Assignment 1: Computational Methods of Optimization (E0-230)

Sacchit Kale
SR No: 22219

September 2024

Contents

1 Convex and Coercive Functions

1.1 Convexity Check for f_1 and f_2

a) The functions f_1 and f_2 were tested for convexity over the interval $[-2, 2]$. The method used to check convexity involved checking criteria of Convexity.

$$f(\lambda x_1 + (1-\lambda)x_2 \leq \lambda f(x_1) + (1-\lambda)f(x_2) \; \forall x \in [-2, 2]$$

.
The interval [-2,2] was divided in interval of equal length , the definition of convexity was checked for each point in the interval for $\lambda = 0.5$
The following results were obtained:

- f_1 is convex on the interval.
- f_2 is convex on the interval.

b)For Strong Convexity the below given strict inequality was checked .
.

$$f(\lambda x_1 + (1-\lambda)x_2 < \lambda f(x_1) + (1-\lambda)f(x_2) \; \forall x \in [-2, 2]$$

The following results were obtained:

- f_1 is not strictly convex on the interval.
- f_2 is strictly convex on the interval.

Since we only have access to the function values of f_1 and f_2 at specific points, and no direct access to their derivatives or internal structure, a grid search method was employed to find the minimum of the functions within a specified interval. The grid search involves evaluating the function at evenly spaced points in the interval and identifying the point where the function achieves its minimum.
Using the grid search method, we identified the point of minimum x^* and the corresponding function value $f(x^*)$ for both f_1 and f_2. The results are as follows:

- For f_1, the minimum point x^* was found to be `[-0.30303030]` with a function value of $f(x^*) =$ `[0.0064]`.Note minima isnt unique
- For f_2, the minimum point x^* was found to be `[-2]` with a function value of $f(x^*) =$ `[ 0.21653645317858033]`.Minima is unique since f_2 is strictly convex.

1.2 Coercivity Check for f_3

a) The function f_3, a quartic polynomial, was tested for coercivity by analyzing its coefficients. To determine the coefficients, a Vandermonde matrix was used, which allows us to express the polynomial in terms of powers of x. For large values of $|x|$, the behavior of the polynomial is dominated by the term with the highest degree (the fourth-order term). If the coefficient of the fourth-order term $a > 0$, this means that as $|x|$ tends to ∞, the term ax^4

grows without bound. Therefore, the function f_3 tends to infinity as $|x|$ increases, which implies that f_3 is coercive,if $a < 0$ f_3 will tend to -∞ hence f_3 would not be coercive. Here $a = 0.125 > 0$ In summary:

- The fourth-order term dominates the polynomial for large $|x|$.
- Since $a > 0$, $f_3(x) \to \infty$ as $|x| \to \infty$, which satisfies the definition of a coercive function.

This reasoning confirms that f_3 is coercive.
b) To find the roots and stationary points of the quartic polynomial f_3, we used the following methods:

- **Roots:** The roots of f_3 were determined by solving the equation $f_3(x) = 0$. This was achieved using numerical methods, specifically by applying a root-finding algorithm such as Python's `numpy.roots()` function, which computes the roots of a polynomial by solving its characteristic equation.
- **Stationary Points:** By definition Stationary points are points where f' is 0 or undefined,here f_3 is a quartic polynomial hence f' will be defined.The stationary points of f_3 were determined by first computing its first derivative, $f_3'(x)$. The stationary points are where $f_3'(x) = 0$. We solved this equation to find the critical points. Further, the second derivative, $f_3''(x)$, was used to classify these points as minima, maxima, or saddle points:
  - If $f_3''(x) > 0$ at a stationary point, it indicates a local minimum.
  - If $f_3''(x) < 0$, it indicates a local maximum.

The stationary points and roots of f_3 were calculated and classified accordingly, providing insight into the function's behavior at critical points.

Roots	Minima	Local Maxima
2.416543, -2.002636, 0.690238, -0.404145	1.7988731352792562	0.14674195725992265

Table 1: Stationary Points of f_3

2 Gradient Descent Methods

2.1 Gradient Descent with Fixed Step Size

Using the function $f_4(x) = \frac{1}{2}x^T Ax + b^T x$, gradient descent was implemented with a fixed step size. The step size $\alpha = 10^{-5}$ was chosen, and the algorithm was run for 10000 iterations. The following results were obtained:

x^*	$f(x^*)$
-4.50065259e-05,-5.0000e-04,-1.00000000e-03,-2.00000e-03, -9.99954827e-03	-0.006772503252769977

Table 2: Results of Constant Gradient Descent

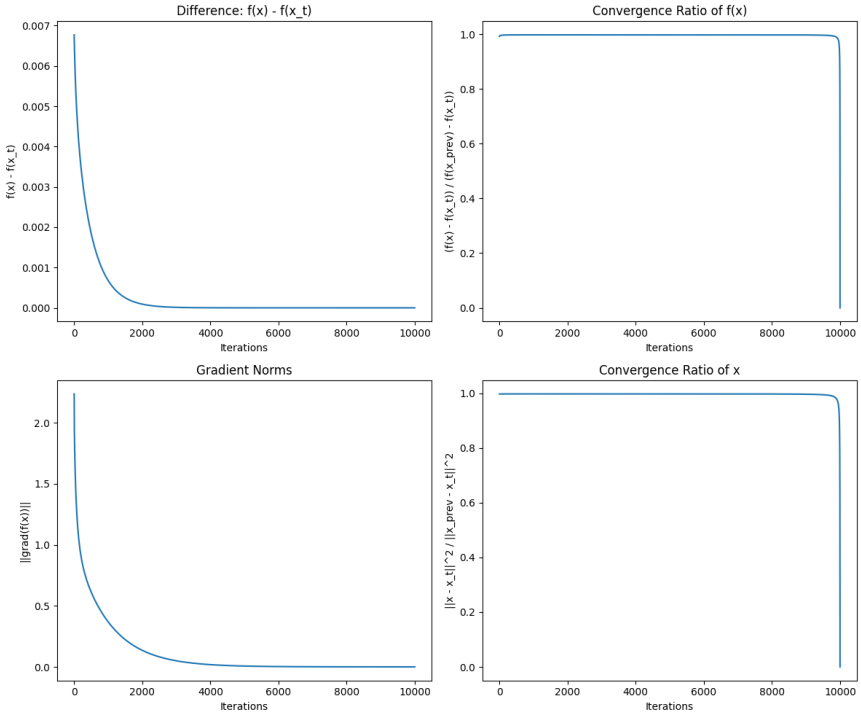

Figure 1: Gradient Descent with Fixed Step Size

2.2 Gradient Descent with Diminishing Step Size

A diminishing step size was applied as $\alpha_k = \frac{10^{-3}}{k+1}$. The results for $T = 10000$ iterations are compared with the fixed step size results.Value of x^* *and* $f(x^*)$ obtained from Diminishing Gradient Descent are different than that obtained from Constant Gradient Descent.The reasons for different values could be slower rate of convergence in Diminishing Gradient Descent and α in Diminishing Gradient Descent becomes too small for meaningful descent .After 10,000 Iterations Diminishing Gradient Descent hasn't converged as $\nabla f(x_T) = 0.37258498993091516$ which is not zero.

x^*	$f(x^*)$
-4.50065259e-05,-5.0000e-04,-1.00e-03,-1.98871635e-03, -6.27461458e-03	-0.006078546604756475

Table 3: Results of Diminishing Gradient Descent

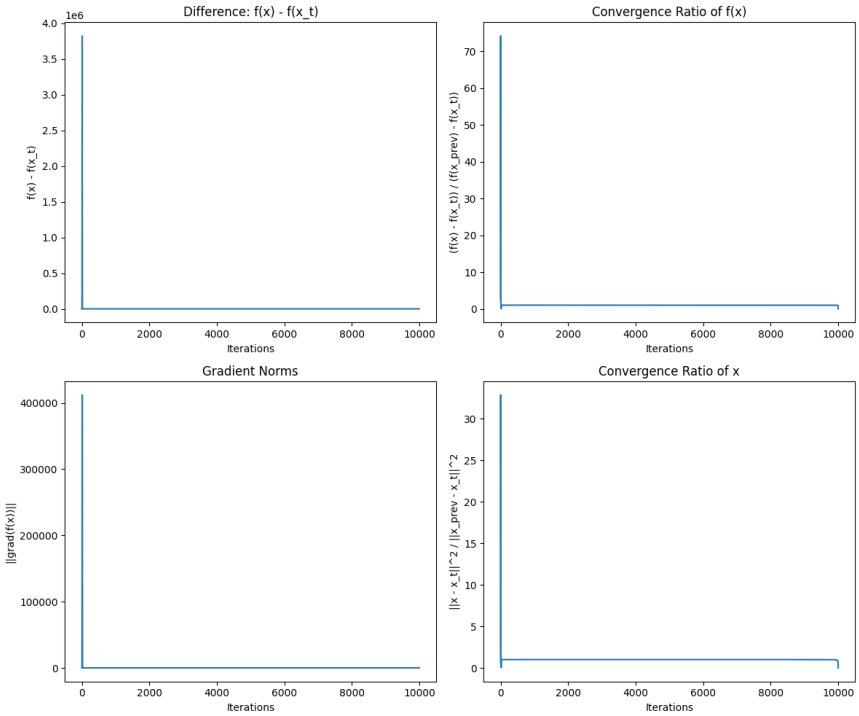

Figure 2: Gradient Descent with Diminishing Step Size

2.3 Inexact Line Search Using Wolfe Conditions

The Wolfe conditions were used to dynamically adjust the step size at each iteration. The function `InExactLineSearch(c1, c2, gamma)` was implemented, and the following results were obtained:

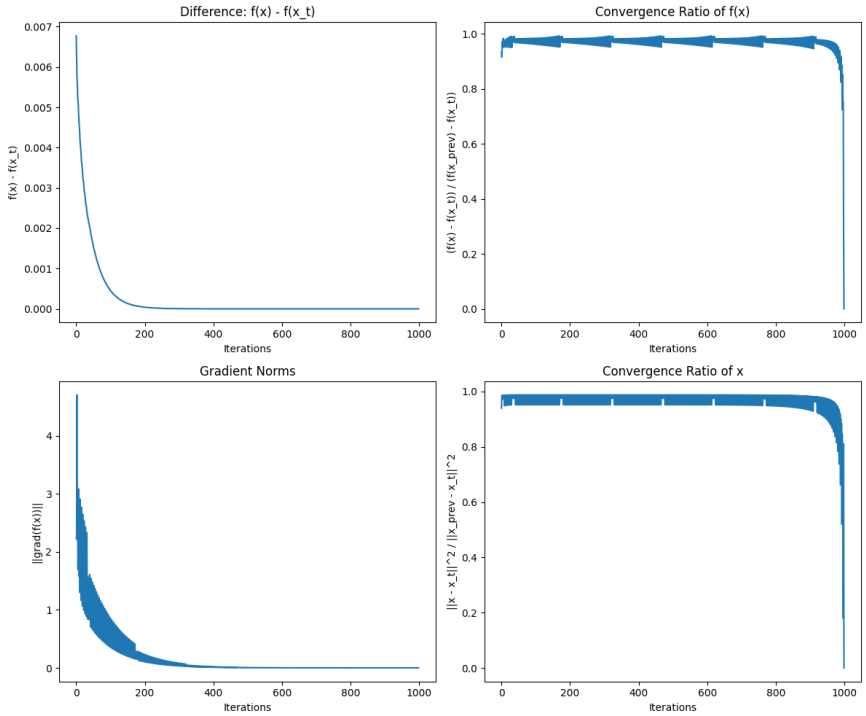

Figure 3: Gradient Descent with Inexact Line Search

x^*	$f(x^*)$
-4.50061313e-05, -5.00000000e-04, -1.00000000e-03, -2.00000000e-03, -9.99994835e-03	-0.006772503262

Table 4: Results of Inexact Line Search

2.4 Exact Line Search

The exact line search was implemented using the formula $\alpha_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T A p_k}$ for 1000 iterations.

x^*	$f(x^*)$
-4.50061313e-05,-5.000e-04,-1.00000e-03,-2.00000e-03, -9.99994835e-03	-0.006772503262838028

Table 5: Results of Exact Line Search

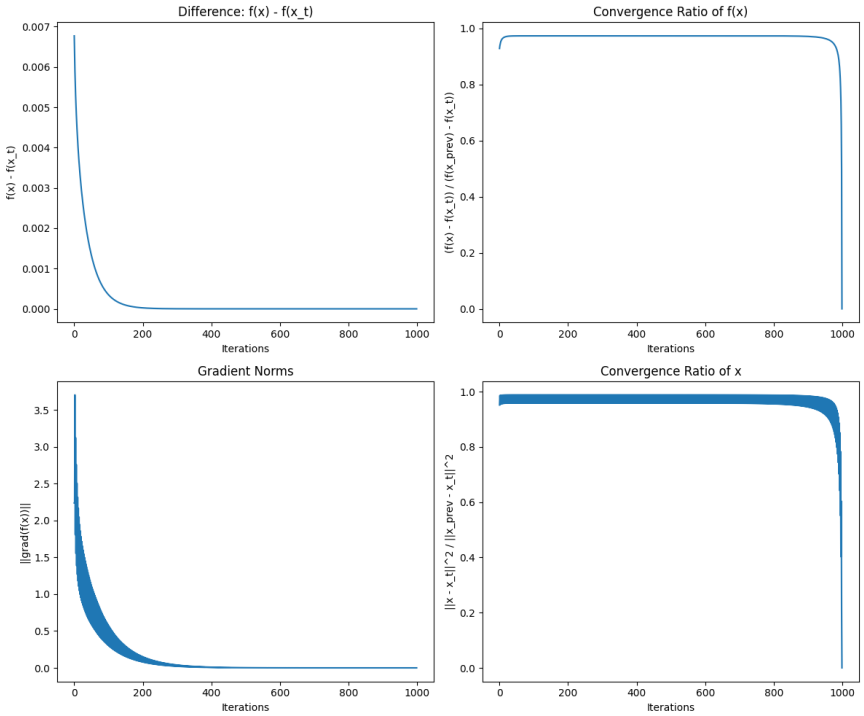

Figure 4: Gradient Descent with Exact Line Search

3 Perturbed Gradient Descent

Perturbed gradient descent was applied to avoid saddle points. The function $f(x, y) = e^{xy}$ was optimized, and the following experiments were conducted:

- Gradient descent with fixed step size
- Gradient descent with decreasing step size
- Gradient descent with fixed step size and fixed noise variance
- Gradient descent with fixed step size and decreasing noise variance
- Gradient descent with decreasing step size and fixed noise variance
- Gradient descent with decreasing step size and decreasing noise variance

3.1 Stationary Points of f(x,y)

o find the stationary points of $f(x, y) = e^{xy}$, we compute the partial derivatives with respect to x and y:

$$f_x(x, y) = \frac{\partial}{\partial x} e^{xy} = ye^{xy}$$

$$f_y(x, y) = \frac{\partial}{\partial y} e^{xy} = xe^{xy}$$

Setting both partial derivatives to zero, we get the system of equations:

$$ye^{xy} = 0$$

$$xe^{xy} = 0$$

Since $e^{xy} \neq 0$ for all real x and y, we conclude that:

$$x = 0 \quad \text{and} \quad y = 0$$

Thus, the stationary point for f(x,y) is (0,0),
To determine what kind of stationary point (0,0) is we need to evaluate the Hessian. The Hessian matrix is composed of the second-order partial derivatives of the function $f(x, y) = e^{xy}$.

First, we compute the first partial derivatives:

$$f_x(x, y) = \frac{\partial}{\partial x} e^{xy} = ye^{xy}$$

$$f_y(x, y) = \frac{\partial}{\partial y} e^{xy} = xe^{xy}$$

Now, we compute the second partial derivatives:

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(ye^{xy}\right) = y^2 e^{xy}$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial y}\left(ye^{xy}\right) = (1 + xy)e^{xy}$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}\left(xe^{xy}\right) = x^2 e^{xy}$$

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial x}\left(xe^{xy}\right) = (1 + xy)e^{xy}$$

Thus, the Hessian matrix is:

$$H(f) = \begin{pmatrix} y^2 e^{xy} & (1 + xy)e^{xy} \\ (1 + xy)e^{xy} & x^2 e^{xy} \end{pmatrix}$$

Now, evaluating at the point $(0, 0)$:

$$H(f)\Big|_{(0,0)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The eigenvalues of this matrix are ± 1 ,hence (0,0) is a saddle point.

3.2 Trajectory of G.D for x=y initial points

Consider the function $f(x, y) = e^{xy}$. Let us show that if we start with $x_0 = y_0$, gradient descent will always lie on the line $x = y$.

The gradient of $f(x, y)$ is given by:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) = (ye^{xy}, xe^{xy})$$

At any point where $x = y$, we have:

$$\nabla f(x, x) = \left(xe^{x^2}, xe^{x^2} \right)$$

The direction of the gradient is proportional in both the x- and y-directions. Therefore, if we start with $x = y$, the update step in gradient descent will maintain $x = y$ because both components of the gradient are equal. Thus, the update rule:

$$x_{k+1} = x_k - \alpha \frac{\partial f}{\partial x}, \quad y_{k+1} = y_k - \alpha \frac{\partial f}{\partial y}$$

will update both x and y by the same amount, keeping $x = y$ in all iterations.

Hence, starting with $x = y$, gradient descent will always lie on the line $x = y$.

3.3 Contour Plot

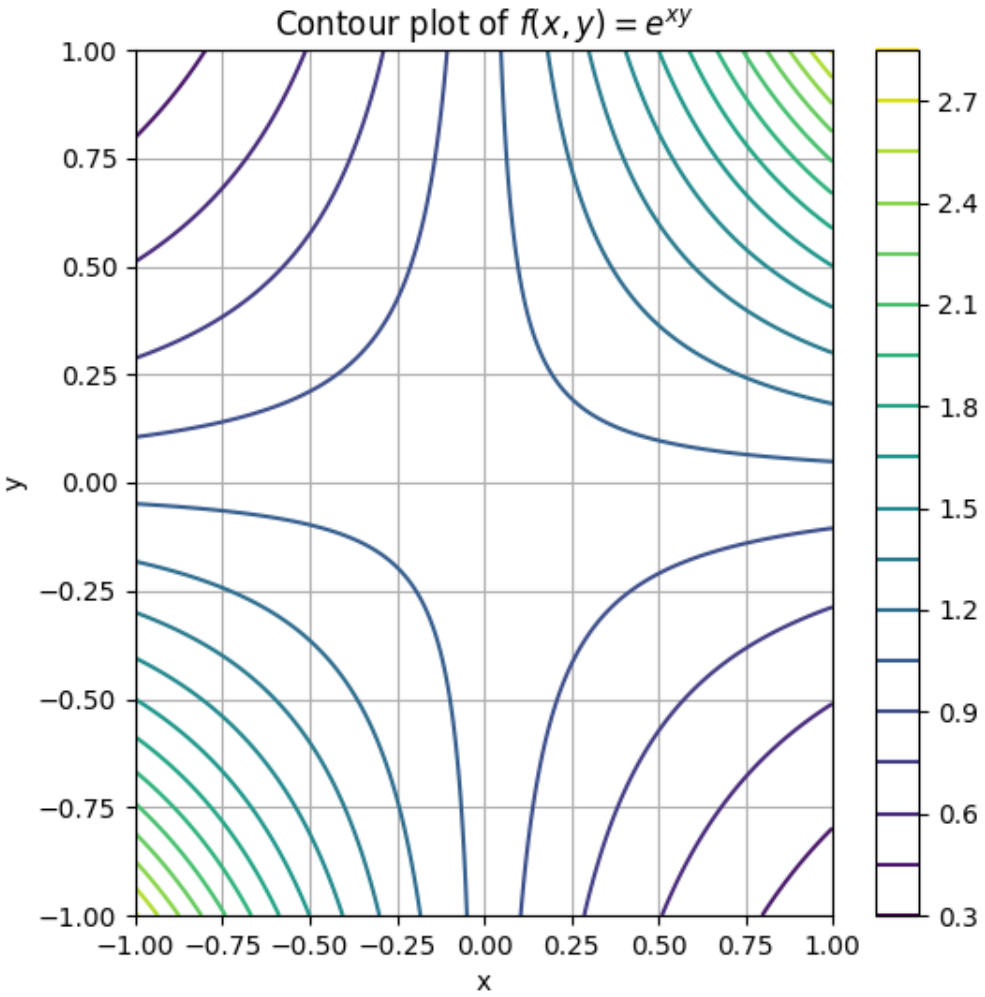

Figure 5: Contour Plot of f = e^{xy}

3.4 Gradient Descent with fixed Step Size

Gradient Descent was ran for 10,000 Iterations with $\alpha = 10^{-3}$, Algorithm Converged to the Saddle Point.

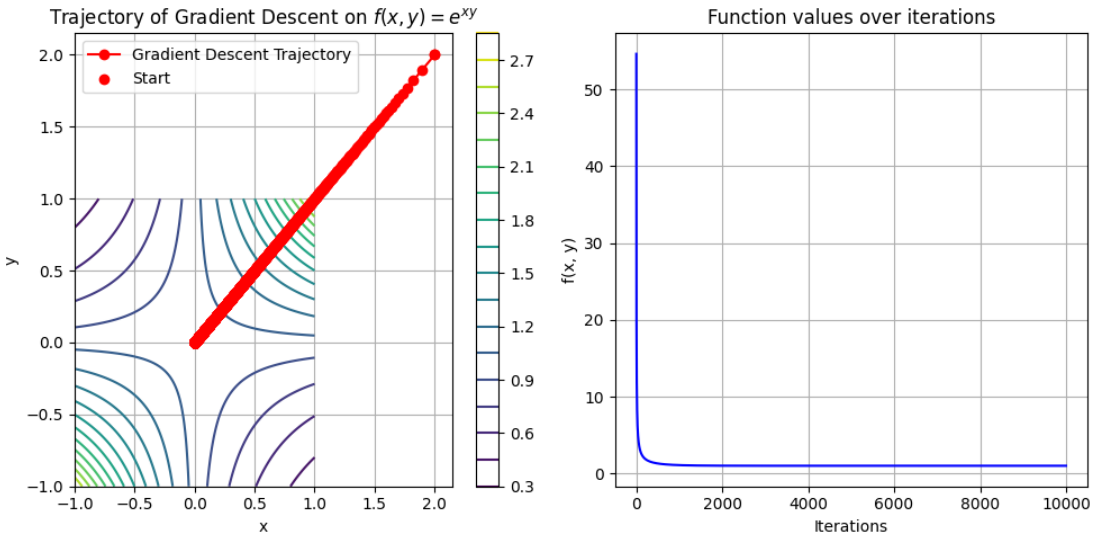

Figure 6: 3.4 Gradient Descent with fixed Step Size Plot

3.5 Gradient Descent with Decreasing Step Sizes

Diminishing Gradient Descent was ran for 10,000 iterations with $\alpha_t = \frac{10^{-3}}{t+1}$,Algorithm didn't Converge as Update Steps became to small.

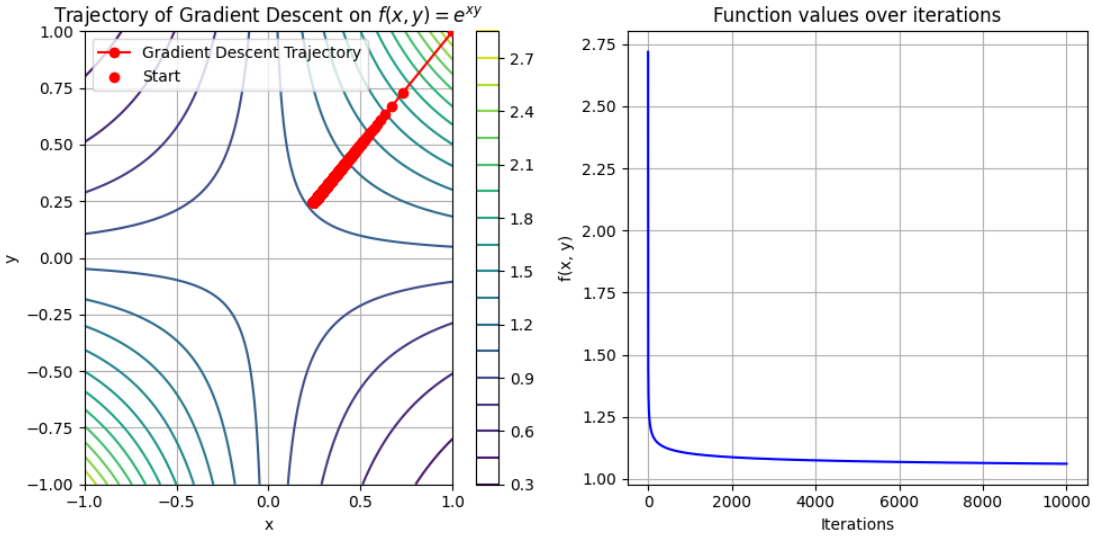

Figure 7: 3.5 Gradient Descent with Decreasing Step Sizes

3.6 Gradient Descent with fixed step size and fixed variance

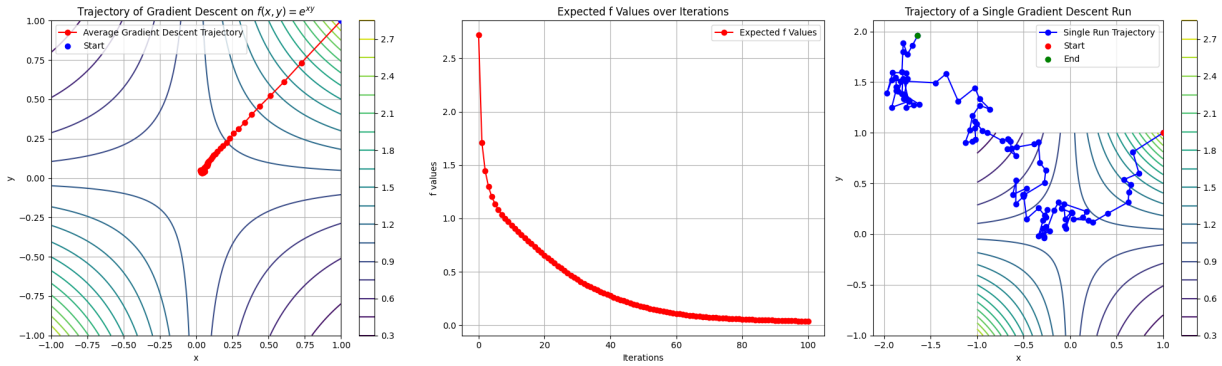

Figure 8: 3.6 Gradient Descent with fixed step size and fixed variance

3.7 Gradient Descent with fixed step size and decreasing variance

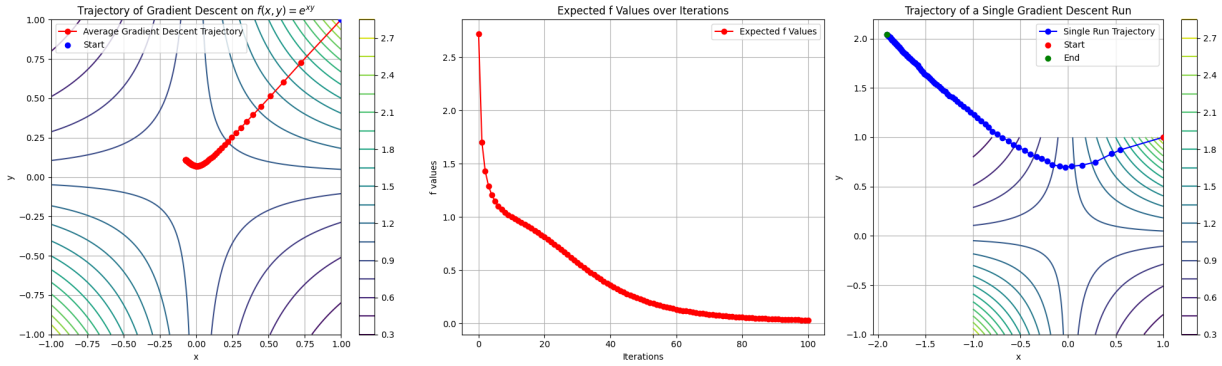

Figure 9: 3.7 Gradient Descent with fixed step size and decreasing variance

3.8 Gradient Descent with fixed variance and decreasing step size

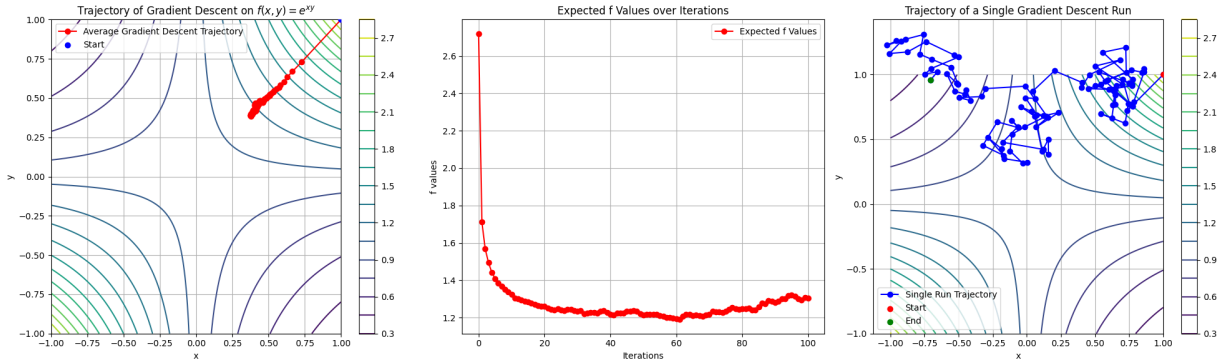

Figure 10: 3.8 Gradient Descent with fixed variance and decreasing step size

3.9 Gradient Descent with decreasing step size and decreasing variance

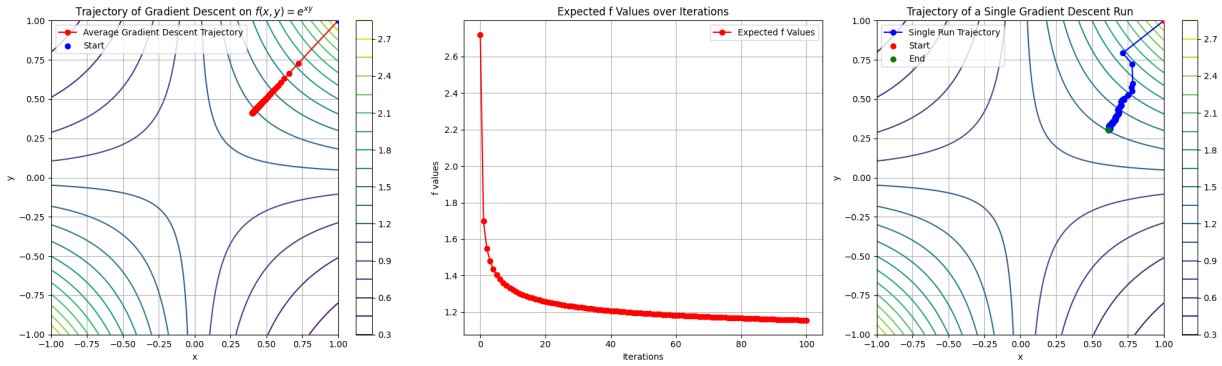

Figure 11: 3.9 Gradient Descent with decreasing step size and decreasing variance

3.10 Relation Between $\mathbb{E}[f^{(t+1)}]$ *and* $\mathbb{E}[f^{(t)}]$

In this question, we analyze how the expected value of the function changes over iterations during perturbed gradient descent. The update rule for perturbed gradient descent is given by:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha_t \left(\nabla f(\theta^{(t)}) + \zeta^{(t)} \right),$$

where α_t is the step size, $\nabla f(\theta^{(t)})$ is the gradient of the function at iteration t, and $\zeta^{(t)}$ is the added noise, often modeled as $\zeta^{(t)} \sim N(0, \sigma^2 I)$.

First-order Approximation

Using the first-order approximation of the function, we have:

$$f(\theta^{(t+1)}) \approx f(\theta^{(t)}) - \alpha_t \|\nabla f(\theta^{(t)})\|^2 + \alpha_t \nabla f(\theta^{(t)})^\top \zeta^{(t)}.$$

Since $\zeta^{(t)}$ is zero-mean noise, taking the expectation of both sides gives:

$$E[f(\theta^{(t+1)})] \approx E[f(\theta^{(t)})] - \alpha_t E[\|\nabla f(\theta^{(t)})\|^2].$$

Second-order Approximation

When we include second-order information using the Hessian $H(\theta) = \nabla^2 f(\theta)$, we expand the function as:

$$f(\theta^{(t+1)}) \approx f(\theta^{(t)}) - \alpha_t \nabla f(\theta^{(t)})^\top \left(\nabla f(\theta^{(t)}) + \zeta^{(t)}\right) + \frac{1}{2}\alpha_t^2 \left(\nabla f(\theta^{(t)}) + \zeta^{(t)}\right)^\top H(\theta^{(t)}) \left(\nabla f(\theta^{(t)}) + \zeta^{(t)}\right).$$

Taking the expectation, we get:

$$E[f(\theta^{(t+1)})] \approx E[f(\theta^{(t)})] - \alpha_t E[\|\nabla f(\theta^{(t)})\|^2] + \frac{1}{2}\alpha_t^2 \left(E[\|\nabla f(\theta^{(t)})\|_H^2] + \sigma^2 \mathrm{Tr}(H(\theta^{(t)}))\right).$$

Here, $\|\nabla f(\theta^{(t)})\|_H^2 = \nabla f(\theta^{(t)})^\top H(\theta^{(t)}) \nabla f(\theta^{(t)})$, and $\sigma^2 \mathrm{Tr}(H(\theta^{(t)}))$ accounts for the noise impact.

The expected change in function value at iteration $t+1$ depends on: - The first-order gradient term $\|\nabla f(\theta^{(t)})\|^2$, - The curvature of the function $\|\nabla f(\theta^{(t)})\|_H^2$, - The noise variance σ^2 and the trace of the Hessian $\mathrm{Tr}(H(\theta^{(t)}))$.

This shows how the second-order approximation modifies the basic first-order relationship by including the curvature and noise effects. Using From 1st Order approximation we get:

$$E[f(\theta^{(t+1)})] \leq E[f(\theta^{(t)})]$$

This implies expected value of f decreasing with t.

.

4 Zeroth Order Optimization

4.1 Extremas of f

The function is given as:

$$f(x) = x(x-1)(x+2)(x-3)$$

To find the extreme points, we first differentiate the function:

$$f'(x) = \frac{d}{dx}\left[x(x-1)(x+2)(x-3)\right] = 4x^3 - 6x^2 + 10x - 6$$

We then set the derivative equal to zero to find the critical points:

$$f'(x) = 4x^3 - 6x^2 + 10x - 6 = 0$$

Solving for x will give us the critical points. These critical points can be further analyzed using the second derivative test to determine the nature of the extreme points.
Solving for x gives x = 2.30277564 ,-1.30277564 , 0.5
$f''(x) = 12x^2 - 12x - 10$,f"(2.30277564) and f"(-1.30277564) > 0 ,f"(0.5) < 0.
This Implies 2.30277564 ,-1.30277564 are local minima and 0.5 is local maxima.
Hence 2.30277564 ,-1.30277564 , 0.5 are extremas of f.

4.2 Fibonacci search and Golden Section Search

a)**Number of iterations required to reach specified precision in Golden Section Search = 21**
x^* = **2.3027522715254074 and f(x^*)= -8.999999992902357**

b)**Number of iterations required to reach specified precision in Fibonacci Search = 21**
x^* = **2.3027522715254074 and f(x^*) =-8.999999992902357**

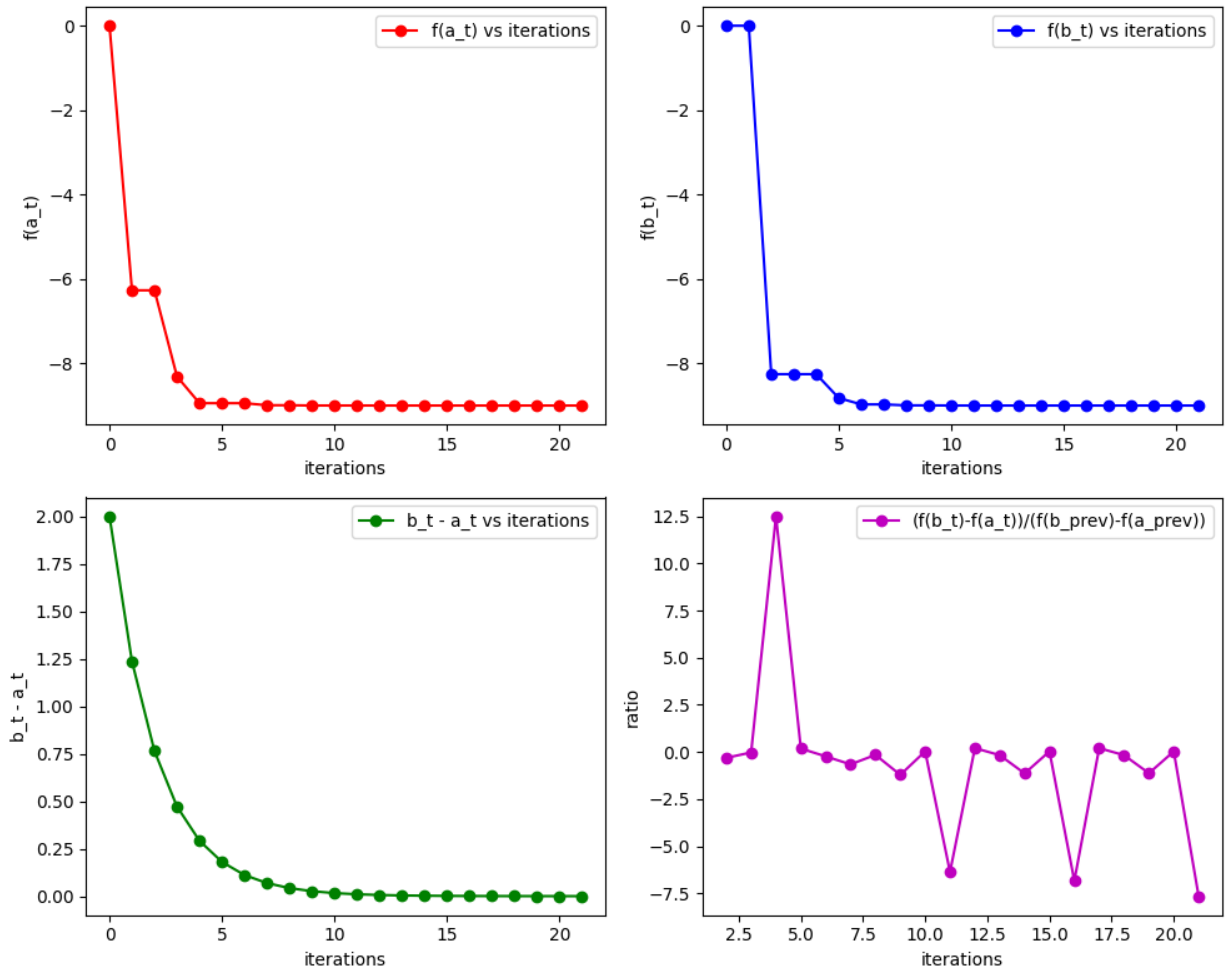

Figure 12: Plots for Q4 Golden Section Search

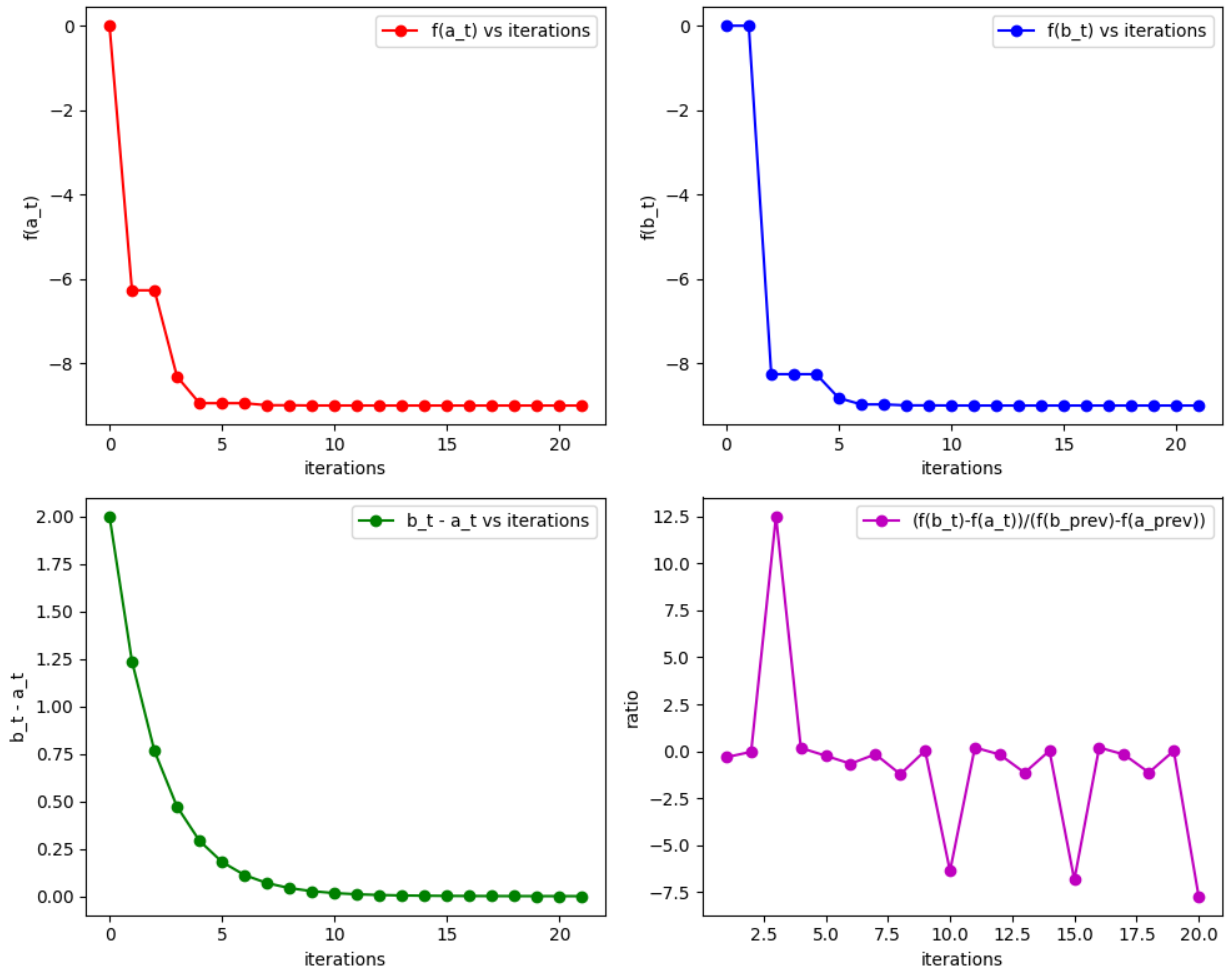

Figure 13: Plots for Q4 Fibonacci Search