it is a homogeneous fourth order polynomial with positive coefficients, bounded from below, Lipschitz continuous, twice differentiable... it does not satisfy a s.p.d. Hessian everywhere for sublevel sets, particularly evidenced by the actual minimiser having a positive semidefinite Hessian, implying that neighbouring points will as well. Hence for our line search choice of WC we cannot guarantee global convergence.

As a final remark, all theoretical content has been taken from the Moodle slides or the book by Jorge Nocedal and Stephen J. Wright, Numerical Optimisation (2006).