Analysis of Gradient Descent Implementations

Tanishq Hindoliya (BSD-BG-2421)

May 10, 2025

Abstract

This report summarizes the results of five R scripts implementing gradient descent for various optimization problems. Each section details the problem, convergence plots, final solution, number of iterations, and observations. The reader may access the used R files from this repo

Contents

1 Problem 1: Linear Regression (1.R)

1.1 Objective

The script `1.R` implements gradient descent to find the optimal parameters (β_0, β_1) for a simple linear regression model $y = \beta_0 + \beta_1 x + \epsilon$. The loss function is $f_0(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$.

1.2 Plots of Convergence

The script generates the following plots. Please ensure these are saved and linked correctly.

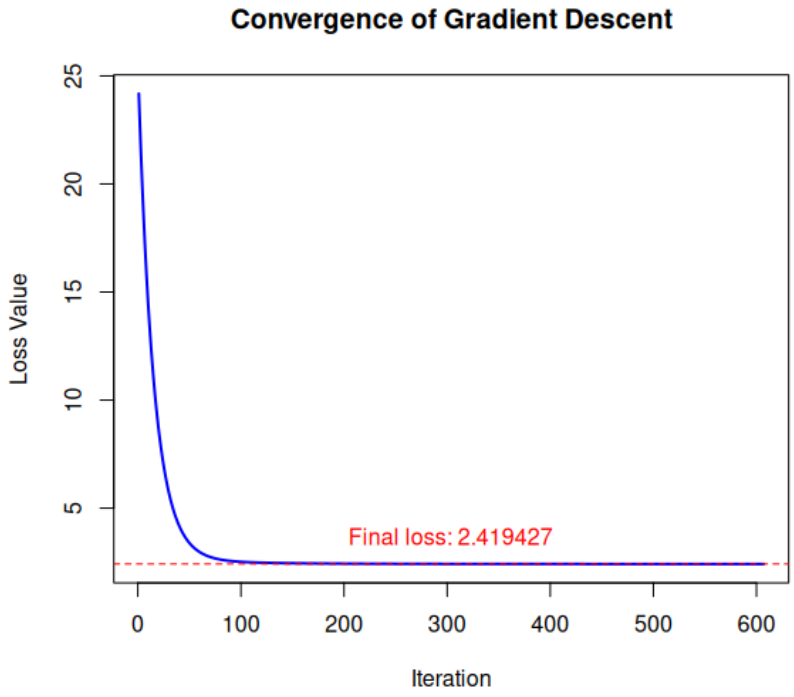

Figure 1: Loss value vs. Iterations for Linear Regression.

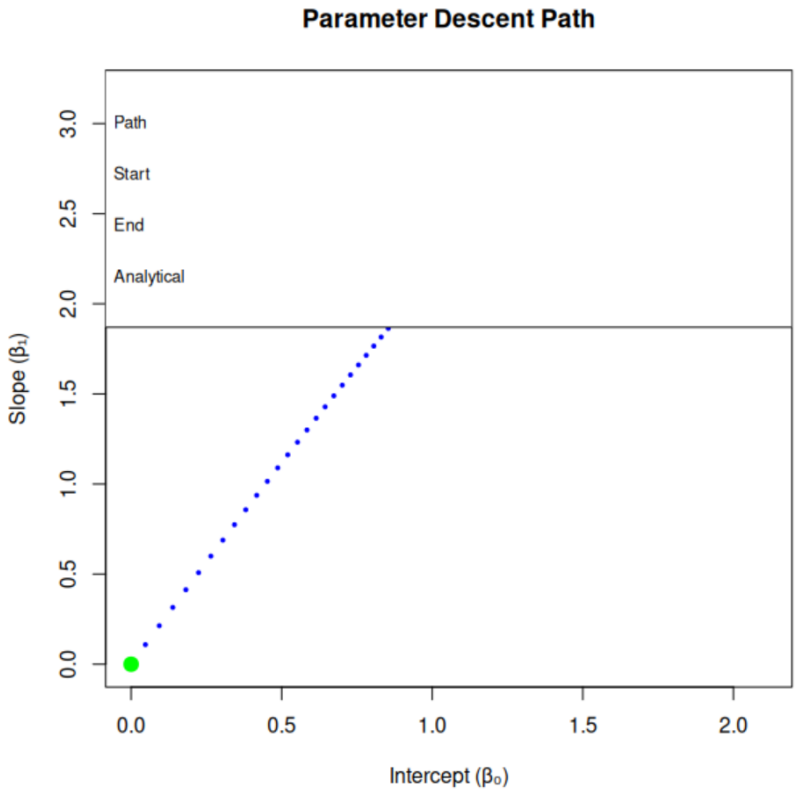

Figure 2: Parameter Descent Path (β_0 vs. β_1) for Linear Regression.

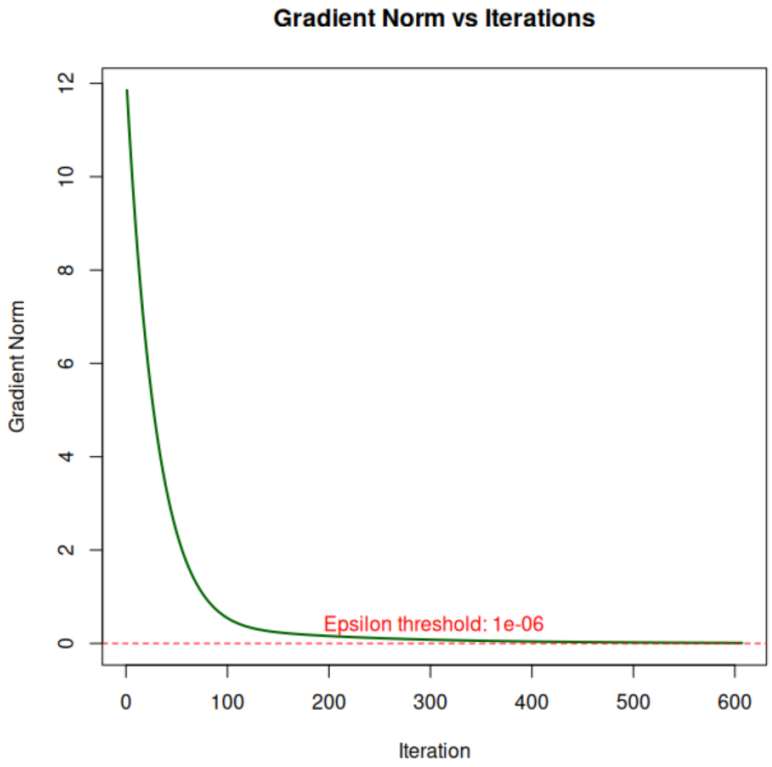

Figure 3: Gradient Norm vs. Iterations for Linear Regression.

1.3 Final Solution and Number of Iterations

The script `1.R` outputs the final parameters and iteration count. Based on its structure, the output will be similar to:

```
Optimal parameters:
Intercept (): 2.099838
Slope (): 3.090087
Final loss value: 2.723825
Number of iterations needed: 722
```

The script also compares these results with R's built-in `lm()` function.

1.4 Observations and Difficulties

- The algorithm converges by monitoring the gradient norm and the change in loss value.
- The learning rate (`eta`) is a critical hyperparameter. A value of 0.01 is used.
- The solution is compared against an analytical solution and R's `lm()` function, providing a good benchmark.
- Potential difficulties with gradient descent, as noted in the script, include:
  - Learning rate selection: too large can cause divergence, too small leads to slow convergence.
  - Sensitivity to initial values and feature scaling (though less critical for this simple 2-parameter problem with well-behaved data).
  - For linear regression, the loss function is convex, so local minima are not an issue.

2 Problem 2: Logistic Regression (2.R)

2.1 Objective

The script `2.R` implements gradient descent to find the optimal parameters (β) for a logistic regression model. It minimizes the negative log-likelihood function.

2.2 Plots of Convergence

The script generates plots for loss convergence and the decision boundary.

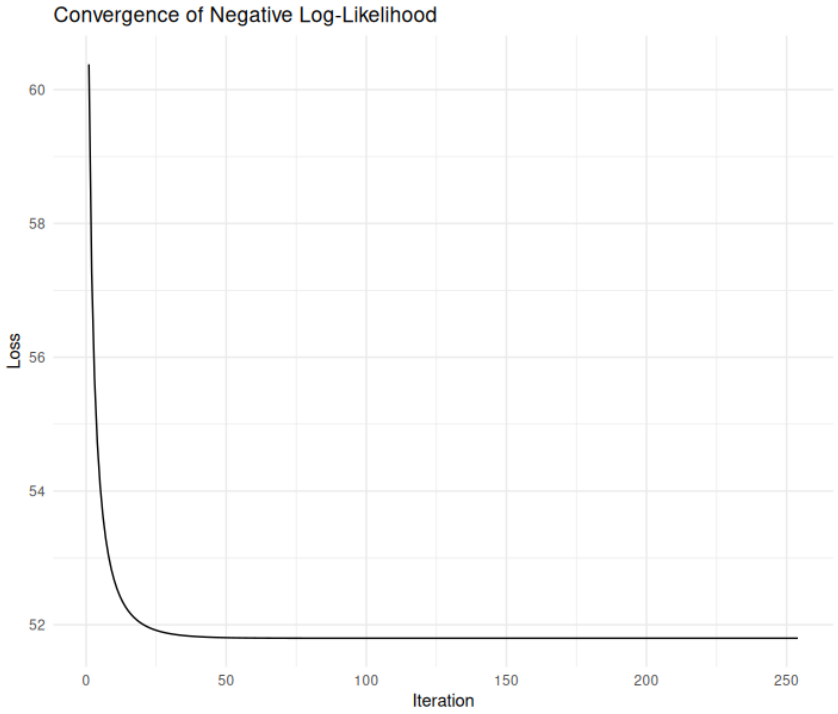

Figure 4: Convergence of Negative Log-Likelihood for Logistic Regression.

Figure 5: Logistic Regression Decision Boundary.

2.3 Final Solution and Number of Iterations

The script `2.R` outputs the final parameters and iteration count. The output format is:

```
Final beta values: 2.384689, -2.855389
Number of iterations required: 254
Final negative log-likelihood: 51.79935
Accuracy: 82%
```

2.4 Observations and Difficulties

- The script uses a sigmoid function and minimizes the negative log-likelihood.
- Convergence is determined when the gradient norm falls below a threshold ϵ.
- The learning rate $\eta = 0.05$ and threshold $\epsilon = 10^{-5}$ are used.
- Model accuracy is calculated as a performance metric.
- Visualizations are created using `ggplot2`.
- Logistic regression also has a convex loss function (negative log-likelihood), ensuring convergence to a global minimum.

3 Problem 3: Quadratic Convex Function Minimization (3.R)

3.1 Objective

The script `3.R` minimizes a quadratic convex function of the form $f(x) = x^T Ax + b^T x$ using gradient descent.

3.2 Plots of Convergence

The script provides plots for loss vs. iterations, the 2D descent path, and a 3D visualization.

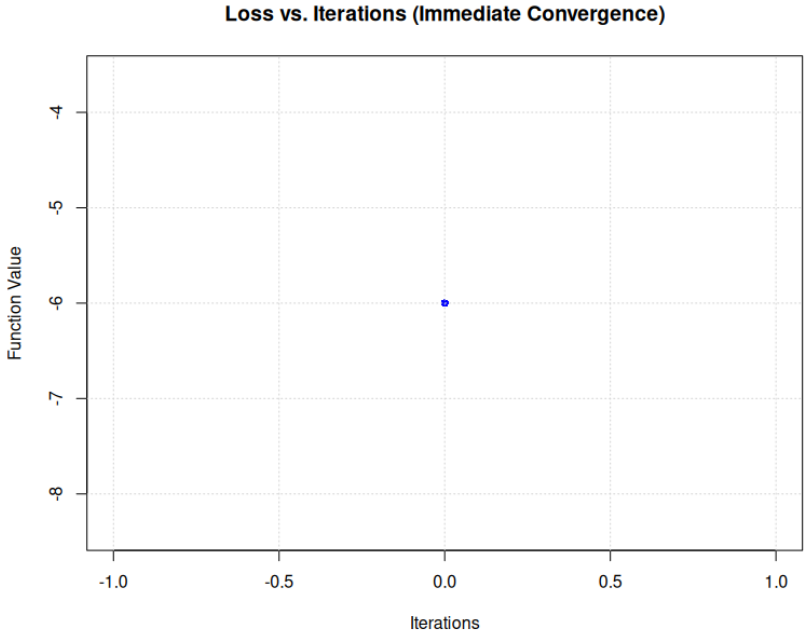

Figure 6: Loss vs. Iterations for Quadratic Function Minimization.

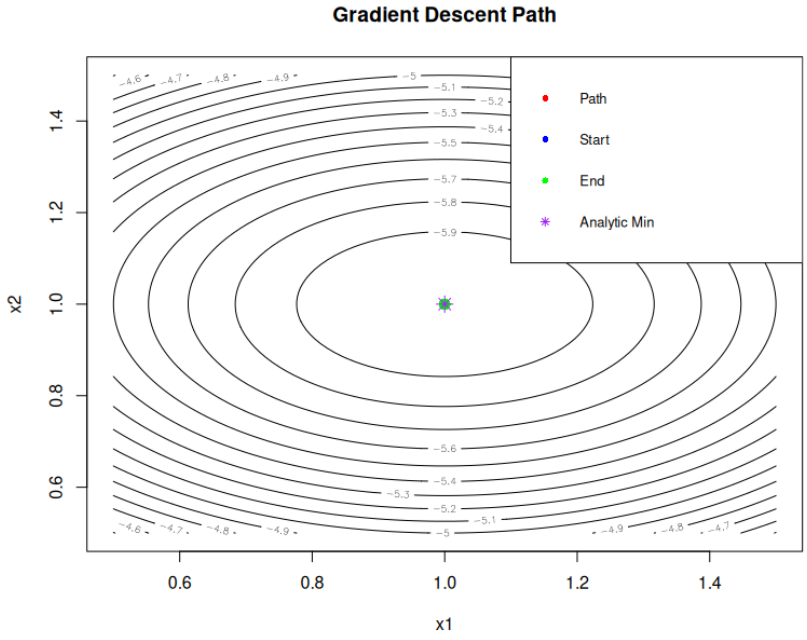

Figure 7: 2D Gradient Descent Path on Contour Plot.

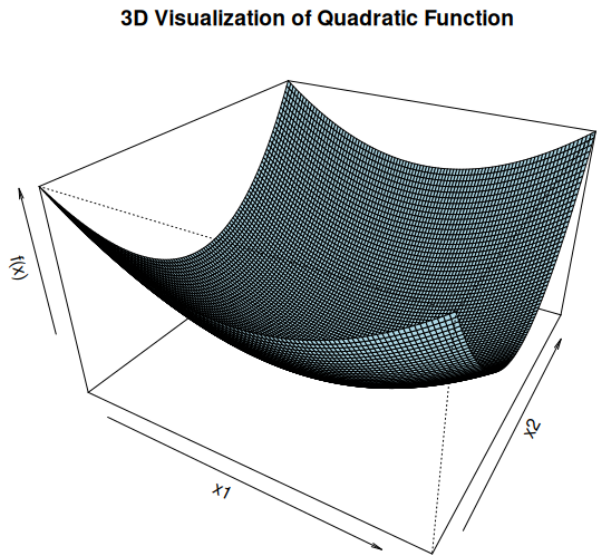

Figure 8: 3D Visualization of the Quadratic Function and Descent Path.

3.3 Final Solution and Number of Iterations

The script `3.R` outputs the final solution and iteration details:

```
Minimum found at x = (1, 1)
Minimum function value: -6
Number of iterations: 0
Converged: TRUE
Final gradient norm: 0
```

An analytical solution is also computed for comparison.

3.4 Observations and Difficulties

- The objective function is convex, guaranteeing that gradient descent finds the global minimum.
- The convergence in immediate
- The learning rate $\eta = 0.1$ and threshold $\epsilon = 10^{-6}$ are used.
- The script effectively visualizes the optimization process in 2D and 3D..

4 Problem 4: Normal Distribution MLE (4.R)

4.1 Objective

The script `4.R` estimates the parameters (μ and σ) of a normal distribution using Maximum Likelihood Estimation (MLE). Gradient descent is applied to minimize the negative log-likelihood function.

4.2 Plots of Convergence

The script generates plots for loss convergence, parameter space descent, and a 3D loss surface.

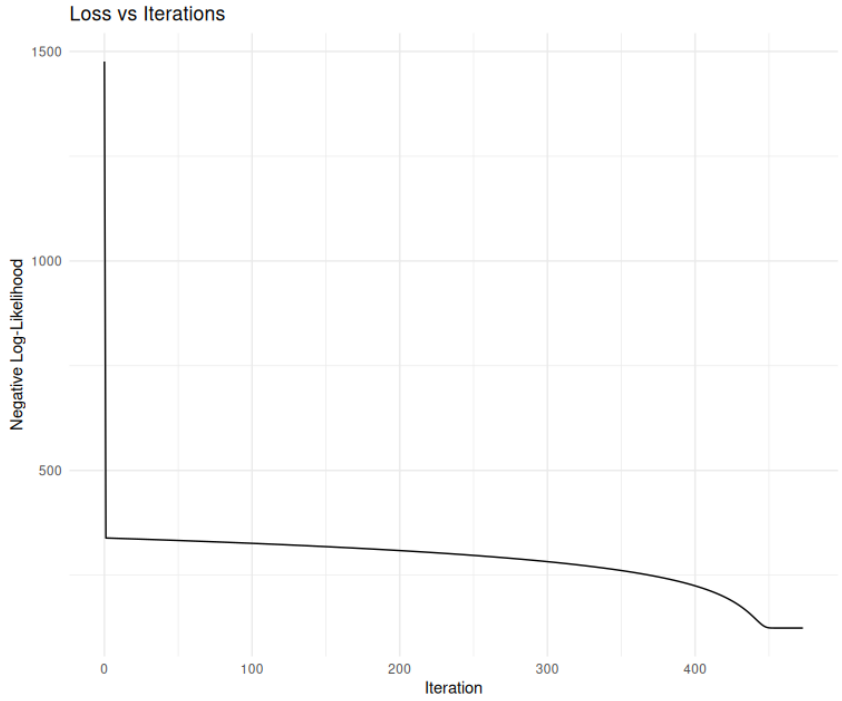

Figure 9: Loss (Negative Log-Likelihood) vs. Iterations for MLE.

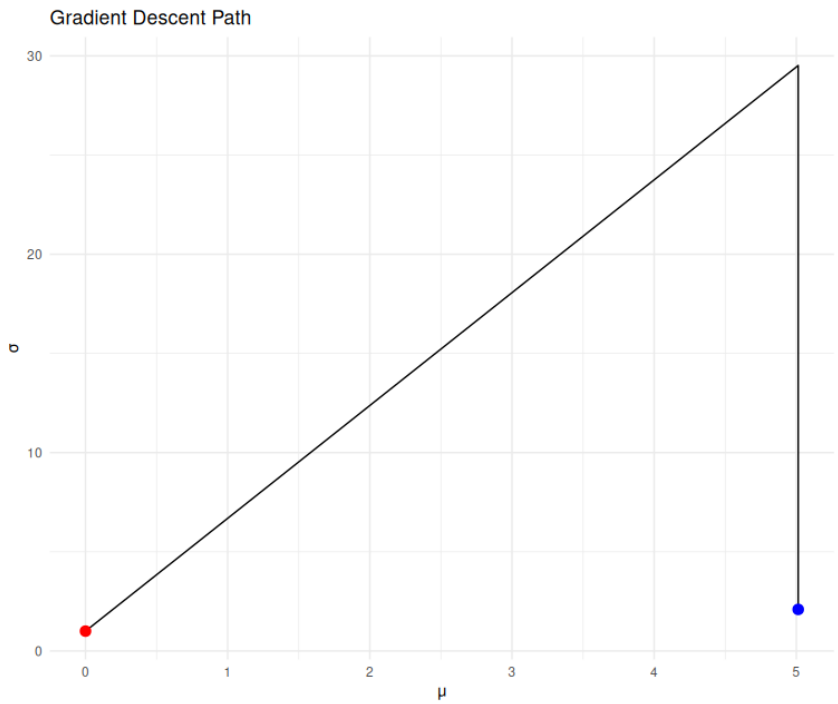

Figure 10: Gradient Descent Path in Parameter Space (μ vs. σ).

4.3 Final Solution and Number of Iterations

The script `4.R` reports the estimated parameters and convergence status:

```
Converged after 474 iterations.
(or Maximum iterations reached without convergence.)

Final parameters:
 = 5.013222
 = 2.093897
```

The script also prints the analytical solution (sample mean and scaled sample standard deviation) for comparison.

4.4 Observations and Difficulties

- The negative log-likelihood for a normal distribution is minimized.
- A crucial step is ensuring σ remains positive, handled by `sigma <- max(sigma, 1e-6)`.
- The learning rate $\eta = 0.01$ and threshold $\epsilon = 10^{-5}$ are used.
- The problem is convex with respect to μ and σ^2, but the parameterization in σ can still be handled well by gradient descent.

5 Problem 5: Rosenbrock Function Optimization (5.R)

5.1 Objective

The script `5.R` applies gradient descent to optimize the Rosenbrock function, a well-known non-convex benchmark function for optimization algorithms. The function is $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$.

5.2 Plots of Convergence

The script visualizes the loss, the 2D path, and a 3D surface plot.

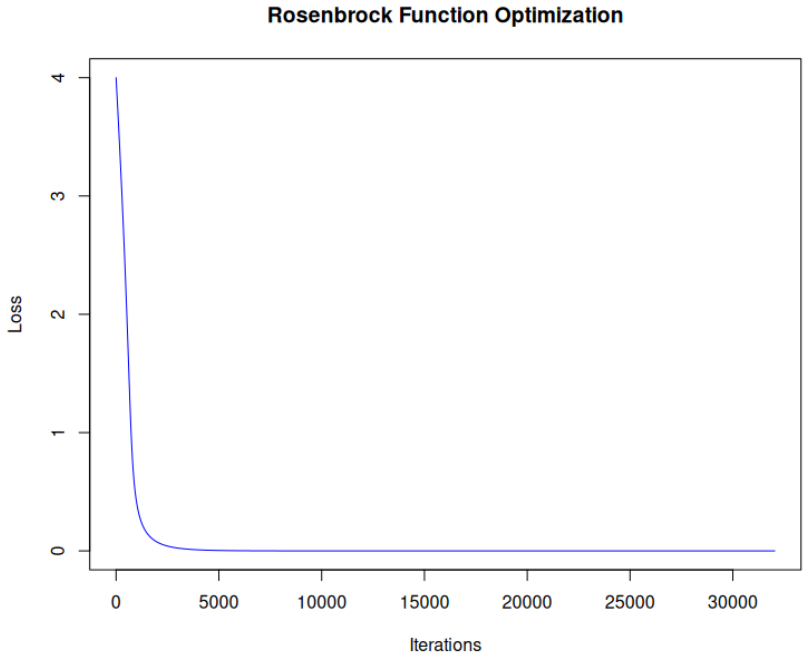

Figure 11: Loss vs. Iterations for Rosenbrock Function Optimization.

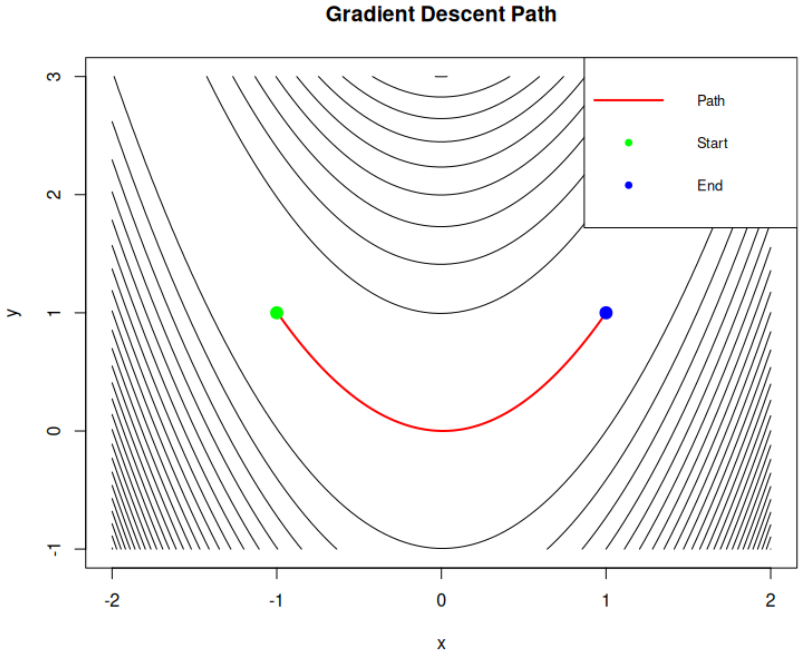

Figure 12: 2D Gradient Descent Path on Rosenbrock Function Contour Plot.

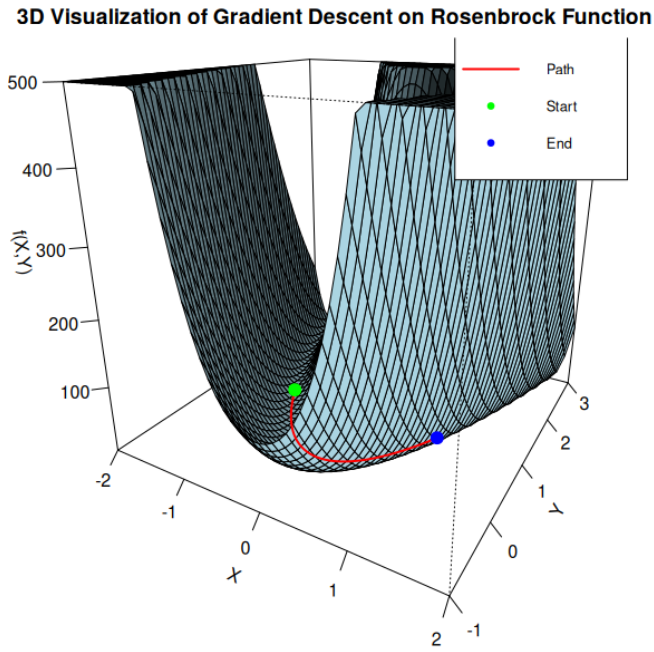

Figure 13: 3D Visualization of Gradient Descent on Rosenbrock Function.

5.3 Final Solution and Number of Iterations

The script `5.R` outputs the final position, loss, and iterations:

```
Final position: x = 0.9999989, y = 0.9999978
Final loss: 1.251652e-12
Total iterations: 32035
```

The global minimum is at $(1, 1)$ with $f(1, 1) = 0$.

5.4 Observations and Difficulties

- The Rosenbrock function has a narrow, parabolic valley, making it challenging for some optimization algorithms.
- The script uses an initial point of $(-1, 1)$, learning rate $\eta = 0.001$, threshold $\epsilon = 10^{-6}$, and a high `max_iterations` limit (100,000).
- The non-convex nature means gradient descent might get stuck in local minima if they existed and were attractive, or converge slowly in flat regions. However, for the standard Rosenbrock function, it typically converges to the global minimum, albeit sometimes slowly.
- The 3D visualization uses R's base `persp` function.
- The choice of learning rate is particularly sensitive for the Rosenbrock function. The small learning rate used suggests an attempt to navigate the valley carefully.