UNIVERSITY
OF TWENTE.

Faculty of Electrical Engineering,
Mathematics and Computer Science

# Trade-offs in the Non-Convex Stochastic Gradient Method: Generalization, Optimization, and Computational Complexity

**Author:**
Andrii Soldatov

**Supervisors:**
Clément Lezane
Sophie Langer

Date of Submission: **January 24, 2025**

Location: **Enschede, The Netherlands**

This document was prepared as part of the Bachelor's Thesis for the degree of Applied Mathematics at University of Twente.

# Abstract

This thesis explores the Stochastic Gradient Method (SGM) and Deterministic Gradient Method (DGM) in the context of non-convex optimization. For SGM, we analyze the generalization bounds and computational complexity, highlighting that the method's performance is significantly influenced by the $\beta$-smoothness of the objective function and the constant step size $\alpha$. The generalization bound for non-convex SGM with constant step sizes $\alpha$ exhibits a complexity of $O(2^n)$, suggesting the need for further research with more flexible step size strategies ($\alpha_t$, where $t$ denotes the iteration number) instead of a fixed $\alpha$. Additionally, the time complexity scales with the number of samples $n$ and the problem's dimensionality $d$, resulting in a complexity of $O(nd)$. In the case of DGM, we investigate the relationship between $\beta$-smoothness and $\eta$-expansiveness. However, this relationship has not been proven to hold for non-convex functions in general, which implies that DGM's performance is highly sensitive to the choice of the constant step size $\alpha$. Furthermore, we examine the optimization error for DGM, which is shown to have a complexity of $O\left(\left(\frac{8\mu}{L}\right)^T\right)$, with $L$ being the Lipschitz constant, $\mu$ the Polyak-Lojasiewicz constant, and $T$ denotes the number of iterations in the DGM process. This exponential complexity is a defining feature of deterministic methods. Additionally, we derive an upper bound for the population risk $R[w]$, where $w$ represents the model parameters. Simulations for both SGM and DGM indicate that for complex non-convex functions, such as $\sum_d \sqrt{|x|}$ with dimensions $d = 1$ and $d = 10$, $\beta$-smoothness decreases significantly as the dimension increases. Despite the drastic decrease in $\beta$, the results show that convergence worsens, with larger jumps and greater inconsistency as the dimensionality increases, indicating that higher dimensions have a detrimental effect on performance.

# Contents

# Table of Notations

| Symbol | Description |
|---|---|
| $S$ | Dataset consisting of $n$ samples, $S = \{z_1, \ldots, z_n\}$. |
| $S'$ | A modified dataset differing from $S$ by a single data point. |
| $R[w]$ | Population risk associated with the model parameter $w$. |
| $R_S[w]$ | Empirical risk computed on the sample dataset $S$. |
| $\epsilon_{\text{gen}}$ | Expected generalization error. |
| $\epsilon_{\text{stab}}$ | Uniform stability error. |
| $\epsilon_{\text{opt}}$ | Optimization error bound. |
| $\epsilon$ | Precision. |
| $\mu$ | Polyak-Lojasiewicz constant. |
| $f(w; z)$ | Loss incurred by model $w$ on data point $z$. |
| $T$ | Total number of iterations in the optimization process. |
| $n$ | Number of data samples in the dataset. |
| $d$ | Dimensionality of the optimization problem. |
| $\nabla F(w)$, $\nabla f(w)$ | Gradient of function $F$ and $f$ with parameter $w$. |
| $G(w)$ | Gradient update rule mapping a point $w \in \mathbb{R}^d$ to another point. |
| $\mathbb{R}^d$ | $d$-dimensional real vector space. |
| $\delta_t$ | The norm of the difference between two weight vectors $w_t$ and $w'_t$ at iteration $t$. |
| $\alpha_t$ | Learning rate (step size) at iteration $t$. |
| $\alpha$ | Constant learning rate. |
| $w_t$, $x_t$ | Model parameter (or data point) at iteration $t$. |
| $f(w_t)$, $F(w_t)$ | Function value at $w_t$, where $w_t$ is the model's parameter at iteration $t$. |
| $F_*$ | Optimal function value (global minimum). |
| $\eta$ | Expansiveness constant. |
| $\beta$ | Smoothness constant. |
| $\langle x, y \rangle$ | Inner product of vectors $x$ and $y$. |
| $\sigma$ | 1. Upper bound for the update rule: $\sup_{w \in \Omega} \|w - G(w)\| \leq \sigma$.<br>2. Standard deviation of Gaussian noise, with $\sigma^2$ as its variance. |

Table 1: Summary of symbols and notations used in this report.

# Introduction

The Stochastic Gradient Method (SGM) is a widely used optimization technique in machine learning, particularly suited for handling large datasets efficiently. Unlike traditional methods that compute gradients using the entire dataset—often a computationally expensive process—SGM approximates the gradient by using a small, randomly selected batch of the data. This approach enables faster updates and makes SGM highly effective for large-scale optimization problems. SGM is especially crucial in fields like deep learning, where models such as neural networks, including those powering AI systems like ChatGPT, are trained. It helps speed up the training process by updating the model's parameters more frequently, with each update being based on a small, randomly selected batch of data. This reduces the time spent per iteration and uses less computing power,

which is essential for building systems that can handle large amounts of data.

This report explores several key aspects of SGM, with a particular focus on its application to non-convex optimization problems. We investigate the stability of SGM through theoretical proofs and examine how $\beta$-smooth and $\eta$-expansive properties of functions influence the performance of the method. Our analysis focuses on deriving generalization bounds and evaluating computational complexity for non-convex functions. For convex functions, existing results serve as a reference to compare and highlight the unique challenges posed by non-convex functions.

In contrast, the Deterministic Gradient Method (DGM) is a classical optimization technique where the gradient is computed using the entire dataset at each step, rather than a random subset as in SGM. DGM is typically used for optimization problems where the objective function is differentiable, and the goal is to find the global minimum of the function. This report focuses on applying DGM to non-convex optimization problems, particularly investigating the relationship between $\beta$-smoothness and $\eta$-expansiveness of the objective function. We derive theoretical results, such as bounds on the optimization error, and analyze how the properties such as smoothness and expansiveness influence the convergence of the method. Additionally, we explore the upper bound for the population risk $R[w]$.

To validate the theoretical findings, we conduct simulations using Python, applying SGM and DGM to both one-dimensional and multi-dimensional functions. These simulations are designed to highlight the practical implications of the theoretical results, particularly in terms of computational efficiency and the impact of dimensionality on the method's performance.

Research questions:

- How does the Stochastic Gradient Method (SGM) perform in optimizing non-convex functions, particularly in terms of the complexities of generalization bounds and computational time?
- What are the theoretical implications of applying SGM to non-convex optimization problems, and how do smoothness and expansiveness properties influence the performance of DGM and SGM?
- How does the dimensionality of the optimization problem affect the behavior and performance of SGM, particularly in non-convex settings?

# Deterministic Gradient Method

**Stability of Randomized Iterative Algorithms [1].**
We consider the general framework of supervised learning. The goal is to learn a model $w$ that minimizes the risk of incorrect predictions. The data points $S = (z_1, \ldots, z_n)$ are drawn independently and identically from an unknown distribution $\mathcal{D}$. The *population risk*, which measures the average loss of a model $w$, is defined as:

$$\mathcal{R}[w] \coloneqq \mathbb{E}_{z\sim\mathcal{D}} f(w; z),$$

where $f$ represents a loss function, and $f(w; z)$ is the loss calculated by the model $w$ when making a prediction based on $z$ and comparing it to the true outcome.

Since the population risk $\mathcal{R}[w]$ cannot be directly calculated (as $\mathcal{D}$ is unknown), a common approach is to use the *empirical risk*, which is computed using the sample data:

$$\mathcal{R}_S[w] \coloneqq \frac{1}{n} \sum_{i=1}^{n} f(w; z_i).$$

The difference between the empirical risk and the population risk is referred to as the *generalization error*, given by:

$$\mathcal{R}_S[w] - \mathcal{R}[w]. \tag{2.1}$$

In cases where $w = \mathcal{A}(S)$ is derived from the dataset $S$ using a potentially randomized algorithm $\mathcal{A}$, it is useful to study the expected generalization error:

$$\epsilon_{\text{gen}} \coloneqq \mathbb{E}_{S,\mathcal{A}}\big[\mathcal{R}_S[\mathcal{A}(S)] - \mathcal{R}[\mathcal{A}(S)]\big], \tag{2.2}$$

where the expectation is taken over both the randomness in the sampling of $S$ and the random behavior of the algorithm $\mathcal{A}$.

To analyze the generalization behavior of such algorithms, we use the concept of *uniform stability*, which measures how sensitive the algorithm $\mathcal{A}$ is to changes in the input data.

**Definition 1** (Uniform Stability)**.** *A randomized algorithm $\mathcal{A}$ is said to be $\epsilon$-uniformly stable if, for any two datasets $S, S' \in Z^n$ that differ in at most one element, the following holds:*

$$\sup_z \mathbb{E}_{\mathcal{A}}\big[f(\mathcal{A}(S); z) - f(\mathcal{A}(S'); z)\big] \leq \epsilon_{stab}. \tag{2.3}$$

*Here, the expectation is taken over the randomness of the algorithm $\mathcal{A}$. The value of $\epsilon_{stab}$ captures the maximum impact of a single change in the data on the output of the algorithm.*

The uniform stability of an algorithm plays a crucial role in bounding its generalization error, ensuring that the model's performance does not vary significantly with small changes in the input dataset.

**Process Overview.**
To understand the process of empirical and population loss computation in randomized algorithms, refer to the concept map in Figure 1. The diagram starts with the population distribution $\mathcal{D}$, from which a sample dataset $S$ is drawn. This dataset is used to compute the empirical loss $\mathcal{R}_S$, which measures the model's error on the training data.

The randomized algorithm $\mathcal{A}$ processes $S$ and outputs $\mathcal{A}(S)$, representing the trained model or parameter set. The diagram highlights how comparing this to $\mathcal{A}(S')$, where $S'$ differs by one element, allows the evaluation of **uniform stability**. Uniform stability measures how much the output $\mathcal{A}(S)$ changes when the input dataset changes slightly. This is key to ensuring that small changes in the dataset do not cause large changes in the model's predictions.

The concept map, Figure 1, also demonstrates the connection between **generalization** and **optimization**. While optimization focuses on minimizing the empirical loss, generalization ensures that the model performs well on unseen data by keeping the difference between empirical loss and population loss (generalization error) small. The diagram shows how uniform stability helps in bounding the generalization error, linking it to the performance of the model on the population distribution.

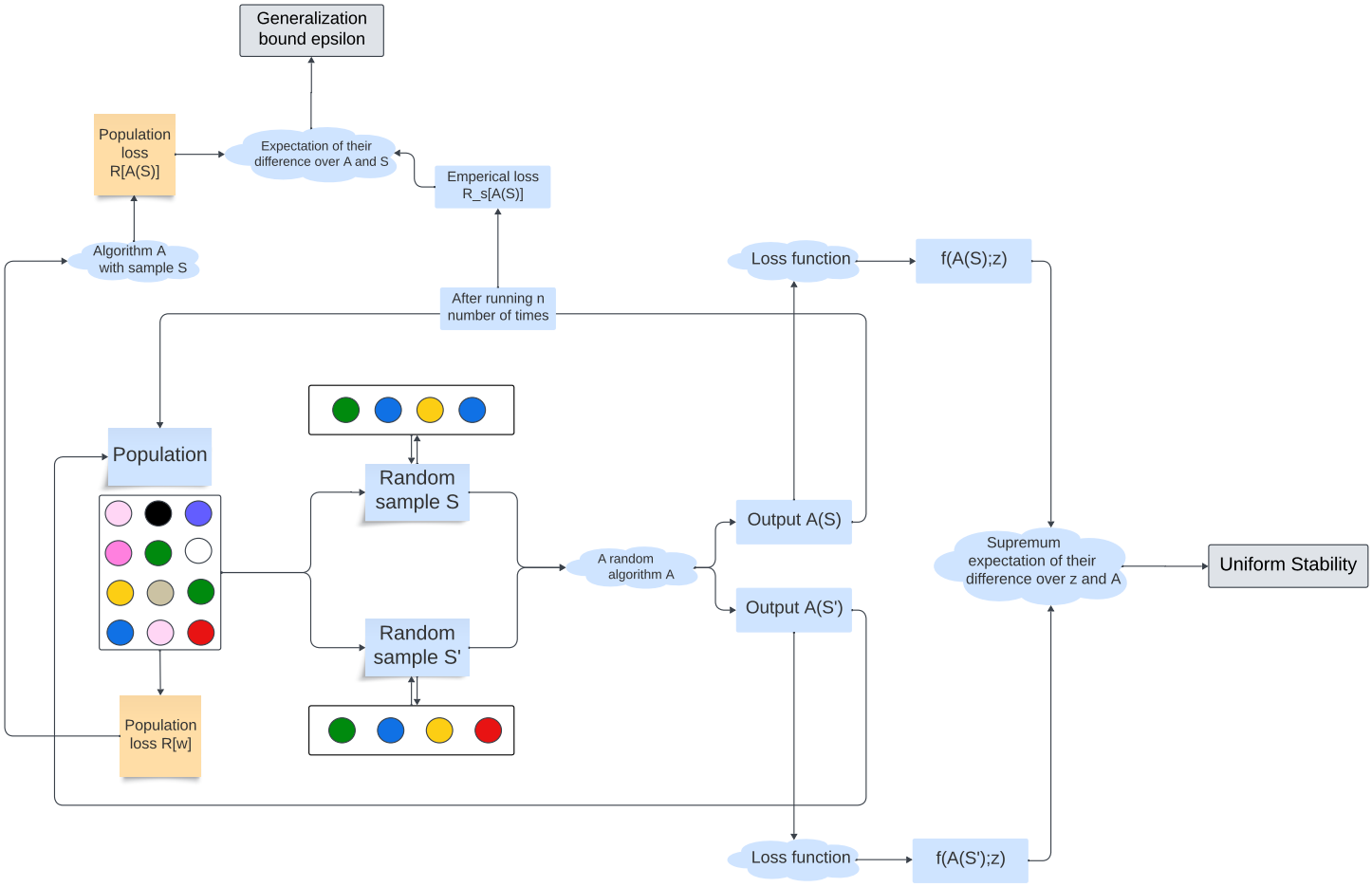


Figure 1: Concept map illustrating the relationship between population loss, empirical loss, dataset sampling, and uniform stability.

**Theorem 1** (Generalization in expectation)**.** *Let $\mathcal{A}$ be $\epsilon$-uniformly stable. Then,*

$$\left|\mathbb{E}_{S,\mathcal{A}}\left[\mathcal{R}_S[\mathcal{A}(S)] - \mathcal{R}[\mathcal{A}(S)]\right]\right| \leq \epsilon_{stab}.$$

*Proof.* Denote by $S = (z_1, \ldots, z_n)$ and $S' = (z'_1, \ldots, z'_n)$ two independent random samples, and let $S^{(i)} = (z_1, \ldots, z_{i-1}, z'_i, z_{i+1}, \ldots, z_n)$ be the sample that is identical to $S$ except in

the $i$-th example, where we replace $z_i$ with $z_i'$. With this notation, we get that

$$\begin{aligned}
\mathbb{E}_{S,\mathcal{A}}\big[\mathcal{R}_S[\mathcal{A}(S)]\big] &= \mathbb{E}_S\mathbb{E}_{\mathcal{A}}\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathcal{A}(S); z_i)\right] \\
&= \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\mathcal{A}}\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathcal{A}(S^{(i)}); z_i')\right] \\
&= \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\mathcal{A}}\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathcal{A}(S); z_i')\right] + \delta \\
&= \mathbb{E}_{\mathcal{A}}\mathbb{E}_S\big[\mathcal{R}[\mathcal{A}(S)]\big] + \delta,
\end{aligned}$$

where we can express $\delta$ as

$$\delta = \mathbb{E}_S\mathbb{E}_{S'}\mathbb{E}_{\mathcal{A}}\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathcal{A}(S^{(i)}); z_i') - \frac{1}{n}\sum_{i=1}^{n} f(\mathcal{A}(S); z_i')\right].$$

Furthermore, taking the supremum over any two data sets $S, S'$ differing in only one sample, we can bound the difference as

$$|\delta| \leq \sup_{S,S',z} \mathbb{E}_{\mathcal{A}}\big[|f(\mathcal{A}(S); z) - f(\mathcal{A}(S'); z)|\big] \leq \epsilon_{stab}, \tag{1}$$

by our assumption on the uniform stability of $\mathcal{A}$. The claim follows. □

*Remark.* Theorem 2.2 proves that if an algorithm is uniformly stable, then its generalization error is small.

## Basic Definitions and Theorems

To proceed with the proofs and derivations related to generalization bounds, optimization errors, and computational time complexities, we introduce the following definitions:

**Update rule definition and its properties [1].**

**Definition 2** (Update Rule)**.** *An **update rule** is a function $G : \Omega \to \Omega$ that maps a point $w \in \Omega$ in the parameter space to another point $G(w)$. One common form of an update rule is the gradient update, where the new point is determined by:*

$$G(w) = w - \alpha\nabla f(w),$$

*where $\alpha \geq 0$ is the step size and $f : \Omega \to \mathbb{R}$ is the function to be optimized.*

**Definition 3.** *An update rule is $\sigma$-bounded if*

$$\sup_{w\in\Omega} \|w - G(w)\| \leq \sigma.$$

**Definition 4** ($\eta$-Expansiveness)**.** *A mapping $G : \Omega \to \mathbb{R}^d$ is said to be $\eta$-expansive if there exists a constant $\eta \geq 0$ such that:*

$$\sup_{v,w\in\Omega} \frac{\|G(v) - G(w)\|}{\|v - w\|} \leq \eta, \quad \forall v, w \in \Omega.$$

UNIVERSITY OF TWENTE.

*Importance.* The Definition 4 condition ensures that the mapping $G$ does not expand the distance between any two points in the domain $\Omega$ by more than a factor of $\eta$.

Applying these properties, we can derive the following lemma that describes how a sequence of updates to a model diverges under perturbations to the training set.

**Lemma 1** (Growth recursion)**.** *Fix an arbitrary sequence of updates $G_1, \ldots, G_T$ and another sequence $G'_1, \ldots, G'_T$. Let $w_0 = w'_0$ be a starting point in $\Omega$ and define $\delta_t = \|w_t - w'_t\|$ where $w_t, w'_t$ are defined recursively through*

$$w_{t+1} = G_t(w_t), \quad w'_{t+1} = G'_t(w'_t). \quad (t > 0)$$

*Then, we have the recurrence relation for $(t > 0)$,*

$$\delta_0 = 0$$

$$\delta_{t+1} \leq \begin{cases} \eta\delta_t, & \text{if } G_t = G'_t \text{ is } \eta\text{-expansive,} \\ \min(\eta, 1)\delta_t + 2\sigma, & \text{if } G_t \text{ and } G'_t \text{ are } \sigma\text{-bounded.} \end{cases}$$

***Proof.*** *The first bound on $\delta_t$ follows directly from the assumption that $G_t = G'_t$ and the definition of expansiveness. For the second bound, recall from Definition 3 that if $G_t$ and $G'_t$ are $\sigma$-bounded, then by the triangle inequality,*

$$\delta_{t+1} = \|G(w_t) - G'(w'_t)\| \leq \|G(w_t) - w_t + w'_t - G'(w'_t)\| + \|w_t - w'_t\|$$

$$\leq \delta_t + \|G(w_t) - w_t\| + \|G(w'_t) - w'_t\| \leq \delta_t + 2\sigma,$$

*which gives half of the second bound. We can alternatively bound $\delta_{t+1}$ as*

$$\delta_{t+1} = \|G_t(w_t) - G'_t(w'_t)\| = \|G_t(w_t) - G_t(w'_t) + G_t(w'_t) - G'_t(w'_t)\|$$

$$\leq \|G_t(w_t) - G_t(w'_t)\| + \|G_t(w'_t) - G'_t(w'_t)\| \leq \eta\delta_t + 2\sigma. \quad \square$$

**Equivalence of smoothness and expansiveness for Deterministic methods [1].**

**Definition 5** ($\beta$-Smooth Function)**.** *A differentiable function $f$ is said to be $\beta$-**smooth** if its gradient is Lipschitz continuous with constant $\beta > 0$. Formally, this means that for all $v, w \in \Omega$, the following inequality holds:*

$$\sup_{v,w \in \Omega} \frac{\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|} \leq \beta,$$

*where $\Omega$ is the domain of $f$.*

*Importance.* The $\beta$-smoothness property ensures that the gradient changes predictably within $\beta$ bound.

**Definition 6** ($L$-Lipschitz Continuity)**.** *A differentiable function $f$ is said to be $L$-**Lipschitz continuous** if its gradient is bounded by $L$ over its domain. Formally, for all $v, w \in \Omega$, the following inequality holds $\|\nabla f(w)\| \leq L$ which implies:*

$$\|f(v) - f(w)\| \leq L\|v - w\|,$$

*where $L > 0$ is the Lipschitz constant and $\Omega$ is the domain of $f$.*

*Importance.* $L$-Lipschitz continuity ensures that the gradient of $f$ does not change too rapidly, which is essential for stability and convergence in optimization algorithms.

**Lemma 2.** *Assume that $f$ is $L$-Lipschitz from the Definition 6. Then, the gradient update $G_{f,\alpha}$, introduced in the Definition 2, is $(\alpha L)$-bounded.*

***Proof.*** *By the Lipschitz assumption, $\|w - G_{f,\alpha}(w)\| = \|\alpha \nabla f(w)\| \leq \alpha L$.* □

We will observe that $\beta$-smoothness and $\eta$-expansiveness are the vital conditions of the function that allow SGM to produce, within finite time, reasonable results. Thus, with the following proof we represent how both of these properties relate to each other.

**Theorem 2** (Smoothness ⇒ expansiveness)**.** *If $f$ is $\beta$-smooth according to Definition 5, then the update rule $G$ from Definition 2, with step size $\alpha > 0$, is $(1 + \alpha\beta)$-expansive.*

***Proof:***
*Using the $\eta$-expansiveness (Definition 4):*

$$\begin{aligned}\sup_{v,w\in\Omega} \frac{\|G(v) - G(w)\|}{\|v - w\|} &= \sup_{v,w\in\Omega} \frac{\|v - \alpha\nabla f(v) - (w - \alpha\nabla f(w))\|}{\|v - w\|} \\ &= \sup_{v,w\in\Omega} \frac{\|(v - w) - \alpha(\nabla f(v) - \nabla f(w))\|}{\|v - w\|}\end{aligned}$$

*Using the triangle inequality:*

$$\sup_{v,w\in\Omega} \frac{\|G(v) - G(w)\|}{\|v - w\|} \leq \sup_{v,w\in\Omega} \left[\frac{\|v - w\|}{\|v - w\|} + \frac{\alpha\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|}\right]$$

$$\leq 1 + \alpha \sup_{v,w\in\Omega} \left[\frac{\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|}\right]$$

*As $f$ is $\beta$-smooth by assumption, this leads to:*

$$\sup_{v,w\in\Omega} \frac{\|G(v) - G(w)\|}{\|v - w\|} \leq 1 + \alpha\beta.$$

*Which shows that $G$ is $(1 + \alpha\beta)$-expansive.*

**Theorem 3** (Expansiveness ⇒ Smoothness)**.** *If the update rule $G$ is $\eta$-expansive according to Definition 4, with step size $\alpha > 0$, then $f$ is $\frac{1+\eta}{\alpha}$-smooth.*

***Proof:***
*Using the definition of the update rule $G$ (Definition 2):*

$$\begin{aligned}\|G(v) - G(w)\| &= \|v - \alpha\nabla f(v) - (w - \alpha\nabla f(w))\| \\ &= \|(v - w) - \alpha(\nabla f(v) - \nabla f(w))\|\end{aligned}$$

*Taking the supremum and using the assumption that $G$ is $\eta$-expansive leads to:*

$$\sup_{v,w\in\Omega} \frac{\|G(v) - G(w)\|}{\|v - w\|} \leq \eta$$

*Substitute $G(v) = v - \alpha\nabla f(v)$ into the supremum:*

$$\sup_{v,w\in\Omega} \frac{\|(v - w) - \alpha(\nabla f(v) - \nabla f(w))\|}{\|v - w\|} \leq \eta$$

*or*

$$\sup_{v,w\in\Omega} \frac{\|\alpha(\nabla f(v) - \nabla f(w)) \ - (v - w)\|}{\|v - w\|} \leq \eta$$

*Applying the triangle inequality:*

$$\sup_{v,w\in\Omega} \frac{\alpha\|\nabla f(v) - \nabla f(w)\| - \|v - w\|}{\|v - w\|} \leq \eta$$

$$\sup_{v,w\in\Omega} \left[\frac{\alpha\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|} - 1\right] \leq \eta$$

$$\sup_{v,w\in\Omega} \left[\frac{\alpha\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|}\right] - 1 \leq \eta$$

*Rewriting:*

$$\alpha \sup_{v,w\in\Omega} \left[\frac{\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|}\right] \leq 1 + \eta$$

$$\sup_{v,w\in\Omega} \frac{\|\nabla f(v) - \nabla f(w)\|}{\|v - w\|} \leq \frac{1 + \eta}{\alpha}.$$

*Thus, $f$ is $\frac{1+\eta}{\alpha}$-smooth.*

*Remark.* These two proofs show that the smoothness of the function $f$ and the expansiveness of the update rule $G$ are tightly linked. If $f$ is $\beta$-smooth, then the update rule will have controlled, bounded expansions, ensuring stable and efficient convergence. Similarly, if $G$ is $\eta$-expansive, it provides an upper bound on the smoothness of $f$, which guarantees that the function behaves in a way that the optimization algorithm can handle effectively. Thus, both of these relations are vital to ensuring that DGM produces meaningful results within a finite number of iterations.

## Population risk and Optimization error of DGM

In this subsection, we analyze how well the Deterministic Gradient Method generalizes and how efficient it is in terms of complexity. The goal is to explain the balance between stability (how consistent the results are with changes in the data) and optimization accuracy (how well DGM minimizes the training error). We first establish an upper bound for the expected risk using both the optimization error and the stability term. Then, we define the optimization error as a way to measure the difference between the training error and the smallest possible empirical error.

## Decomposing Population Risk with Optimization Error

The *optimization error* quantifies the gap between the empirical risk of the model parameter $w$ and the minimal empirical risk achievable by the optimal parameters $w_\star^S$ in expectation. It is defined as:

$$\epsilon_{\text{opt}}(w) \coloneqq \mathbb{E}\left[R_S[w] - R_S[w_\star^S]\right], \tag{2}$$

where $w_\star^S = \arg\min_w R_S[w]$. Here, $w_\star^S$ represents the model parameters that achieve the minimum value of the empirical risk $R_S[w]$. The notation $\arg\min_w R_S[w]$ refers to the argument $w$ that minimizes $R_S[w]$, i.e., the set of weights $w$ that minimizes the training error for the given dataset $S$.

Building upon Theorem 1, the following result establishes an upper bound for the expected risk of a model produced by DGM.

**Theorem 4** (Population risk upper bound)**.** *The expected population risk of a model $w$ learned by DGM is bounded as:*

$$\mathbb{E}_{S,\mathcal{A}}[R[w]] \leq \mathbb{E}_S[R_S[w_\star^S]] + \epsilon_{opt} + \epsilon_{stab}, \tag{3}$$

*where $\epsilon_{stab}$ denotes the stability term, $\epsilon_{opt}$ represents the optimization error, and the expectations are taken over both the dataset sampling $S$ and the randomness of the algorithm $\mathcal{A}$.*

***Proof.*** *In the context of supervised learning, where $S = (z_1, \ldots, z_n)$ is sampled i.i.d. from an unknown population distribution $\mathcal{D}$, the population risk $R[w]$ measures the expected loss:*

$$R[w] = \mathbb{E}_{z\sim\mathcal{D}}[f(w; z)].$$

*Given that $w = \mathcal{A}(S)$ is generated by a randomized algorithm $\mathcal{A}$, the expected population risk can be expressed as:*

$$\mathbb{E}_{S,\mathcal{A}}[R[w]] = \mathbb{E}_{S,\mathcal{A}}\big[\mathbb{E}_{z\sim\mathcal{D}}[f(\mathcal{A}(S); z)]\big].$$

*By Theorem 1, the population risk can be bounded in terms of the empirical risk $\mathbb{E}_{S,\mathcal{A}}[R_S[w]]$ and the stability term $\epsilon_{stab}$, which measures how sensitive the algorithm is to small changes in the dataset:*

$$\mathbb{E}_{S,\mathcal{A}}[R[w]] \leq \mathbb{E}_{S,\mathcal{A}}[R_S[w]] + \epsilon_{stab}.$$

*Now, consider the empirical risk $\mathbb{E}_{S,\mathcal{A}}[R_S[w]]$. By introducing the optimal empirical risk $R_S[w_\star^S]$, where $w_\star^S$ minimizes the empirical risk over $S$, we can further break down the bound:*

$$\mathbb{E}_{S,\mathcal{A}}[R_S[w]] \leq \mathbb{E}_S[R_S[w_\star^S]] + \epsilon_{opt},$$

*where $\epsilon_{opt}$ represents the optimization error of the algorithm $\mathcal{A}$ when approximating $w_\star^S$. Combining these two bounds gives:*

$$\mathbb{E}_{S,\mathcal{A}}[R[w]] \leq \mathbb{E}_S[R_S[w_\star^S]] + \epsilon_{opt} + \epsilon_{stab}.$$

*This result accounts for both the randomness in sampling from the population and the randomness inherent in the algorithm $\mathcal{A}$, completing the proof.* □

We decomposed the population risk into three components: the minimum training error, the optimization error, and the stability term. The optimization error measures the gap between the empirical risk of the trained model and the empirical risk minimizer. Understanding this term is important for identifying areas where the optimization process can be refined, eventually leading to improved generalization and efficiency of DGM.

## Optimization Error in Non-Convex DGM.

Similar to the proof of the Theorem 4 from the report [2], we conduct the proof for the optimization error of the DGM.

**Theorem 5** (Optimization Error Upper Bound for SGM)**.** *Let $F(w)$ be a smooth function that satisfies the Polyak-Lojasiewicz (PL) Inequality [2] with some constant $\mu > 0$ (the PL constant), meaning that:*

$$\frac{1}{2}\|\nabla F(w)\|^2 \geq \mu\left(F(w) - F_\star\right), \quad \forall w.$$

*Here, $F_\star$ is the minimum value of $F(w)$, and $w_\star$ is the corresponding minimizer. Additionally, assume that $F(w)$ is L-Lipschitz continuous [2], i.e., for all $w, v \in \mathbb{R}^d$, we have:*

$$F(v) \leq F(w) + \langle \nabla F(w), v - w\rangle + \frac{L}{2}\|v - w\|^2.$$

*Suppose we use the iterative update:*

$$w_{t+1} = w_t - \alpha_t \nabla F(w_t),$$

*where $\alpha_t = \alpha = \frac{2}{L}$ is a constant step size. Then the optimization error after $T$ iterations is bounded as:*

$$F(w_T) - F_\star \leq \frac{L}{2}\|w_1 - w_\star\|^2 \left(2 - \frac{8\mu}{L}\right)^{T-1}.$$

***Proof.*** *We begin with the following assumptions:*

$$w_{t+1} = w_t - \alpha_t \nabla F(w_t),$$

$$\|\nabla F(w_t)\|^2 \geq 2\mu(F(w_t) - F_\star),$$

$$|F(w) - F(v) - \langle \nabla F(v), w - v\rangle| \leq \frac{L}{2}\|w - v\|^2.$$

*Now, using the update rule for $w_{t+1}$, we have:*

$$|F(w_{t+1}) - F(w_t) - \langle \nabla F(w_t), w_{t+1} - w_t \rangle| \leq \frac{L}{2}\|w_{t+1} - w_t\|^2,$$

$$F(w_{t+1}) - F(w_t) + \alpha_t \|\nabla F(w_t)\|^2 \leq \frac{L\alpha_t^2}{2}\|\nabla F(w_t)\|^2,$$

$$F(w_{t+1}) - F(w_t) + \left(\alpha_t - \frac{L\alpha_t^2}{2}\right)\|\nabla F(w_t)\|^2 \leq 0.$$

*If $\left(\alpha_t - \frac{L\alpha_t^2}{2}\right) \geq 0$, then:*

$$F(w_{t+1}) - F(w_\star) \leq 2\left(1 - \mu\alpha_t - \frac{\mu L\alpha_t^2}{2}\right)(F(w_t) - F_\star).$$

*Taking a constant step size $\alpha_t = \alpha = \frac{2}{L}$ that minimizes the expression $\left(1 - \mu\alpha_t - \frac{\mu L\alpha_t^2}{2}\right)$, the inequality simplifies as follows:*

$$F(w_{t+1}) - F_\star \leq \left(2 - \frac{8\mu}{L}\right)(F(w_t) - F_\star).$$

*By applying this iteratively from $t = 1$ to $T - 1$, we obtain the telescopic product:*

$$F(w_T) - F_\star \leq \left(2 - \frac{8\mu}{L}\right)^{T-1}(F(w_1) - F_\star).$$

***Smoothness Property:***
*Given the smoothness property of $F$, we have:*

$$F(w) \leq F(w_\star) + \langle \nabla F(w_\star), w - w_\star \rangle + \frac{L}{2}\|w - w_\star\|^2.$$

*Lemma 2.22 from [3] says that $\nabla F(w_\star) = 0$, thus this simplifies to:*

$$F(w_1) - F_\star \leq \frac{L}{2}\|w_1 - w_\star\|^2.$$

*Substituting this into the earlier bound gives:*

$$F(w_T) - F_\star \leq \frac{L}{2}\|w_1 - w_\star\|^2\left(2 - \frac{8\mu}{L}\right)^{T-1}.$$

□

This result highlights the exponential convergence of the function value towards $F_\star$, modulated by the distance of the initial point $w_1$ from $w_\star$ and constants $L$ and $\mu$. This exponential bound arises due to the deterministic nature of the update rule; further details are provided in the *Discussion* section.

Additionally, [3] stated a Corollary 3.5 which can be exploited to calculate the optimization error complexity for convex functions. We apply the same corollary to state

that non-convex functions have optimization error complexity as $O((\frac{8\mu}{L})^T)$.

**Computational time complexity.**
Referring to [1] the computational time complexity for convex DGM and SGM are $O(n^{1.5})$ and $O(n)$ respectively, where $n$ represents the number of samples or data points. While based on the Appendix code the computational complexity for the non-convex SGM is $O(Td)$, where $T$ is the number of iterations for the SGM and $d$ is the dimension of the non-convex function. We could choose $T = n$, with $n$ as the number of samples and obtain $O(nd)$.

# Stochastic Gradient Method

## Generalization bound for convex SGM

**Definition 7** (Convex Function)**.** *A function $f : \Omega \to \mathbb{R}$ is convex if, for all $u, v \in \Omega$, we have*

$$f(u) \geq f(v) + \langle \nabla f(v), u - v \rangle.$$

According to [1], the generalization bound, from the equation (2.2), is given by:

$$\epsilon_{stab} \leq \frac{2L^2}{n} \sum_{t=1}^{T} (\alpha_t)^{T-t}.$$

where:

- $\alpha_t$: Step size at iteration $t$.
- $L$: Lipschitz constant of the gradient of the loss function.
- $n$: Number of samples in the dataset.
- $T$: Total number of iterations.
- $\epsilon_{\text{stab}}$: Uniform stability measure from the Definition 1.

This result highlights several key dependencies. The bound inversely depends on the number of samples $n$, implying that larger datasets generally improve generalization. Additionally, the learning rate $\alpha_t$ plays a crucial role in the stability of the method, as it exponentially affects the bound through the term $(\alpha_t)^{T-t}$. This suggests that careful tuning of $\alpha_t$ over iterations is essential to control the generalization error, particularly in longer training processes.

## Generalization bound for non-convex functions in SGM

The definition of the non-convexity can be derived from the definition of convexity (Definition 7):

**Definition 8** (Non-Convex Function)**.** *A function $f : \Omega \to \mathbb{R}$ is non-convex if it does not satisfy the convexity condition. This means that there exist points $u, v \in \Omega$ where the inequality*

$$f(u) < f(v) + \langle \nabla f(v), u - v \rangle$$

*holds.*

Together with the application of Lemma 3 [1], which is essential for deriving the generalization bound for non-convex SGM, the resulting bound is expected to resemble the form of the convex generalization bound but with increased sophistication:

**Lemma 3.** *Assume that $f$ is $\beta$-smooth. Then, the following properties hold:*

1. *According to the Definition 2, the update rule $G_{f,\alpha}$ is $(1 + \alpha\beta)$-expansive.*
2. *Assume in addition that $f$ is convex from the Definition 7. Then, for any $\alpha \leq \frac{2}{\beta}$, the gradient update $G_{f,\alpha}$ is 1-expansive.*

**Theorem 6** (Generalization bound of non-convex SGM)**.** *Assume that the loss function $f(\cdot, z)$ is $\beta$-smooth, non-convex, and $L$-Lipschitz for every $z$. Suppose that we run SGM with a constant step size $\alpha_t = \alpha$ for $T$ steps. Then SGM satisfies*

$$\epsilon_{stab} \leq \frac{2L^2\alpha}{n} \sum_{t=1}^{T} (1 + \alpha\beta)^{T-t}.$$

*where $\epsilon_{stab}$ is a uniform stability measure.*

***Proof.*** *Suppose $S$ and $S'$ are two samples of size $n$, differing in only a single example. Consider the gradient updates $G_1, \ldots, G_T$ and $G'_1, \ldots, G'_T$ induced by running SGM on samples $S$ and $S'$, respectively.*

*Fix an example $z \in Z$ and apply the Lipschitz condition on $f(\cdot, z)$ to get*

$$\mathbb{E}|f(w_t, z) - f(w'_t, z)| \leq L\mathbb{E}[\delta_t], \tag{1}$$

*where $\delta_t = \|w_t - w'_t\|$. Observe that at step $t$, with probability $1 - \frac{1}{n}$, the example selected by SGM is the same in both $S$ and $S'$. In this case, $G_t = G'_t$. From Lemma 3.1, we can use the fact that the update rule $G_t$ is $(1 + \alpha\beta)$-expansive due to non-convexity and the constant step size.*

*With probability $\frac{1}{n}$, the selected example is different, in which case we use that both $G_t$ and $G'_t$ are $\alpha L$-bounded as a consequence of Lemma 2. Hence, we can apply Lemma 1 and linearity of expectation to conclude that for every $t$,*

$$\mathbb{E}[\delta_{t+1}] \leq \Big(1 - \frac{1}{n}\Big)\mathbb{E}[(1 + \alpha\beta)\delta_t] + \frac{1}{n}(\mathbb{E}[(1 + \alpha\beta)\delta_t + 2\alpha L]) =$$

$$= (1 + \alpha\beta)\mathbb{E}[\delta_t] + \frac{2\alpha L}{n}$$

*Expanding this recursion we derive the inequality:*

$$\mathbb{E}[\delta_T] \leq \frac{2L}{n} \sum_{t=0}^{T-1} (1 + \alpha\beta)^{T-t-1}\alpha.$$

*Here $\delta_0 = 0$ and by appending $T$'s step leads to:*

$$\mathbb{E}[\delta_T] \leq \frac{2L}{n} \sum_{t=1}^{T} (1 + \alpha\beta)^{T-t} \alpha.$$

*Plugging this back into Equation (1), we obtain*

$$\mathbb{E}|f(w_T, z) - f(w'_T, z)| \leq \frac{2L^2\alpha}{n} \sum_{t=1}^{T} (1 + \alpha\beta)^{T-t}.$$

*Since this bound holds for all $S$, $S'$, and $z$, we obtain the desired bound on the uniform stability.* □

This bound, while similar to the convex case, shows a more complex dependence on smoothness and expansiveness. The term $(1 + \alpha\beta)^{T-t}$ can grow exponentially if $\alpha\beta > 0$, which poses a significant challenge when analyzing the stability of SGM for non-convex functions. This exponential growth highlights the instability introduced by large step sizes or high smoothness constants $\beta$ in non-convex optimization.

**Complexity of the Generalization Bound**
The complexity of the generalization bound of SGM can be derived from Theorem 6:

$$\epsilon_{\text{stab}} \leq \frac{2L^2\alpha}{n} \sum_{t=1}^{T} (1 + \alpha\beta)^{T-t}.$$

By selecting specific parameter values, such as the step size $\alpha = \frac{1}{\beta}$ and the number of iterations $T = n$, the expression simplifies, leading to:

$$\epsilon_{\text{stab}} \in O(2^n).$$

This result highlights the exponential growth of the generalization bound complexity with respect to $n$, emphasizing the limitations of using fixed step sizes.

## Summary table

We conclude the final results using the results of [1] and the results for non-convex SGM in the Table 2.

| | **DGM-convex** | **SGD-convex** | **SGD non-convex** |
|---|---|---|---|
| Computational time: | $O(n^{1.5})$ | $O(nd)$ | $O(nd)$ |
| Generalization bound: | $O(1/\sqrt{n})$ | $O(1/\sqrt{n})$ | $O(2^n)$ |

Table 2: Comparison of DGM and SGM for Convex and Non-Convex Functions

# Discussion/Extensions to SGM

## Deterministic Nature of Updates and Optimization Error Bounds

The proof of Theorem 5 establishes the iterative update rule for $w_{t+1}$ as:

$$w_{t+1} = w_t - \alpha_t \nabla F(w_t),$$

where the deterministic nature of the gradient update dictates the trajectory of $w_t$. A vivid illustration of the deterministic nature of the gradient update in 3D is represented by [4]. Unlike stochastic methods that introduce randomness, deterministic updates ensure that the sequence of iterates $w_t$ is uniquely determined by the initial value $w_1$ and the step size $\alpha_t$.
This deterministic path yields a telescoping product in the error bound:

$$F(w_T) - F_\star \leq \frac{L}{2}\|w_1 - w_\star\|^2 \left(1 - \frac{4\mu}{L}\right)^{T-1},$$

where the convergence rate is influenced by the Polyak-Lojasiewicz (PL) constant $\mu$ and the smoothness constant $L$.

The term $\left(1 - \frac{4\mu}{L}\right)^{T-1}$ arises directly from the deterministic nature of the update rule, where each step deterministically drives the iterate closer to the optimal minimizer $w_\star$. This predictable trajectory leads to an exponential decay in the optimization error. More precisely, as it was shown after proving Theorem 5 the optimization error is of the form:

$$F(w_T) - F_\star \in O\left(\left(\frac{4\mu}{L}\right)^T\right),$$

indicating exponential growth in the error over time. This exponential growth is a direct consequence of the absence of random fluctuations or dynamic adjustments in the step size, which would otherwise influence the convergence rate.

In contrast, for Stochastic Gradient Methods, deriving optimization error bounds requires distinct approaches due to the inherent stochasticity in the gradient estimates. Several methodologies have been proposed to handle the challenges posed by this randomness:

- **Variance Decomposition**: This approach involves decomposing the error into deterministic and stochastic components, which helps in accounting for the variance introduced by gradient sampling. A relevant discussion of this technique can be found in [5], where they explore how variance reduction strategies can improve convergence.

- **Averaging Techniques**: Averaging over multiple iterates or gradient estimates helps reduce noise and improve convergence stability. For instance, [6] discuss how mini-batching and averaging strategies can enhance parallel Stochastic Gradient Descent(SGD) for regression.

- **Expected Error Analysis**: This method focuses on estimating the expected error by making specific assumptions about the distribution of the stochastic gradients. For example, [7] examine how such assumptions can be used to analyze the generalization error of SGD in classification problems, particularly under low-noise conditions.

These approaches typically rely on the smoothness assumptions of the objective function (e.g., Lipschitz continuity as defined in Definition 6, and the Polyak-Lojasiewicz inequality as stated in Theorem 5), along with the properties of the stochastic gradients. A more detailed exploration of these methods within the context of SGM requires further analysis, which falls outside the scope of the deterministic framework discussed here.

## Possibility for Other Superior Methods to SGM

Proximal Stochastic Gradient Algorithms, such as ProxSVRG, have gained attention due to their effectiveness in optimizing non-convex functions with non-smooth regularizers. These methods address the main limitations of standard Stochastic Gradient Methods, including their difficulty in efficiently handling non-smooth terms and their reduced convergence speed when optimizing functions with sharp variations or irregular structures [8]. By including proximal operators, ProxSVRG can effectively deal with non-smooth regularizers (e.g., $\ell_1$-norms or total variation penalties), ensuring more stable updates and avoiding oscillations [8]. As a result, ProxSVRG achieves faster convergence and better scalability compared to SGM, which often relies on assumptions of smoothness and can become inefficient when these assumptions are violated [1].

**Complexity and Performance.**
According to the report by [8], where in the Table 1 they illustrateed the complexities as follows:

- **ProxSVRG [8]:**
  - **Stochastic First-Order Oracle (SFO) Complexity:** $O(n + n^{2/3}\epsilon^{-2})$.
  - **Proximal Oracle (PO) Complexity:** $O(\epsilon^{-2})$.

  ProxSVRG exhibits an efficient dependency on the accuracy parameter $\epsilon$, achieving $\epsilon^{-2}$ for the proximal oracle and leveraging minibatch sizes for scalability.

Comparing it to the complexities of the SGM for the non-convex case we refer to [9] where it was stated that:

- **Non-Convex Stochastic Gradient Descent [9]:**
  - **Complexity:** $O(\epsilon^{-4})$.

  The SGM complexity scales worse with $\epsilon$, reflecting a slower convergence rate when high precision is required.

**Challenges and Future Directions.**
While ProxSVRG+ offers significant advantages, it also presents certain challenges. The increased computational cost per iteration can be problematic in environments with limited resources. Moreover, ensuring that the required smoothness conditions are satisfied in practical applications can be difficult, which may limit the method's applicability in some real-world scenarios.

Future research could focus on the following:

- Developing adaptive methods that reduce the dependence on strict smoothness assumptions.
- Looking into combined methods that merge ProxSVRG with other algorithms to improve the trade-off between the time needed per iteration and the overall speed of convergence.
- Expanding ProxSVRG to work with a wider variety of non-smooth and non-convex regularizers, making it more flexible for different types of problems.

# Simulation

In this section, we analyze the behavior of Stochastic Gradient Methods together with Deterministic Gradient Method through experiments that examine smoothness, expansiveness and other theoretical properties. We aim to validate key theorems, explore the impact of noise and dimensionality, and highlight practical discrepancies with theoretical predictions.

**Behavior of DGM on Smooth and Non-Smooth Functions.**
On the comparison between the behavior of DGM applied to the 1-smooth function $f(x) = x^2$ and the non-smooth function $f(x) = \sqrt{x}$ on the interval $[0, 1]$.

- **1-Smooth Function:** The function $f(x) = x^2$ was tested, which is convex and 1-smooth. During the simulation, DGM did not converge exactly within the predefined number of iterations, yielding an average result of approximately $w_t \approx 0.2663$ for the 1000-point setup. After reducing the interval to 500 points, the result improved to $w_t = 0.0931$. However, this outcome is influenced by the randomness of selecting a specific point on the interval $[0, 1]$. In this case, fewer iterations yielded more accurate results in less time, which aligns with the findings of [1], who suggested that for Gradient Methods, fewer iterations with sufficiently small steps are often sufficient to achieve good results without needing many iterations.

- **Non-Smooth Function:** The function $f(x) = \sqrt{x}$, which is not beta-smooth, was also tested. Despite its non-smooth nature, SGM converged to the exact solution after appropriately adjusting the dataset size. This behavior is attributed to the numerical methods in Python packages, which approximate the gradient of non-smooth functions. On the interval $[0, 1]$, the gradient approximations caused monotonic jumps toward zero, facilitating convergence. This outcome reflects the interaction between the algorithm and the numerical properties of the domain and gradient approximations.

- **Challenges with Non-Smooth Functions:** Simulations revealed that certain function calls for non-smooth functions, particularly those implementing $\eta$-expansiveness, led to infinite running times. To ensure efficiency, these function calls should be either removed or appropriately adjusted for handling non-smooth functions in the DGM framework.

These two examples underline the importance of the $\beta$-smoothness property, which ensures that the simulation can progress smoothly, converge effectively, and terminate within a reasonable time frame. Additionally, they illustrate how $\eta$-expansiveness impacts the performance of DGM. Although $\eta$-expansiveness is crucial in theory, its application to non-smooth functions introduces challenges, such as causing infinite idle time in the algorithms. This highlights the need for more effective methods to manage $\eta$-expansiveness in practice, particularly when dealing with non-smooth functions.

## Simulation with Stochastic Gradient Method and Deterministic Gradient Method

To analyze the stochastic aspect of the gradient descent method, we simulate its behavior by introducing Gaussian noise to the gradients at each iteration. We observe how randomness or stochasticity affects convergence and explore its impact on uniform stability. The procedure is as follows:

1. The function $f(x) = \sqrt{|x|}$ is selected for analysis.
2. A random starting point $x_0$ is chosen uniformly from the interval $[-1, 1]$.
3. $n$ samples of Gaussian noise with zero mean and variance $\sigma^2 = 0.01$ are generated.
4. Over $T = 1000$ steps of stochastic gradient descent:
   - Compute the exact gradient $f'(x_t)$.
   - Choose a noise sample $s_t$ uniformly from the $n$ samples.
   - Compute a stochastic gradient $g_t = f'(x_t) + s_t$.
   - Perform the descent step $x_{t+1} = x_t - \alpha_t g_t$.

Using parameters $T = 1000$, $n = 100$, and $\sigma = 0.1$ and step size $\alpha_t = \alpha = 0.001$ the experiment is repeated three times with random initial values. The table below represents the results of the simulation:

| **Sample** | **Initial $x_t$ (Iteration 0)** | **Final $x_t$ (Iteration 1000)** |
|---|---|---|
| Sample 1 | -0.01034 | 0.01509 |
| Sample 2 | -0.69577 | -0.01286 |
| Sample 3 | 0.53296 | -0.01369 |

Table 3: Results of Stochastic Gradient Descent

**SGM Plots.**
The following figures illustrate the behavior of the functions $f(x) = x^2$ and $f(x) = \sqrt{|x|}$ under different conditions. The Figure 2 and Figure 3 show $f(x) = x^2$ without any noise(DGM) and with noise(SGM). The second set of figures shows $f(x) = \sqrt{|x|}$ with Gaussian noise applied to the gradient at each iteration.

**Function Analysis Results.**
For the function $f(x) = x^2$, the results are as follows:

- The function is convex.
- $\beta -$ smoothness =2.0.
- $\gamma -$ expansiveness = 1.0.

For the function $f(x) = \sqrt{|x|}$, the results are as follows:

- The function is non-convex.
- $\beta -$ smoothness $= 225,020,049$.
- $\gamma -$ expansiveness $= 14,434$.

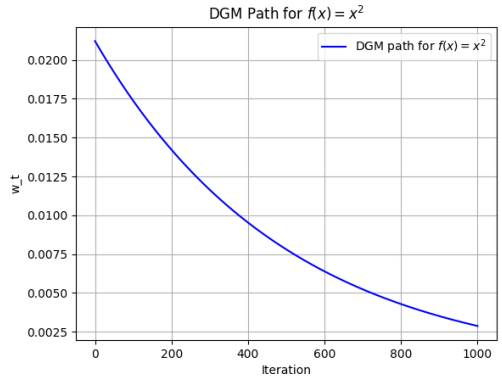


Figure 2: DGM:$f(x) = x^2$.

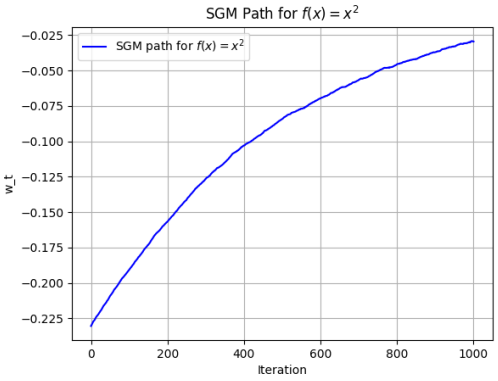


Figure 3: SGM:$f(x) = x^2$

Figure 4: Comparison of DGM and SGD behavior for $f(x) = x^2$.

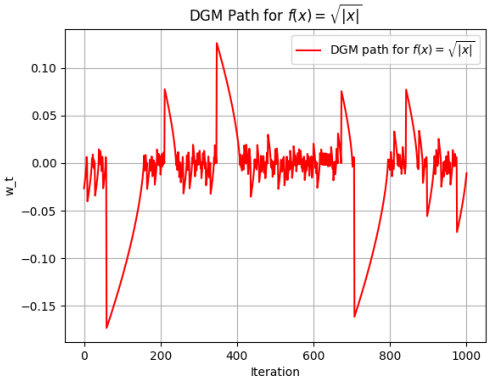


Figure 5: DGM of $f(x) = \sqrt{|x|}$

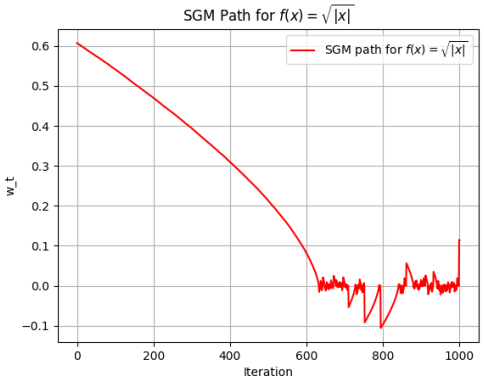


Figure 6: SGM of $f(x) = \sqrt{|x|}$

Figure 7: Comparison of DGM and SGD behaviors for $f(x) = \sqrt{|x|}$.

**Validation of Theorem 2 and Theorem 3 through Simulation.**
According to Theorem 2, if $f$ is $\beta$-smooth, then $G$ is $(1+\alpha\beta)$-expansive. For the function $f(x) = x^2$, we have $\beta = 2.0$ and step size $\alpha = 0.001$, which results in an expansiveness of:

$$1 + \alpha\beta = 1 + 2.0 \times 0.001 = 1.002 > 1.0.$$

This is consistent with the theory, confirming that the function is convex and exhibits the expected behavior.

For the non-convex function $f(x) = \sqrt{|x|}$, with $\beta = 225,020,049$ and $\alpha = 0.001$, we compute the expansiveness as:

$$1 + \alpha\beta = 1 + 225,020,049 \times 0.001 = 225,022 > 14,434$$

which also aligns with the expected outcome of a highly expansive function, as predicted by Theorem 2.

Additionally, from Theorem 3, for $G$ being $\eta$-expansive, we expect $f$ to be $\frac{1+\eta}{\alpha}$-smooth, with the same step size $\alpha = 0.001$.

For the convex function $f(x) = x^2$, we found that the expansiveness of $G$ is $\eta = 1.0$. Substituting this value into the relation:

$$\beta = \frac{1+\eta}{\alpha} = \frac{1+1.0}{0.001} = \frac{2}{0.001} = 2000.0 > 2.0.$$

This large value of $\beta = 2000.0$ contrasts with the actual smoothness of the function, which is $\beta = 2.0$, suggesting that there are additional factors at play that influence the smoothness of the function, such as the step size or the specific nature of the function.

For the non-convex function $f(x) = \sqrt{|x|}$, we found the expansiveness of $G$ to be $\eta = 14,434$. Substituting this value into the relation:

$$\beta = \frac{1+\eta}{\alpha} = \frac{1+14,434}{0.001} = \frac{14,435}{0.001} = 14,435,000 < 225,020,049$$

The expected value of $\beta = 14,435,000$ is much smaller than the observed value of $\beta = 225,020,049$, indicating a discrepancy. The actual smoothness for the non-convex function is much higher than predicted by the relation, suggesting that the theoretical relation does not fully capture the complexities of the non-convex function's gradient, which may be more irregular and erratic than expected.

*Summary.* The discrepancies between theoretical and actual values of $\beta$-smoothness can be attributed to the limitations of the relations that describe smoothness and expansiveness. For the convex function $f(x) = x^2$, the expansiveness was consistent with the theory, but the expected $\beta$-smoothness from Theorem 3 was much higher than the actual value, suggesting that factors such as step size influence smoothness. For the non-convex function $f(x) = \sqrt{|x|}$, the expansiveness value was much larger than expected, and substituting the observed expansiveness into the relation produced a smaller $\beta$, indicating that the theoretical relations do not fully capture the complexities of non-convex functions.

*Adjustment of Findings:* The discrepancies between theoretical and observed values of $\beta$-smoothness can be explained by the sensitivity of the relations to the step size $\alpha$. Specifically, by adjusting the step size $\alpha$ (e.g., from 0.001 to 0.1), we can align the theoretical and simulated results, which shows that the conditions of the theorems are highly dependent on a "reasonably chosen" step size $\alpha$. This adjustment allows the relations to hold, but emphasizes the importance of carefully selecting the step size $\alpha$ in simulations and practical applications to ensure consistency with theoretical expectations.

These results suggest that while the relations provide a vague estimate, they do not fully capture the behavior of these functions, essentially for non-convex functions where irregularities in the gradient can have a significant impact.

## Multidimensional SGM

We simulate the SGM for two functions in $d = 10$ dimensions:

- $F_1(x) = \sum_{i=1}^{d} x_i^2$ (Euclidean norm squared).
- $F_2(x) = \sum_{i=1}^{d} \sqrt{|x_i|}$ (sum of square roots of absolute values).

Two scenarios are considered: with Gaussian noise(SGM) and without noise(DGM). The results are presented below.

**Trajectories with Gaussian Noise(SGM).**
The SGM trajectories for the functions are shown in Figure 8.

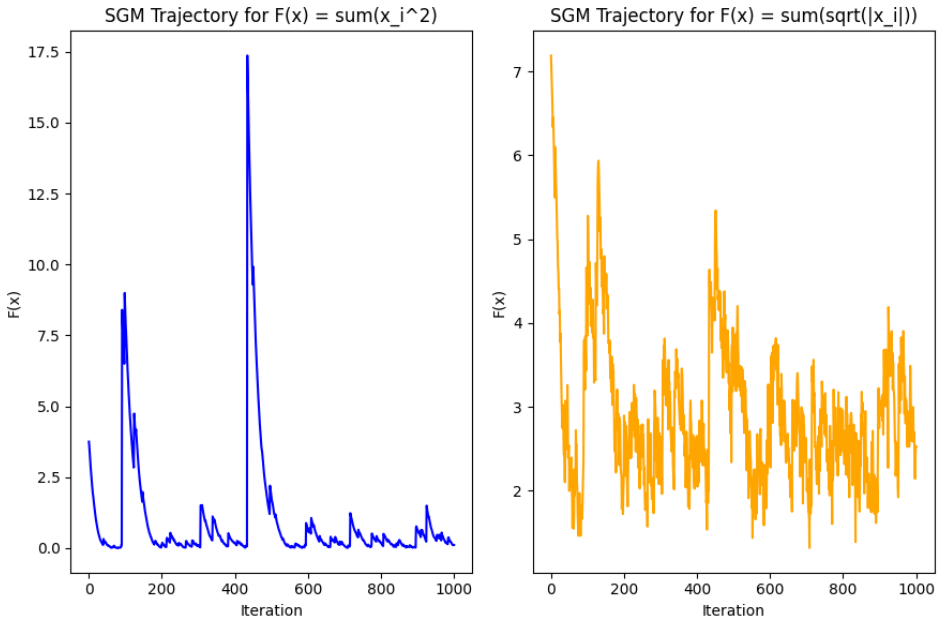


Figure 8: SGM Trajectories for $d = 10$. Left: $F_1(x) = \sum_{i=1}^{d} x_i^2$. Right: $F_2(x) = \sum_{i=1}^{d} \sqrt{|x_i|}$.

**Trajectories Without Noise(DGM).**
The DGM trajectories for the functions are shown in Figure 9.

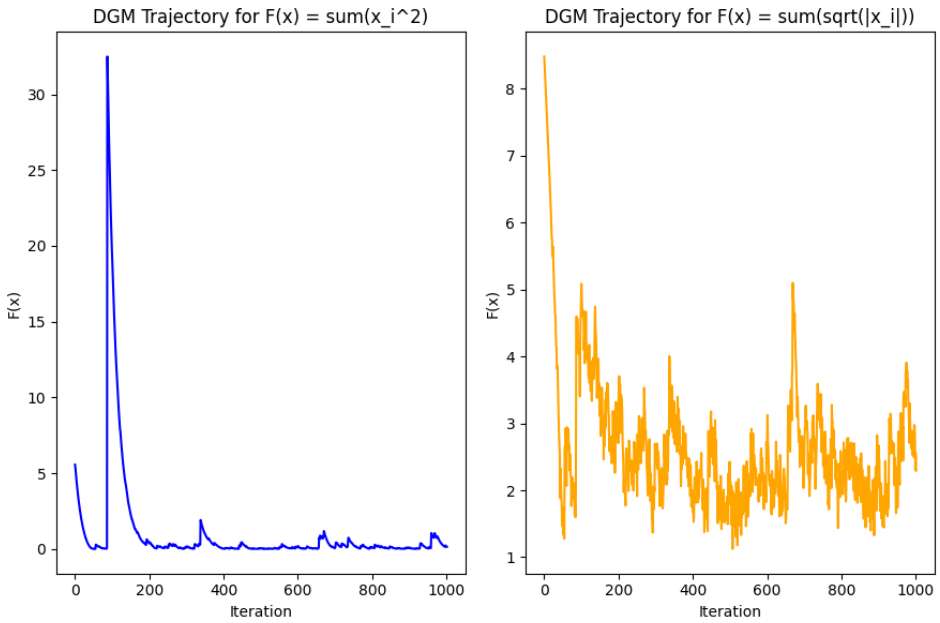


Figure 9: DGM Trajectories for $d = 10$. Left: $F_1(x) = \sum_{i=1}^{d} x_i^2$. Right: $F_2(x) = \sum_{i=1}^{d} \sqrt{|x_i|}$.

**Analysis of Convergence Behavior.**
The results from the Figure 8 and Figure 9 clearly show differences in the convergence behavior of the SGM and DGM for the two test functions, $F_1(x) = \sum_{i=1}^{d} x_i^2$ and $F_2(x) = \sum_{i=1}^{d} \sqrt{|x_i|}$, under identical experimental conditions.

For $F_1(x)$, which is convex, with $\beta = 2.00$, the smoothness properties remain well-suited for optimization even as the dimension increases to $d = 10$. This is consistent with the observations for $d = 1$, where the updates remained stable, and both SGM and DGM reliably converged to the global minimum. Table 5 further confirms this, showing that $w_t$ approaches zero as the number of iterations progresses.

In contrast, for $F_2(x)$, which is non-convex, with $\beta = 60.28$, both methods continue to exhibit poor convergence behavior as $d$ increases. The instability in the updates, caused by the large $\beta$-value, which was first observed for $d = 1$, becomes more pronounced at higher dimensions. This results in inconsistency and jumps in $w_t$, even after 1000 iterations.

This extension to $d = 10$ reinforces our earlier conclusion: the $\beta$-smoothness of the function has a significant impact on the optimization process. While $F_1(x)$ benefits from a manageable $\beta$, the large $\beta$ of $F_2(x)$ hampers optimization efficiency and makes convergence impossible. Notably, the performance worsens as the dimensionality increases, confirming that higher dimensions exacerbate the difficulty of convergence.

**Table of the results.**
Take the parameters for the table: $T = 1000$, $d = 10$, and $\sigma = 0.1$. We do again the experiment 3 times with random initial values. The tables below represent the results of the simulation:

| Sample | Initial $x_t$ (Iteration 0) | Final $x_t$ (Iteration 1000) |
|---|---|---|
| Sample 1 | 0.235272 | -0.000134 |
| Sample 2 | 0.583524 | 0.059467 |
| Sample 3 | -0.232673 | 0.058326 |

Table 4: Results of Stochastic Gradient Descent for $F_1(x) = \sum_{i=1}^{d} x_i^2$

| Sample | Initial $x_t$ (Iteration 0) | Final $x_t$ (Iteration 1000) |
|---|---|---|
| Sample 1 | 0.224963 | 0.432106 |
| Sample 2 | 0.576978 | 0.038964 |
| Sample 3 | -0.222307 | 0.037623 |

Table 5: Results of Stochastic Gradient Descent for $F_2(x) = \sum_{i=1}^{d} \sqrt{|x_i|}$

# Conclusion

This thesis explored the application of the Stochastic Gradient Method (SGM) and Deterministic Gradient Method (DGM) to non-convex optimization problems, with a focus on their generalization bounds, computational complexity, and the impact of smoothness ($\beta$) and expansiveness ($\eta$) properties on performance.

The analysis revealed that the performance of both SGM and DGM is highly influenced by the properties of the objective function, particularly its $\beta$-smoothness and $\eta$-expansiveness. For SGM, the generalization bound complexity for non-convex functions with constant step sizes is $O(2^n)$, highlighting the limitations of fixed step sizes and the need for further research on adaptive step size strategies ($\alpha_t$). Additionally, the time complexity of SGM scales with the number of samples $n$ and the problem's dimensionality $d$, resulting in a complexity of $O(nd)$.

For the DGM, we explored the relationship between $\beta$-smoothness and $\eta$-expansiveness but found that this relationship has not been proven for non-convex functions in general. This suggests that DGM's performance is particularly sensitive to the choice of constant step size $\alpha$.

Simulations for both SGM and DGM validated several theoretical findings, especially concerning stability and optimization performance, and demonstrated the critical role of parameter tuning, such as the step size ($\alpha_t$). The experiments highlighted the complex relationship between smoothness ($\beta$) and expansiveness ($\eta$), providing practical insights into how these properties influence optimization behavior. The results also showed that higher dimensionality exacerbates convergence issues, with larger jumps and greater instability, confirming that dimensionality has a detrimental impact on performance.

In conclusion, this thesis analyzes the performance of SGM and DGM in non-convex optimization, highlighting both their advantages and limitations. While the results provide useful insights, they also point to gaps in the analysis, particularly around generalization bounds and optimization error for non-convex functions. Future work should focus on improving theoretical frameworks for adaptive step size $\alpha_t$ schedules and investigating

their practical impact in large-scale, high-dimensional optimization problems.

# Bibliography

# References


[1] Y. S. Moritz Hardt, Benjamin Recht, "Train faster, generalize better: Stability of stochastic gradient descent," *arXiv preprint arXiv:1509.01240*, 2016, accessed: 2024-10-10. [Online]. Available: https://arxiv.org/abs/1509.01240

[2] M. S. Hamed Karimi, Julie Nutini, "Linear convergence of gradient and proximal-gradient methods under the polyak lojasiewicz condition," *arXiv preprint arXiv:1608.04636*, 2016. [Online]. Available: https://arxiv.org/pdf/1608.04636

[3] R. M. G. Guillaume Garrigos, "Handbook of convergence theorems for (stochastic) gradient methods," *arXiv preprint arXiv:2301.11235*, March 12, 2024. [Online]. Available: https://arxiv.org/pdf/2301.11235

[4] A. Kathuria, *Intro to optimization in deep learning: Gradient Descent*. DigitalOcean, 2024. [Online]. Available: https://www.digitalocean.com/community/tutorials/intro-to-optimization-in-deep-learning-gradient-descent

[5] D. Driggs, M. Ehrhardt, and C.-B. Schönlieb, "Accelerating variance-reduced stochastic gradient methods," *Mathematical Programming*, vol. 191, pp. 671–715, 2022. [Online]. Available: https://doi.org/10.1007/s10107-020-01566-2

[6] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification," *Journal of Machine Learning Research*, vol. 18, no. 223, pp. 1–42, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-595.html

[7] S. Yashima, "Generalization error analysis of stochastic gradient descent on classification problems under low noise condition," Master's Thesis, January 2020, supervisor: Associate Professor Taiji Suzuki.

[8] J. L. Zhize Li, "Simple and optimal stochastic gradient methods for nonsmooth nonconvex optimization," *JMLR: v23/21-0028*, August, 2022. [Online]. Available: https://jmlr.org/papers/v23/21-0028.html

[9] P. R. Ahmed Khaled, "Better theory for sgd in the nonconvex world," *arXiv preprint arXiv:1608.04636*, July 27, 2020. [Online]. Available: https://arxiv.org/pdf/2002.03329


# Appendix

```
import numpy as np
import matplotlib.pyplot as plt

d = 10
```

```
eta = 0.01
iterations = 1001
sigma = 0.1  # Noise standard deviation


def F1(x):
    return np.sum(x ** 2)  # Euclidean norm squared


def grad_F1(x): # O(d)
    return 2 * x


def F2(x):
    return np.sum(np.sqrt(np.abs(x)))


def grad_F2(x): # O(d)
    # Gradient of F2: 0.5 * sign(x) / sqrt(|x|)
    grad = np.zeros_like(x)
    non_zero_indices = x != 0  # division by zero avoided
    grad[non_zero_indices] = 0.5 * np.sign(x[non_zero_indices]) /
        np.sqrt(np.abs(x[non_zero_indices]))
    return grad


# Lipschitz constant and smoothness checks
def check_conditions(func_grad, points=1000):
    u = np.random.uniform(-1, 1, size=(points, d))
    v = np.random.uniform(-1, 1, size=(points, d))

    # Compute the supremum of ||\nabla f(u) - \nabla f(v)|| / ||u
        - v||
    diff_grad = np.linalg.norm(func_grad(u) - func_grad(v), axis
       =1)
    diff_points = np.linalg.norm(u - v, axis=1)
    valid = diff_points > 0
    ratios = diff_grad[valid] / diff_points[valid]
    return np.max(ratios)


def check_convexity(func, points=1000):
    u = np.random.uniform(-1, 1, size=(points, d))
    v = np.random.uniform(-1, 1, size=(points, d))
    theta = np.random.uniform(0, 1, size=(points, 1))  # Random
       theta values in [0, 1]

    # Evaluate convexity condition
    lhs = func(theta * u + (1 - theta) * v)
    rhs = theta.flatten() * func(u) + (1 - theta.flatten()) *
       func(v)
```

```
51
52      # Explicit check for known problematic cases
53      u_explicit = np.array([1])
54      v_explicit = np.array([-1])
55      theta_explicit = 1
56      lhs_explicittt = func(theta_explicit * u_explicit + (1 -
            theta_explicit) * v_explicit)
57      rhs_explicit = theta_explicit * func(u_explicit) + (1 -
            theta_explicit) * func(v_explicit)
58
59      return np.all(lhs <= rhs) and lhs_explicittt <= rhs_explicit
60
61
62  # checks for the functions
63  sup_grad_F1 = check_conditions(grad_F1)
64  sup_grad_F2 = check_conditions(grad_F2)
65
66  is_convex_F1 = check_convexity(F1)
67  is_convex_F2 = check_convexity(F2)
68
69
70  print("Function F1:")
71  print(f"Supremum of ||\nabla f(u) - \nabla f(v)|| / ||u - v||: {
        sup_grad_F1:.6f}")
72  print(f"Function is {'convex' if is_convex_F1 else 'non-convex'}"
        )
73  print("Function is beta-smooth")
74
75  print("\nFunction F2:")
76  print(f"Supremum of ||\nabla f(u) - \nabla f(v)|| / ||u - v||: {
        sup_grad_F2:.6f}")
77  print(f"Function is {'convex' if is_convex_F2 else 'non-convex'}"
        )
78  print("Function is beta-smooth")
79
80  # initialize random starting point
81  x_t = np.random.uniform(-1, 1, size=d)
82
83
84  trajectory_F1 = [F1(x_t)]
85  trajectory_F2 = [F2(x_t)]
86
87  # Log iterations
88  print("\nLogging iterations:")
89
90  for _ in range(iterations):  # O(T)
91
92      noise = np.random.normal(0, sigma, size=d)
93
94      # Gradient step for F1
95      grad_f1 = grad_F1(x_t)  # O(d)
```

```
    stochastic_grad_f1 = grad_f1
    x_t = x_t - eta * stochastic_grad_f1
    trajectory_F1.append(F1(x_t))
    print(f"Iteration {_}: F1 w_t = {x_t[0]:.6f}, Gradient = {
        grad_f1[0]:.6f}, Noise = {noise[0]:.6f}, "
          f"Stochastic Gradient = {stochastic_grad_f1[0]:.6f}")

    grad_f2 = grad_F2(x_t)
    stochastic_grad_f2 = grad_f2
    x_t = x_t - eta * stochastic_grad_f2
    trajectory_F2.append(F2(x_t))
    print(f"Iteration {_}: F2 w_t = {x_t[0]:.6f}, Gradient = {
        grad_f2[0]:.6f}, Noise = {noise[0]:.6f}, "
          f"Stochastic Gradient = {stochastic_grad_f2[0]:.6f}")

plt.figure(figsize=(9, 6))

plt.subplot(1, 2, 1)
plt.plot(trajectory_F1, color='blue')
plt.title("SGM Trajectory for F(x) = sum(x_i^2)")
plt.xlabel("Iteration")
plt.ylabel("F(x)")

plt.subplot(1, 2, 2)
plt.plot(trajectory_F2, color='orange')
plt.title("SGM Trajectory for F(x) = sum(sqrt(|x_i|))")
plt.xlabel("Iteration")
plt.ylabel("F(x)")

plt.tight_layout()
plt.show()
```