Final Project on Numerical Algorithms

PageRank, Minimizers, and Face Recognition

Vourvachakis S. Georgios

Department of Materials Science and Engineering
University of Crete

Jan. 16, 2025

Summary

1 PageRank Algorithm

2 Gradient-based Minimizers

3 Spectral Face Recognition System

PageRank Algorithm

A General-purpose Platform

Network-based Formulation

The word wide web can be interpreted as a *directed graph* $\mathcal{G} = (V, E)$ consisting of V nodes (i.e, web pages) and E directed edges (i.e., links) connecting them.

The PageRank of a page $A \in V$ is given as follows ($n \leq |V| - 1$):

$$PR(A) = (1 - d) + d \sum_{i \in [n]} PR(T_i)/C(T_i)$$

- $\{T_i\}_{i=1}^{n} \subseteq V$ are A's *parent nodes* (i.e., citations or hyperlinks),
- $d \in [0, 1]$ is a damping factor, indicating the probability of following a link on the current page,
- and $C(A)$ is the *out-degree* of node A.

A General-purpose Platform

Network-based Formulation

The word wide web can be interpreted as a *directed graph* $\mathcal{G} = (V, E)$ consisting of V nodes (i.e, web pages) and E directed edges (i.e., links) connecting them.

The PageRank of a page $A \in V$ is given as follows ($n \leq |V| - 1$):

$$PR(A) = (1 - d) + d \sum_{i \in [n]} PR(T_i)/C(T_i)$$

- $\{T_i\}_{i=1}^{n} \subseteq V$ are A's *parent nodes* (i.e., citations or hyperlinks),
- $d \in [0, 1]$ is a damping factor, indicating the probability of following a link on the current page,
- and $C(A)$ is the *out-degree* of node A.

Stochastic Link Matrix

Spectral Perspective

$PR(A)$ is a *component of the principal eigenvector* of the normalized link matrix of the web graph, reflecting the stationary distribution of a Markov process where a random surfer navigates the web [Random Surfer Model].

Given $N = |V|$ web pages, the **PageRank Matrix** is constructed:

$$M = d \cdot A + (1 - d) \frac{1}{N} \mathbb{1}_{N \times N}$$

where A is the *Normalized Adjacency Matrix* with

$$A_{ij} = \begin{cases} \frac{1}{C(j \to i)} & \text{if there is a link from page } j \text{ to page } i \\ 0 & \text{otherwise .} \end{cases}$$

PageRank Algorithm: Power Iteration Method

Objective of PageRank Algorithm

Assign a **rank** to each node based on the link structure of the graph. The rank of a node reflects its **relative importance** within the graph.

PageRank Workflow:

1. Construct the normalized adjacency matrix A.
2. Create the PageRank matrix M.
3. Apply the *Power Method* to find the normalized dominant eigenvector[a], $\mathbf{v_{max}} \in \{\mathbf{v} \mid M\mathbf{v} = \lambda_{max}\mathbf{v}, \|\mathbf{v}\|_p = 1\}$, containing the ranking for each web page. We obtain the results across norms: $p \in \{1, 2, \infty\}$. [b]

[a] stationary distribution of a random walk on the graph.

[b] dominant eigenvalue estimate: $\lambda_{max} = \{\mathbf{v}_{max}^\top M \mathbf{v_{max}} \mid \|\mathbf{v}_{max}\|_p = 1\}$.

PageRank Algorithm: Power Iteration Method

Objective of PageRank Algorithm

Assign a **rank** to each node based on the link structure of the graph. The rank of a node reflects its **relative importance** within the graph.

PageRank Workflow:

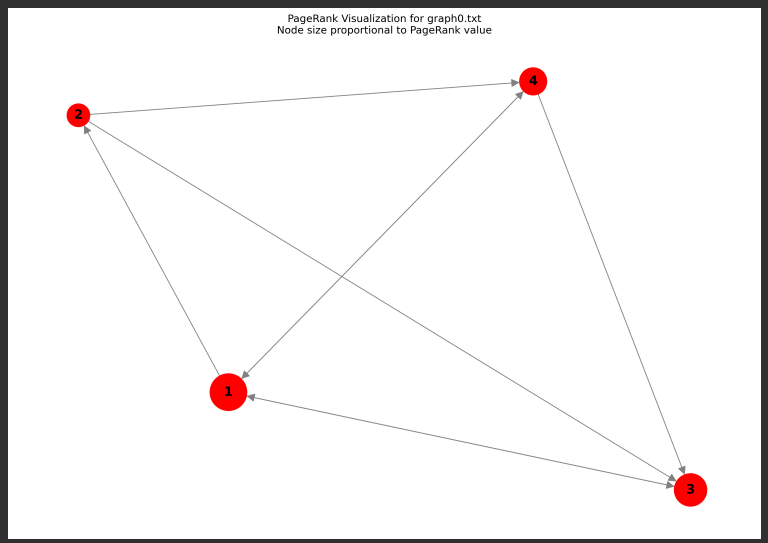
1. Construct the normalized adjacency matrix A.
2. Create the PageRank matrix M.
3. Apply the *Power Method* to find the normalized dominant eigenvector[a], $\mathbf{v_{max}} \in \{\mathbf{v} \mid M\mathbf{v} = \lambda_{max}\mathbf{v}, \|\mathbf{v}\|_p = 1\}$, containing the ranking for each web page. We obtain the results across norms: $p \in \{1, 2, \infty\}$. [b]

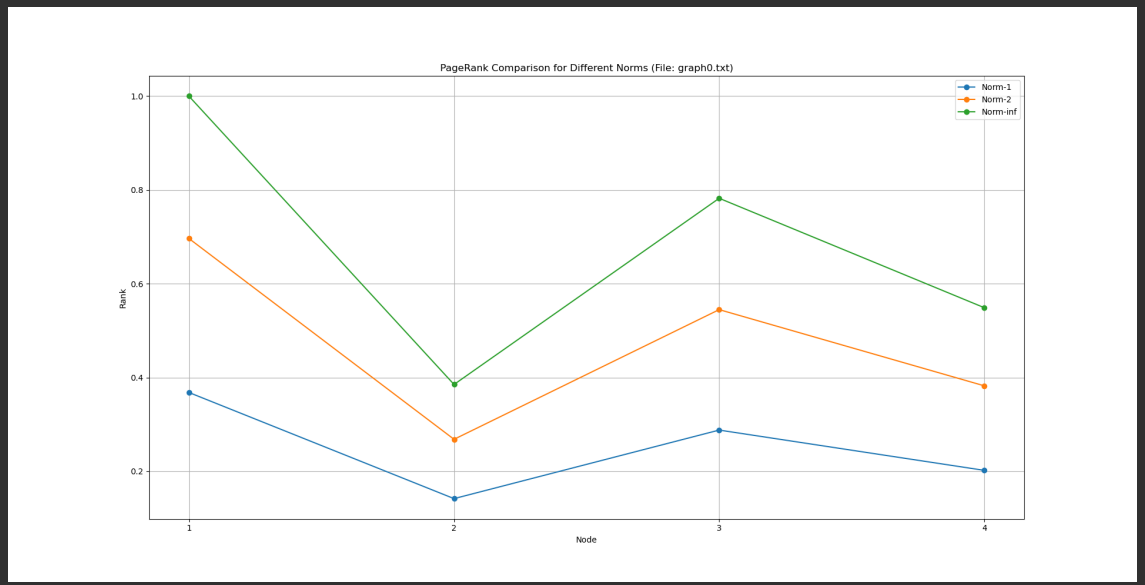
[a] stationary distribution of a random walk on the graph.

[b] dominant eigenvalue estimate: $\lambda_{max} = \{\mathbf{v}_{max}^\top M \mathbf{v_{max}} \mid \|\mathbf{v}_{max}\|_p = 1\}$.

PageRank Analysis Results (graph0.txt)

Norm	Eigenvalue	Iterations	Sum of Ranks $\sum_{i\in[N]} \lvert v_{max,i}\rvert$
L1	0.279	31	1.000
L2	1.000[1]	45	1.000
L∞	2.061	31	2.716

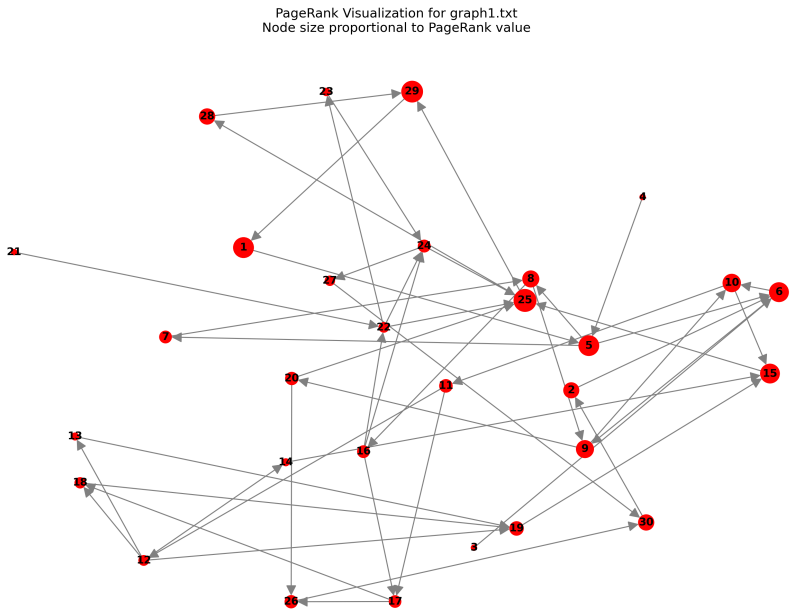


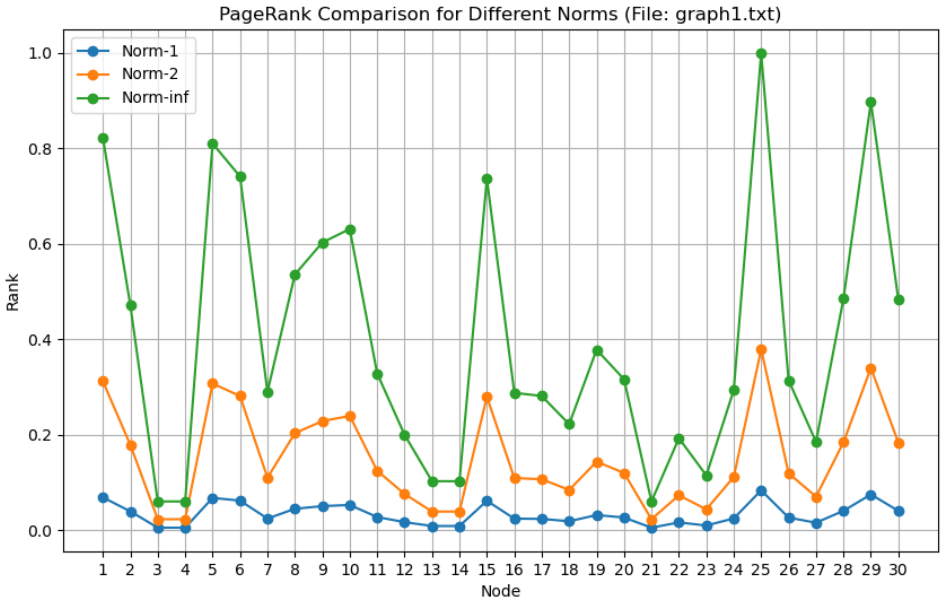


[1] for $p = 2$: $\lambda_{max} := \rho(M) = \max_{\|\mathbf{v}\|_2=1} \|M\mathbf{v}\|_2 = \sqrt{\lambda_{max}(M^\top M)} \approx 1$

PageRank Analysis Results (graph1.txt)

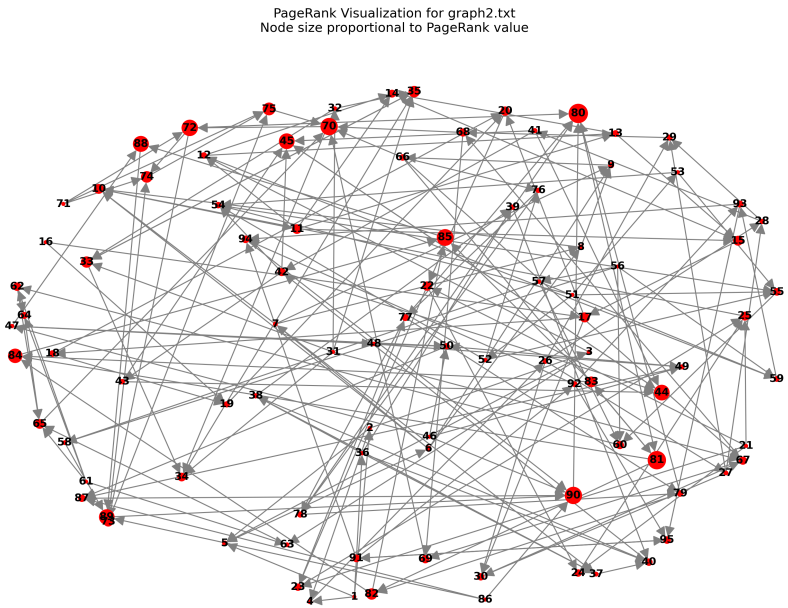
Norm	Eigenvalue	Iterations	Sum of Ranks
L1	0.048	67	1.000
L2	1.000	67	4.551
L∞	6.958	68	12.005

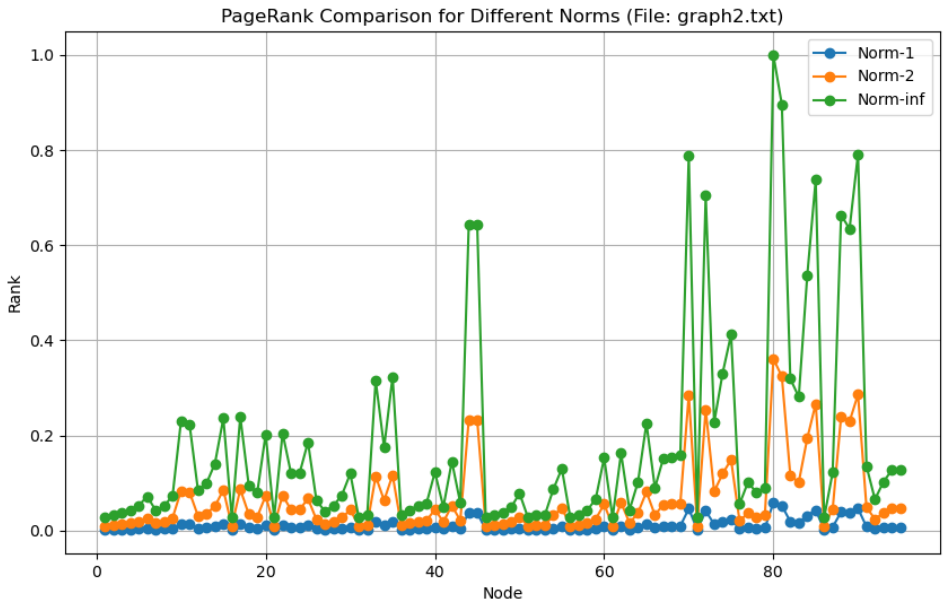




PageRank Analysis Results (graph2.txt)

Norm	Eigenvalue	Iterations	Sum of Ranks
L1	0.026	120	1.000
L2	1.000	125	6.1985433148
L∞	7.662	128	17.157





PageRank Algorithm: NetworkX API

Comparative Analysis:
Custom PageRank Algorithm vs $\mathrm{networkx.pagerank(G, alpha = d)}$

▷ Both methods identified the same highest-ranked nodes.

	Graph0	Graph1	Graph2
Maximum difference	0.632	0.917	0.942
Average difference	0.429	0.367	0.170
Power Method eigenvalue	2.061	6.958	7.662
Power Method iterations	31	68	128

$\mathrm{NetworkX}$ performance characteristics:

- More efficient memory usage through sparse matrix representation.
- Slightly faster convergence in most cases.
- Better handling of edge cases (e.g., dangling/isolated nodes)

Gradient-based Minimizers

Unconstrained Convex Optimization

Empirical Risk Minimization

Given objective $f : \mathbb{R}^n \to \mathbb{R}$, find global minimizer $x^* \in \mathbb{R}^n$ s.t.:

$$x^* = \text{argmin}_{x \in \mathbb{R}^n} f(x).$$

Armijo-Goldstein condition (1966)

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

▷ *Plain GD* for $\tau = 1.0$ and $c = 0.0$.

Algorithm 1 Gradient Descent with Backtracking Line Search

Input: Function $f(x)$, maximum iterations K, tolerance ϵ

Output: Local minimizer x_k

Choose random $x_0 \in \mathbb{R}^n$
Choose α_0, $(\tau, c) \in (0, 1)^2$
Initialize counter $k = 0$
while $k < K$ ***and*** $\|\nabla f(x_k)\| > \epsilon$ ***and*** $\|x_k - x_{k-1}\| > \epsilon$ **do**
  $p_k = -\nabla f(x_k)$
  Set $m = \nabla f(x_k)^T p_k$; $j = 0$; $t = cm$
  while $f(x_k + \alpha_j p_k) > f(x_k) + \alpha_j t$ **do**
    $\alpha_{j+1} = \tau \alpha_j \;\; ; \;\; j = j + 1$
  end
  $x_{k+1} = x_k + \alpha_j p_k$
  $k = k + 1$
end

Unconstrained Convex Optimization

Empirical Risk Minimization

Given objective $f : \mathbb{R}^n \to \mathbb{R}$, find global minimizer $x^* \in \mathbb{R}^n$ s.t.:

$$x^* = \text{argmin}_{x \in \mathbb{R}^n} f(x).$$

Armijo-Goldstein condition (1966)

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

▷ *Plain GD* for $\tau = 1.0$ and $c = 0.0$.

Algorithm 2 Gradient Descent with Backtracking Line Search

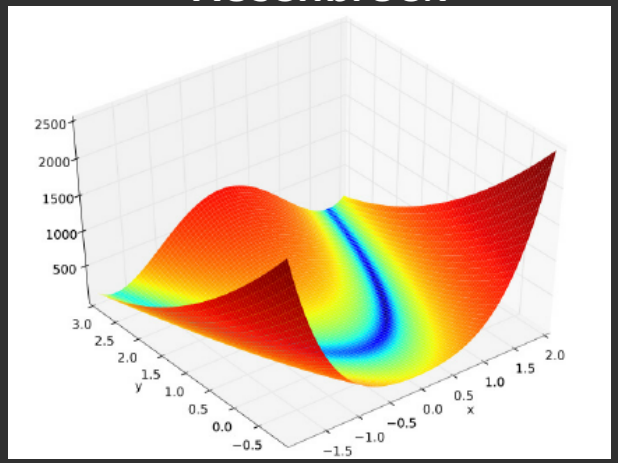
Input: Function $f(x)$, maximum iterations K, tolerance ϵ

Output: Local minimizer x_k

Choose random $x_0 \in \mathbb{R}^n$
Choose α_0, $(\tau, c) \in (0, 1)^2$
Initialize counter $k = 0$
while $k < K$ ***and*** $\|\nabla f(x_k)\| > \epsilon$ ***and*** $\|x_k - x_{k-1}\| > \epsilon$ **do**
  $p_k = -\nabla f(x_k)$
  Set $m = \nabla f(x_k)^T p_k$; $j = 0$; $t = cm$
  while $f(x_k + \alpha_j p_k) > f(x_k) + \alpha_j t$ **do**
    $\alpha_{j+1} = \tau \alpha_j \;\; ; \;\; j = j + 1$
  end
  $x_{k+1} = x_k + \alpha_j p_k$
  $k = k + 1$
end

Test Functions

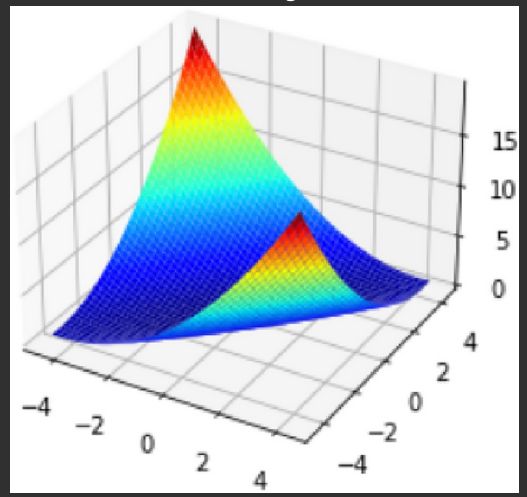
Rosenbrock



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

Valley-shaped

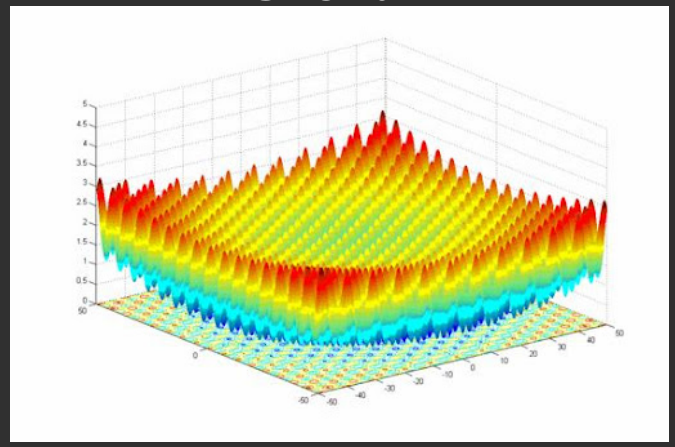
Matyas



$$f(x_1, x_2) = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2$$

Plate-shaped

Griewank



$$f(x_1, x_2) = 1 + \frac{1}{4000} \cdot (x_1^2 + x_2^2) - \cos(x_1) \cos\left(\frac{x_2}{\sqrt{2}}\right)$$

Many Local Minima

Implementation Parameters

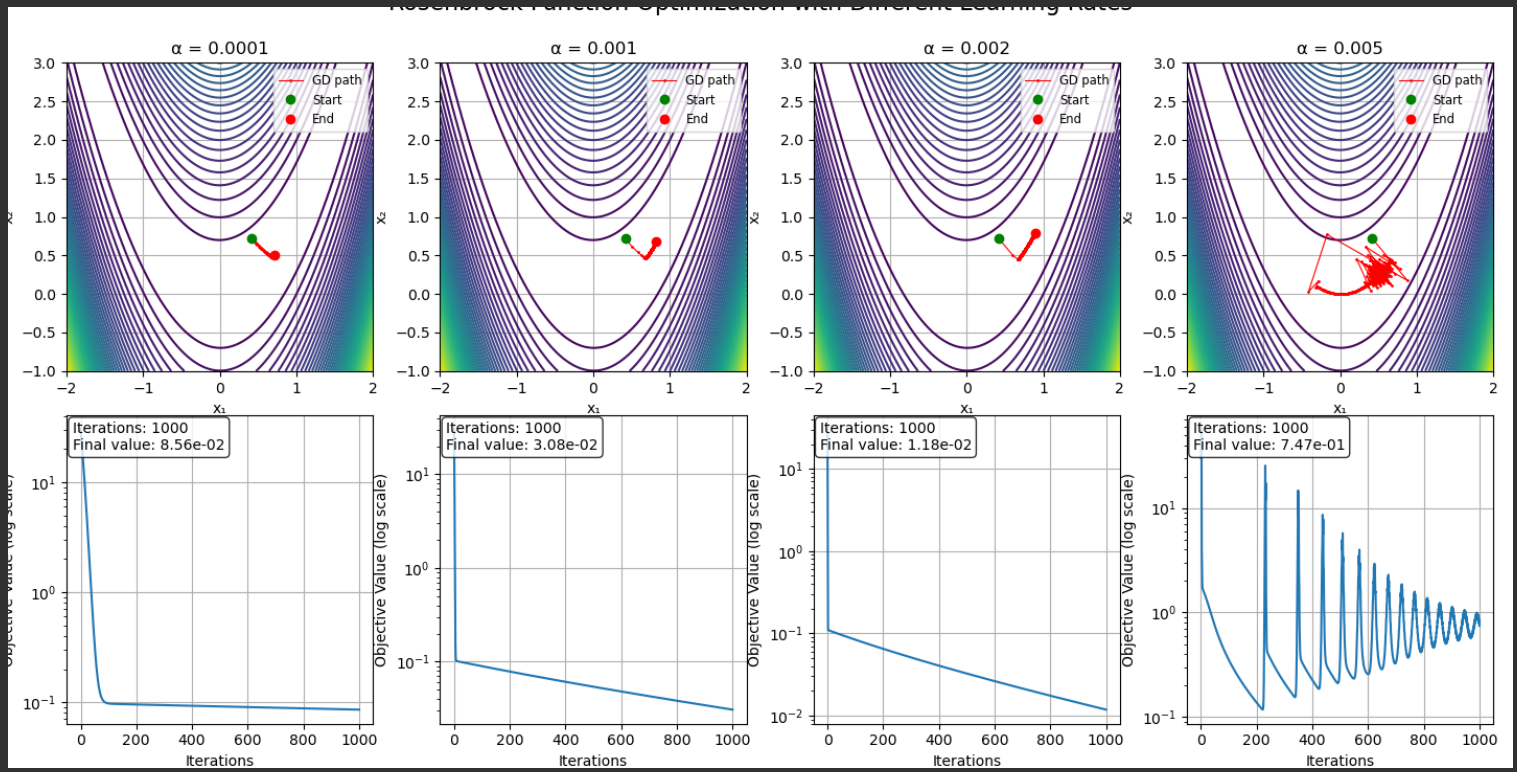
Fixed Step Gradient Descent	
Parameter	**Value**
α (Matyas, Griewank)	0.1
α (Rosenbrock)[2]	0.001
τ	1.0
c	0.0
Backtracking Line Search	
Initial α_0	1.0
Reduction τ	0.1
Decrease param c	0.0001

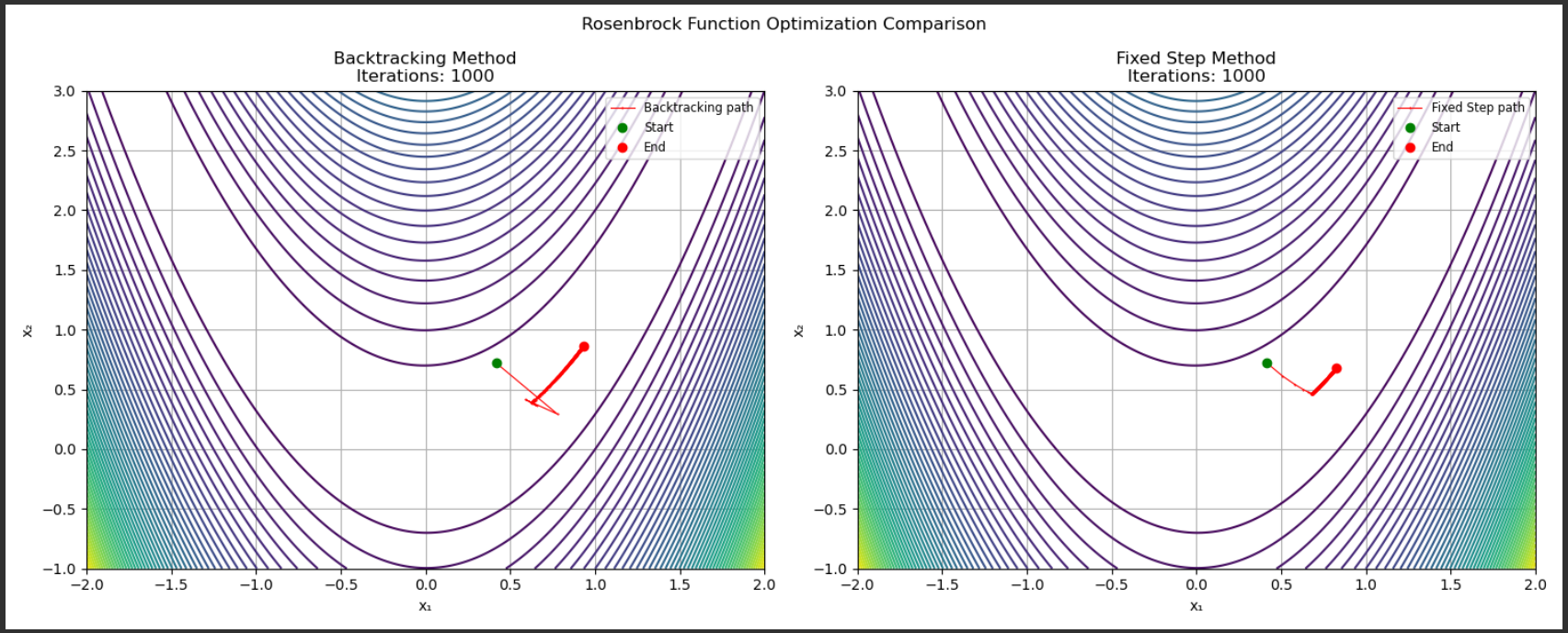
Table: Gradient Descent and Backtracking Line Search Parameters

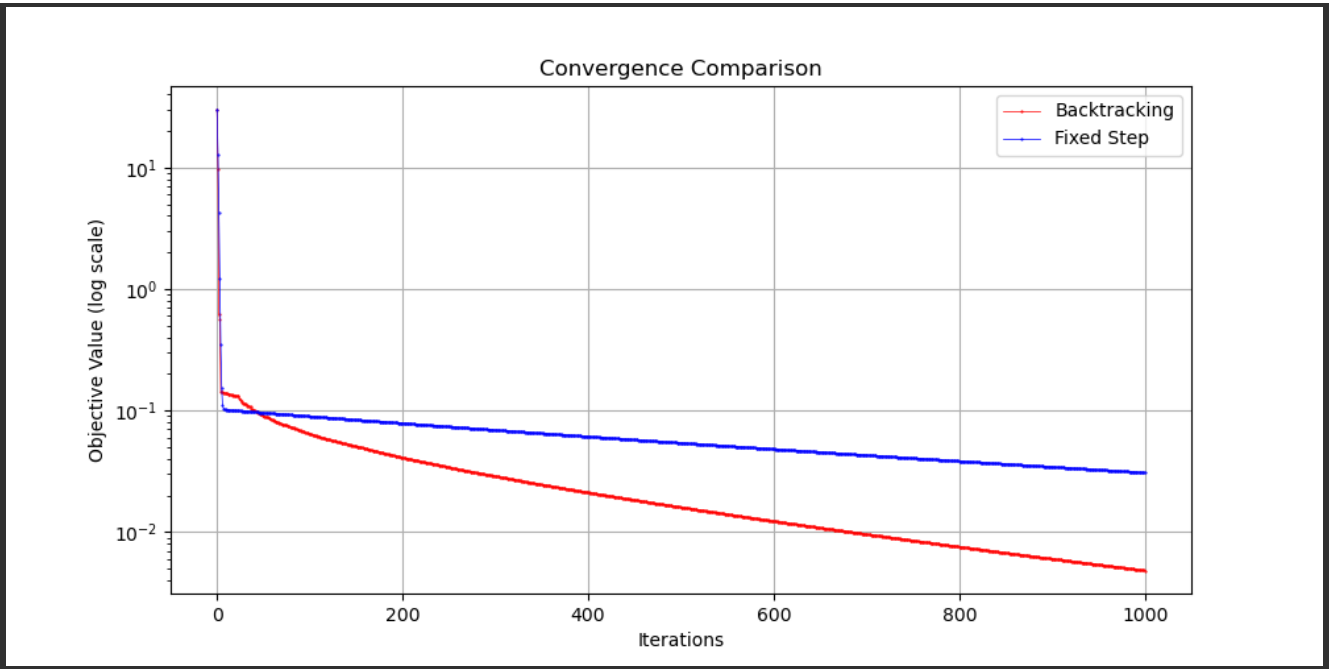
[2] for Rosebrock's function the step-size α is reduced due to stability issues [$\mathrm{RuntimeWarning}$: overflow encountered in scalar power of $f(x_1, x_2)$ and in scalar multiply of $\partial f/\partial x_1$.]

Gradient Descent: Results for Rosenbrock Function

α	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	**Iter**	**Status**
0.0001	$[0.7077, 0.4994]$	0.0856	1000	k_{max}
0.001	$[0.8246, 0.6792]$	0.0308	1000	k_{max}
0.002	$[0.8914, 0.7942]$	0.0118	1000	k_{max}
0.005	$[0.5297, 0.2080]$	0.7472	1000	k_{max}

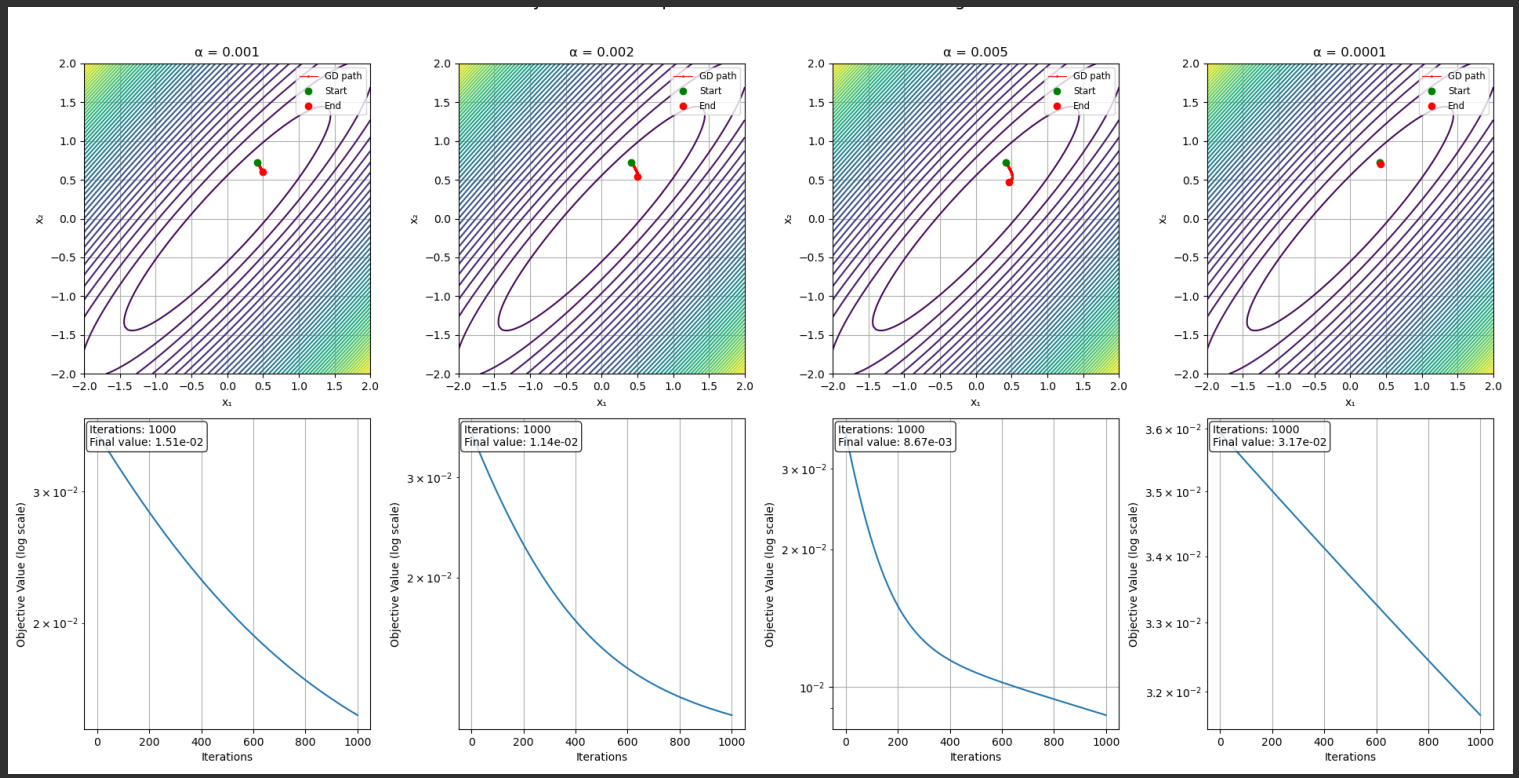


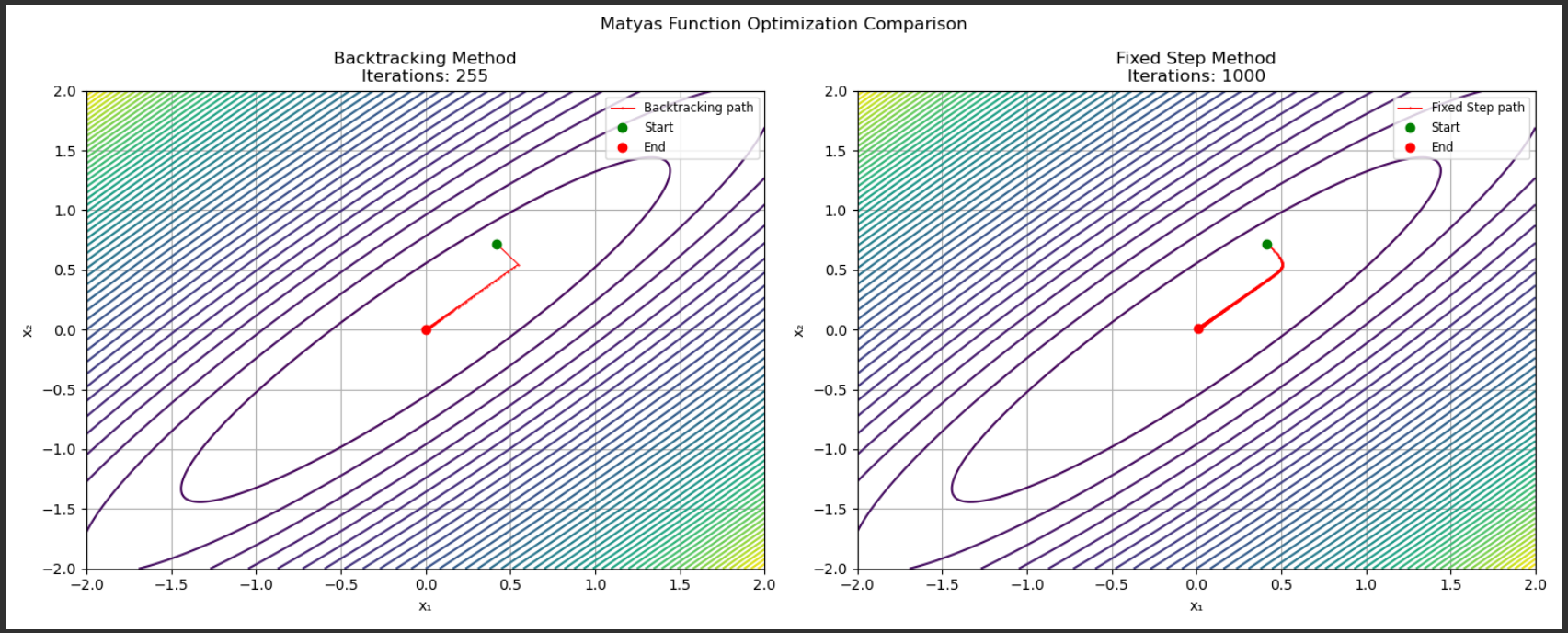

Rosenbrock Function Optimization Comparison
Backtracking Method
Iterations: 1000
Backtracking path
Start
End
Fixed Step Method
Iterations: 1000
Fixed Step path
Start
End
x₁
x₂

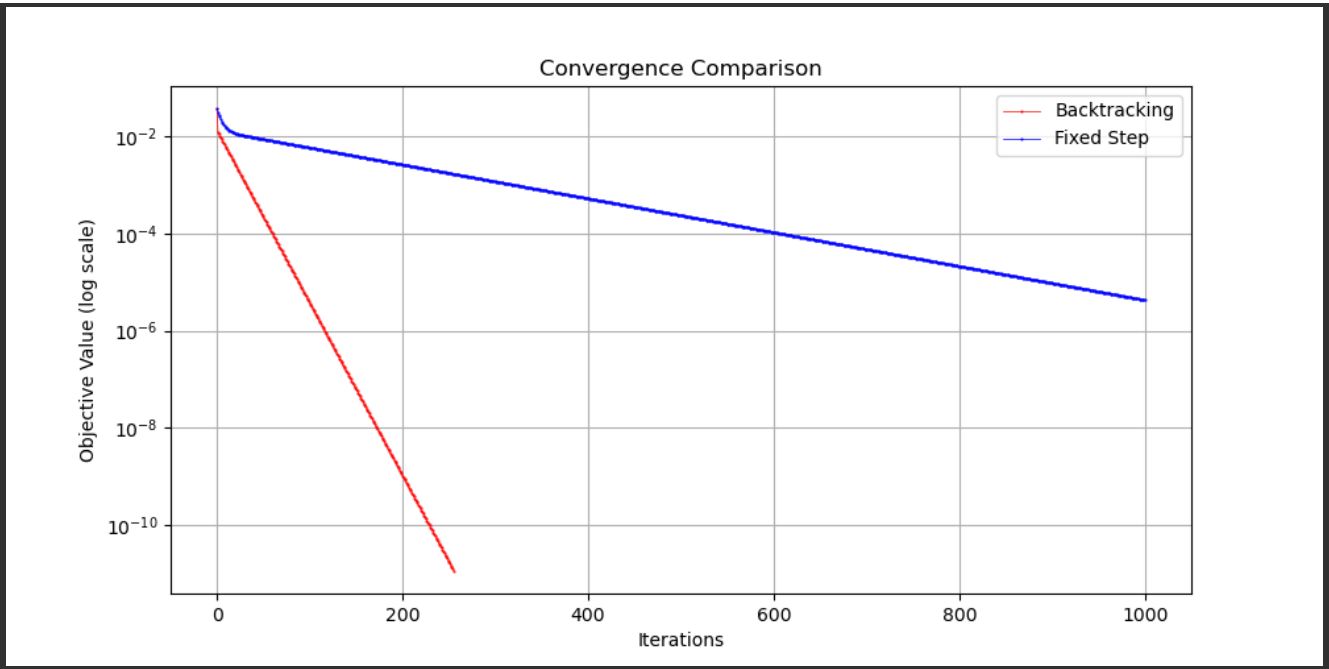

Convergence Comparison
Backtracking
Fixed Step
Objective Value (log scale)
Iterations

Gradient Descent: Results for Matyas Function

α	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	**Iter**	**Status**
0.0001	$[0.4292, 0.7036]$	0.0317	1000	k_{max}
0.001	$[0.4906, 0.6021]$	0.015	1000	k_{max}
0.002	$[0.5045, 0.5454]$	0.0114	1000	k_{max}
0.005	$[0.4646, 0.4666]$	0.0087	1000	k_{max}

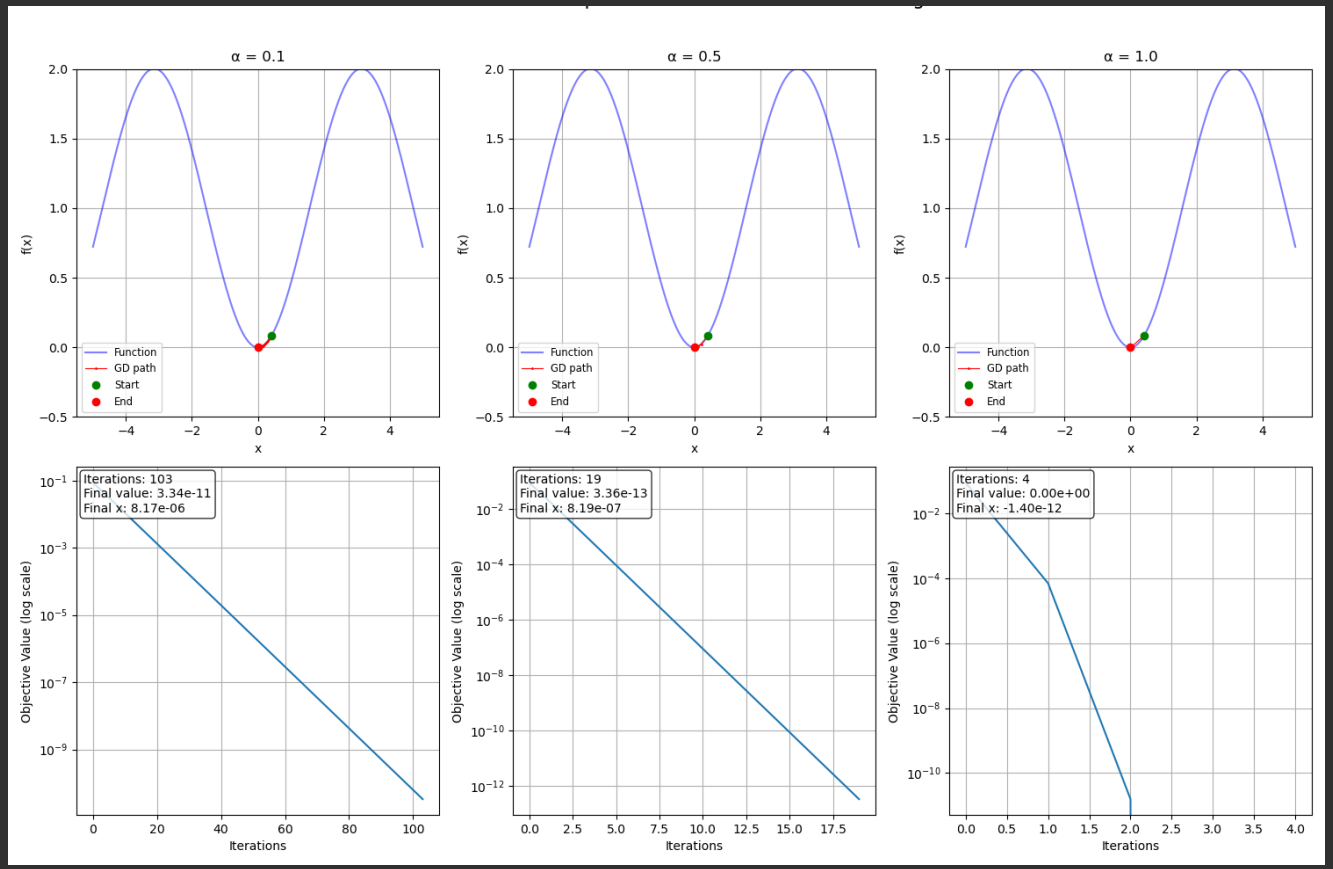


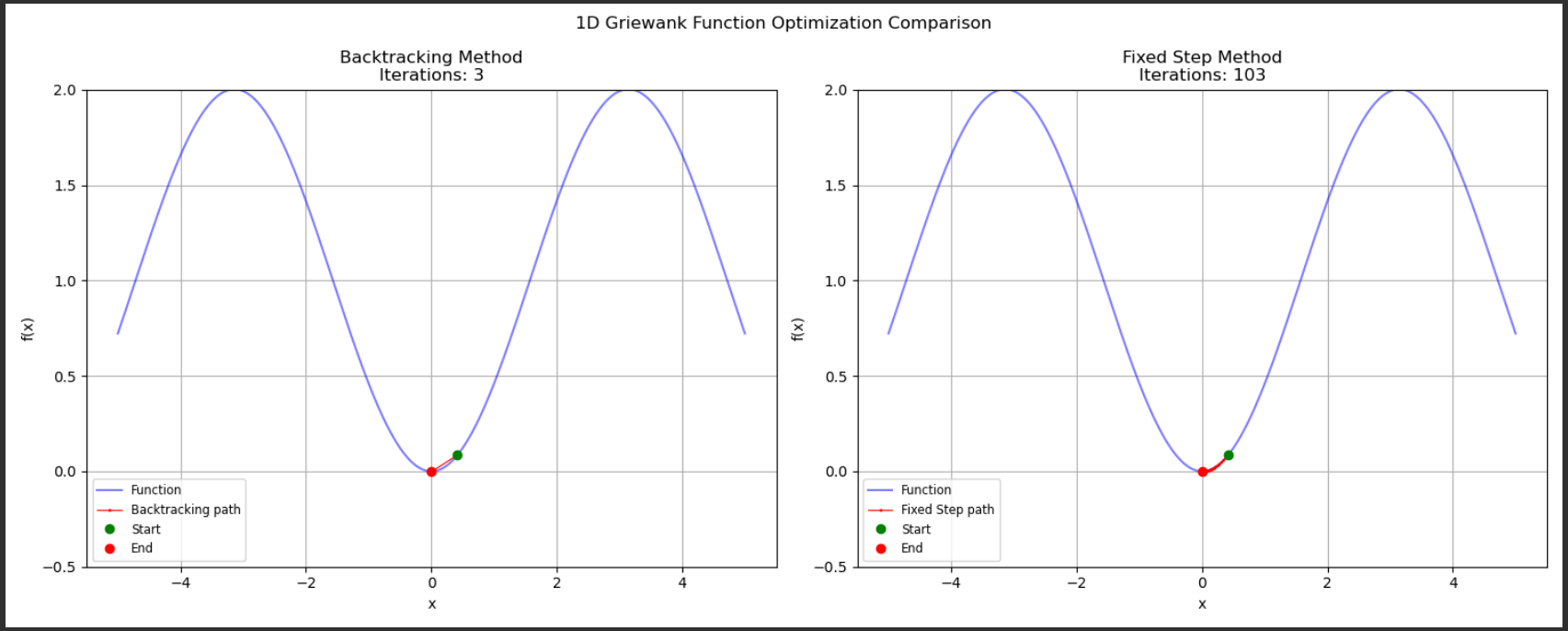

Matyas Function Optimization Comparison
Backtracking Method
Iterations: 255
Backtracking path
Start
End
Fixed Step Method
Iterations: 1000
Fixed Step path
Start
End
x₁
x₂

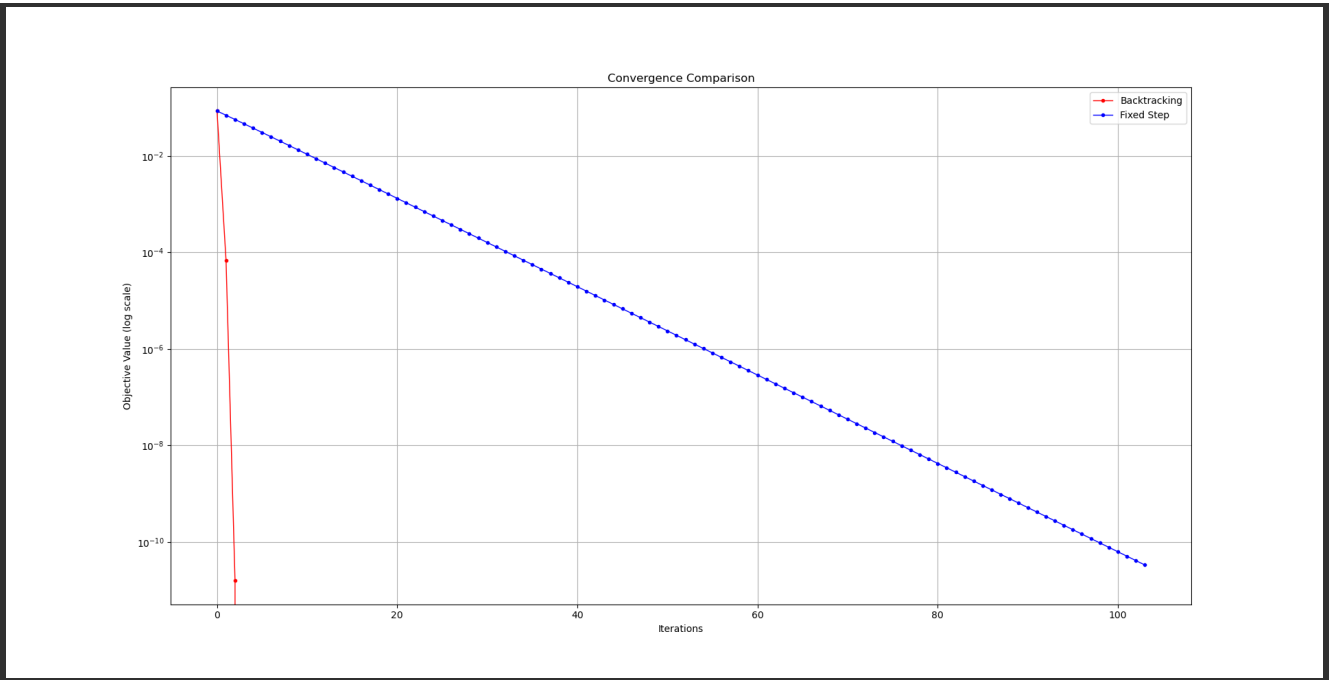

Convergence Comparison
Backtracking
Fixed Step
Objective Value (log scale)
Iterations

Gradient Descent: Results for 1D Griewank Function

α	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	**Iter**	**Status**
0.1	8.1673e-06	3.3370e-11	103	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$
0.5	8.18850599e-07	3.3551e-13	19	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$
1.0	-1.4037444e-12	0.0 (underflow)	2	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$

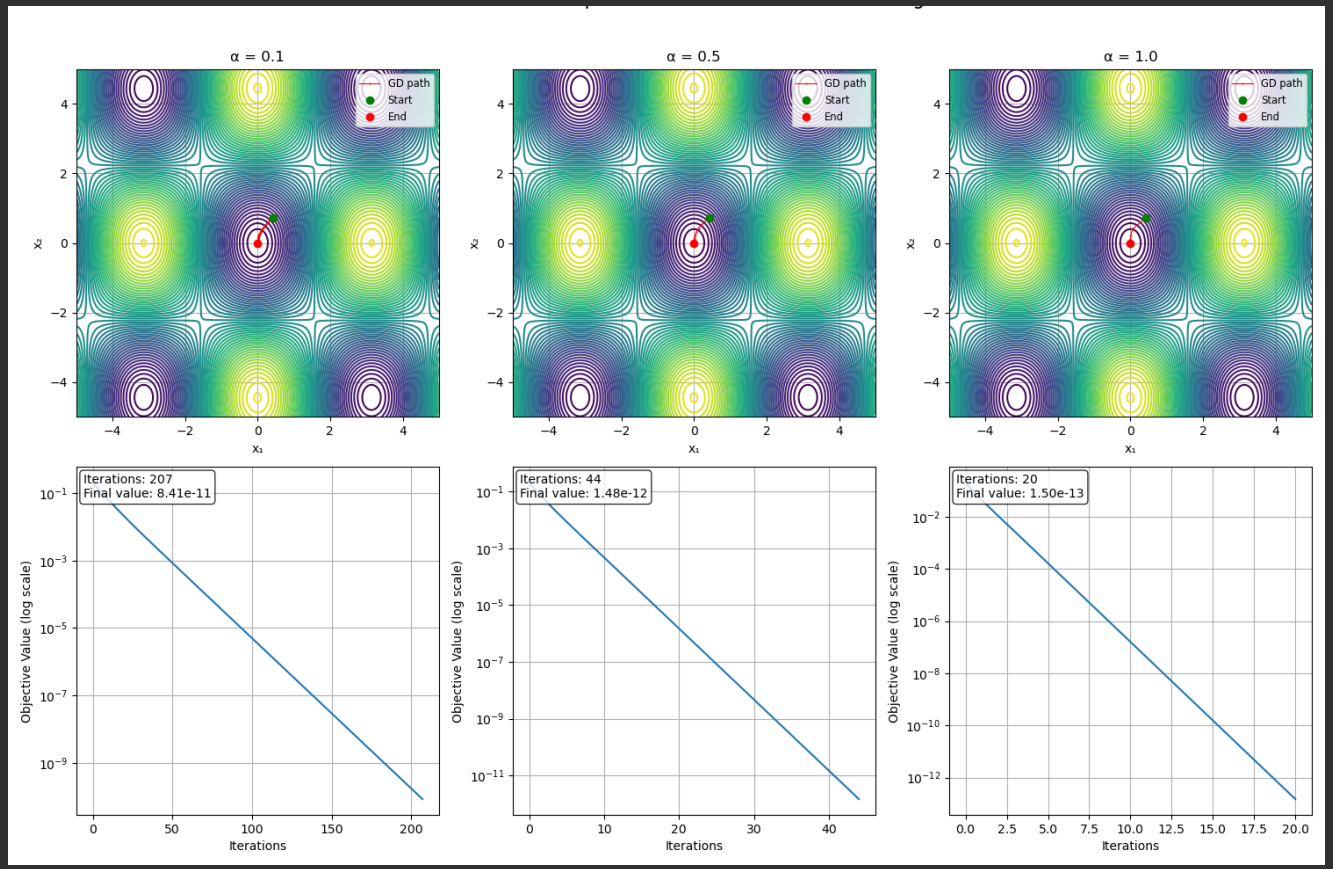


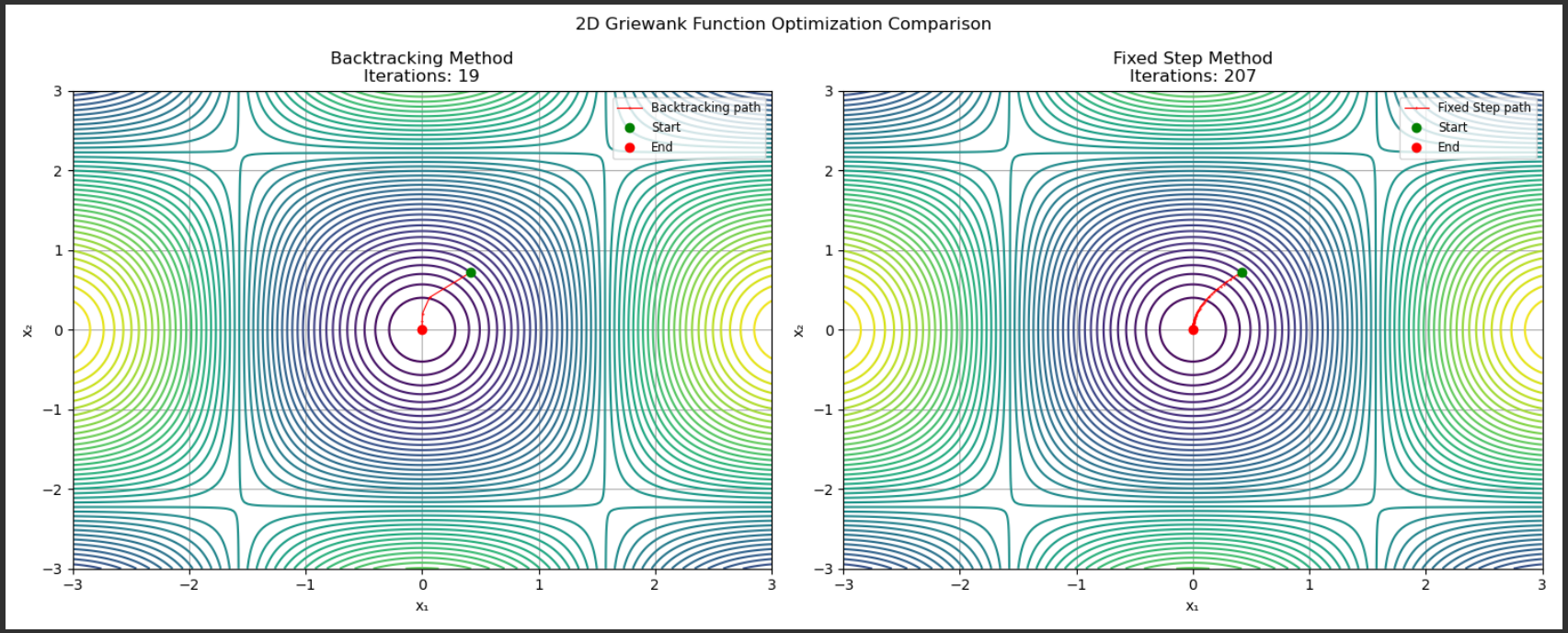

1D Griewank Function Optimization Comparison
Backtracking Method
Iterations: 3
Fixed Step Method
Iterations: 103
f(x)
x
Function
Backtracking path
Fixed Step path
Start
End

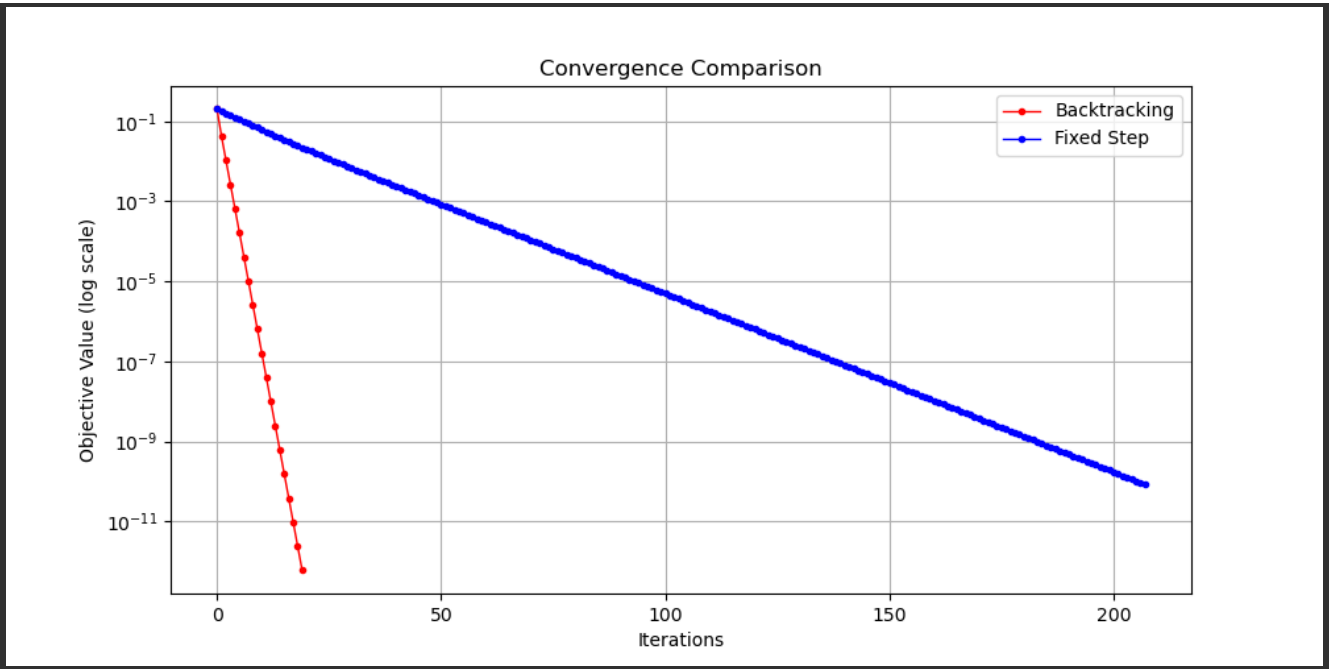

Convergence Comparison
Backtracking
Fixed Step
Objective Value (log scale)
Iterations

Gradient Descent: Results for 2D Griewank Function

α	$\mathbf{x}^*$	$f(\mathbf{x}^*)$	**Iter**	**Status**
0.1	[2.6117e-13, 1.8523e-05]	8.5863e-11	189	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$
0.5	[5.3672e-17, 2.2625e-06]	1.2811e-12	41	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$
1.0	[−1.6360e-61, 7.7283e-07]	1.4955e-13	20	$\|\|x_{k+1} - x_k\|\| \leq \epsilon$




2D Griewank Function Optimization Comparison
Backtracking Method
Iterations: 19
Backtracking path
Start
End
Fixed Step Method
Iterations: 207
Fixed Step path
Start
End
x₁
x₂


Convergence Comparison
Backtracking
Fixed Step
Objective Value (log scale)
Iterations

Backtracking vs. Fixed Step Methods: Key Insights

- **Efficiency:** Backtracking requires fewer iterations, especially excelling in Griewank functions.
- **Stability:** Adapts step sizes automatically, avoiding issues like the fixed step's need for $\alpha = 0.001$ in Rosenbrock.
- **Path Characteristics:** Produces direct paths to the optimum, unlike the zigzagging of fixed steps.
- **Cost-Benefit:** Though backtracking involves more evaluations per iteration, fewer total iterations reduce overall cost.

Spectral Face Recognition System

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\text{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\text{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\text{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\operatorname{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\operatorname{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Mathematical Framework

Objective

Find the principal components of the distribution of faces, treating an image as an object in a high dimensional "face space".

Given a set of N face images, each with $m \times n = M$ pixels, we represent each face as an M-vector. The face recognition problem can be formulated as follows:

1. Form matrix $S = [\{\mathbf{f}_i\}_{i\in[N]}]$, where each f_i is a face vector.
2. Compute mean face $\bar{\mathbf{f}} = \frac{1}{N}\sum_{i\in[N]} \mathbf{f}_i$.
3. Normalize the face matrix, S: $\alpha_i = \mathbf{f}_i - \bar{\mathbf{f}}$, forming matrix $A = [\{\alpha_i\}_{i\in[N]}]$.
4. Perform SVD factorization: $A_{(M\times N)} = U_{(M\times M)}\Sigma_{(M\times N)}V^{\top}_{(N\times N)}$.
5. Compute the coordinate vector of a given face $\mathbf{f}_q$:[a] $\mathbf{x}_q = [\{\mathbf{u}_i\}_{i=1}^{r}]^{\top}(\mathbf{f}_q - \bar{\mathbf{f}})$.

[a] since $\operatorname{span}\{\{\mathbf{u}_i\}_{i=1}^{r}\} = \mathcal{R}(A) \subseteq \mathbb{R}^{M\times 1}$.

Normalization Improvements

These adjustments make the thresholds more interpretable and less dependent on image size:

▷ **Pixel Value Normalization**: Images are normalized to the $[0, 1]$ range:

$$\mathbf{f}_{normalized} = \mathbf{f}/255.$$

▷ **Distance Normalization**: Distances are normalized by image dimensions:

$$\epsilon_f = \frac{\|(\mathbf{f} - \bar{\mathbf{f}}) - \mathbf{f}_p\|_2}{\sqrt{M}}, \quad \epsilon_i = \frac{\|\mathbf{x} - \mathbf{x}_i\|_2}{\sqrt{M}}.$$

Hyperparameter Search Methodology

Conditional Grid Search

Let the set of candidate values for ϵ_0 and ϵ_1 be denoted by $\mathcal{E}_0 = [e_0^1, e_0^2, \ldots, e_0^m]$ and $\mathcal{E}_1 = [e_1^1, e_1^2, \ldots, e_1^n]$, respectively. The condition $\epsilon_0 > \epsilon_1$, restricts the grid search to valid pairs (e_0^i, e_1^j) such that $e_0^i > e_1^j$. Complexity reduction: $\mathcal{O}(m \times n) \rightarrow \mathcal{O}(m \times \log(n))$.

This empirical procedure yields optimal thresholds:

- $\epsilon_0 = 25$: The threshold distinguishing **known from unknown faces**.
- $\epsilon_1 = 20$: The threshold distinguishing **faces from non-faces**.

Improper image size

The image $26.\text{jpg}$ in training data had dimensions of 320×319 pixels, thus it has being rescaled properly at preprocessing step.

Classification Results

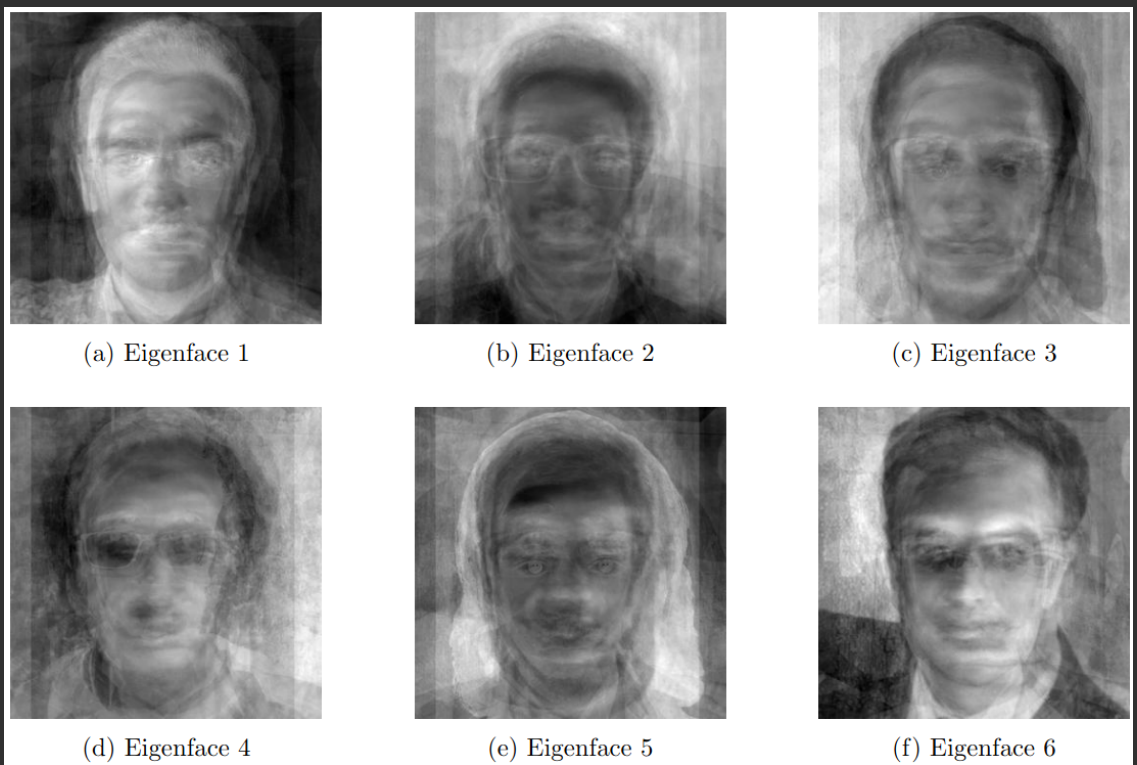
Test Image	$\epsilon_{\mathbf{f}}$	$\min_{\mathbf{i}\in[\mathbf{N}]}(\epsilon_{\mathbf{i}})$	**Classification**
1.jpg	5.22×10^{-13}	8.41×10^{-15}	Known (1)
3.jpg	7.62×10^{-13}	1.30×10^{-14}	Known (3)
5.jpg	4.65×10^{-13}	9.45×10^{-15}	Known (5)
11.jpg	5.63×10^{-13}	7.92×10^{-15}	Known (11)
25.jpg	8.21×10^{-13}	9.94×10^{-15}	Known (25)
38.jpg	0.1660	0.0192	Known (36)
noface.jpg	37.98	52.25	Not face
U1.jpg	45.31	49.23	Not face
U2.jpg	39.07	56.57	Not face
U3.jpg	1.04×10^{-12}	$0\ (underflow)$	Known (34)

Proper Identification

Note that the U3.jpg is indeed an image contained on the training set (U3.jpg = 34.jpg) and it is correctly identified.

Potential Improvements

- **Preprocessing Enhancements**: Background removal and Central Cropping.
- **Feature Space Optimization**: Weight eigenfaces by eigenvalues to emphasize discriminative features.
- **Hyperparameter tuning**: Dynamic determination of thresholds.



(a) Eigenface 1 (b) Eigenface 2 (c) Eigenface 3

(d) Eigenface 4 (e) Eigenface 5 (f) Eigenface 6

Thanks for your attention

Q & A section