

A Hamiltonian Systems Approach to Neural Network Optimization

by

Joshua Joseph George

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2026

© Joshua Joseph George 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We propose and analyze structure-preserving methods for first-order optimization of Lipschitz smooth objectives by interpreting the dynamics as a dissipative Hamiltonian system, in which the model parameters evolve jointly with an auxiliary momentum variable. This formulation induces a natural energy dissipation mechanism that motivates the design of optimization algorithms that inherit a discrete energy decay property. We develop discrete gradient (DG) methods that preserve an exact discrete time energy decay property, ensuring monotone dissipation independent of stepsize. Building on this framework, we introduce variants which empirically reduce oscillations, improve runtime, and improve robustness to ill-conditioned problems.

To address the computational cost of the implicit DG methods, we propose semi-implicit discrete gradient (SIDG) schemes obtained by linearizing the DG updates and incorporating curvature through L-BFGS Hessian approximations, which are used to efficiently solve the resulting linear systems. These schemes retain key structure-preserving properties while significantly reducing computational cost, yielding a practical balance between stability and efficiency. We establish monotone energy decay, boundedness of iterates, and sublinear convergence to first-order stationary points.

Numerical experiments on ill-conditioned least-squares problems, regularized logistic regression, physics-informed neural networks, and CIFAR-10 image classification demonstrate good performance despite ill-conditioning and competitive performance as compared to widely used optimizers such as ADAM, Stochastic gradient descent, and L-BFGS.

Acknowledgements

This thesis would not be possible without the support of many people, some of whom I am able to acknowledge here.

First and foremost, I thank Jesus Christ, through whom all things are possible. I am grateful for his guidance and blessings throughout my entire life.

I would like to express my sincere gratitude to my supervisors, Dr. Kirsten Morris and Dr. David C. Del Rey Fernández, for their consistent support, insightful feedback, and guidance. Their mentorship has been invaluable, and their suggestions have greatly improved both this thesis and my overall research approach. I am grateful to Dr. Jun Liu and Dr. Giang Tran for taking the time to read and comment on this thesis.

I am especially thankful to my family. To my mother, Leena, for her endless love, unwavering support, and for always being there for me through every challenge. I am deeply grateful for the morals and values she instilled in me, which have helped me stay true to myself. To my father, Joseph, for his constant encouragement and support. To my elder brother, Robert, whose advice has always been enlightening and who has been an ideal role model.

To my graduate friends, Anand, Thomas, and Aneel, for all the memorable moments throughout this journey. From research discussions and idea sharing to countless hangouts, calls, and banter, they made graduate school memorable and enjoyable.

To Vanessa, for her unwavering presence and support. Through both the highs and lows, she has always been there to listen, encourage, and push me to grow. I wish her nothing but the very best.

To my undergraduate friends, Vedant, Akrash, Kailash, Gurveer, Coleton, Davidson, Almer, Vedd, and Phong, for their continued friendship and support over the years. Whether it was late-night calls, board game nights, gaming sessions, or trips to Toronto, Edmonton, and Vancouver, they provided a much-needed outlet and kept me grounded. I am grateful for the long discussions, the time we spent taking courses together, and for always listening to my occasional rants about life and research. I always look back on those times fondly.

Finally, to my friends from the tennis club, Alex, Ian, and Jordan, for the great late-night sessions, dinners, and endless memes, and for being my biggest fans. I am still waiting for the day one of you beats me.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
List of Figures	viii
List of Tables	ix
1 Introduction	1
2 Background	6
2.1 Neural Networks	7
2.2 Optimization Methods	8
2.3 Discretizations	10
2.4 Basic Assumptions	11
2.5 Smoothness of common objective functions	13
2.5.1 Least squares	14
2.5.2 Regularized Logistic Regression	15
2.5.3 Local smoothness of neural network objectives	16
2.6 Stationary Points	18

3	Dissipative Hamiltonian Systems	20
3.1	Dissipative Hamiltonian Formulation of Optimization	21
3.1.1	Example	23
3.2	Momentum Gradient Descent (MGD)	24
3.3	Momentum Gradient Descent with δ (MGD- δ)	27
3.4	Discrete Gradient methods	28
3.4.1	Discrete gradient updates (DG)	30
3.4.2	Convergence analysis of the DG method	36
3.4.3	Discrete gradient updates with δ (DG- δ)	43
3.4.4	Convergence analysis of the DG- δ method	46
3.4.5	Limit points of the DG schemes	47
3.5	Semi-Implicit Discrete Gradient Method	49
3.5.1	Convergence analysis of the SIDG method	53
3.5.2	Semi-Implicit Discrete-Gradient- δ updates	61
4	Results	65
4.1	Least Squares	66
4.1.1	Experimental Setup	66
4.1.2	Results	67
4.1.3	Discussion	67
4.2	Regularized Logistic Regression	71
4.2.1	Experimental Setup	71
4.2.2	Results	74
4.2.3	Discussion	74
4.3	PINN Experiment	77
4.3.1	Experimental setup	77
4.3.2	Results	78
4.3.3	Discussion	79

4.4	CIFAR-10 Experiments	82
4.4.1	Experimental Setup	82
4.4.2	Results	82
4.4.3	Discussion	83
5	Conclusion	86
5.1	Conclusion	86
5.2	Future Work	88
	References	90
	Appendices	93
A	Background on Lyapunov stability	94
B	Convergence analysis of Momentum Gradient Descent	96
C	Convergence analysis of DG-δ	101
D	Experiments	104
D.1	Regularized Logistic Regression additional plots	104
D.1.1	Tables for well-conditioned case	104
D.2	CIFAR-10 additional plots	106
E	Additional Material	107
E.1	Inner Products and Norms	107
E.2	Convexity and Smoothness Definitions	108
E.3	Proof of the Descent Lemma	108
E.4	Well-posedness and Fixed-Point Theory	109

List of Figures

4.1	Loss versus iterations for increasing condition numbers. As κ increases, explicit methods slow down significantly, while implicit methods remain more robust to ill-conditioning.	68
4.2	Final loss versus step size for $\kappa = 10^6$. The explicit methods become unstable as h increases, while the implicit methods remain stable over a much larger range of step sizes.	70
4.3	Time to tolerance versus iterations for increasing condition numbers. The δ -implicit variants achieve a significantly better tradeoff between iteration count and runtime, with the advantage becoming more pronounced as the condition number increases.	71
4.4	Illustration of the effect of the margin parameter on label separability for a 2-D example. For $m = 1$, the labels are less separable and lie closer to the decision boundary. For $m = 4$, the labels become more separable. The line $a^\top \theta_\star = 0$ denotes the decision boundary of the classifier.	73
4.5	Comparison of all methods on the PINN inverse problem. Top row: training metrics. Bottom row: $\mathcal{L}_{\text{rel}}$ metrics. The dashed line indicates the target tolerance $\text{rel}L_2 \leq 10^{-1}$. We removed GD from the plots for clarity.	80
4.6	Comparison of all methods on CIFAR-10. Top row: training metrics. Bottom row: validation metrics	85
D.1	Comparison of the best methods on CIFAR-10. Top row: training metrics. Bottom row: validation metrics	106

List of Tables

4.1	Iterations and time (seconds) required to reach the target tolerance for least squares problems of increasing condition number. As the problem becomes more ill-conditioned, MGD slows dramatically and fails at $\kappa = 10^6$, whereas the implicit methods remain robust. The δ -variants, especially DG- δ and SIDG- δ , achieve the lowest iteration counts and fastest times in the hardest regimes.	67
4.2	Regularized logistic regression results for the ill-conditioned, high-margin regime ($m = 4, \kappa = 1000$). All methods achieve similar final losses and accuracies. However, among the structure preserving methods SIDG and SIDG- δ require far fewer iterations than first-order methods and are competitive with L-BFGS, while DG incurs higher runtime due to inner iterations. Furthermore, the SIDG methods achieve strong generalization performance, with test loss and accuracy comparable to Adam. Results are reported as mean $\pm$ standard deviation across random seeds.	74
4.3	Regularized logistic regression results for the ill-conditioned, low-margin regime ($m = 1, \kappa = 1000$), where label noise increases problem difficulty. All methods again reach similar solutions, but SIDG maintains the lowest iteration counts among structure-preserving methods and remains competitive with L-BFGS. Dashes indicate failure to converge. Results are reported as mean $\pm$ standard deviation across random seeds.	75
4.4	PINN performance across seeds (mean $\pm$ standard deviation). L-BFGS achieves the best overall performance in terms of accuracy and convergence speed, though with higher variability. Among the structure-preserving methods, SIDG and SIDG- δ provide the best balance between accuracy and efficiency, requiring significantly fewer iterations than explicit methods while maintaining competitive runtime. DG methods are substantially slower due to expensive implicit updates.	79

4.5	CIFAR-10 generalization performance at the best validation epoch (mean $\pm$ standard deviation across seeds, sorted by test accuracy). DG- δ achieves the strongest overall generalization performance, attaining the highest test accuracy and lowest validation loss. Adam also performs strongly with competitive accuracy and lower runtime, while SIDG methods achieve comparable performance with slightly lower accuracy but better training optimization.	83
4.6	CIFAR-10 training and runtime performance (mean $\pm$ standard deviation across seeds). SIDG methods achieve the lowest training loss, while DG methods incur higher runtime due to more expensive updates. SGD attains the highest training accuracy, but doesn't have competitive generalization metrics.	84
D.1	Regularized logistic regression results for the well-conditioned high-margin case ($m = 4, \kappa = 1$). Results are reported as mean $\pm$ standard deviation across random seeds.	104
D.2	Regularized logistic regression results for the well-conditioned low-margin case ($m = 1, \kappa = 1$). Results are reported as mean $\pm$ standard deviation across random seeds.	105

Chapter 1

Introduction

Training a machine learning model can be formulated as an optimization problem. Given a dataset, the goal is to determine the model parameters $\theta \in \mathbb{R}^n$ that yield the most accurate predictions. We introduce a *loss function* $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, that measures how well the model performs for given parameters θ . The resulting optimization problem is

$$\min_{\theta \in \mathbb{R}^n} \mathcal{L}(\theta). \tag{1.1}$$

Classic examples include least-squares loss in regression and cross-entropy loss in classification. The specific form varies, but the problem is always the same: find parameters that drive the loss down. However, $\mathcal{L}$ is rarely convex (see Definition [E.3](#)) or well-conditioned (see Definition [2.13](#)). Modern loss landscapes are high-dimensional and non-convex, with steep cliffs and flat plateaus that make minimization difficult.

In practice, we rarely solve these problems *exactly*. Hence, rather than solving [\(1.1\)](#) directly, one starts from an initial guess θ_0 and iteratively updates this guess. Each iteration is designed to reduce the loss.

The central question is: *in which direction should the parameters move, and by how much?* One of the ways through which this is answered is the gradient. It points in the direction of the steepest increase, so moving in the negative gradient direction reduces the loss most rapidly at the local level. This gives the classical gradient descent update

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} \mathcal{L}(\theta_k), \tag{1.2}$$

where $h > 0$ is the step size which controls the magnitude of each update. Repeating this update produces a sequence of iterates that drives the parameters toward a local minima. Many successful methods build on these same principles. Some use information from previous steps to

accelerate convergence, others adapt their updates to the local geometry. A classic example is the momentum-type update

$$\theta_{k+1} = \theta_k - h\nabla_{\theta}\mathcal{L}(\theta_k) + \beta(\theta_k - \theta_{k-1}), \quad (1.3)$$

where $\beta \in [0, 1)$ controls how strongly previous motion influences the next step. Momentum reduces oscillations in directions of high curvature while carrying information from previous updates, which can accelerate convergence.

The loss function can be imagined as a landscape; high values form hills, low values form valleys, and training corresponds to moving downhill. As the step size h is taken smaller and smaller, the iterates begin to resemble a smooth trajectory. In the limit, one obtains a *gradient flow*

$$\dot{\theta}(t) = -\nabla_{\theta}\mathcal{L}(\theta(t)), \quad (1.4)$$

which describes the motion of a ball rolling along the loss landscape. In this continuous-time setting, the loss $\mathcal{L}(\theta)$ plays the role of an energy for the system. Differentiating (1.4) yields

$$\frac{d}{dt}\mathcal{L}(\theta(t)) = \langle \nabla_{\theta}\mathcal{L}(\theta(t)), \dot{\theta}(t) \rangle = -\|\nabla_{\theta}\mathcal{L}(\theta(t))\|^2 \leq 0, \quad (1.5)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product and $\|\cdot\|$ the associated norm (see Appendix E.1-E.2). Consequently, the **energy** decreases monotonically over time. This property is referred to as *energy dissipation*: the system continuously loses energy as it evolves toward lower-energy states.

Momentum methods admit a similar interpretation. They resemble the motion of a ball moving across the loss landscape while carrying inertia. This naturally leads to second-order dynamical systems, in which optimization is understood through the interplay of energy and dissipation. In particular, classical momentum methods can be viewed as discretizations of the damped second-order differential equation

$$\ddot{\theta}(t) + \gamma\dot{\theta}(t) + \nabla_{\theta}\mathcal{L}(\theta(t)) = 0, \quad (1.6)$$

where $\gamma > 0$ is a damping parameter that controls the rate of energy dissipation. A convenient formulation of this system is obtained by introducing the velocity variable $\omega := \dot{\theta}$, leading to the first-order system

$$\dot{\theta} = \omega, \quad \dot{\omega} = -\nabla_{\theta}\mathcal{L}(\theta) - \gamma\omega. \quad (1.7)$$

In this setting, one defines the total energy

$$\mathcal{H}(\theta, \omega) := \mathcal{L}(\theta) + \frac{1}{2}\|\omega\|^2, \quad (1.8)$$

where $\mathcal{L}(\theta)$ represents potential energy and $\frac{1}{2}\|\omega\|^2$ kinetic energy. A direct computation shows that along trajectories of this system,

$$\frac{d}{dt}\mathcal{H}(\theta(t), \omega(t)) = -\gamma\|\omega(t)\|^2 \leq 0, \quad (1.9)$$

so the total energy decreases monotonically. This dissipation mechanism drives the system toward equilibrium points corresponding to minimizers of the loss.

While continuous-time dynamics provide useful theoretical insight, optimization algorithms are implemented in discrete time. This is achieved through discretizations, where we replace the continuous trajectory $\theta(t)$ with a sequence of iterates $\{\theta_k\}_{k \geq 0}$, where $k \in \mathbb{N}$ denotes the iteration index and $h > 0$ is a fixed step size that controls the magnitude of each update.

Standard explicit discretizations do not preserve the continuous dissipation law exactly (1.5). When the gradient flow is discretized by explicit Euler, the descent estimate agrees with the continuous energy decay only up to truncation error; for sufficiently *nice* objectives (smooth landscapes, see Definition 2.5) we have

$$\mathcal{L}(\theta_{k+1}) - \mathcal{L}(\theta_k) = -h\|\nabla_{\theta}\mathcal{L}(\theta_k)\|^2 + \mathcal{O}(h^2).$$

The higher-order term is harmless for very small step sizes, but for larger steps it can cause the system to *gain* energy instead of losing it. This may lead to oscillations or even blow-up of the iterates. Consequently, stability is no longer guaranteed by the discretization itself, and must instead be enforced through careful step-size tuning.

This is especially problematic in ill-conditioned or non-convex problems. The challenge is therefore not only to move downhill quickly, but to do so in a way that preserves the underlying dissipative structure of the continuous system.

This motivates a natural question: *can we design optimizers that are stable by construction, by preserving the underlying structure of the physical system?* A class of methods that answer this question is the Discrete Gradient (DG) methods. Rather than using the gradient directly, they replace it with a discrete analog chosen so that the updates satisfy an exact discrete energy decay law at every step, regardless of step size. In this way, the method mirrors the dissipation property of the continuous system at the discrete level.

The price is that such methods are *implicit*. The next iterate is not given by an explicit formula, but must be found by solving a nonlinear equation

$$\theta_{k+1} = \Phi(\theta_{k+1}), \quad (1.10)$$

at every iteration, where Φ is a nonlinear function. This can be done via fixed-point iteration or Newton's method, but in large-scale machine learning problems, where the parameter dimension

is high, and many iterations are required, the cost of these nonlinear solves can be computationally expensive.

To summarize the discussion so far: implicit structure-preserving methods are stable but expensive; explicit methods are cheap but fragile. Now the question becomes: *can we design optimizers that are stable by construction, by preserving the underlying structure of the physical system **while being computationally efficient**?* This thesis is aimed at developing methods that fit this middle ground.

Quasi-Newton methods, such as L-BFGS, are one natural candidate. Rather than forming the true Hessian or its inverse, they build an approximation from the gradient information gathered during previous iterations, capturing useful information without the full cost of a second-order method. To be specific, one adds the curvature to the plain gradient step

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} \mathcal{L}(\theta_k),$$

to obtain

$$\theta_{k+1} = \theta_k - h H_k \nabla_{\theta} \mathcal{L}(\theta_k),$$

where $H_k \approx \nabla_{\theta}^2 \mathcal{L}(\theta_k)^{-1}$ is a cheap approximation to the inverse Hessian, built by storing only the last m gradient difference vectors rather than forming and inverting an $n \times n$ matrix. The search direction $H_k \nabla_{\theta} \mathcal{L}(\theta_k)$ therefore captures curvature information, providing an approximation to Newton-type updates at substantially reduced computational cost.

The main idea developed in this thesis is to combine this approximation with the implicit structure-preserving updates (DG methods). Rather than solving the full implicit problem exactly, we approximate the expensive inner solve using an L-BFGS-type inverse Hessian information. This produces the *semi-implicit discrete gradient* (SIDG) methods studied in this thesis that are designed to preserve dissipative structure more faithfully than standard explicit schemes while avoiding a full nonlinear solve at every iteration. The goal is stability by design and efficiency by construction, i.e., the guarantees of structure-preserving integration without the computational cost associated with it.

Contributions

The main contributions of this thesis are summarized as follows:

- **Structure-preserving optimization methods.** We use discrete gradient methods for optimization problems. Although discrete-gradient methods have been explored in this context, we formulate them within a dissipative Hamiltonian framework, ensuring that the resulting algorithms inherit an exact discrete energy dissipation property for updating model parameters.

- **New damping variants for momentum methods.** We introduce an additional damping parameter δ within the dissipative Hamiltonian formulation, leading to the modified system

$$\dot{\theta} = \omega - \delta \nabla_{\theta} \mathcal{L}(\theta), \quad \dot{\omega} = -\nabla_{\theta} \mathcal{L}(\theta) - \gamma \omega.$$

Discretizations of this system yield new optimization algorithms, including a damped momentum gradient method (MGD- δ) and a structure-preserving discrete gradient method (DG- δ), that improve robustness across a wide range of problems.

- **New Semi-Implicit structure-preserving methods.** We develop semi-implicit discrete-gradient (SIDG) schemes based on linearization of the discrete gradient. These methods retain key stability properties while significantly reducing computational cost and runtime.
- **Convergence and stability analysis.** Under standard smoothness assumptions, we establish convergence guarantees and analyze the stability properties of the newly proposed methods, including results for non-convex objectives.
- **Numerical experiments.** We evaluate the proposed methods on several optimization problems, including ill-conditioned least-squares problems, ℓ_2 -regularized logistic regression, physics-informed neural networks (PINNs), and a classification task on the CIFAR-10 dataset. We compare against widely used optimizers such as Adam, stochastic gradient descent, and L-BFGS.

Organization of the Thesis

The thesis is organized as follows.

Chapter 2 introduces background on machine learning, first-order optimization methods, and the standard assumptions used in nonconvex analysis. Chapter 3 presents the dissipative Hamiltonian formulation of optimization and derives classical momentum methods from continuous-time dynamics. We further develop structure-preserving optimization methods, including the DG and DG- δ schemes, as well as the SIDG and SIDG- δ methods. This chapter also establishes their well-posedness and provides theoretical analysis, including discrete energy identities, convergence guarantees, and gradient convergence rates. Chapter 4 presents numerical experiments on a range of optimization and machine learning problems like ill-conditioned least squares, regularized logistic regression, physics-informed neural networks, and image classification on CIFAR-10. The proposed methods are compared against standard optimizers such as stochastic gradient descent, Adam, and L-BFGS. The appendices provide additional theoretical background, detailed proofs, and supplementary experimental results.

Overall, this thesis introduces structure-preserving optimization methods within a dissipative Hamiltonian framework, along with a new semi-implicit scheme that improves practical performance.

Chapter 2

Background

This chapter introduces the background needed for the development and analysis of the proposed optimization methods. Since the applications considered in this thesis arise from supervised machine learning, we begin by recalling how training can be formulated as the minimization of an empirical loss function. We then review neural networks, introduce classical optimization methods, and state the main assumptions used later in the convergence analysis. Finally, we discuss smoothness properties of several common objective functions and recall basic notions concerning stationary points in nonconvex optimization.

Many machine learning methods can be formulated as optimization problems involving the minimization of a differentiable loss function (1.1). In supervised learning, one is given a dataset of input-output pairs $\{(x_i, y_i)\}_{i=1}^N$, where N denotes the number of data points, and aims to learn a mapping from inputs to outputs. This setting constitutes the class of problems considered in this thesis, where the objective function arises from empirical risk minimization and is given by

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{i=1}^N l(f(x_i; \theta), y_i), \quad (2.1)$$

where $f(x_i; \theta)$ denotes the prediction with parameters θ and $l(\cdot, \cdot)$ is a loss function measuring the prediction error for a single data sample. Training a model, therefore, amounts to solving the optimization problem

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\theta). \quad (2.2)$$

The resulting optimization problem is typically non-convex over the parameter space. The losses depend on the task involved and can be either the standard mean squared error (MSE) for regression or the cross-entropy loss for classification LeCun et al. [12].

2.1 Neural Networks

Neural networks are a widely used approach for constructing machine learning models of the form $f(x; \theta)$, where θ denotes the trainable parameters. They have been successfully applied to a broad range of tasks, including regression and classification LeCun et al. [12]. A neural network defines a mapping

$$f(x; \theta) : \mathbb{R}^n \rightarrow \mathbb{R}^m,$$

where $x \in \mathbb{R}^n$ denotes the input data and θ represents the trainable parameters of the model. Feedforward neural networks are typically constructed as compositions of layers,

$$f(x; \theta) = f_L \circ f_{L-1} \circ \cdots \circ f_1(x),$$

where each layer takes the form

$$f_\ell(z) := \sigma_\ell(W_\ell z + b_\ell).$$

Here, W_ℓ and b_ℓ denote the weight matrix and bias vector, while σ_ℓ is a nonlinear activation function such as ReLU, sigmoid, or hyperbolic tangent. The parameters of the network are given by

$$\theta = \{W_\ell, b_\ell\}_{\ell=1}^L.$$

Training a neural network involves minimizing the empirical loss introduced earlier (2.1).

Neural network training is typically performed using gradient-based optimization methods. In this setting, backpropagation is used to efficiently compute the gradient of the loss function with respect to the model parameters via the chain rule. These gradients are then used within an optimization algorithm, such as gradient descent or its variants, to iteratively update the parameters. The learning process can be viewed as constituting two computational phases:

- **Forward pass:** compute predictions $\hat{y} = f(x; \theta)$.
- **Backward pass:** compute gradients of the loss $\nabla_\theta \mathcal{L}(\theta)$ and updating the parameters using an optimization algorithm.

Because neural networks involve compositions of nonlinear functions and often contain very high-dimensional parameter spaces, the resulting optimization problems are typically nonconvex and large-scale. Consequently, the development of efficient and *stable* optimization algorithms plays a central role in modern machine learning. Training such models requires iterative optimization algorithms, which motivates the review of classical first-order methods in the next section. The notion of stability will be made precise later in Section 2.4.

2.2 Optimization Methods

The classical gradient descent (GD) method updates parameters according to

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} \mathcal{L}(\theta_k), \quad (2.3)$$

where $h > 0$ denotes the step size, which in the machine learning community is referred to as the *learning rate*.

The GD method can be traced back to the work of Cauchy et al. [2]. Its convergence properties for smooth objectives are well understood Nesterov [16]. Stochastic variants Robbins and Monro [22] further enable scalability to high-dimensional problems and large datasets and form the foundation of modern machine learning optimization methods Bottou et al. [1].

In large-scale learning problems, computing gradients over the entire dataset may be computationally expensive. Stochastic gradient descent (SGD) addresses this issue by evaluating the gradient using a randomly selected sample $i_k \in \{1, \dots, N\}$,

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} l(f(x_{i_k}; \theta_k), y_{i_k}). \quad (2.4)$$

In practice, stochastic gradients are often computed using a subset (mini-batch) of the data rather than a single sample. Let $B_k = \{i_1, \dots, i_m\} \subset \{1, \dots, N\}$ denote a mini-batch of $m = |B_k|$ randomly selected indices. The mini-batch stochastic gradient descent update is given by

$$\theta_{k+1} = \theta_k - \frac{h}{|B_k|} \sum_{i \in B_k} \nabla_{\theta} l(f(x_i; \theta_k), y_i). \quad (2.5)$$

By averaging over a small subset of the data, mini-batch methods yield more accurate gradient estimates than those obtained from a single sample, while remaining far cheaper per iteration than full-batch gradient descent Bottou et al. [1].

Momentum methods further accelerate gradient-based optimization by incorporating information from previous iterates. A classic example is the heavy-ball method introduced by Polyak [19],

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} \mathcal{L}(\theta_k) + \beta(\theta_k - \theta_{k-1}), \quad (2.6)$$

where $\beta \in [0, 1)$ is the momentum parameter. This update admits an interpretation from a dynamical systems perspective. As previously discussed, it can be viewed as a time discretization of the damped second-order differential equation

$$\ddot{\theta}(t) + \gamma \dot{\theta}(t) + \nabla_{\theta} \mathcal{L}(\theta(t)) = 0, \quad (2.7)$$

where γ is some function of β . Heavy-ball methods can accelerate convergence for ill-conditioned problems. For quadratic functions that are μ -strongly convex (E.4) and its gradient is L -Lipchitz (see Definition (E.5)), where $\mu, L > 0$, the optimal parameters are

$$h = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \quad \beta = \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^2,$$

where $\kappa = L/\mu$ denotes the condition number Lessard et al. [14]. For quadratic problems, these parameters yield the optimal worst-case convergence rate of the heavy-ball method.

While first-order optimization methods are widely adopted for their low computational overhead, they often suffer from slow convergence in ill-conditioned settings because they do not explicitly account for the curvature of the objective function. Accounting for curvature is important because it captures how the gradient changes across directions, allowing the method to appropriately scale updates and avoid overly small steps in flat directions and overly large steps in steep directions. Second-order methods address this limitation by incorporating Hessian information. A classic example is Newton's method, which scales the gradient by the inverse Hessian to adapt the update direction to the local geometry and is given by

$$\theta_{k+1} = \theta_k - \nabla_{\theta}^2 \mathcal{L}(\theta_k)^{-1} \nabla_{\theta} \mathcal{L}(\theta_k). \quad (2.8)$$

By accounting for local curvature, Newton's method enables rapid, often quadratic convergence near a minimizer Nocedal and Wright [17]. However, the computational cost of computing and inverting the Hessian matrix makes it impractical for high-dimensional problems.

To bridge the gap between first-order efficiency and second-order speed, quasi-Newton methods construct approximations of the inverse Hessian using only gradient information. In particular, the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm Nocedal and Wright [17] maintains a sequence of matrices $H_k \approx \nabla_{\theta}^2 \mathcal{L}(\theta_k)^{-1}$ and updates the parameters via:

$$\theta_{k+1} = \theta_k - h_k H_k \nabla_{\theta} \mathcal{L}(\theta_k), \quad (2.9)$$

where h_k is the step-size typically chosen via a line search. Rather than forming H_k explicitly, L-BFGS builds this approximation using a limited history of past updates. Specifically, it stores pairs

$$s_k = \theta_{k+1} - \theta_k, \quad y_k = \nabla_{\theta} \mathcal{L}(\theta_{k+1}) - \nabla_{\theta} \mathcal{L}(\theta_k),$$

which capture curvature information along recent trajectories. These pairs are utilized in an efficient recursive procedure to compute the product $H_k \nabla_{\theta} \mathcal{L}(\theta_k)$ without the need for large-scale matrix storage. By approximating curvature in this way, quasi-Newton methods significantly improve convergence on ill-conditioned problems while retaining a computational cost comparable to first-order methods. This concept forms the foundation for the semi-implicit methods developed in this work.

Building on the dynamical interpretation introduced above, optimization algorithms can be viewed more generally as discretizations of continuous-time dynamical systems. This perspective is particularly natural for momentum methods, which admit second-order differential equation models involving gradient forces and damping, helping to explain phenomena such as acceleration, oscillation, and the role of damping parameters.

A notable modern connection between optimization and dynamical systems is the work of Su et al. [23], who derived a second-order ordinary differential equation as the small-step-size limit of Nesterov’s accelerated gradient method, showing that this continuous-time perspective clarifies both accelerated convergence and the oscillatory behavior often observed in practice.

A central analytical tool in this framework is the use of *Lyapunov functions* (see Appendix A), namely energy-like quantities that decrease along trajectories of the dynamics. Such functions provide a systematic way to establish boundedness, monotonicity, and convergence rates in both continuous and discrete time. In the context of accelerated and momentum methods, Wilson et al. [26] showed that Lyapunov functions provide a unifying interpretation of estimate sequences and clarify the relationship between continuous-time models and discrete-time algorithms.

For optimization problems arising in machine learning, Hamiltonian and port-Hamiltonian formulations provide an additional structure-preserving viewpoint with inherent energy dissipation. In particular, Massaroli et al. [15] proposed a port-Hamiltonian approach to neural network training, showing how dissipation and energy-based modeling can be incorporated into optimization dynamics. This perspective is relevant to the present thesis, in which the proposed methods are derived from dissipative Hamiltonian systems and analyzed using discrete energy dissipation laws.

2.3 Discretizations

We consider different discretizations of continuous-time dynamics, including explicit, implicit, and semi-implicit methods. Consider the gradient flow

$$\dot{\theta}(t) = -\nabla_{\theta}\mathcal{L}(\theta(t)).$$

Definition 2.1 (Explicit methods). A method is called explicit if the next iterate is computed solely using information from the current iterate. For example, the explicit Euler discretization of the gradient flow is given by

$$\theta_{k+1} = \theta_k - h\nabla_{\theta}\mathcal{L}(\theta_k),$$

which corresponds to the classical GD method.

Definition 2.2 (Implicit methods). A method is called implicit if the next iterate is defined through an equation involving the unknown future state. For example, the implicit Euler discretization is given by

$$\theta_{k+1} = \theta_k - h \nabla_{\theta} \mathcal{L}(\theta_{k+1}),$$

which requires solving a nonlinear equation at each iteration.

Definition 2.3 (Semi-implicit methods). A method is called semi-implicit if it combines explicit and implicit updates, treating some terms using current information and others using future information.

Explicit methods are typically computationally inexpensive but may require restrictive step sizes for stability. In contrast, implicit methods are often more stable (see Definition 2.9) and can admit larger step sizes, but require solving nonlinear equations at each iteration. Semi-implicit methods provide a middle ground, aiming to retain stability (see Definition 2.9) properties while reducing computational cost.

2.4 Basic Assumptions

Throughout this thesis, we consider optimization problems satisfying several standard assumptions commonly used in nonconvex optimization. Unless otherwise stated, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$.

Before defining global L -smoothness, we recall the notion of global Lipschitz continuity.

Definition 2.4 (Global Lipschitz continuity). A function $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$ is called globally L -Lipschitz continuous if there exists a constant $L > 0$ such that

$$\|\mathcal{L}(\theta) - \mathcal{L}(\tilde{\theta})\| \leq L \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in \mathbb{R}^n.$$

Assumption 2.5 (Global L -smoothness). A continuously differentiable function $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be globally L -smooth, with $L > 0$, if

$$\|\nabla_{\theta} \mathcal{L}(\theta) - \nabla_{\theta} \mathcal{L}(\tilde{\theta})\| \leq L \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in \mathbb{R}^n.$$

Equivalently, $\nabla_{\theta} \mathcal{L}$ is globally Lipschitz continuous with constant L .

For twice continuously differentiable functions, a sufficient condition for global L -smoothness is that

$$\|\nabla_{\theta}^2 \mathcal{L}(\theta)\| \leq L, \quad \forall \theta \in \mathbb{R}^n.$$

Assumption 2.6 (*Compact sub-level sets*). For every $c \in \mathbb{R}$ the sub-level set

$$\{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \leq c\}$$

is compact. A sufficient condition for this property is that $\mathcal{L}$ is coercive, that is,

$$\lim_{\|\theta\| \rightarrow \infty} \mathcal{L}(\theta) = \infty.$$

Assumption 2.7 (*Existence of stationary points*). The set of critical points

$$\{\theta \in \mathbb{R}^n : \nabla_{\theta} \mathcal{L}(\theta) = 0\}$$

is nonempty.

Under Assumptions 2.5-2.6, the function $\mathcal{L}$ is bounded below. Indeed, since sub-level sets are compact and nonempty, and $\mathcal{L}$ is continuous, it attains its minimum on each sub-level set. In particular, $\inf_{\theta} \mathcal{L}(\theta) > -\infty$.

These assumptions are standard in the analysis of first-order optimization algorithms and ensure that the optimization problem is well-posed. In particular, smoothness enables the use of descent inequalities, while compact sub-level sets guarantee boundedness of the iterates generated by gradient-based methods. Without such a structure, optimization problems may fail to admit solutions. For example, the function $f(x) = x$ has no minimum on $\mathbb{R}$.

Lemma 2.8 (Descent lemma). *If $\mathcal{L}$ is L -smooth, then for all $\theta, \tilde{\theta} \in \mathbb{R}^n$*

$$\mathcal{L}(\tilde{\theta}) \leq \mathcal{L}(\theta) + \nabla_{\theta} \mathcal{L}(\theta)^{\top} (\tilde{\theta} - \theta) + \frac{L}{2} \|\tilde{\theta} - \theta\|^2. \quad (2.10)$$

The descent lemma Nesterov [16] is a standard consequence of L -smoothness. For completeness, a proof is provided in Appendix E.3. This is a fundamental tool in the analysis of gradient-based optimization methods. In particular, it enables the derivation of descent guarantees and convergence rates under suitable step-sizes for the above first-order methods (2.3), (2.4), (2.5), and (2.6).

Furthermore, we introduce several qualitative notions, namely, *stability*, *robustness*, *overfitting*, *condition number*, and *ill-conditioning* which describe the practical behavior of optimization methods. These notions are defined below in the context of this work.

Definition 2.9 (Stability). An optimization method is said to be stable if the sequence of iterates $\{\theta_k\}_{k \in \mathbb{N}}$ remains bounded and does not exhibit divergent or highly oscillatory behavior under appropriate step-size choices.

Definition 2.10 (Robustness). An optimization method is said to be robust if its convergence behavior is relatively insensitive to problem conditioning, noise, or slight variations in hyperparameters such as the step size.

Definition 2.11 (Overfitting). In supervised learning, overfitting refers to the phenomenon where a model fits the training data well but performs poorly on unseen data.

Definition 2.12 (Condition number). Let $\mathcal{L}$ be a function that is μ -strongly convex and L -smooth with $\mu, L > 0$. The condition number of $\mathcal{L}$ is defined as

$$\kappa := \frac{L}{\mu}.$$

In the strongly convex quadratic case, where

$$\mathcal{L}(\theta) = \frac{1}{2}\theta^\top Q\theta + b^\top \theta + c,$$

with $Q \in \mathbb{R}^{n \times n}$ symmetric positive definite, this coincides with

$$\kappa = \frac{\lambda_{\max}(Q)}{\lambda_{\min}(Q)},$$

where $L = \lambda_{\max}(Q)$ and $\mu = \lambda_{\min}(Q)$ denote the largest and smallest eigenvalues of Q , respectively.

Definition 2.13 (Conditioning of an Optimization Problem). A well-conditioned optimization problem is one whose condition number κ is close to 1. In this case, small perturbations in the input lead to small changes in the solution, reflecting similar curvature across directions.

An optimization problem is said to be *ill-conditioned* if its condition number κ is large. In such cases, the objective function exhibits significant variation in curvature across directions, which leads to slow convergence of first-order methods and increased sensitivity to the step size (see Remark 2.15). In particular, small perturbations in the input can result in large changes in the solution.

2.5 Smoothness of common objective functions

To validate the theoretical results developed in this work, it is necessary to verify the smoothness assumption for the objective functions encountered in practice. We therefore consider commonly used loss functions in optimization and machine learning and explicitly characterize their smoothness constants.

2.5.1 Least squares

We begin with the least squares objective

$$\mathcal{L}(\theta) := \frac{1}{2N} \|A\theta - b\|^2, \quad A \in \mathbb{R}^{N \times N}, \quad b \in \mathbb{R}^N. \quad (2.11)$$

This problem plays a fundamental role in optimization, providing a simple yet representative setting in which the behavior of optimization methods can be precisely analyzed. It allows explicit characterization of smoothness, conditioning, and step-size restrictions in terms of the spectrum of the data matrix.

Expanding (2.11), the objective can be written in the quadratic form

$$\mathcal{L}(\theta) = \frac{1}{2} \theta^\top Q \theta - c^\top \theta + \frac{1}{2N} \|b\|^2, \quad Q := \frac{1}{N} A^\top A, \quad c := \frac{1}{N} A^\top b.$$

Since Q is symmetric positive semidefinite, the geometry of the problem is governed by its spectrum. In particular, the largest eigenvalue of Q determines the smoothness of the objective, as shown below.

Proposition 2.14. *The function $\mathcal{L}$ in (2.11) is globally L -smooth with*

$$L = \frac{1}{N} \|A\|^2 = \lambda_{\max}(Q).$$

Proof. A direct computation shows that

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N} A^\top (A\theta - b).$$

Hence, for any $\theta, \tilde{\theta} \in \mathbb{R}^n$,

$$\nabla_\theta \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}(\tilde{\theta}) = \frac{1}{N} A^\top A(\theta - \tilde{\theta}).$$

Taking norms and using $\|A^\top A\| = \|A\|^2$, we obtain

$$\|\nabla_\theta \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}(\tilde{\theta})\| \leq \frac{1}{N} \|A\|^2 \|\theta - \tilde{\theta}\|.$$

Since $Q = \frac{1}{N} A^\top A$ is symmetric positive semi-definite, its operator norm satisfies $\|Q\| = \lambda_{\max}(Q)$, yielding the result. $\square$

Remark 2.15 (Condition number and step-size). This characterization highlights that the smoothness constant is determined by the largest eigenvalue of Q . Consequently, for explicit methods applied to (2.11), stability imposes a step-size restriction of the form

$$h \lesssim \frac{1}{\lambda_{\max}(Q)},$$

which becomes increasingly restrictive as the condition number grows. This dependence on the spectrum of Q is a key factor governing the behavior of optimization methods in ill-conditioned settings.

2.5.2 Regularized Logistic Regression

We consider another fundamental problem in machine learning: logistic regression, which is widely used for binary classification. Given data points $a_i \in \mathbb{R}^n$ with labels $y_i \in \{-1, 1\}$, the goal is to learn a parameter vector θ such that the sign of $a_i^\top \theta$ predicts the label y_i . The logistic loss defined below penalizes misclassified points.

To improve stability and prevent overfitting, one often adds an ℓ_2 regularization term LeCun et al. [12], which penalizes large parameter values and promotes well-conditioned optimization problems. The loss function is defined as

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{i=1}^N \log(1 + e^{-y_i a_i^\top \theta}) + \frac{\lambda}{2} \|\theta\|^2, \quad (2.12)$$

where $\lambda \geq 0$ is the regularization parameter. Let $A \in \mathbb{R}^{N \times n}$ be the data matrix whose i -th row is $a_i^\top$.

Proposition 2.16. $\mathcal{L}$ in (2.12) is globally L -smooth with

$$L = \frac{\|A\|^2}{4N} + \lambda.$$

Proof. Let $\phi(t) = \log(1 + e^{-t})$. Then $\phi''(t) = \frac{e^t}{(1+e^t)^2}$, which is maximised at $t = 0$ with value $1/4$, so $0 \leq \phi''(t) \leq \frac{1}{4}$ for all t . Differentiating (2.12) twice yields,

$$\nabla_\theta^2 \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \phi''(y_i a_i^\top \theta) a_i a_i^\top + \lambda I \preceq \frac{1}{4N} A^\top A + \lambda I.$$

Taking the norm and using $\|A^\top A\| = \|A\|^2$ results in,

$$\|\nabla_\theta^2 \mathcal{L}(\theta)\| \leq \frac{\|A\|^2}{4N} + \lambda, \quad \forall \theta,$$

which proves the claim. □

2.5.3 Local smoothness of neural network objectives

We now consider objective functions arising in neural networks. In this setting, the empirical loss is

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N l(f(x_i; \theta), y_i),$$

where $f(x; \theta)$ is a neural network and $l(\cdot, \cdot)$ is a loss function.

In contrast to quadratic least squares and regularized logistic regression, such objectives are generally not globally L -smooth over the entire parameter space. This is because the derivatives of neural networks depend on the magnitude of the parameters and involve compositions across layers, which can grow rapidly with depth and are not uniformly bounded. As a result, global Lipschitz constants are typically not available Herrera et al. [9].

However, under mild regularity assumptions, one can establish *local smoothness* on bounded regions of the parameter space, which is sufficient for the analysis of first-order methods.

Definition 2.17 (Local L -smoothness). A function $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be locally L -smooth on a set $K \subset \mathbb{R}^n$ if there exists a constant $L_K > 0$ such that

$$\|\nabla_{\theta} \mathcal{L}(\theta) - \nabla_{\theta} \mathcal{L}(\tilde{\theta})\| \leq L_K \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in K.$$

Proposition 2.18 (Local smoothness on compact sets). Let $\mathcal{L} \in C^2(\mathbb{R}^n)$, and let

$$S_c := \{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \leq c\}$$

be a compact sub-level set. Then there exists a constant $L_c > 0$ such that

$$\|\nabla_{\theta} \mathcal{L}(\theta) - \nabla_{\theta} \mathcal{L}(\tilde{\theta})\| \leq L_c \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in S_c.$$

That is, $\mathcal{L}$ is locally L -smooth on S_c .

Proof. Since $\mathcal{L} \in C^2(\mathbb{R}^n)$, the Hessian $\nabla_{\theta}^2 \mathcal{L}$ is continuous. As S_c is compact, $\nabla_{\theta}^2 \mathcal{L}$ is bounded on S_c , so there exists $M > 0$ such that

$$\|\nabla_{\theta}^2 \mathcal{L}(\theta)\| \leq M, \quad \forall \theta \in S_c.$$

Applying the mean value theorem to $\nabla_{\theta} \mathcal{L}$ then gives

$$\|\nabla_{\theta} \mathcal{L}(\theta) - \nabla_{\theta} \mathcal{L}(\tilde{\theta})\| \leq M \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in S_c.$$

Setting $L_c := M$ completes the proof. □

We now show that neural network objectives satisfy the above conditions under mild assumptions. Since the dataset $\{x_i\}_{i=1}^N$ is finite, the inputs are bounded, i.e., there exists a constant $c > 0$ such that $\|x_i\| \leq c$ for all i .

Proposition 2.19 (*Local smoothness of neural network loss*). *Consider the empirical risk*

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N l(f(x_i; \theta), y_i),$$

where $f(x; \theta)$ is a feedforward neural network. Assume that:

- each activation function $\sigma_\ell \in C^2(\mathbb{R})$,
- the loss function $l(\cdot, \cdot) \in C^2(\mathbb{R})$.

Then $\mathcal{L} \in C^2(\mathbb{R}^n)$. In particular, for every compact set $K \subset \mathbb{R}^n$, there exists a constant $L_K > 0$ such that

$$\|\nabla_\theta \mathcal{L}(\theta) - \nabla_\theta \mathcal{L}(\tilde{\theta})\| \leq L_K \|\theta - \tilde{\theta}\|, \quad \forall \theta, \tilde{\theta} \in K.$$

For a detailed derivation of these bounds and explicit constants for multi-layer feedforward networks, see Herrera et al. [9].

Proof. Each layer of the network has the form

$$z \mapsto \sigma_\ell(W_\ell z + b_\ell).$$

If $\sigma_\ell \in C^2$, then this map is C^2 with respect to its inputs and parameters. Since a finite composition of C^2 maps is again C^2 , it follows that $\theta \mapsto f(x_i; \theta)$ is C^2 for each fixed input x_i . Now because $l(\cdot, \cdot)$ is also C^2 , the mapping

$$\theta \mapsto l(f(x_i; \theta), y_i)$$

is C^2 . Taking a finite sum over i shows that $\mathcal{L} \in C^2(\mathbb{R}^n)$.

The result then follows from Proposition 2.18. □

Typical examples of smooth activation functions include sigmoid, tanh, sinh, softplus, and GELU. For such choices, the above assumptions are satisfied. Popular examples of non-smooth activations are ReLU.

2.6 Stationary Points

In nonconvex optimization, convergence guarantees for first-order methods are typically expressed in terms of stationary points rather than global minimizers. We therefore introduce several standard definitions used throughout the analysis of nonconvex optimization problems.

Definition 2.20 (Stationary point). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function. A point $x^* \in \mathbb{R}^n$ is called a *stationary point* if

$$\nabla f(x^*) = 0.$$

Definition 2.21 (Local minimum). A point $x^* \in \mathbb{R}^n$ is a *local minimum* of f if there exists $r > 0$ such that

$$f(x^*) \leq f(x), \quad \forall x \in B_r(x^*),$$

where $B_r(x^*)$ denotes the open ball of radius r centered at x^* .

Definition 2.22 (Strict local minimum). A point $x^* \in \mathbb{R}^n$ is called a *strict local minimum* if there exists $r > 0$ such that

$$f(x^*) < f(x), \quad \forall x \in B_r(x^*) \setminus \{x^*\}.$$

Definition 2.23 (Saddle point). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function. A point $x^* \in \mathbb{R}^n$ is called a *saddle point* if it is a stationary point,

$$\nabla f(x^*) = 0,$$

but x^* is neither a local minimum nor a local maximum.

Definition 2.24 (Strict saddle point). A stationary point x^* of a twice differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called a *strict saddle point* if the Hessian matrix $\nabla^2 f(x^*)$ has at least one negative eigenvalue, i.e.,

$$\lambda_{\min}(\nabla^2 f(x^*)) < 0.$$

In nonconvex optimization landscapes, stationary points may correspond to local minima, local maxima, or saddle points. Gradient-based methods may converge to saddle points rather than minimizers. Nevertheless, recent theoretical results provide strong guarantees regarding the typical behavior of gradient descent.

For smooth nonconvex objectives satisfying the *strict saddle* property, gradient descent with random initialization almost surely avoids strict saddle points and converges to a local minimizer. This result was established by Lee et al. [13] using tools from dynamical systems and stable manifold theory.

Although saddle points are unstable for gradient descent dynamics, escaping their neighborhoods can still require significant time. In the worst case, gradient descent may require exponential time to escape certain saddle regions Du et al. [6]. Several mechanisms have been proposed to accelerate escape from saddle points. For example, adding random perturbations leads to algorithms such as perturbed gradient descent, which provably escapes saddle points in polynomial time Jin et al. [10]. Stochastic gradient methods also introduce inherent noise that helps push iterates along directions of negative curvature Daneshmand et al. [5]. Momentum-based gradient methods introduce inertial dynamics that amplify directions of negative curvature, which can facilitate escape from saddle points. For example, accelerated gradient methods have been shown to escape saddle points faster than gradient descent in certain nonconvex settings Jin et al. [11].

Remark 2.25. While this thesis does not focus on the analysis of saddle point escape times, the above results provide important context for the behavior of the optimization methods we discussed. In particular, under suitable step-size conditions and the strict saddle property, gradient-based algorithms converge almost surely to local minimizers. Moreover, variations such as stochastic gradient descent and momentum methods can influence the efficiency with which saddle regions are escaped, thereby affecting practical convergence behavior.

Chapter 3

Dissipative Hamiltonian Systems

The goal of this chapter is to formalize the dissipative Hamiltonian formulation of optimization. This framework provides for a principled way to model optimization dynamics as energy-based systems with dissipation, and serves as the foundation for the approaches developed in subsequent chapters. In classical mechanics, the state of a system is described by its position and momentum variables, while the *Hamiltonian* is a scalar function representing the total energy of the system.

Let $\theta \in \mathbb{R}^n$ denote the parameter vector and let $\omega \in \mathbb{R}^n$ denote an auxiliary momentum variable. In simple mechanical systems, one often has $\omega := \dot{\theta}$. We define the Hamiltonian as

$$\mathcal{H}(\theta, \omega) := \mathcal{L}(\theta) + \frac{1}{2}\|\omega\|^2, \quad (3.1)$$

where $\mathcal{L}(\theta)$ represents the potential energy and $\frac{1}{2}\|\omega\|^2$ is the kinetic energy. The evolution of the system is governed by Hamilton's equations,

$$\dot{\theta} = \nabla_{\omega} \mathcal{H}(\theta, \omega), \quad (3.2)$$

$$\dot{\omega} = -\nabla_{\theta} \mathcal{H}(\theta, \omega). \quad (3.3)$$

These dynamics conserve total energy. Indeed,

$$\frac{d}{dt} \mathcal{H} = \nabla_{\theta} \mathcal{H}^{\top} \dot{\theta} + \nabla_{\omega} \mathcal{H}^{\top} \dot{\omega} = \nabla_{\theta} \mathcal{H}^{\top} \nabla_{\omega} \mathcal{H} - \nabla_{\omega} \mathcal{H}^{\top} \nabla_{\theta} \mathcal{H} = 0. \quad (3.4)$$

This conservation property is fundamental in classical mechanics, but it also makes pure Hamiltonian dynamics unsuitable for optimization as the total energy is conserved. To remedy this, we introduce dissipation by adding a damping term to the momentum equation,

$$\dot{\theta} = \nabla_{\omega} \mathcal{H}(\theta, \omega), \quad (3.5)$$

$$\dot{\omega} = -\nabla_{\theta} \mathcal{H}(\theta, \omega) - \gamma \nabla_{\omega} \mathcal{H}(\theta, \omega), \quad (3.6)$$

which corresponds to a particle moving in the potential landscape $\mathcal{L}(\theta)$ subject to friction or damping $\gamma > 0$. The energy now satisfies

$$\frac{d}{dt}\mathcal{H}(\theta(t), \omega(t)) = -\gamma\|\omega\|^2 \leq 0, \quad (3.7)$$

driving the system toward lower-energy states. In particular, suitable dissipation ensures that the energy decreases monotonically along trajectories, and under standard assumptions (e.g., compact sub-level sets), the dynamics converge to equilibria corresponding to stationary points of the loss function. This is the core mechanism behind momentum-based optimization methods such as the heavy-ball method and Nesterov acceleration.

3.1 Dissipative Hamiltonian Formulation of Optimization

Following Massaroli et al. [15], we interpret the parameter dynamics of an optimization problem as a dissipative Hamiltonian system. In this setting, we consider the Hamiltonian (3.1) which corresponds to the total energy of the system. Let $\theta \in \mathbb{R}^n$ denote the model parameters, $\mathcal{L}(\theta)$ the loss function, and $\omega \in \mathbb{R}^n$ an auxiliary momentum variable.

Define the state $\xi := (\theta, \omega) \in \mathbb{R}^{2n}$. The dynamics are constructed from the skew-symmetric interconnection matrix J and dissipation matrix R ,

$$J := \begin{bmatrix} O_n & I_n \\ -I_n & O_n \end{bmatrix}, \quad R := \begin{bmatrix} O_n & O_n \\ O_n & R_{22} \end{bmatrix}, \quad R_{22} = R_{22}^\top \succeq 0, \quad (3.8)$$

where $O_n, I_n \in \mathbb{R}^{n \times n}$ are the zero and identity matrices, respectively. This yields the autonomous dissipative Hamiltonian system

$$\dot{\xi} = (J - R)\nabla\mathcal{H}(\xi). \quad (3.9)$$

The skew-symmetric matrix J encodes the conservative Hamiltonian structure, while $R \succeq 0$ introduces dissipation. The dynamics are,

$$\dot{\theta} = \omega, \quad (3.10)$$

$$\dot{\omega} = -\nabla_{\theta}\mathcal{L}(\theta) - R_{22}\omega. \quad (3.11)$$

More generally, the dissipation matrix in a dissipative Hamiltonian system may take the block form

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \quad R = R^\top \succeq 0.$$

Different choices of these blocks correspond to different damping mechanisms in the dynamics. In this work, we consider two particular structures. The first corresponds to the standard choice

$$R = \begin{bmatrix} O_n & O_n \\ O_n & R_{22} \end{bmatrix},$$

which introduces dissipation only in the momentum equation. The second allows additional damping in the parameter dynamics,

$$R = \begin{bmatrix} R_{11} & O_n \\ O_n & R_{22} \end{bmatrix}.$$

The effect of this additional dissipation term will be explored in later sections.

A positive definite function whose value is non-increasing along trajectories is commonly referred to as a *Lyapunov function* in dynamical systems. For more background on Lyapunov functions and stability, refer to [A](#).

Proposition 3.1. *The Hamiltonian $\mathcal{H}$ is non-increasing along trajectories of (3.9), and hence serves as a Lyapunov function for the system.*

Proof. Taking the derivative along trajectories gives

$$\dot{\mathcal{H}} = \nabla_{\theta} \mathcal{H}^{\top} \dot{\theta} + \nabla_{\omega} \mathcal{H}^{\top} \dot{\omega}.$$

Substituting the system equations yields

$$\dot{\mathcal{H}} = \nabla_{\theta} \mathcal{H}^{\top} \omega + \nabla_{\omega} \mathcal{H}^{\top} (-\nabla_{\theta} \mathcal{H} - R_{22} \nabla_{\omega} \mathcal{H}).$$

Since $\nabla_{\omega} \mathcal{H} = \omega$, the first two terms cancel, giving

$$\dot{\mathcal{H}} = -\nabla_{\omega} \mathcal{H}^{\top} R_{22} \nabla_{\omega} \mathcal{H}.$$

Since $R_{22} \succeq 0$, it follows that $\dot{\mathcal{H}} \leq 0$. □

The identity

$$\dot{\mathcal{H}}(\theta, \omega) = -\nabla_{\omega} \mathcal{H}(\theta, \omega)^{\top} R_{22} \nabla_{\omega} \mathcal{H}(\theta, \omega) \leq 0 \tag{3.12}$$

shows that the Hamiltonian satisfies a continuous-time dissipation law. In particular, the total energy is non-increasing along trajectories.

The equilibria of the dissipative Hamiltonian system are of the form

$$(\theta^*, 0) \quad \text{with} \quad \nabla_{\theta} \mathcal{L}(\theta^*) = 0.$$

Indeed, at equilibrium we must have $\dot{\theta} = 0$ and $\dot{\omega} = 0$. From $\dot{\theta} = \omega$ it follows that $\omega = 0$, and substituting into (3.11) yields $\nabla_{\theta} \mathcal{L}(\theta) = 0$.

Now, we recall that a function $\mathcal{L}(\theta) \in \mathcal{C}^1$ is said to have compact sub-level sets if, for every $c \in \mathbb{R}$, the set

$$\{\theta : \mathcal{L}(\theta) \leq c\},$$

is compact.

Theorem 3.2. *Assume that $\mathcal{L}$ has compact sub-level sets and that $R_{22} \succ 0$. Then every trajectory of the dissipative Hamiltonian system (3.9) converges to the set*

$$E = \{(\theta^*, 0) : \nabla_{\theta} \mathcal{L}(\theta^*) = 0\}.$$

Proof. Since $\dot{\mathcal{H}} \leq 0$, the Hamiltonian is non-increasing along trajectories. Hence every solution remains in the sub-level set

$$\{(\theta, \omega) \in \mathbb{R}^n \times \mathbb{R}^n : \mathcal{H}(\theta, \omega) \leq \mathcal{H}(\theta(0), \omega(0))\}.$$

Let $c := \mathcal{H}(\theta(0), \omega(0))$. Since $\mathcal{H}(\theta, \omega) = \mathcal{L}(\theta) + \frac{1}{2}\|\omega\|^2$, any (θ, ω) satisfying $\mathcal{H}(\theta, \omega) \leq c$ must satisfy $\mathcal{L}(\theta) \leq c$ and $\|\omega\| \leq \sqrt{2c}$. Therefore, this sub-level set is contained in

$$\{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \leq c\} \times \{\omega \in \mathbb{R}^n : \|\omega\| \leq \sqrt{2c}\}.$$

The first set is compact by assumption, and the second is a closed and bounded subset of $\mathbb{R}^n$, hence compact. It follows that the sub-level sets of $\mathcal{H}$ are compact, and thus trajectories are bounded. Moreover,

$$\dot{\mathcal{H}} = -\omega^\top R_{22} \omega,$$

and since $R_{22} \succ 0$, we have

$$\dot{\mathcal{H}} = 0 \iff \omega = 0.$$

The largest invariant subset of $\{(\theta, \omega) : \dot{\mathcal{H}} = 0\}$ is therefore

$$E = \{(\theta^*, 0) : \nabla_{\theta} \mathcal{L}(\theta^*) = 0\},$$

since $\omega = 0$ and $\dot{\omega} = 0$ imply $\nabla_{\theta} \mathcal{L}(\theta) = 0$. The result then follows from LaSalle's invariance principle (see Theorem A). $\square$

The above theorem shows that the continuous-time dissipative Hamiltonian dynamics converge to the set of stationary points of the loss function. When the objective satisfies the strict saddle property (see Definition 2.24), saddle points correspond to unstable equilibria. Consequently, gradient-based dynamics initialized randomly almost surely converge to a local minimizer rather than a saddle point Lee et al. [13].

3.1.1 Example

To illustrate the above framework, consider the setting of Massaroli et al. [15] for training neural networks. We use the Hamiltonian defined in (3.1). For a supervised learning problem with data pair (x, y) , the loss function may be written as

$$\mathcal{L}(\theta) = \frac{\alpha}{2} \|y - f(x, \theta)\|^2 + \frac{\beta}{2} \|\theta\|^2,$$

where the second term corresponds to ℓ_2 regularization. The gradient of the Hamiltonian is,

$$\nabla \mathcal{H}(\theta, \omega) = \begin{pmatrix} \nabla_{\theta} \mathcal{L}(\theta) \\ \omega \end{pmatrix},$$

where

$$\nabla_{\theta} \mathcal{L}(\theta) = \alpha \langle \nabla_{\theta} f(x, \theta), f(x, \theta) - y \rangle + \beta \theta.$$

and $\omega = \dot{\theta}$. The first term of the gradient is the data loss $\frac{\alpha}{2} \|y - f(x, \theta)\|^2$ which is exactly the backpropagated gradient of the network output with respect to the parameters, scaled by α . Setting the gradient to 0, we obtain

$$\dot{\theta}^* = 0, \quad \nabla_{\theta} f(x, \theta)^{\top} (f(x, \theta^*) - y) + \beta \theta^* = 0, \quad (3.13)$$

where θ^* is a stationary point. Since the $\frac{1}{2} \|\dot{\theta}\|^2$ term is convex, minimization of $\dot{\theta}$ leads to $\dot{\theta}^* = 0$. The stationary points of the Hamiltonian satisfy

$$\dot{\theta}^* = 0, \quad \nabla_{\theta} \mathcal{L}(\theta^*) = 0.$$

Thus, equilibria of the dissipative Hamiltonian system are of the form $(\theta^*, 0)$, where θ^* is a stationary point of the loss function.

Note that in general a stationary point need not be a minimizer. However, every minimizer θ^* of $\mathcal{L}$ corresponds to an equilibrium $(\theta^*, 0)$ of the associated dissipative Hamiltonian dynamics.

3.2 Momentum Gradient Descent (MGD)

We now consider a simple time discretization of the continuous-time dissipative Hamiltonian system

$$\dot{\theta} = \omega, \quad (3.14)$$

$$\dot{\omega} = -\nabla_{\theta} \mathcal{L}(\theta) - \gamma \omega, \quad (3.15)$$

where $\gamma > 0$ is a damping parameter. Let $h > 0$ denote the time step and define $t_k = kh$. For $\theta_k := \theta(t_k)$ and $\omega_k := \omega(t_k)$, different time discretizations of (3.14)-(3.15) lead to different momentum-type optimization algorithms.

To obtain momentum gradient descent, we use a semi-implicit scheme by treating the momentum equation explicitly while updating the parameter variable using the new momentum value. This gives the semi-implicit scheme

$$\omega_{k+1} = \omega_k - h \nabla_{\theta} \mathcal{L}(\theta_k) - h \gamma \omega_k, \quad (3.16)$$

$$\theta_{k+1} = \theta_k + h \omega_{k+1}. \quad (3.17)$$

Equivalently,

$$\omega_{k+1} = (1 - h\gamma)\omega_k - h\nabla_{\theta}\mathcal{L}(\theta_k). \quad (3.18)$$

As before, we eliminate ω_k . From (3.17) at step $k - 1$ we have

$$\omega_k = \frac{\theta_k - \theta_{k-1}}{h}. \quad (3.19)$$

Substituting (3.18) into (3.17) gives

$$\theta_{k+1} = \theta_k + h((1 - h\gamma)\omega_k - h\nabla_{\theta}\mathcal{L}(\theta_k)).$$

Using (3.19), we obtain

$$\theta_{k+1} = \theta_k + (1 - h\gamma)(\theta_k - \theta_{k-1}) - h^2\nabla_{\theta}\mathcal{L}(\theta_k). \quad (3.20)$$

Equation (3.20) can be rewritten in the classical form

$$\theta_{k+1} = \theta_k + \beta(\theta_k - \theta_{k-1}) - \alpha\nabla_{\theta}\mathcal{L}(\theta_k), \quad (3.21)$$

by setting

$$\beta = 1 - h\gamma, \quad \alpha = h^2. \quad (3.22)$$

From the condition $0 < h\gamma \leq 1$, it follows that

$$0 \leq \beta < 1. \quad (3.23)$$

Equation (3.21) coincides with the classical momentum gradient descent method Ochs et al. [18]. Thus, the heavy-ball iteration arises naturally as a semi-implicit discretization of the dissipative Hamiltonian dynamics (3.14)-(3.15). For further discussion on the role of the momentum parameter, see Qian [20]. The resulting scheme is summarized in Algorithm 1.

Algorithm 1 Momentum Gradient Descent (MGD) when $R = \text{diag}(O_n, \gamma I_n)$

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, damping $\gamma > 0$, step size $h > 0$

Require: Iterations K

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ (often $\omega_0 = 0$)

1: **for** $k = 0, 1, \dots, K - 1$ **do**

2: Velocity update:

$$\omega_{k+1} \leftarrow (1 - \gamma h) \omega_k - h \nabla_{\theta} \mathcal{L}(\theta_k)$$

3: Position update:

$$\theta_{k+1} \leftarrow \theta_k + h \omega_{k+1}$$

4: **end for**

5: **return** θ_K

As shown below, like other explicit first-order methods, such schemes generally guarantee a decrease of the objective only under suitable step-size restrictions that depend on the loss function.

We now summarize the convergence behavior of MGD under the standard assumptions introduced in Section 2.4.

Theorem 3.3 (Convergence of MGD, Ochs et al. [18]). *Suppose that $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable with an L -Lipschitz continuous gradient (2.5), and that its sub-level sets are compact (2.6). Let $0 \leq \beta < 1$ and $\alpha > 0$ satisfy*

$$\alpha < \frac{2(1 - \beta)}{L}.$$

Then the iterates $\{\theta_k\}_{k \geq 0}$ generated by (3.21) satisfy

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 = \mathcal{O}\left(\frac{1}{K}\right).$$

This result shows that MGD achieves a sublinear $\mathcal{O}(1/K)$ best-iterate convergence rate in terms of the gradient norm.

Remark 3.4. With $\alpha = h^2$ and $\beta = 1 - h\gamma$ as in (3.22), the condition

$$\alpha < \frac{2(1 - \beta)}{L}$$

is equivalent to

$$h < \frac{2\gamma}{L}.$$

Theorem 3.3 implies that, for any tolerance $\varepsilon > 0$, there exists an index $0 \leq k \leq K$ such that

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \leq \varepsilon,$$

whenever

$$K = \mathcal{O}(\varepsilon^{-2}).$$

Thus, MGD requires at most $\mathcal{O}(\varepsilon^{-2})$ iterations to compute an ε -stationary point, which is the standard nonconvex best-iterate complexity bound.

3.3 Momentum Gradient Descent with δ (MGD- δ)

In the previous section, we showed that momentum gradient descent arises from a semi-implicit discretization of the dissipative (3.9) Hamiltonian system

$$\dot{\theta} = \omega, \quad \dot{\omega} = -\nabla_{\theta}\mathcal{L}(\theta) - \gamma\omega.$$

We now consider a different dissipative Hamiltonian system that introduces additional damping in the parameter dynamics. A nonzero block R_{11} is added to the dissipation matrix. In particular, for $R_{11} \equiv \delta I$, where $\delta > 0$ is an additional damping parameter, we obtain

$$\dot{\theta} = \omega - \delta \nabla_{\theta}\mathcal{L}(\theta), \tag{3.24}$$

$$\dot{\omega} = -\nabla_{\theta}\mathcal{L}(\theta) - \gamma\omega. \tag{3.25}$$

Using the same semi-implicit discretization as in Section 3.2, we get the updates:

$$\omega_{k+1} = (1 - h\gamma)\omega_k - h\nabla_{\theta}\mathcal{L}(\theta_k), \tag{3.26}$$

$$\theta_{k+1} = \theta_k + h\omega_{k+1} - h\delta\nabla_{\theta}\mathcal{L}(\theta_k). \tag{3.27}$$

The resulting scheme is summarized in Algorithm 2 below.

Algorithm 2 Momentum Gradient Descent with δ -damping (MGD- δ) when $R = \text{diag}(\delta I_n, \gamma I_n)$

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, parameters $\delta > 0$, $\gamma > 0$, step size $h > 0$

Require: Iterations K

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ (often $\omega_0 = 0$)

1: **for** $k = 0, 1, \dots, K - 1$ **do**

2: Velocity update:

$$\omega_{k+1} \leftarrow (1 - \gamma h)\omega_k - h\nabla_{\theta}\mathcal{L}(\theta_k)$$

3: Position update :

$$\theta_{k+1} \leftarrow \theta_k + h(\omega_{k+1} - \delta \nabla_{\theta}\mathcal{L}(\theta_k))$$

4: **end for**

5: **return** θ_K

Convergence analysis for the MGD- δ scheme is not provided in this work. In Chapter 4, we present numerical experiments demonstrating that MGD- δ often exhibits faster convergence and improved stability compared to standard MGD across a range of optimization tasks.

3.4 Discrete Gradient methods

Implicit discretizations, such as implicit gradient descent or proximal point methods, are known to be unconditionally stable with respect to the step size. However, they do not, in general, preserve the dissipative structure (3.12) of the underlying Hamiltonian system Grimm et al. [8]. As shown in the previous section, standard discretizations of dissipative Hamiltonian dynamics may fail to retain the monotonic energy decay observed in continuous time, unless restrictive step-size conditions are imposed. This limitation motivates the use of structure-preserving discretizations, developed in geometric numerical integration, which are designed to preserve key qualitative properties of the continuous system. In particular, discrete gradient methods enforce a discrete energy dissipation law by construction, ensuring monotonic energy decay at the discrete level Celledoni and Høiseth [3], Ehrhardt et al. [7].

The main idea is to replace the usual gradient $\nabla\mathcal{H}$ by a two-point quantity that depends on both the current state and the next state. This quantity is designed so that the change in energy between two consecutive iterates is represented exactly by an inner product, thereby giving a *discrete* analogue of the chain rule. Because it plays the role of the gradient at the discrete level, it is called a *discrete gradient*. This construction leads to numerical schemes that inherit an exact discrete-time dissipation law, that mirrors the continuous-time energy decay for any step size.

Recent work has established well-posedness and convergence guarantees for discrete gradient methods in optimization. For smooth objectives, discrete-gradient methods admit well-posed implicit updates and enjoy convergence guarantees, including sublinear $\mathcal{O}(1/K)$ rates in the convex setting and linear convergence under the Polyak–Łojasiewicz condition Ehrhardt et al. [7]. In this section, we apply this framework to the dissipative Hamiltonian system (3.9), which yields a well-posed optimization scheme with convergence guarantees for L -smooth, possibly non-convex objectives.

We begin with the formal definition.

Definition 3.5 (Discrete Gradient). A function

$$\bar{\nabla}\mathcal{H} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

is called a *discrete gradient* of a differentiable function $\mathcal{H} : \mathbb{R}^n \rightarrow \mathbb{R}$ if, for all $\xi, \tilde{\xi} \in \mathbb{R}^n$, it satisfies

$$\mathcal{H}(\tilde{\xi}) - \mathcal{H}(\xi) = \bar{\nabla}\mathcal{H}(\xi, \tilde{\xi})^\top (\tilde{\xi} - \xi), \quad (3.28)$$

$$\bar{\nabla}\mathcal{H}(\xi, \xi) = \nabla\mathcal{H}(\xi). \quad (3.29)$$

The first property is a discrete chain rule, while the second ensures consistency with the analytic gradient.

Let $h > 0$ denote the time step and define $t_k = kh$. For $\xi_k := \xi(t_k)$, the discrete-gradient discretization of the dissipative Hamiltonian system (3.9) is given by

$$\frac{\xi_{k+1} - \xi_k}{h} = (J - R)\bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}), \quad (3.30)$$

The importance of this construction is that, due to (3.28), the scheme preserves the dissipation structure exactly at the discrete level. In particular, the numerical method satisfies a discrete energy law that directly mirrors the continuous dissipation law (3.12).

Proposition 3.6 (Discrete energy dissipation). *Let ξ_{k+1} be defined by (3.30). Then*

$$\mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k) = -h \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1})^\top \tilde{R}(\xi_k, \xi_{k+1}) \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}) \leq 0. \quad (3.31)$$

In particular, for every $h > 0$,

$$\frac{\mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k)}{h} \leq 0. \quad (3.32)$$

Proof. Using the discrete chain rule (3.28),

$$\mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k) = \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1})^\top (\xi_{k+1} - \xi_k).$$

Substituting (3.30) gives

$$\mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k) = h \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1})^\top (\tilde{J}(\xi_k, \xi_{k+1}) - \tilde{R}(\xi_k, \xi_{k+1})) \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}).$$

Since $\tilde{J}$ is skew-symmetric, $\bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1})^\top \tilde{J}(\xi_k, \xi_{k+1}) \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}) = 0$ and we obtain

$$\mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k) = -h \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1})^\top \tilde{R}(\xi_k, \xi_{k+1}) \bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}) \leq 0,$$

as $\tilde{R}$ is positive semidefinite. □

An important example of a discrete gradient is the *averaged vector field* (AVF) discrete gradient Ehrhardt et al. [7], defined by

$$\bar{\nabla}\mathcal{H}(\xi_k, \xi_{k+1}) := \int_0^1 \nabla\mathcal{H}((1-s)\xi_k + s\xi_{k+1}) ds. \quad (3.33)$$

We now verify that this indeed defines a discrete gradient.

Proposition 3.7. *The AVF discrete gradient (3.33) satisfies the two properties (3.28)-(3.29).*

Proof. Let

$$\rho(s) = (1-s)\xi_k + s\xi_{k+1}, \quad \rho'(s) = \xi_{k+1} - \xi_k.$$

By the fundamental theorem of calculus,

$$\begin{aligned} \mathcal{H}(\xi_{k+1}) - \mathcal{H}(\xi_k) &= \int_0^1 \frac{d}{ds} \mathcal{H}(\rho(s)) ds \\ &= \int_0^1 \nabla \mathcal{H}(\rho(s))^\top \rho'(s) ds \\ &= \left(\int_0^1 \nabla \mathcal{H}(\rho(s)) ds \right)^\top (\xi_{k+1} - \xi_k). \end{aligned}$$

This proves (3.28). For (3.29), if $\xi_{k+1} = \xi_k = \xi$, then

$$\bar{\nabla} \mathcal{H}(\xi, \xi) = \int_0^1 \nabla \mathcal{H}(\xi) ds = \nabla \mathcal{H}(\xi).$$

□

Now that we have introduced discrete gradients, we return to our underlying optimization problem. Recall that, for $\xi = (\theta, \omega) \in \mathbb{R}^{2n}$, we consider the Hamiltonian (3.1). We now apply the discrete-gradient framework to this Hamiltonian to derive structure-preserving optimization updates. Recall that the dissipative Hamiltonian structure is defined by the matrices

$$J = \begin{bmatrix} O_n & I_n \\ -I_n & O_n \end{bmatrix}, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \quad R = R^\top \succeq 0.$$

3.4.1 Discrete gradient updates (DG)

We first consider the standard dissipation structure in which damping acts only in the momentum dynamics, corresponding to

$$R = \begin{bmatrix} O_n & O_n \\ O_n & \gamma I_n \end{bmatrix} \succ 0, \quad \gamma > 0.$$

The discrete gradient discretization of the dissipative Hamiltonian system is obtained via (3.30). Let $\xi_k := (\theta_k, \omega_k)$ and $\xi_{k+1} := (\theta_{k+1}, \omega_{k+1})$. The scheme then takes the form

$$\theta_{k+1} = \theta_k + h \bar{\nabla}_\omega \mathcal{H}(\xi_k, \xi_{k+1}), \tag{3.34}$$

$$\omega_{k+1} = \omega_k - h \left(\bar{\nabla}_\theta \mathcal{H}(\xi_k, \xi_{k+1}) + \gamma \bar{\nabla}_\omega \mathcal{H}(\xi_k, \xi_{k+1}) \right). \tag{3.35}$$

Because the Hamiltonian is separable in θ and ω , the two discrete-gradient components are

$$\bar{\nabla}_\theta \mathcal{H}(\xi_k, \xi_{k+1}) = \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta_{k+1} - \theta_k)) ds, \quad (3.36)$$

$$\bar{\nabla}_\omega \mathcal{H}(\xi_k, \xi_{k+1}) = \frac{1}{2}(\omega_k + \omega_{k+1}). \quad (3.37)$$

Indeed, for the ω term, i.e., $\bar{\nabla}_\omega \mathcal{H}(\xi_k, \xi_{k+1})$, we have

$$\begin{aligned} \bar{\nabla}_\omega \mathcal{H}(\omega_k, \omega_{k+1}) &= \int_0^1 (\omega_k + s(\omega_{k+1} - \omega_k)) ds \\ &= \omega_k + \left(\int_0^1 s ds \right) (\omega_{k+1} - \omega_k) = \frac{1}{2}(\omega_k + \omega_{k+1}). \end{aligned}$$

Substituting (3.36)-(3.37) into (3.34)-(3.35) yields the DG updates

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}), \quad (3.38)$$

$$\omega_{k+1} = \omega_k - h(\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) + \gamma \frac{1}{2}(\omega_k + \omega_{k+1})). \quad (3.39)$$

This scheme is implicit (see Definition 2.3), since the discrete gradient depends on the unknown iterate θ_{k+1} .

In practice, it is convenient to rewrite this system as a fixed-point equation in the variable θ_{k+1} . Starting from the DG updates

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}), \quad (3.40)$$

$$\omega_{k+1} = \omega_k - h(\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) + \gamma \frac{1}{2}(\omega_k + \omega_{k+1})), \quad (3.41)$$

we first solve (3.41) for ω_{k+1} . Rearranging terms gives

$$(1 + \frac{h\gamma}{2})\omega_{k+1} = (1 - \frac{h\gamma}{2})\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}).$$

Substituting this relation into (3.40) yields

$$\theta_{k+1} = \theta_k + h \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})}{2 + h\gamma}.$$

Since the discrete gradient depends on θ_{k+1} , this defines a nonlinear fixed-point (see Definition E.11) problem. Define the discrete gradient

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta - \theta_k)) ds,$$

and introduce

$$m(\theta) := \frac{2\omega_k - h\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)}{2 + h\gamma}.$$

Then the DG step can be written as the fixed-point equation

$$\theta_{k+1} = \Phi_k(\theta_{k+1}), \quad \Phi_k(\theta) := \theta_k + h m(\theta). \quad (3.42)$$

The following lemma 3.8 establishes that the fixed-point problem (3.42) is well posed (see Definition E.10). In the lemma, the index k denotes the outer DG iteration. Since the DG step is implicit, the update θ_{k+1} is computed via an inner fixed-point iteration, which we index by j . The iterates $\theta^{(j)}$ therefore approximate the fixed point θ_{k+1} .

To establish well-posedness, we first require the following Lipschitz estimates for the discrete gradient.

Lemma 3.8 (Ehrhardt et al. [7]). *If $\mathcal{L}$ is L -smooth, then for all $\theta, \tilde{\theta} \in \mathbb{R}^n$,*

$$\|\bar{\nabla}_\theta \mathcal{L}(\theta, \tilde{\theta}) - \nabla_\theta \mathcal{L}(\theta)\| \leq \frac{L}{2} \|\tilde{\theta} - \theta\|. \quad (3.43)$$

Moreover, the following holds

$$\|\bar{\nabla}_\theta \mathcal{L}(\theta, \tilde{\theta}) - \bar{\nabla}_\theta \mathcal{L}(\theta, \theta')\| \leq \frac{L}{2} \|\tilde{\theta} - \theta'\|. \quad (3.44)$$

Proof. Using the definition of the AVF discrete gradient and the L -smoothness of $\mathcal{H}$,

$$\begin{aligned} \|\bar{\nabla}_\theta \mathcal{L}(\theta, \tilde{\theta}) - \nabla_\theta \mathcal{L}(\theta)\| &= \left\| \int_0^1 (\nabla_\theta \mathcal{L}(\theta + s(\tilde{\theta} - \theta)) - \nabla_\theta \mathcal{L}(\theta)) ds \right\| \\ &\leq \int_0^1 \|\nabla_\theta \mathcal{L}(\theta + s(\tilde{\theta} - \theta)) - \nabla_\theta \mathcal{L}(\theta)\| ds \\ &\leq \int_0^1 Ls \|\tilde{\theta} - \theta\| ds = \frac{L}{2} \|\tilde{\theta} - \theta\|. \end{aligned}$$

This proves (3.43). The proof of (3.44) is analogous:

$$\begin{aligned} \|\bar{\nabla}_\theta \mathcal{L}(\theta, \tilde{\theta}) - \bar{\nabla}_\theta \mathcal{L}(\theta, \theta')\| &= \left\| \int_0^1 (\nabla_\theta \mathcal{L}(\theta + s(\tilde{\theta} - \theta)) - \nabla_\theta \mathcal{L}(\theta + s(\theta' - \theta))) ds \right\| \\ &\leq \int_0^1 Ls \|\tilde{\theta} - \theta'\| ds = \frac{L}{2} \|\tilde{\theta} - \theta'\|. \end{aligned}$$

□

Lemma 3.9 (Well-posedness of DG updates (one step)). *Assume that $\nabla_\theta \mathcal{L}$ is L -Lipschitz (L -smooth). Fix $\gamma > 0$, $h > 0$, and suppose $(\theta_k, \omega_k) \in \mathbb{R}^n \times \mathbb{R}^n$ is given. Define*

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta - \theta_k)) ds, \quad m(\theta) := \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)}{2 + h\gamma},$$

and the fixed-point map

$$\Phi_k(\theta) := \theta_k + h m(\theta).$$

Let

$$\Lambda := \frac{Lh^2}{2(2 + h\gamma)}.$$

If $\Lambda < 1$, then Φ_k is a contraction on $\mathbb{R}^n$ and therefore admits a unique fixed point $\theta_{k+1} \in \mathbb{R}^n$ satisfying

$$\theta_{k+1} = \Phi_k(\theta_{k+1}).$$

Set $m_k := m(\theta_{k+1})$ and define $\omega_{k+1} := 2m_k - \omega_k$. Then $(\theta_{k+1}, \omega_{k+1})$ satisfies the DG updates (3.38)–(3.39). Moreover, the fixed-point iterations $\theta^{(j+1)} = \Phi_k(\theta^{(j)})$ converge linearly to θ_{k+1} :

$$\|\theta^{(j)} - \theta_{k+1}\| \leq \Lambda^j \|\theta^{(0)} - \theta_{k+1}\|.$$

Proof. By (3.44),

$$\left\| \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) - \bar{\nabla}_\theta \mathcal{L}(\theta_k, \tilde{\theta}) \right\| \leq \frac{L}{2} \|\theta - \tilde{\theta}\|.$$

Hence,

$$\|\Phi_k(\theta) - \Phi_k(\tilde{\theta})\| = \frac{h^2}{2 + h\gamma} \left\| \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) - \bar{\nabla}_\theta \mathcal{L}(\theta_k, \tilde{\theta}) \right\| \leq \Lambda \|\theta - \tilde{\theta}\|.$$

If $\Lambda < 1$, Φ_k is a contraction on $(\mathbb{R}^n, \|\cdot\|)$. By the Banach fixed-point theorem (see Theorem E.13), it admits a unique fixed point θ_{k+1} . Set $m_k := m(\theta_{k+1})$. Then

$$\theta_{k+1} - \theta_k = h m_k.$$

From the definition of m_k , we obtain

$$\omega_{k+1} - \omega_k = -h(\bar{\nabla}_\theta \mathcal{L}_k + \gamma m_k),$$

with $m_k = \frac{1}{2}(\omega_k + \omega_{k+1})$. Hence, $(\theta_{k+1}, \omega_{k+1})$ satisfies the DG updates (3.38)–(3.39). Finally, using the contraction property and $\Phi_k(\theta_{k+1}) = \theta_{k+1}$,

$$\|\theta^{(j+1)} - \theta_{k+1}\| = \|\Phi_k(\theta^{(j)}) - \Phi_k(\theta_{k+1})\| \leq \Lambda \|\theta^{(j)} - \theta_{k+1}\|.$$

Iterating yields $\|\theta^{(j)} - \theta_{k+1}\| \leq \Lambda^j \|\theta^{(0)} - \theta_{k+1}\|$. □

The fixed-point representation above leads to the implementation shown in Algorithm 3.

Lemma 3.9 provides a sufficient stepsize condition $0 < h < h_{\max}$ ensuring that the DG step exists uniquely for every iteration $k = 0, 1, \dots, K$. The method may still perform well for larger values of h , but formal guarantees are no longer ensured.

The contraction condition $\Lambda < 1$ yields

$$Lh^2 - 2\gamma h - 4 < 0.$$

Define $p(h) := Lh^2 - 2\gamma h - 4$. Since $p(0) = -4 < 0$, the inequality $p(h) < 0$ holds for $0 < h < h_{\max}$, where $h_{\max}$ is the positive root of $p(h) = 0$. This gives us

$$h_{\max} = \frac{2\gamma + \sqrt{4\gamma^2 + 16L}}{2L} = \frac{\gamma + \sqrt{\gamma^2 + 4L}}{L}.$$

For comparison, the maximum stepsize ensuring stability of classical momentum gradient descent is typically of order γ/L (Remark B.5), so the DG method admits a substantially larger guaranteed stepsize.

Algorithm 3 Discrete-Gradient updates ($R = \text{diag}(O_n, \gamma I_n)$)

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, damping $\gamma > 0$, step size $h > 0$

Require: Outer iterations K , inner iterations J , inner tolerance ε

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ (often $\omega_0 = 0$)

- 1: **for** $k = 0, 1, \dots, K - 1$ **do**
- 2: Define the discrete gradient (AVF):

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta - \theta_k)) ds$$

- 3: Define the map $m(\theta)$ and fixed-point operator Φ_k :

$$m(\theta) := \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)}{2 + h\gamma}, \quad \Phi_k(\theta) := \theta_k + h m(\theta)$$

{If $\frac{Lh^2}{2(2+h\gamma)} < 1$, then Φ_k is a contraction (lemma 3.9).}

- 4: Then θ_{k+1} solves the fixed-point equation $\theta_{k+1} = \Phi_k(\theta_{k+1})$.

- 5: **Inner fixed point iteration :**

- 6: $\theta^{(0)} \leftarrow \theta_k + \frac{2h}{2 + h\gamma} \omega_k$ {initial guess}

- 7: $\hat{\theta} \leftarrow \theta^{(0)}$

- 8: **for** $j = 0, 1, \dots, J - 1$ **do**

- 9: $\hat{\theta} \leftarrow \Phi_k(\theta^{(j)})$

- 10: **if** $\|\hat{\theta} - \theta^{(j)}\| \leq \varepsilon$ **then**

- 11: **break**

- 12: **end if**

- 13: $\theta^{(j+1)} \leftarrow \hat{\theta}$

- 14: **end for**

- 15: **Accept:** $\theta_{k+1} \leftarrow \hat{\theta}$

- 16: $\bar{\nabla}_\theta \mathcal{L}_k \leftarrow \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})$

- 17: $m_k \leftarrow \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}_k}{2 + h\gamma}$

- 18: $\omega_{k+1} \leftarrow 2m_k - \omega_k$

- 19: **end for**

- 20: **return** θ_K
-

3.4.2 Convergence analysis of the DG method

In this section, we analyze the convergence properties of the DG updates (3.38)-(3.39). Throughout, we assume that the implicit DG step is solved exactly, so that the update relations hold. By Lemma 3.9, this is guaranteed whenever

$$\frac{Lh^2}{2(2+h\gamma)} < 1.$$

For convenience, define

$$\mathcal{H}_k := \mathcal{H}(\theta_k, \omega_k) = \mathcal{L}(\theta_k) + \frac{1}{2}\|\omega_k\|^2, \quad \bar{\nabla}_\theta \mathcal{L}_k := \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}), \quad m_k := \frac{\omega_k + \omega_{k+1}}{2}.$$

For the DG method with

$$R = \begin{bmatrix} O_n & O_n \\ O_n & \gamma I_n \end{bmatrix} \succ 0, \quad \gamma > 0.$$

Proposition 3.6 gives the discrete energy law

$$\mathcal{H}_{k+1} - \mathcal{H}_k = -\frac{h\gamma}{4}\|\omega_{k+1} + \omega_k\|^2 = -h\gamma\|m_k\|^2 \leq 0. \quad (3.45)$$

Hence the sequence $\{\mathcal{H}_k\}_{k \geq 0}$ is nonincreasing.

Recall, since our function $\mathcal{L}$ is L -smooth (2.5) and has compact sub-level sets (2.6), the function $\mathcal{L}$ is bounded below. Indeed, since sub-level sets are compact and nonempty, and $\mathcal{L}$ is continuous, it attains its minimum on each sub-level set. In particular, $\inf_\theta \mathcal{L}(\theta) = \mathcal{L}_{\inf} > -\infty$. Now we have some preliminary estimates.

Lemma 3.10. *For every $K \geq 0$,*

$$\frac{1}{K+1} \sum_{k=0}^K \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{h\gamma(K+1)}. \quad (3.46)$$

Consequently,

$$\sum_{k=0}^{\infty} \|m_k\|^2 < \infty, \quad m_k \rightarrow 0,$$

and

$$\min_{0 \leq k \leq K} \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{h\gamma(K+1)}. \quad (3.47)$$

Proof. Summing (3.45) from $k = 0$ to K gives

$$h\gamma \sum_{k=0}^K \|m_k\|^2 = \mathcal{H}_0 - \mathcal{H}_{K+1}.$$

Since

$$\mathcal{H}_{K+1} = \mathcal{L}(\theta_{K+1}) + \frac{1}{2}\|\omega_{K+1}\|^2 \geq \mathcal{L}_{\inf},$$

we obtain

$$h\gamma \sum_{k=0}^K \|m_k\|^2 \leq \mathcal{H}_0 - \mathcal{L}_{\inf}.$$

Dividing by $h\gamma(K+1)$ yields (3.46). The summability and convergence $m_k \rightarrow 0$ follow immediately. The bound (3.47) follows from

$$\min_{0 \leq k \leq K} \|m_k\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|m_k\|^2.$$

□

Corollary 3.11. *For every $K \geq 0$,*

$$\frac{1}{K+1} \sum_{k=0}^K \|\theta_{k+1} - \theta_k\|^2 \leq \frac{h(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma(K+1)}. \quad (3.48)$$

Consequently,

$$\sum_{k=0}^{\infty} \|\theta_{k+1} - \theta_k\|^2 < \infty, \quad \theta_{k+1} - \theta_k \rightarrow 0,$$

and

$$\min_{0 \leq k \leq K} \|\theta_{k+1} - \theta_k\|^2 \leq \frac{h(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma(K+1)}. \quad (3.49)$$

Proof. From (3.38),

$$\theta_{k+1} - \theta_k = h m_k.$$

Hence

$$\|\theta_{k+1} - \theta_k\|^2 = h^2 \|m_k\|^2.$$

The result follows directly from Lemma 3.10. □

Lemma 3.12. *Under the assumption that the sub-level sets of $\mathcal{L}$ are compact, the sequences $\{\theta_k\}_{k \geq 0}$ and $\{\omega_k\}_{k \geq 0}$ generated by the DG method are bounded. Consequently, the sequence $\{(\theta_k, \omega_k)\}_{k \geq 0}$ admits convergent subsequences. Moreover, there exists a constant $G < \infty$ such that*

$$\|\bar{\nabla}_\theta \mathcal{L}_k\| \leq G, \quad \forall k \geq 0.$$

Proof. Since $\mathcal{H}_k \leq \mathcal{H}_0$ for all k , we have

$$\mathcal{L}(\theta_k) \leq \mathcal{H}_k \leq \mathcal{H}_0.$$

Thus, $\{\theta_k\}_{k \geq 0}$ lies in the sub-level set

$$\{\theta \in \mathbb{R}^n : \mathcal{L}(\theta) \leq \mathcal{H}_0\},$$

which is compact our assumption. In particular, $\{\theta_k\}_{k \geq 0}$ is bounded. Moreover,

$$\frac{1}{2} \|\omega_k\|^2 = \mathcal{H}_k - \mathcal{L}(\theta_k) \leq \mathcal{H}_0 - \mathcal{L}_{\inf},$$

so the sequence $\{\omega_k\}_{k \geq 0}$ is bounded as well. The existence of convergent subsequences follows from the Bolzano-Weierstrass theorem.

Now, since both θ_k and θ_{k+1} belong to the compact sub-level set, the entire line segment

$$\theta_k + s(\theta_{k+1} - \theta_k), \quad s \in [0, 1],$$

is contained in a compact subset. Hence, there is a $R > 0$ with $\|\theta_k\| \leq R$ for all k . Since $\{\theta_k\}_{k \geq 0}$ is contained in a compact subset of $\mathbb{R}^n$ and $\nabla_\theta \mathcal{L}$ is continuous, it follows that $\nabla_\theta \mathcal{L}$ is bounded on this set. Hence, there exists a constant $G < \infty$ such that

$$\|\nabla_\theta \mathcal{L}(\theta)\| \leq G := \max_{\|\theta\| \leq R} \|\nabla_\theta \mathcal{L}(\theta)\|$$

for all θ on these segments. Using the definition of the AVF discrete gradient,

$$\bar{\nabla}_\theta \mathcal{L}_k = \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta_{k+1} - \theta_k)) ds,$$

we obtain

$$\|\bar{\nabla}_\theta \mathcal{L}_k\| \leq \int_0^1 \|\nabla_\theta \mathcal{L}(\theta_k + s(\theta_{k+1} - \theta_k))\| ds \leq \int_0^1 G ds = G.$$

□

We now use the preceding lemmas to derive convergence rates for the DG method.

Theorem 3.13 (*Best-iterate rate for the discrete gradient*). Suppose that $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable with an L -Lipschitz continuous gradient (2.5), and that its sub-level sets are compact (2.6). Let $\{(\theta_k, \omega_k)\}_{k \geq 0}$ be generated by the DG method with $h > 0$ and $\gamma > 0$. Then there exists a constant $C < \infty$ such that for every $K \geq 0$,

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})\|^2 \leq \frac{C(h, L, \gamma)}{K+1}, \quad (3.50)$$

where

$$C := \frac{G\sqrt{8(\mathcal{H}_0 - \mathcal{L}_{\inf})}}{h} + \frac{\gamma(\mathcal{H}_0 - \mathcal{L}_{\inf})}{h} + \left|1 - \frac{\gamma h}{2}\right| \frac{\sqrt{5}L(\mathcal{H}_0 - \mathcal{L}_{\inf})}{h\gamma} + \frac{hG^2}{2h} + \frac{5L^2h^2(\mathcal{H}_0 - \mathcal{L}_{\inf})}{4h\gamma}. \quad (3.51)$$

In particular,

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})\|^2 = \mathcal{O}(1/K).$$

Proof. We proceed in several steps.

Step 1: Basic identities. Recall that the DG updates can be written as

$$\theta_{k+1} - \theta_k = hm_k, \quad \omega_{k+1} - \omega_k = -h(\bar{\nabla}_\theta \mathcal{L}_k + \gamma m_k). \quad (3.52)$$

By the discrete chain rule,

$$\mathcal{L}(\theta_{k+1}) - \mathcal{L}(\theta_k) = \bar{\nabla}_\theta \mathcal{L}_k^\top (\theta_{k+1} - \theta_k) = h \bar{\nabla}_\theta \mathcal{L}_k^\top m_k. \quad (3.53)$$

Moreover, by Lemma 3.10,

$$\min_{0 \leq k \leq K} \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{h\gamma(K+1)}. \quad (3.54)$$

Step 2: Summation identity for the discrete gradient. Taking the inner product of the second relation in (3.52) with $\bar{\nabla}_\theta \mathcal{L}_k$ gives

$$\bar{\nabla}_\theta \mathcal{L}_k^\top (\omega_{k+1} - \omega_k) = -h \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 - h\gamma \bar{\nabla}_\theta \mathcal{L}_k^\top m_k.$$

Using (3.53) to eliminate $\bar{\nabla}_\theta \mathcal{L}_k^\top m_k$, we obtain

$$h \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 = -\bar{\nabla}_\theta \mathcal{L}_k^\top (\omega_{k+1} - \omega_k) - \gamma(\mathcal{L}(\theta_{k+1}) - \mathcal{L}(\theta_k)).$$

Summing from $k = 0$ to K yields

$$h \sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 = -\sum_{k=0}^K \bar{\nabla}_\theta \mathcal{L}_k^\top (\omega_{k+1} - \omega_k) - \gamma(\mathcal{L}(\theta_{K+1}) - \mathcal{L}(\theta_0)). \quad (3.55)$$

Since $\mathcal{L}(\theta_{K+1}) \geq \mathcal{L}_{\inf}$, it follows that

$$-\gamma(\mathcal{L}(\theta_{K+1}) - \mathcal{L}(\theta_0)) \leq \gamma(\mathcal{L}(\theta_0) - \mathcal{L}_{\inf}) \leq \gamma(\mathcal{H}_0 - \mathcal{L}_{\inf}). \quad (3.56)$$

Hence,

$$h \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 \leq -\sum_{k=0}^K \bar{\nabla}_{\theta} \mathcal{L}_k^{\top} (\omega_{k+1} - \omega_k) + \gamma(\mathcal{H}_0 - \mathcal{L}_{\inf}). \quad (3.57)$$

We want a bound on the left side of the above equation. Now consider the first term on the right-hand side.

Step 3: Using Summation by parts.

$$\sum_{k=0}^K \bar{\nabla}_{\theta} \mathcal{L}_k^{\top} (\omega_{k+1} - \omega_k) = \bar{\nabla}_{\theta} \mathcal{L}_K^{\top} \omega_{K+1} - \bar{\nabla}_{\theta} \mathcal{L}_0^{\top} \omega_0 - \sum_{k=0}^{K-1} (\bar{\nabla}_{\theta} \mathcal{L}_{k+1} - \bar{\nabla}_{\theta} \mathcal{L}_k)^{\top} \omega_{k+1}. \quad (3.58)$$

Substituting (3.58) into (3.57) and using the triangle inequality gives

$$\begin{aligned} h \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 &\leq |\bar{\nabla}_{\theta} \mathcal{L}_K^{\top} \omega_{K+1}| + |\bar{\nabla}_{\theta} \mathcal{L}_0^{\top} \omega_0| \\ &\quad + \left| \sum_{k=0}^{K-1} (\bar{\nabla}_{\theta} \mathcal{L}_{k+1} - \bar{\nabla}_{\theta} \mathcal{L}_k)^{\top} \omega_{k+1} \right| + \gamma(\mathcal{H}_0 - \mathcal{L}_{\inf}). \end{aligned} \quad (3.59)$$

By Lemma 3.12, we have

$$\|\bar{\nabla}_{\theta} \mathcal{L}_k\| \leq G, \quad \|\omega_k\| \leq \sqrt{2(\mathcal{H}_0 - \mathcal{L}_{\inf})},$$

for all k , so the first two terms on the right-hand side of (3.59) satisfy

$$|\bar{\nabla}_{\theta} \mathcal{L}_K^{\top} \omega_{K+1}| + |\bar{\nabla}_{\theta} \mathcal{L}_0^{\top} \omega_0| \leq 2G\sqrt{2(\mathcal{H}_0 - \mathcal{L}_{\inf})}. \quad (3.60)$$

We are left with finding a bound for the third term in (3.59).

Step 4: Estimate of the term. By Lemma 3.8,

$$\|\bar{\nabla}_{\theta} \mathcal{L}_k - \nabla_{\theta} \mathcal{L}(\theta_k)\| \leq \frac{L}{2} \|\theta_{k+1} - \theta_k\| = \frac{Lh}{2} \|m_k\|.$$

Therefore,

$$\begin{aligned} \|\bar{\nabla}_{\theta} \mathcal{L}_{k+1} - \bar{\nabla}_{\theta} \mathcal{L}_k\| &\leq \|\bar{\nabla}_{\theta} \mathcal{L}_{k+1} - \nabla_{\theta} \mathcal{L}(\theta_{k+1})\| + \|\nabla_{\theta} \mathcal{L}(\theta_{k+1}) - \nabla_{\theta} \mathcal{L}(\theta_k)\| + \|\nabla_{\theta} \mathcal{L}(\theta_k) - \bar{\nabla}_{\theta} \mathcal{L}_k\| \\ &\leq \frac{Lh}{2} \|m_{k+1}\| + Lh \|m_k\| + \frac{Lh}{2} \|m_k\| \\ &= \frac{Lh}{2} \|m_{k+1}\| + \frac{3Lh}{2} \|m_k\|. \end{aligned} \quad (3.61)$$

Squaring, summing, and using $(a + b)^2 \leq 2a^2 + 2b^2$ gives

$$\sum_{k=0}^{K-1} \|\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k\|^2 \leq 5L^2 h^2 \sum_{k=0}^K \|m_k\|^2 \leq \frac{5L^2 h(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma},$$

where the last step follows from Lemma 3.10. Now use the identity

$$\omega_{k+1} = \left(1 - \frac{\gamma h}{2}\right) m_k - \frac{h}{2} \bar{\nabla}_\theta \mathcal{L}_k, \quad (3.62)$$

which follows directly from the DG updates. Then

$$\begin{aligned} & \left| \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top \omega_{k+1} \right| \\ & \leq \left| 1 - \frac{\gamma h}{2} \right| \left| \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top m_k \right| + \frac{h}{2} \left| \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top \bar{\nabla}_\theta \mathcal{L}_k \right|. \end{aligned} \quad (3.63)$$

For the first sum, Cauchy-Schwarz and (3.62) imply

$$\begin{aligned} \left| \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top m_k \right| & \leq \left(\sum_{k=0}^{K-1} \|\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k\|^2 \right)^{1/2} \left(\sum_{k=0}^{K-1} \|m_k\|^2 \right)^{1/2} \\ & \leq \frac{\sqrt{5} L(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma}. \end{aligned} \quad (3.64)$$

For the second sum, use the identity

$$(a - b)^\top b = \frac{1}{2} (\|a\|^2 - \|b\|^2 - \|a - b\|^2),$$

with $a = \bar{\nabla}_\theta \mathcal{L}_{k+1}$ and $b = \bar{\nabla}_\theta \mathcal{L}_k$, to obtain

$$\begin{aligned} \left| \frac{h}{2} \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top \bar{\nabla}_\theta \mathcal{L}_k \right| & \leq \frac{h}{4} (\|\bar{\nabla}_\theta \mathcal{L}_K\|^2 + \|\bar{\nabla}_\theta \mathcal{L}_0\|^2) + \frac{h}{4} \sum_{k=0}^{K-1} \|\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k\|^2 \\ & \leq \frac{hG^2}{2} + \frac{5L^2 h^2 (\mathcal{H}_0 - \mathcal{L}_{\inf})}{4\gamma}. \end{aligned} \quad (3.65)$$

Combining (3.63), (3.64), and (3.65), results in

$$\left| \sum_{k=0}^{K-1} (\bar{\nabla}_\theta \mathcal{L}_{k+1} - \bar{\nabla}_\theta \mathcal{L}_k)^\top \omega_{k+1} \right| \leq \left| 1 - \frac{\gamma h}{2} \right| \frac{\sqrt{5} L(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma} + \frac{hG^2}{2} + \frac{5L^2 h^2 (\mathcal{H}_0 - \mathcal{L}_{\inf})}{4\gamma}. \quad (3.66)$$

Step 5: Conclusion. Substituting (3.60) and (3.66) into (3.59) yields

$$h \sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 \leq C,$$

where

$$C := 2G\sqrt{2(\mathcal{H}_0 - \mathcal{L}_{\inf})} + \gamma(\mathcal{H}_0 - \mathcal{L}_{\inf}) + \left|1 - \frac{\gamma h}{2}\right| \frac{\sqrt{5}L(\mathcal{H}_0 - \mathcal{L}_{\inf})}{\gamma} + \frac{hG^2}{2} + \frac{5L^2h^2(\mathcal{H}_0 - \mathcal{L}_{\inf})}{4\gamma}.$$

Dividing by $h(K+1)$ and using

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2$$

we conclude that

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 \leq \frac{C/h}{K+1}.$$

Renaming C/h as C proves (3.50). $\square$

Corollary 3.14 (*Best-iterate rate for the true gradient*). *Under the assumptions of Theorem 3.13, there exists a constant $C' < \infty$ such that the true gradient satisfies*

$$\min_{0 \leq k \leq K} \|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq \frac{\tilde{C}(\gamma, L, h)}{K+1}, \quad (3.67)$$

where

$$\tilde{C}(\gamma, L, h) := 2C + \frac{L^2h(\mathcal{H}_0 - \mathcal{L}_{\inf})}{2\gamma}. \quad (3.68)$$

In particular,

$$\min_{0 \leq k \leq K} \|\nabla_\theta \mathcal{L}(\theta_k)\|^2 = \mathcal{O}(1/K).$$

Proof. By Lemma 3.8, we have, for every k ,

$$\|\nabla_\theta \mathcal{L}(\theta_k)\| \leq \|\bar{\nabla}_\theta \mathcal{L}_k\| + \|\bar{\nabla}_\theta \mathcal{L}_k - \nabla_\theta \mathcal{L}(\theta_k)\| \leq \|\bar{\nabla}_\theta \mathcal{L}_k\| + \frac{Lh}{2} \|m_k\|. \quad (3.69)$$

Using $(a+b)^2 \leq 2a^2 + 2b^2$, we obtain

$$\|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq 2\|\bar{\nabla}_\theta \mathcal{L}_k\|^2 + \frac{L^2h^2}{2} \|m_k\|^2. \quad (3.70)$$

Summing (3.70) from $k = 0$ to K gives

$$\sum_{k=0}^K \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq 2 \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 + \frac{L^2 h^2}{2} \sum_{k=0}^K \|m_k\|^2. \quad (3.71)$$

Using the estimates from Theorem 3.13,

$$\sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 \leq C, \quad (3.72)$$

where

$$C = \frac{G\sqrt{8(\mathcal{H}_0 - \mathcal{L}_{\inf})}}{h} + \frac{\gamma(\mathcal{H}_0 - \mathcal{L}_{\inf})}{h} + \left|1 - \frac{\gamma h}{2}\right| \frac{\sqrt{5} L(\mathcal{H}_0 - \mathcal{L}_{\inf})}{h\gamma} + \frac{hG^2}{2h} + \frac{5L^2 h^2 (\mathcal{H}_0 - \mathcal{L}_{\inf})}{4h\gamma},$$

and

$$\sum_{k=0}^K \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\gamma h}, \quad (3.73)$$

we obtain

$$\sum_{k=0}^K \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq 2C + \frac{L^2 h^2}{2} \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\gamma h}. \quad (3.74)$$

Finally, since

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2, \quad (3.75)$$

we conclude that

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{1}{K+1} \left(2C + \frac{L^2 h (\mathcal{H}_0 - \mathcal{L}_{\inf})}{2\gamma} \right). \quad (3.76)$$

Thus (3.67) holds with

$$\tilde{C}(\gamma, L, h) = 2C + \frac{L^2 h (\mathcal{H}_0 - \mathcal{L}_{\inf})}{2\gamma}. \quad (3.77)$$

□

3.4.3 Discrete gradient updates with δ (DG- δ)

We now consider the modified dissipative Hamiltonian system introduced in Chapter 3, where additional damping is present in the parameter dynamics. This corresponds to the dissipation matrix

$$R = \begin{bmatrix} \delta I_n & O_n \\ O_n & \gamma I_n \end{bmatrix} \succ 0, \quad \delta > 0, \gamma > 0.$$

Applying the discrete gradient discretization (3.30) to (3.9), and proceeding as in the DG case, we obtain the DG- δ updates

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}) - h\delta \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}), \quad (3.78)$$

$$\omega_{k+1} = \omega_k - h \left(\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) + \gamma \frac{1}{2}(\omega_k + \omega_{k+1}) \right). \quad (3.79)$$

As in the DG scheme, the method is implicit since the discrete gradient depends on θ_{k+1} . The additional δ term introduces extra damping in the parameter dynamics, analogous to the MGD- δ method derived in section 3.4.3. As will be shown in Chapter 4, this may lead to better convergence in terms of time to reach tolerance and runtime.

The DG- δ updates (3.78)-(3.79) also define an implicit nonlinear system. As in the DG case, it is useful to rewrite this system as a fixed-point equation in the variable θ_{k+1} . Rearranging (3.79) gives

$$(1 + \frac{h\gamma}{2})\omega_{k+1} = (1 - \frac{h\gamma}{2})\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}).$$

Hence,

$$\omega_{k+1} = \frac{(2 - h\gamma)\omega_k - 2h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})}{2 + h\gamma}. \quad (3.80)$$

Substituting (3.80) into (3.78) yields

$$\theta_{k+1} = \theta_k + h \left(\frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})}{2 + h\gamma} - \delta \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) \right).$$

Since the discrete gradient depends on θ_{k+1} , this again defines a nonlinear fixed-point problem. The following lemma gives a sufficient condition for this fixed-point problem to be well posed.

Lemma 3.15 (Well-posedness of DG- δ updates (one step)). *Assume that $\nabla_\theta \mathcal{L}$ is L -Lipschitz. Fix $\gamma > 0$, $\delta > 0$, $h > 0$, and suppose $(\theta_k, \omega_k) \in \mathbb{R}^n \times \mathbb{R}^n$ is given. Define*

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta - \theta_k)) ds, \quad m(\theta) := \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)}{2 + h\gamma},$$

and

$$\Phi_{k,\delta}(\theta) := \theta_k + h(m(\theta) - \delta \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)).$$

Let

$$\Lambda_\delta := \frac{Lh^2}{2(2 + h\gamma)} + \frac{Lh\delta}{2}.$$

If $\Lambda_\delta < 1$, then $\Phi_{k,\delta}$ is a contraction on $\mathbb{R}^n$ and therefore admits a unique fixed point $\theta_{k+1} \in \mathbb{R}^n$ satisfying

$$\theta_{k+1} = \Phi_{k,\delta}(\theta_{k+1}).$$

Then $(\theta_{k+1}, \omega_{k+1})$ satisfies the DG- δ update relations (3.78)-(3.79). Moreover, the fixed-point iterations $\theta^{(j+1)} = \Phi_{k,\delta}(\theta^{(j)})$ converge linearly to θ_{k+1} :

$$\|\theta^{(j)} - \theta_{k+1}\| \leq \Lambda_\delta^j \|\theta^{(0)} - \theta_{k+1}\|.$$

Proof. The proof is similar to the proof of Lemma 3.9. □

The contraction condition $\Lambda_\delta < 1$ is equivalent to

$$\frac{Lh^2}{2(2+h\gamma)} + \frac{Lh\delta}{2} < 1.$$

Rearranging gives

$$L(1+\delta\gamma)h^2 + 2(L\delta - \gamma)h - 4 < 0.$$

Hence, the sufficient stepsize bound is

$$h_{\max} = \frac{\gamma - L\delta + \sqrt{(L\delta - \gamma)^2 + 4L(1 + \delta\gamma)}}{L(1 + \delta\gamma)}.$$

Again, the fixed-point representation defines the implementation shown in Algorithm 4.

Algorithm 4 Discrete-Gradient- δ updates ($R = \text{diag}(\delta I_n, \gamma I_n)$)

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, damping $\gamma > 0, \delta > 0$, step size $h > 0$

Require: Outer iterations K , inner iterations J , inner tolerance ε

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ (often $\omega_0 = 0$)

1: **for** $k = 0, 1, \dots, K - 1$ **do**

2: Define the discrete gradient:

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta) := \int_0^1 \nabla_\theta \mathcal{L}(\theta_k + s(\theta - \theta_k)) ds$$

3: Define the map $m(\theta) : \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta)}{2 + h\gamma}$

4: Define the DG- δ fixed-point operator $\Phi_{k,\delta}(\theta) := \theta_k + h(m(\theta) - \delta \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta))$
 {If $\frac{Lh^2}{2(2+h\gamma)} + \frac{Lh\delta}{2} < 1$, then Φ_k is a contraction (lemma 3.15)}

5: Then θ_{k+1} solves the fixed-point equation $\theta_{k+1} = \Phi_{k,\delta}^j(\theta_{k+1})$.

6: **Inner fixed point iteration:**

7: $\theta^{(0)} \leftarrow \theta_k + \frac{2h}{2 + h\gamma} \omega_k$ {initial guess}

8: $\hat{\theta} \leftarrow \theta^{(0)}$

9: **for** $j = 0, 1, \dots, J - 1$ **do**

10: $\hat{\theta} \leftarrow \Phi_{k,\delta}^j(\theta^{(j)})$

11: **if** $\|\hat{\theta} - \theta^{(j)}\| \leq \varepsilon$ **then**

12: **break**

13: **end if**

14: $\theta^{(j+1)} \leftarrow \hat{\theta}$

15: **end for**

16: **Accept:** $\theta_{k+1} \leftarrow \hat{\theta}$

17: $\bar{\nabla}_\theta \mathcal{L}_k \leftarrow \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1})$

18: $m_k \leftarrow \frac{2\omega_k - h \bar{\nabla}_\theta \mathcal{L}_k}{2 + h\gamma}$

19: $\omega_{k+1} \leftarrow 2m_k - \omega_k$

20: **end for**

21: **return** θ_K

3.4.4 Convergence analysis of the DG- δ method

In this section, we state the convergence guarantees for the DG- δ method. Since the proofs are simple modifications of those for the DG method, the proofs are deferred to Appendix C. Again,

since our function $\mathcal{L}$ is L -smooth (2.5) and has compact sub-level sets (2.6), the function $\mathcal{L}$ is bounded below and we have $\inf_{\theta} \mathcal{L}(\theta) = \mathcal{L}_{\inf} > -\infty$.

Theorem 3.16 (*Best-iterate rate for DG- δ*). *Under the assumption that $\mathcal{L}$ is L -smooth and has compact sublevel sets, let $\{(\theta_k, \omega_k)\}_{k \geq 0}$ be generated by the DG- δ method with parameters $\delta > 0$, $\gamma > 0$, and $h > 0$. Then, for every $K \geq 0$,*

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\delta h(K+1)}. \quad (3.81)$$

In particular,

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})\|^2 = \mathcal{O}(1/K).$$

Corollary 3.17 (*Best-iterate rate for the true gradient for DG- δ*). *Under the assumptions of Theorem 3.16, for every $K \geq 0$, the true gradient satisfies*

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{C(\gamma, \delta, L, h)}{K+1}, \quad (3.82)$$

where

$$C(\gamma, \delta, L, h) := \left(\frac{2 \left(1 + \frac{L\delta h}{2}\right)^2}{\delta h} + \frac{L^2 h}{2\gamma} \right) (\mathcal{H}_0 - \mathcal{L}_{\inf}). \quad (3.83)$$

In particular,

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 = \mathcal{O}(1/K).$$

3.4.5 Limit points of the DG schemes

We now discuss the convergence of the parameter sequence itself. The previous results show that both the discrete gradients $\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})$ and the true gradients $\nabla_{\theta} \mathcal{L}(\theta_k)$ converge to zero, and that consecutive parameter updates become small. Since the sequence $\{\theta_k\}_{k \geq 0}$ is bounded, it admits convergent subsequences. We show that every accumulation point of these iterates is a stationary point of the objective.

Corollary 3.18 (*Subsequence convergence of the parameters*). *Assume that the hypotheses of Lemma 3.12 and Corollary 3.11 hold. Then there exists a subsequence $\{\theta_{k_j}\}_{j \geq 0}$ and a point $\bar{\theta} \in \mathbb{R}^n$ such that*

$$\theta_{k_j} \rightarrow \bar{\theta}.$$

Moreover,

$$\theta_{k_j+1} \rightarrow \bar{\theta}.$$

Proof. By Lemma 3.12, the sequence $\{\theta_k\}_{k \geq 0}$ is bounded. Hence, by the Bolzano–Weierstrass theorem, it admits a convergent subsequence $\{\theta_{k_j}\}_{j \geq 0}$ with

$$\theta_{k_j} \rightarrow \bar{\theta}.$$

From Corollary 3.11,

$$\|\theta_{k+1} - \theta_k\| \rightarrow 0.$$

Therefore,

$$\|\theta_{k_j+1} - \bar{\theta}\| \leq \|\theta_{k_j+1} - \theta_{k_j}\| + \|\theta_{k_j} - \bar{\theta}\| \rightarrow 0.$$

and we get $\theta_{k_j+1} \rightarrow \bar{\theta}$. □

Theorem 3.19 (*Stationarity of limit points*). *Let $\{\theta_k\}_{k \geq 0}$ be generated by the DG or DG- δ method. From Corollary 3.14 and 3.17 we have,*

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \rightarrow 0.$$

Then every limit point $\bar{\theta}$ of $\{\theta_k\}_{k \geq 0}$ is a stationary point of $\mathcal{L}$, i.e.,

$$\nabla_{\theta} \mathcal{L}(\bar{\theta}) = 0.$$

Proof. Let $\bar{\theta}$ be any limit point of $\{\theta_k\}_{k \geq 0}$. Then there exists a subsequence $\{\theta_{k_j}\}_{j \geq 0}$ such that

$$\theta_{k_j} \rightarrow \bar{\theta}.$$

Since

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \rightarrow 0,$$

we also have

$$\nabla_{\theta} \mathcal{L}(\theta_{k_j}) \rightarrow 0.$$

By continuity of $\nabla_{\theta} \mathcal{L}$,

$$\nabla_{\theta} \mathcal{L}(\theta_{k_j}) \rightarrow \nabla_{\theta} \mathcal{L}(\bar{\theta}).$$

Hence,

$$\nabla_{\theta} \mathcal{L}(\bar{\theta}) = 0.$$

Thus, every limit point is stationary. □

3.5 Semi-Implicit Discrete Gradient Method

Although the discrete gradient (DG) scheme preserves the dissipation structure exactly, each iteration requires solving a nonlinear fixed-point problem (3.42), which can be computationally expensive. To reduce this cost, we introduce a semi-implicit approximation obtained by linearizing the gradient inside the discrete gradient.

Recall that the DG updates (3.38)-(3.39) can be written as

$$\omega_{k+1} = \frac{2 - h\gamma}{2 + h\gamma} \omega_k - \frac{2h}{2 + h\gamma} \bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}), \quad (3.84)$$

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}). \quad (3.85)$$

To avoid solving the nonlinear discrete-gradient equation, we approximate the gradient along the segment between θ_k and θ_{k+1} . Using a first-order Taylor approximation around θ_k , we write

$$\nabla_\theta \mathcal{L}(\theta_k + s(\theta_{k+1} - \theta_k)) \approx \nabla_\theta \mathcal{L}(\theta_k) + s \nabla_\theta^2 \mathcal{L}(\theta_k)(\theta_{k+1} - \theta_k). \quad (3.86)$$

Integrating with respect to $s \in [0, 1]$ gives

$$\bar{\nabla}_\theta \mathcal{L}(\theta_k, \theta_{k+1}) \approx \nabla_\theta \mathcal{L}(\theta_k) + \frac{1}{2} \nabla_\theta^2 \mathcal{L}(\theta_k)(\theta_{k+1} - \theta_k). \quad (3.87)$$

Using

$$\theta_{k+1} - \theta_k = \frac{h}{2}(\omega_k + \omega_{k+1}), \quad (3.88)$$

and substituting (3.87) and (3.88) into (3.84), we obtain

$$\left(\left(1 + \frac{h\gamma}{2} \right) I_n + \frac{h^2}{4} \nabla_\theta^2 \mathcal{L}(\theta_k) \right) \omega_{k+1} = \left(\left(1 - \frac{h\gamma}{2} \right) I_n - \frac{h^2}{4} \nabla_\theta^2 \mathcal{L}(\theta_k) \right) \omega_k - h \nabla_\theta \mathcal{L}(\theta_k). \quad (3.89)$$

The system (3.89) is linear, but it still requires the Hessian $\nabla_\theta^2 \mathcal{L}(\theta_k)$, which is expensive to compute in large-scale problems. To avoid forming the Hessian explicitly, we replace it by a quasi-Newton approximation B_k . In particular, L-BFGS constructs B_k from recent iterate and gradient differences, called curvature pairs,

$$s_k = \theta_{k+1} - \theta_k, \quad y_k = \nabla_\theta \mathcal{L}(\theta_{k+1}) - \nabla_\theta \mathcal{L}(\theta_k),$$

and applies a limited-memory BFGS update using only the most recent pairs; see Nocedal and Wright [17]. Thus, we use

$$B_k \approx \nabla_\theta^2 \mathcal{L}(\theta_k),$$

rather than forming the exact Hessian.

Replacing $\nabla_\theta^2 \mathcal{L}(\theta_k)$ by B_k in (3.89) gives

$$\left(\left(1 + \frac{h\gamma}{2} \right) I_n + \frac{h^2}{4} B_k \right) \omega_{k+1} = \left(\left(1 - \frac{h\gamma}{2} \right) I_n - \frac{h^2}{4} B_k \right) \omega_k - h \nabla_\theta \mathcal{L}(\theta_k). \quad (3.90)$$

The parameter update is then

$$\theta_{k+1} = \theta_k + \frac{h}{2} (\omega_k + \omega_{k+1}). \quad (3.91)$$

Equations (3.90)–(3.91) define the semi-implicit discrete-gradient method.

It is well known that BFGS-type updates preserve symmetry and positive definiteness provided the curvature condition $s_k^\top y_k > 0$ is enforced; see Nocedal and Wright [17]. However, this property alone does not guarantee uniform spectral bounds on B_k . To ensure the well-posedness of the SIDG iteration and derive the convergence analysis, we impose the following assumption.

Assumption 3.20. For each $k \geq 0$, the matrix $B_k \in \mathbb{R}^{n \times n}$ is symmetric positive definite and satisfies

$$m_B I_n \preceq B_k \preceq M_B I_n$$

for some constants $0 < m_B \leq M_B < \infty$ independent of k .

This assumption ensures that the curvature model used in the semi-implicit update remains uniformly positive definite along the iterates.

Lemma 3.21 (Well-posedness of the SIDG updates). *Suppose Assumption 3.20 holds and define*

$$A_k := \left(1 + \frac{h\gamma}{2} \right) I_n + \frac{h^2}{4} B_k.$$

Then A_k is symmetric positive definite for every $k \geq 0$. Consequently, the linear system (3.90) admits a unique solution ω_{k+1} , and the update (3.91) uniquely defines θ_{k+1} .

Proof. Since B_k is symmetric, A_k is symmetric. For any $v \neq 0$,

$$v^\top A_k v = \left(1 + \frac{h\gamma}{2} \right) \|v\|^2 + \frac{h^2}{4} v^\top B_k v.$$

Using Assumption 3.20, we have

$$v^\top B_k v \geq m_B \|v\|^2.$$

Therefore,

$$v^\top A_k v \geq \left[\left(1 + \frac{h\gamma}{2} \right) + \frac{h^2}{4} m_B \right] \|v\|^2 > 0.$$

Thus, A_k is symmetric positive definite and invertible. Hence the linear system (3.90) admits a unique solution ω_{k+1} . The update (3.91) then uniquely defines θ_{k+1} . $\square$

In the DG method, the implicit update is defined by a nonlinear fixed-point equation, which is typically solved using fixed-point iteration. In contrast, the SIDG method replaces this nonlinear problem with a linear system obtained via a linearization of the discrete gradient. This linear system is then solved efficiently using iterative solvers such as conjugate gradient. This significantly reduces the computational cost (see Chapter 4) while retaining key stability properties of the original method.

The resulting scheme is given in Algorithm 5.

Algorithm 5 Semi-Implicit Discrete-Gradient updates ($R = \text{diag}(O_n, \gamma I_n)$)

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, damping $\gamma > 0$, step size $h > 0$

Require: Outer iterations K , CG iterations J , CG tolerance ε_{cg}

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ (often $\omega_0 = 0$)

Require: Initial Hessian approximation $B_0 \succ 0$

1: **for** $k = 0, 1, \dots, K - 1$ **do**

2: Compute the gradient: $g_k \leftarrow \nabla_{\theta} \mathcal{L}(\theta_k)$.

3: Define the linear operator A_k and right-hand side b_k :

$$A_k := \left(1 + \frac{h\gamma}{2}\right) I_n + \frac{h^2}{4} B_k.$$

$$b_k := \left[\left(1 - \frac{h\gamma}{2}\right) I_n - \frac{h^2}{4} B_k \right] \omega_k - h g_k.$$

4: **Solve the linear system** $A_k \omega_{k+1} = b_k$ **using conjugate gradient.**

5: $\omega^{(0)} \leftarrow \omega_k$ {initial guess}

6: **for** $j = 0, 1, \dots, J - 1$ **do**

7: Perform conjugate gradient for $A_k \omega = b_k$ starting from $\omega^{(j)}$

8: **if** $\|A_k \omega^{(j+1)} - b_k\| \leq \varepsilon_{\text{cg}}$ **then**

9: **break**

10: **end if**

11: **end for**

12: **Accept:** $\omega_{k+1} \leftarrow \omega^{(j+1)}$.

13: Update the parameters:

$$\theta_{k+1} \leftarrow \theta_k + \frac{h}{2} (\omega_k + \omega_{k+1}).$$

14: Compute the new gradient:

$$g_{k+1} \leftarrow \nabla_{\theta} \mathcal{L}(\theta_{k+1}).$$

15: Form the curvature pair: $s_k \leftarrow \theta_{k+1} - \theta_k$, $y_k \leftarrow g_{k+1} - g_k$.

16: **if** $s_k^T y_k > 0$ **then**

17: Store the curvature pair (s_k, y_k) in the L-BFGS memory.

18: Update the Hessian approximation B_{k+1} using the stored curvature pairs.

19: **else**

20: Skip the curvature update and set $B_{k+1} = B_k$.

21: **end if**

22: **end for**

23: **return** θ_K

3.5.1 Convergence analysis of the SIDG method

In this section, we analyze the convergence properties of the semi-implicit discrete gradient (SIDG) method (3.90)-(3.91) where the linear system is solved *exactly* at each iteration. In practice, we compute ω_{k+1} approximately using the conjugate gradient method. Extending the analysis to this inexact setting would require controlling the residual errors, which we do not pursue here.

As in the DG analysis, we work under the standard assumptions (2.5)-(2.6). In addition, we assume uniform spectral bounds on the Hessian approximation B_k , as stated in Assumption 3.20.

The SIDG iteration (3.90) can be written equivalently as

$$(I_n + \beta B_k)\omega_{k+1} = (\alpha I_n - \beta B_k)\omega_k - \mu \nabla_{\theta} \mathcal{L}(\theta_k), \quad (3.92)$$

where

$$\alpha := \frac{2 - h\gamma}{2 + h\gamma}, \quad \beta := \frac{h^2}{2(2 + h\gamma)}, \quad \mu := \frac{2h}{2 + h\gamma}. \quad (3.93)$$

The parameter update is

$$\theta_{k+1} = \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}). \quad (3.94)$$

For convenience, define

$$m_k := \omega_k + \omega_{k+1}. \quad (3.95)$$

Then (3.94) becomes

$$\theta_{k+1} - \theta_k = \frac{h}{2}m_k. \quad (3.96)$$

We first note that Lemmas 3.22-3.29 develop the intermediate results needed to establish Theorem 3.30. We begin with a convenient reformulation of the SIDG update.

Lemma 3.22 (*Equivalent form of the velocity update*). *Let $\{(\theta_k, \omega_k)\}_{k \geq 0}$ be generated by (3.92)-(3.94). Then, for every $k \geq 0$,*

$$\omega_{k+1} - \alpha\omega_k + \beta B_k m_k + \mu \nabla_{\theta} \mathcal{L}(\theta_k) = 0. \quad (3.97)$$

Equivalently,

$$\mu \nabla_{\theta} \mathcal{L}(\theta_k) = (1 + \alpha)\omega_k - (I_n + \beta B_k)m_k. \quad (3.98)$$

Proof. Follows from rearranging (3.92) and using $m_k = \omega_k + \omega_{k+1}$. $\square$

Lemma 3.23 (*Useful identity for the gradient*). *For every $k \geq 0$,*

$$\mu \nabla_{\theta} \mathcal{L}(\theta_k)^{\top} m_k = -\frac{1 + \alpha}{2}(\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - \frac{1 - \alpha}{2}\|m_k\|^2 - \beta m_k^{\top} B_k m_k. \quad (3.99)$$

Proof. Take the inner product of (3.97) with m_k :

$$(\omega_{k+1} - \alpha\omega_k)^\top m_k + \beta m_k^\top B_k m_k + \mu \nabla_\theta \mathcal{L}(\theta_k)^\top m_k = 0. \quad (3.100)$$

Now

$$(\omega_{k+1} - \alpha\omega_k)^\top m_k = (\omega_{k+1} - \alpha\omega_k)^\top (\omega_k + \omega_{k+1}),$$

so

$$\begin{aligned} (\omega_{k+1} - \alpha\omega_k)^\top (\omega_k + \omega_{k+1}) &= \omega_{k+1}^\top \omega_k + \|\omega_{k+1}\|^2 - \alpha\|\omega_k\|^2 - \alpha\omega_k^\top \omega_{k+1} \\ &= \|\omega_{k+1}\|^2 - \alpha\|\omega_k\|^2 + (1 - \alpha)\omega_k^\top \omega_{k+1}. \end{aligned}$$

Also,

$$\|m_k\|^2 = \|\omega_k + \omega_{k+1}\|^2 = \|\omega_k\|^2 + \|\omega_{k+1}\|^2 + 2\omega_k^\top \omega_{k+1},$$

hence,

$$\omega_k^\top \omega_{k+1} = \frac{1}{2} \left(\|m_k\|^2 - \|\omega_k\|^2 - \|\omega_{k+1}\|^2 \right).$$

Substituting gives

$$(\omega_{k+1} - \alpha\omega_k)^\top m_k = \frac{1 + \alpha}{2} (\|\omega_{k+1}\|^2 - \|\omega_k\|^2) + \frac{1 - \alpha}{2} \|m_k\|^2.$$

Plugging this into (3.100) and rearranging yields (3.99). $\square$

We next derive the discrete energy law.

Lemma 3.24 (*One-step descent inequality*). *Suppose $\mathcal{L}$ is L -smooth (2.5) and assumption 3.20 holds. Then, for every $k \geq 0$,*

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) - \frac{1}{2} (\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - \frac{h\gamma}{4} \|m_k\|^2 - \frac{h^2}{8} m_k^\top B_k m_k + \frac{Lh^2}{8} \|m_k\|^2. \quad (3.101)$$

Proof. Since $\mathcal{L}$ is L -smooth, the descent lemma 2.8 gives

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \nabla_\theta \mathcal{L}(\theta_k)^\top (\theta_{k+1} - \theta_k) + \frac{L}{2} \|\theta_{k+1} - \theta_k\|^2. \quad (3.102)$$

Using (3.96),

$$\theta_{k+1} - \theta_k = \frac{h}{2} m_k,$$

so

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \frac{h}{2} \nabla_\theta \mathcal{L}(\theta_k)^\top m_k + \frac{Lh^2}{8} \|m_k\|^2. \quad (3.103)$$

Now Lemma 3.23 yields

$$\mu \nabla_{\theta} \mathcal{L}(\theta_k)^{\top} m_k = -\frac{1+\alpha}{2} (\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - \frac{1-\alpha}{2} \|m_k\|^2 - \beta m_k^{\top} B_k m_k.$$

Dividing by μ , multiplying by $h/2$, and simplifying the coefficients

$$\frac{h(1+\alpha)}{2(2\mu)} = \frac{1}{2}, \quad \frac{h(1-\alpha)}{2(2\mu)} = \frac{h\gamma}{4}, \quad \frac{h\beta}{2\mu} = \frac{h^2}{8},$$

we obtain

$$\frac{h}{2} \nabla_{\theta} \mathcal{L}(\theta_k)^{\top} m_k = -\frac{1}{2} (\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - \frac{h\gamma}{4} \|m_k\|^2 - \frac{h^2}{8} m_k^{\top} B_k m_k.$$

Substituting into (3.103) gives (3.101). $\square$

Theorem 3.25 (*Energy decay for SIDG*). Suppose the iterates are generated by the exact SIDG updates (3.92)-(3.94). Suppose $\mathcal{L}$ is L -smooth and has compact sublevel sets, and Assumption 3.20 holds. Define

$$C(\gamma, L, m_B, h) := \frac{h\gamma}{4} - \frac{h^2}{8} (L - m_B). \quad (3.104)$$

If $C(\gamma, L, m_B, h) > 0$, then for every $k \geq 0$,

$$\mathcal{H}_{k+1} \leq \mathcal{H}_k - C(\gamma, L, m_B, h) \|m_k\|^2. \quad (3.105)$$

Consequently, the sequence $\{\mathcal{H}_k\}_{k \geq 0}$ is nonincreasing. Moreover, for every $K \geq 0$,

$$\frac{1}{K+1} \sum_{k=0}^K \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{C(\gamma, L, m_B, h)(K+1)}. \quad (3.106)$$

and therefore

$$\sum_{k=0}^{\infty} \|m_k\|^2 < \infty, \quad m_k \rightarrow 0.$$

In addition,

$$\min_{0 \leq k \leq K} \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{C(\gamma, L, m_B, h)(K+1)}. \quad (3.107)$$

Finally,

$$\frac{1}{K+1} \sum_{k=0}^K \|\theta_{k+1} - \theta_k\|^2 \leq \frac{h^2(\mathcal{H}_0 - \mathcal{L}_{\inf})}{4C(\gamma, L, m_B, h)(K+1)}, \quad (3.108)$$

and

$$\min_{0 \leq k \leq K} \|\theta_{k+1} - \theta_k\|^2 \leq \frac{h^2(\mathcal{H}_0 - \mathcal{L}_{\inf})}{4C(\gamma, L, m_B, h)(K+1)}. \quad (3.109)$$

Proof. By Assumption 3.20,

$$m_k^\top B_k m_k \geq m_B \|m_k\|^2.$$

Applying this to Lemma 3.24, we obtain

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) - \frac{1}{2}(\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - \left[\frac{h\gamma}{4} - \frac{h^2}{8}(L - m_B) \right] \|m_k\|^2.$$

By the definition of $C(\gamma, L, m_B, h)$,

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) - \frac{1}{2}(\|\omega_{k+1}\|^2 - \|\omega_k\|^2) - C(\gamma, L, m_B, h) \|m_k\|^2.$$

Adding $\frac{1}{2}\|\omega_{k+1}\|^2$ to both sides gives

$$\mathcal{H}_{k+1} \leq \mathcal{H}_k - C(\gamma, L, m_B, h) \|m_k\|^2,$$

which proves (3.105). Now, summing from $k = 0$ to K yields

$$C(\gamma, L, m_B, h) \sum_{k=0}^K \|m_k\|^2 \leq \mathcal{H}_0 - \mathcal{H}_{K+1}.$$

Since

$$\mathcal{H}_{K+1} = \mathcal{L}(\theta_{K+1}) + \frac{1}{2}\|\omega_{K+1}\|^2 \geq \mathcal{L}_{\inf},$$

we obtain

$$C(\gamma, L, m_B, h) \sum_{k=0}^K \|m_k\|^2 \leq \mathcal{H}_0 - \mathcal{L}_{\inf},$$

which implies (3.106). This shows that $\sum_{k=0}^\infty \|m_k\|^2 < \infty$, hence, $m_k \rightarrow 0$. The best-iterate bound (3.107) follows from

$$\min_{0 \leq k \leq K} \|m_k\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|m_k\|^2.$$

Finally, by (3.96),

$$\|\theta_{k+1} - \theta_k\|^2 = \frac{h^2}{4} \|m_k\|^2,$$

so (3.108) and (3.109) follow immediately from (3.106) and (3.107). $\square$

Corollary 3.26 (*Explicit step-size condition*). *The condition $C(\gamma, L, m_B, h) > 0$ is equivalent to*

$$\frac{h\gamma}{4} > \frac{h^2}{8}(L - m_B).$$

Therefore:

1. if $L > m_B$, then it is sufficient to choose

$$0 < h < \frac{2\gamma}{L - m_B};$$

2. if $L \leq m_B$, then $C(\gamma, L, m_B, h) > 0$ for every $h > 0$.

We now derive several bounds that will be used in the gradient-rate theorem.

Lemma 3.27 (*Gradient bound in terms of ω_k and m_k*). For every $k \geq 0$,

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \leq \frac{2}{h} \|\omega_k\| + \left(\frac{2 + h\gamma}{2h} + \frac{hM_B}{4} \right) \|m_k\|. \quad (3.110)$$

Proof. Follow from taking norm on both sides of (3.98) and using $B_k \preceq M_B I_n$ and (3.93). $\square$

Lemma 3.28. For every $k \geq 0$,

$$\|\omega_{k+1} - \omega_k\| \leq \left(\frac{2 + h\gamma}{4} + \frac{Lh^2}{4} + \frac{M_B h^2}{8} \right) \|m_k\| + \left(\frac{2 + h\gamma}{4} + \frac{M_B h^2}{8} \right) \|m_{k+1}\|. \quad (3.111)$$

Proof. Write (3.98) at indices k and $k + 1$:

$$\mu \nabla_{\theta} \mathcal{L}(\theta_k) = (1 + \alpha) \omega_k - (I_n + \beta B_k) m_k,$$

$$\mu \nabla_{\theta} \mathcal{L}(\theta_{k+1}) = (1 + \alpha) \omega_{k+1} - (I_n + \beta B_{k+1}) m_{k+1}.$$

Subtracting gives

$$(1 + \alpha)(\omega_{k+1} - \omega_k) = \mu(\nabla_{\theta} \mathcal{L}(\theta_{k+1}) - \nabla_{\theta} \mathcal{L}(\theta_k)) + (I_n + \beta B_{k+1}) m_{k+1} - (I_n + \beta B_k) m_k.$$

Hence,

$$(1 + \alpha) \|\omega_{k+1} - \omega_k\| \leq \mu \|\nabla_{\theta} \mathcal{L}(\theta_{k+1}) - \nabla_{\theta} \mathcal{L}(\theta_k)\| + \|I_n + \beta B_{k+1}\| \|m_{k+1}\| + \|I_n + \beta B_k\| \|m_k\|. \quad (3.112)$$

Since $\mathcal{L}$ is L -smooth,

$$\|\nabla_{\theta} \mathcal{L}(\theta_{k+1}) - \nabla_{\theta} \mathcal{L}(\theta_k)\| \leq L \|\theta_{k+1} - \theta_k\| = \frac{Lh}{2} \|m_k\|,$$

where we used (3.96). Also,

$$\|I_n + \beta B_k\| \leq 1 + \beta M_B.$$

Substituting into (3.112),

$$(1 + \alpha) \|\omega_{k+1} - \omega_k\| \leq \left(\frac{\mu L h}{2} + 1 + \beta M_B \right) \|m_k\| + (1 + \beta M_B) \|m_{k+1}\|.$$

Therefore,

$$\|\omega_{k+1} - \omega_k\| \leq A(h) \|m_k\| + B(h) \|m_{k+1}\|,$$

where

$$A(h) := \frac{\frac{\mu L h}{2} + 1 + \beta M_B}{1 + \alpha}, \quad B(h) := \frac{1 + \beta M_B}{1 + \alpha}.$$

Simplifying constants using (3.93), we obtain

$$A(h) = \frac{2 + h\gamma}{4} + \frac{Lh^2}{4} + \frac{M_B h^2}{8}, \quad B(h) = \frac{2 + h\gamma}{4} + \frac{M_B h^2}{8},$$

which proves (3.111). $\square$

Corollary 3.29 (*Bound on the velocity*). *For every $k \geq 0$,*

$$\|\omega_k\|^2 \leq \left(\frac{1}{2} + \left(\frac{2 + h\gamma}{4} + \frac{Lh^2}{4} + \frac{M_B h^2}{8} \right)^2 \right) \|m_k\|^2 + \left(\frac{2 + h\gamma}{4} + \frac{M_B h^2}{8} \right)^2 \|m_{k+1}\|^2. \quad (3.113)$$

Proof. Directly follows $m_k = \omega_k + \omega_{k+1}$ which gives us $\omega_k = \frac{1}{2}(m_k - (\omega_{k+1} - \omega_k))$, taking norm on both sides and using Lemma 3.28. $\square$

We can now prove the main nonconvex gradient-rate estimate.

Theorem 3.30 (*Best-iterate rate for the gradient*). *Suppose the assumptions of Theorem 3.25 hold, and assume $C(\gamma, L, m_B, h) > 0$. Then for every $K \geq 1$,*

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{C(h, \gamma, L, m_B, M_B)}{K + 1} (\mathcal{H}_0 - \mathcal{L}_{\inf}), \quad (3.114)$$

where

$$C(h, \gamma, L, m_B, M_B) := \frac{1}{C(\gamma, L, m_B, h)} \left[\frac{8}{h^2} \left(\frac{1}{2} + \left(\frac{2 + h\gamma}{4} + \frac{Lh^2}{4} + \frac{M_B h^2}{8} \right)^2 \right. \right. \quad (3.115) \\ \left. \left. + \left(\frac{2 + h\gamma}{4} + \frac{M_B h^2}{8} \right)^2 \right) + 2 \left(\frac{2 + h\gamma}{2h} + \frac{M_B h}{4} \right)^2 \right].$$

Proof. We prove the theorem in two steps.

Step 1: Square sum of $\{\omega_k\}_{k \geq 0}$. Summing the estimate of Lemma 3.29 over $k \geq 0$, and using

$$\sum_{k=0}^{\infty} \|m_{k+1}\|^2 \leq \sum_{k=0}^{\infty} \|m_k\|^2,$$

we obtain

$$\sum_{k=0}^{\infty} \|\omega_k\|^2 \leq \left[\frac{1}{2} + \left(\frac{2+h\gamma}{4} + \frac{Lh^2}{4} + \frac{M_B h^2}{8} \right)^2 + \left(\frac{2+h\gamma}{4} + \frac{M_B h^2}{8} \right)^2 \right] \sum_{k=0}^{\infty} \|m_k\|^2.$$

By Theorem 3.25,

$$\sum_{k=0}^{\infty} \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{C(\gamma, L, m_B, h)},$$

hence,

$$\sum_{k=0}^{\infty} \|\omega_k\|^2 < \infty.$$

In particular, $\omega_k \rightarrow 0$.

Step 2: Square sum of $\{\nabla_{\theta} \mathcal{L}(\theta_k)\}$. By Lemma 3.27,

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \leq \frac{2}{h} \|\omega_k\| + \left(\frac{2+h\gamma}{2h} + \frac{hM_B}{4} \right) \|m_k\|.$$

Squaring and using $(a+b)^2 \leq 2a^2 + 2b^2$,

$$\|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{8}{h^2} \|\omega_k\|^2 + 2 \left(\frac{2+h\gamma}{2h} + \frac{hM_B}{4} \right)^2 \|m_k\|^2.$$

Summing over $k \geq 0$,

$$\sum_{k=0}^{\infty} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{8}{h^2} \sum_{k=0}^{\infty} \|\omega_k\|^2 + 2 \left(\frac{2+h\gamma}{2h} + \frac{hM_B}{4} \right)^2 \sum_{k=0}^{\infty} \|m_k\|^2.$$

Substituting the Step 1 bound and the estimate from Theorem 3.25, we obtain

$$\sum_{k=0}^{\infty} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq C(h, \gamma, L, m_B, M_B)(\mathcal{H}_0 - \mathcal{L}_{\inf}),$$

where $C(h, \gamma, L, m_B, M_B)$ is given by (3.115). In particular,

$$\sum_{k=0}^{\infty} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 < \infty,$$

and therefore $\nabla_{\theta} \mathcal{L}(\theta_k) \rightarrow 0$.

Step 3: Best-iterate $\mathcal{O}(1/K)$ bound. For every $K \geq 1$,

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2.$$

Using the previous estimate,

$$\sum_{k=0}^K \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \sum_{k=0}^{\infty} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq C(h, \gamma, L, m_B, M_B)(\mathcal{H}_0 - \mathcal{L}_{\inf}),$$

and hence,

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{C(h, \gamma, L, m_B, M_B)}{K+1} (\mathcal{H}_0 - \mathcal{L}_{\inf}).$$

This proves (3.114). □

Remark 3.31 (Step-size restriction comparison to MGD). From Remark B.5, the step-size restriction for MGD is

$$0 < h < \frac{2\gamma}{L}.$$

By Corollary 3.26, the SIDG method requires

$$0 < h < \frac{2\gamma}{L - m_B}, \quad \text{when } L > m_B,$$

while if $L \leq m_B$, there is no step-size restriction and the energy decay condition holds for every $h > 0$. Hence, whenever $L > m_B$, the SIDG admissible step size is less restrictive than that of MGD, since

$$L - m_B < L \implies \frac{2\gamma}{L - m_B} > \frac{2\gamma}{L}.$$

Thus, SIDG allows a larger range of stable step sizes than MGD under the same smoothness and damping parameters. In the regime $L \leq m_B$, SIDG is even more robust, as the dissipation estimate holds for arbitrary step size.

Under the compact sub-level set assumption (2.6), and since $\{\mathcal{H}_k\}$ is non increasing (Theorem 3.25), the sequence $\{\theta_k\}_{k \geq 0}$ remains in a compact sub-level set and is therefore bounded.

Moreover, by Theorem 3.25, we have $m_k \rightarrow 0$, and hence from (3.96),

$$\|\theta_{k+1} - \theta_k\| \rightarrow 0.$$

Consequently, $\{\theta_k\}_{k \geq 0}$ admits accumulation points. Furthermore, by Theorem 3.30, we have $\|\nabla_{\theta} \mathcal{L}(\theta_k)\| \rightarrow 0$, and therefore every such limit point is a stationary point of $\mathcal{L}$.

3.5.2 Semi-Implicit Discrete-Gradient- δ updates

We now consider the case

$$R = \text{diag}(\delta I_n, \gamma I_n) \succ 0, \quad \delta > 0, \quad \gamma > 0.$$

As in the SIDG construction, we replace the discrete gradient by the linearized approximation

$$\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) \approx \nabla_{\theta} \mathcal{L}(\theta_k) + \frac{1}{2} \nabla_{\theta}^2 \mathcal{L}(\theta_k) (\theta_{k+1} - \theta_k). \quad (3.116)$$

For the DG- δ scheme, the parameter update is

$$\theta_{k+1} - \theta_k = \frac{h}{2} (\omega_k + \omega_{k+1}) - h\delta \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}). \quad (3.117)$$

Substituting (3.117) into (3.116) gives

$$\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) \approx \nabla_{\theta} \mathcal{L}(\theta_k) + \frac{1}{2} \nabla_{\theta}^2 \mathcal{L}(\theta_k) \left(\frac{h}{2} (\omega_k + \omega_{k+1}) - h\delta \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) \right).$$

Therefore,

$$\left(I_n + \frac{h\delta}{2} \nabla_{\theta}^2 \mathcal{L}(\theta_k) \right) \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) = \nabla_{\theta} \mathcal{L}(\theta_k) + \frac{h}{4} \nabla_{\theta}^2 \mathcal{L}(\theta_k) (\omega_k + \omega_{k+1}). \quad (3.118)$$

Substituting the same linearized approximation into the DG- δ momentum update yields

$$\begin{aligned} \left(\left(1 + \frac{h\gamma}{2} \right) I_n + \left(\left(1 + \frac{h\gamma}{2} \right) \frac{h\delta}{2} + \frac{h^2}{4} \right) \nabla_{\theta}^2 \mathcal{L}(\theta_k) \right) \omega_{k+1} &= \left(\left(1 - \frac{h\gamma}{2} \right) I_n \right. \\ &\quad + \left(\left(1 - \frac{h\gamma}{2} \right) \frac{h\delta}{2} - \frac{h^2}{4} \right) \nabla_{\theta}^2 \mathcal{L}(\theta_k) \Big) \omega_k \\ &\quad \left. - h \nabla_{\theta} \mathcal{L}(\theta_k) \right). \end{aligned} \quad (3.119)$$

As before, computing the exact Hessian is expensive in large-scale problems. We therefore replace $\nabla_{\theta}^2 \mathcal{L}(\theta_k)$ by the quasi-Newton Hessian approximation B_k . This gives the practical SIDG- δ momentum system

$$\begin{aligned} \left(\left(1 + \frac{h\gamma}{2} \right) I_n + \left(\left(1 + \frac{h\gamma}{2} \right) \frac{h\delta}{2} + \frac{h^2}{4} \right) B_k \right) \omega_{k+1} = & \left(\left(1 - \frac{h\gamma}{2} \right) I_n \right. \\ & + \left(\left(1 - \frac{h\gamma}{2} \right) \frac{h\delta}{2} - \frac{h^2}{4} \right) B_k \right) \omega_k \\ & - h \nabla_{\theta} \mathcal{L}(\theta_k). \end{aligned} \quad (3.120)$$

Similarly, the corresponding approximation of (3.118) is

$$\left(I_n + \frac{h\delta}{2} B_k \right) \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) = \nabla_{\theta} \mathcal{L}(\theta_k) + \frac{h}{4} B_k (\omega_k + \omega_{k+1}). \quad (3.121)$$

Finally, the parameter update is

$$\theta_{k+1} = \theta_k + \frac{h}{2} (\omega_k + \omega_{k+1}) - h\delta \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}). \quad (3.122)$$

Equations (3.120)–(3.122) define the semi-implicit discrete-gradient- δ method.

Lemma 3.32 (Well-posedness of the SIDG- δ updates). *Suppose Assumption 3.20 holds and define*

$$A_k := \left(1 + \frac{h\gamma}{2} \right) I_n + \left(\left(1 + \frac{h\gamma}{2} \right) \frac{h\delta}{2} + \frac{h^2}{4} \right) B_k,$$

and

$$C_k := I_n + \frac{h\delta}{2} B_k.$$

Then A_k and C_k are symmetric positive definite for every $k \geq 0$. Consequently, the linear systems (3.120) and (3.121) admit unique solutions ω_{k+1} and $\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})$, respectively. Hence, the update (3.122) uniquely defines θ_{k+1} .

Proof. Since B_k is symmetric by Assumption 3.20, both A_k and C_k are symmetric. Let $v \in \mathbb{R}^n$ with $v \neq 0$. Then

$$v^{\top} A_k v = \left(1 + \frac{h\gamma}{2} \right) \|v\|^2 + \left(\left(1 + \frac{h\gamma}{2} \right) \frac{h\delta}{2} + \frac{h^2}{4} \right) v^{\top} B_k v.$$

Using Assumption 3.20,

$$v^{\top} B_k v \geq m_B \|v\|^2.$$

Hence,

$$v^\top A_k v \geq \left[\left(1 + \frac{h\gamma}{2}\right) + \left(\left(1 + \frac{h\gamma}{2}\right) \frac{h\delta}{2} + \frac{h^2}{4} \right) m_B \right] \|v\|^2 > 0.$$

Therefore A_k is symmetric positive definite.

Similarly,

$$v^\top C_k v = \|v\|^2 + \frac{h\delta}{2} v^\top B_k v \geq \left(1 + \frac{h\delta}{2} m_B\right) \|v\|^2 > 0.$$

Thus C_k is also symmetric positive definite. Hence both linear systems (3.120) and (3.121) admit unique solutions. The update (3.122) then uniquely defines θ_{k+1} . $\square$

A rigorous convergence analysis for the SIDG- δ scheme is not provided in this work. In Chapter 4, we present numerical experiments that use SIDG- δ . Again, the resulting scheme is given in Algorithm 6.

Algorithm 6 Semi-Implicit Discrete-Gradient- δ updates ($R = \text{diag}(\delta I_n, \gamma I_n)$)

Require: Loss $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}$, damping $\gamma > 0$, additional damping $\delta > 0$, step size $h > 0$

Require: Outer iterations K , CG iterations J , CG tolerance ε_{cg}

Require: Initial state $(\theta_0, \omega_0) \in \mathbb{R}^n \times \mathbb{R}^n$ and Hessian approximation $B_0 \succ 0$

- 1: **for** $k = 0, 1, \dots, K - 1$ **do**
- 2: Compute the gradient: $g_k \leftarrow \nabla_{\theta} \mathcal{L}(\theta_k)$.
- 3: Define the linear operator A_k and right-hand side b_k :

$$A_k := \left(1 + \frac{h\gamma}{2}\right) I_n + \left(\left(1 + \frac{h\gamma}{2}\right) \frac{h\delta}{2} + \frac{h^2}{4}\right) B_k,$$

$$b_k := \left[\left(1 - \frac{h\gamma}{2}\right) I_n + \left(\left(1 - \frac{h\gamma}{2}\right) \frac{h\delta}{2} - \frac{h^2}{4}\right) B_k\right] \omega_k - h g_k.$$

- 4: **Solve the linear system** $A_k \omega_{k+1} = b_k$ **using conjugate gradient.**
 - 5: $\omega^{(0)} \leftarrow \omega_k$ {initial guess}
 - 6: **for** $j = 0, 1, \dots, J - 1$ **do**
 - 7: Perform conjugate gradient for $A_k \omega = b_k$ starting from $\omega^{(j)}$
 - 8: **if** $\|A_k \omega^{(j+1)} - b_k\| \leq \varepsilon_{\text{cg}}$ **then**
 - 9: **break**
 - 10: **end if**
 - 11: **end for**
 - 12: **Accept:** $\omega_{k+1} \leftarrow \omega^{(j+1)}$.
 - 13: Define $C_k := I_n + \frac{h\delta}{2} B_k$, $r_k := g_k + \frac{h}{4} B_k(\omega_k + \omega_{k+1})$.
 - 14: **Solve the linear system** $C_k \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) = r_k$ **using conjugate gradient.**
 - 15: $\bar{\nabla}_{\theta}^{(0)} \leftarrow g_k$ {initial guess}
 - 16: Perform conjugate gradient for $C_k \bar{\nabla}_{\theta} = r_k$
 - 17: **Accept:** $\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1}) \leftarrow \bar{\nabla}_{\theta}^{(j+1)}$.
 - 18: Update the parameters: $\theta_{k+1} \leftarrow \theta_k + \frac{h}{2}(\omega_k + \omega_{k+1}) - h\delta \bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})$.
 - 19: Compute the new gradient: $g_{k+1} \leftarrow \nabla_{\theta} \mathcal{L}(\theta_{k+1})$.
 - 20: Form the curvature pair: $s_k \leftarrow \theta_{k+1} - \theta_k$, $y_k \leftarrow g_{k+1} - g_k$.
 - 21: **if** $s_k^{\top} y_k > 0$ **then**
 - 22: Store the curvature pair (s_k, y_k) in the L-BFGS memory.
 - 23: Update the Hessian approximation B_{k+1} using the curvature pairs.
 - 24: **else**
 - 25: Skip the curvature update and set $B_{k+1} = B_k$.
 - 26: **end if**
 - 27: **end for**
 - 28: **return** θ_K
-

Chapter 4

Results

In this chapter, we evaluate the proposed structure-preserving optimization methods across four experiments. Our goal is to assess how these methods compare to established baselines, including SGD, Adam, and L-BFGS, in terms of convergence speed, generalization, and robustness to ill-conditioning.

We consider four experiments. First, we study the synthetic least squares problem, where the condition number can be controlled explicitly. This allows us to isolate the effect of ill-conditioning and examine how step-size restrictions and convergence rates scale across methods.

Second, we consider ℓ_2 -regularized logistic regression, which provides a simple yet informative setting in which both the conditioning of the feature matrix and the separability of the class labels can be controlled explicitly.

Third, we evaluate the methods on a physics-informed neural network (PINN) for solving a linear advection equation, a nonconvex setting where the optimization landscape is shaped by both the network architecture and the underlying partial differential equation (PDE) structure.

Finally, we consider a large-scale deep learning task on the CIFAR-10 dataset using a ResNet architecture. This experiment evaluates the practical performance of the methods in modern training settings, and we include both optimization and generalization metrics.

Across these experiments, the proposed structure-preserving methods, particularly the semi-implicit schemes (SIDG), emerge as the most consistent method offering strong robustness to ill-conditioning, significantly fewer iterations than explicit methods, and a favorable balance between computational cost and solution quality. The addition of the δ -term improves stability and convergence in many cases, especially for explicit and fully implicit methods. Moreover, the proposed methods achieve competitive or superior performance compared to widely used optimizers such as Adam and L-BFGS.

The remainder of this chapter presents these experiments in detail.

4.1 Least Squares

In this section, we study the behavior of the proposed optimization methods on synthetic least squares problems, where the conditioning of the objective can be controlled explicitly. This allows us to isolate the effect of ill-conditioning on optimization performance. We compare explicit methods (MGD and MGD- δ) with the proposed structure-preserving methods (DG, DG- δ , SIDG, and SIDG- δ). Our goal is to understand how step-size restrictions, iteration complexity, and runtime scale with the condition number of the problem.

4.1.1 Experimental Setup

We study the least squares problem Ehrhardt et al. [7]

$$\mathcal{L}(\theta) = \frac{1}{2n} \|A\theta - b\|^2,$$

which is convex and L -smooth. Defining

$$Q := \frac{1}{n} A^\top A, \quad c := \frac{1}{n} A^\top b,$$

we can write this in a quadratic form as follows:

$$\mathcal{L}(\theta) = \frac{1}{2} \theta^\top Q \theta - c^\top \theta + \frac{1}{2N} \|b\|^2.$$

The conditioning of the quadratic is governed by the eigenvalues of Q . If Q is positive definite, we define the condition number as

$$\kappa(Q) := \frac{\lambda_{\max}(Q)}{\lambda_{\min}(Q)},$$

measures how unevenly curved the objective is across different directions. When $\kappa(Q) \approx 1$, the curvature is roughly the same in every direction and the problem is easy to optimize. As $\kappa(Q)$ grows, some directions become much steeper or change more sharply than others, leading to the gradient changing rapidly (*stiff* directions). As mentioned before, explicit schemes can be highly sensitive to parameter selection. For these schemes, the stability 2.9 region shrinks as $\kappa(Q)$ increases (2.15), forcing the use of smaller step sizes, which leads to slower convergence. Hence, our goal is to understand how this maximum stable step size shrinks as $\kappa(Q)$ increases for all of our proposed schemes.

To study this, we generate synthetic least squares problems with condition numbers

$$\kappa \in \{1, 10^2, 10^4, 10^6\},$$

and compare MGD, MGD- δ , DG, DG- δ , SIDG, and SIDG- δ . We construct A using the singular value decomposition

$$A = U\Sigma V^\top,$$

where U and V are random orthonormal matrices and the singular values of Σ are chosen so that $\text{cond}(A) = \kappa$. All results are averaged over 5 independent runs with different random seeds. We report the mean and standard deviation of the iteration counts and runtime across these runs.

4.1.2 Results

The least squares experiments show the following. As the condition number increases, the explicit method MGD becomes increasingly limited by step-size restrictions, leading to a sharp growth in iteration counts and eventual failure at $\kappa = 10^6$. The implicit methods DG and SIDG are much more robust to ill-conditioning, requiring far fewer iterations than MGD in the ill-conditioned regimes. Lastly, adding the δ -term improves all method families, with DG- δ and SIDG- δ giving the strongest overall performance in both iterations and wall-clock time. Table 4.1 summarises these trends.

Table 4.1: Iterations and time (seconds) required to reach the target tolerance for least squares problems of increasing condition number. As the problem becomes more ill-conditioned, MGD slows dramatically and fails at $\kappa = 10^6$, whereas the implicit methods remain robust. The δ -variants, especially DG- δ and SIDG- δ , achieve the lowest iteration counts and fastest times in the hardest regimes.

Method	$\kappa = 1$	$\kappa = 10^2$	$\kappa = 10^4$	$\kappa = 10^6$
MGD	10 / 0.0033	778 / 0.1302	4049 / 0.4789	–
MGD- δ	10 / 0.0017	684 / 0.1015	1855 / 0.2447	4129 / 0.8910
DG	15 / 0.0034	243 / 0.0505	1008 / 0.2440	1113 / 0.4160
DG- δ	10 / 0.0019	78 / 0.0178	163 / 0.0503	113 / 0.0377
SIDG	15 / 0.0042	243 / 0.0527	1008 / 0.2085	1113 / 0.3423
SIDG- δ	10 / 0.0034	78 / 0.0204	163 / 0.0399	113 / 0.0494

4.1.3 Discussion

In the well-conditioned case $\kappa = 1$, all methods converge rapidly and achieve identical performance (see Table 4.1). All methods reach the target tolerance within 10–15 iterations. In this regime, the optimization problem is not stiff, and step-size restrictions are mild. As a result, the advantages of implicit methods are not pronounced, and all methods exhibit similar convergence behavior.

The effect of ill-conditioning becomes clear as κ increases. For explicit methods, stability is tied to the largest eigenvalue of the quadratic. In particular, we have

$$h \lesssim \frac{1}{\lambda_{\max}(Q)},$$

which follows directly from proposition 2.14 and Theorem B.4. This means that we have to take a step size that satisfies this inequality, which means that the step size is far too small to make significant updates. This is exactly what causes slow convergence in ill-conditioned quadratics.

This is clearly visible in the experiments. For MGD, the best stable step size decreases from $h = 1$ at $\kappa = 1$ to $h = 0.3$ at $\kappa = 10^2$ and $\kappa = 10^4$, and the iteration count grows sharply, from 10 to 778 to 4049. At $\kappa = 10^6$, MGD does not reach the target tolerance within the given number of max iterations. Thus, even though MGD is effective in the well-conditioned case, its performance degrades rapidly as the problem becomes more ill-conditioned.

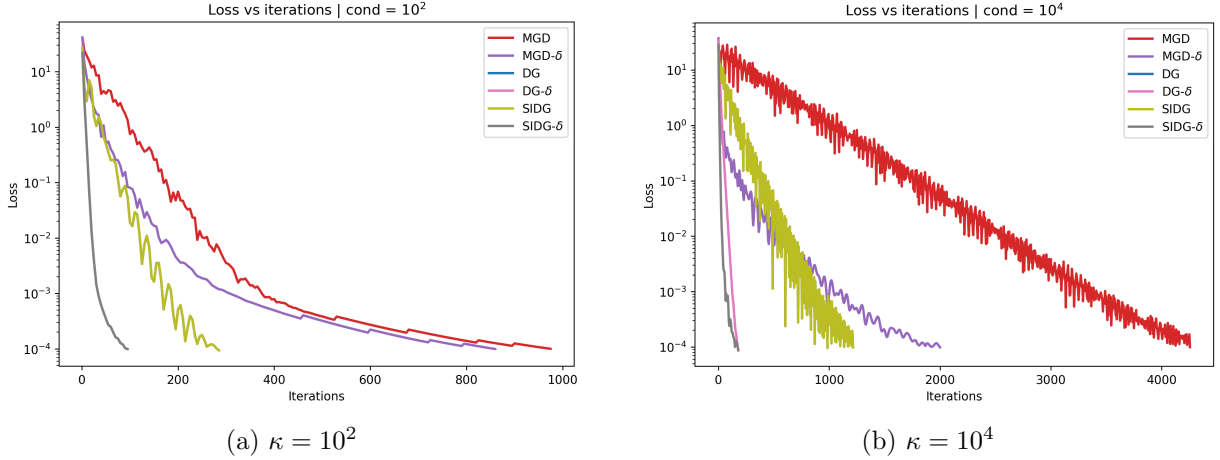


Figure 4.1: Loss versus iterations for increasing condition numbers. As κ increases, explicit methods slow down significantly, while implicit methods remain more robust to ill-conditioning.

At $\kappa = 10^2$, MGD requires 778.00 ± 160.55 iterations to reach the tolerance, while DG and SIDG require only 243.00 ± 26.94 iterations. At $\kappa = 10^4$, this gap widens further: MGD requires 4049.00 ± 124.47 iterations, whereas DG and SIDG require 1008.00 ± 112.68 . This is a substantial improvement already, even before adding the δ -term.

Effect of the δ -parameter: fewer iterations and less oscillation

Across all families of methods, adding δ improves the convergence behavior, and this is evident from the plots. The δ -variants reduce the number of iterations required to reach the tolerance, as summarised in Table 4.1.

At $\kappa = 10^6$, MGD fails to reach tolerance entirely, while MGD- δ still converges in 4129 iterations. Thus, the δ -term not only accelerates convergence but can restore robustness in regimes where the baseline method breaks down.

Second, the δ -variants often produce smoother loss curves. In the ill-conditioned cases, the non- δ methods exhibit visible oscillatory behavior (see Figure 4.1). However, the δ -variants often show a clearer and nearly linear decay on the plots. This is visible for DG- δ and SIDG- δ (gray and pink curves), whose trajectories frequently appear as almost straight descending lines compared to the DG and SIDG curves (blue and yellow curves). Hence, the δ -term not only reduces the iteration count but it also suppresses oscillations and produces a cleaner, more strongly damped descent.

This behavior is also visible for MGD- δ (purple curve). Compared with MGD (red curve), the MGD- δ curves tend to oscillate less and descend more steadily, which is consistent with the interpretation of δ as adding extra damping to the dynamics.

Step-size restrictions and robustness to ill-conditioning

A key observation from the step-size sweeps is that DG, DG- δ , SIDG, and SIDG- δ remain stable for much larger values of h than the explicit methods (see Figure 4.2). In the reported experiments, the implicit methods are stable even at large step sizes ($h = 10000$), while the explicit methods become unstable much earlier.

This larger stability region is one of the main advantages of the implicit structure-preserving schemes. Since they are not forced to resolve the stiffest direction with a very small explicit step, they can take much larger effective steps and therefore remain efficient even when the condition number is large. This is visible at $\kappa = 10^6$ (see Table 4.1). MGD does not converge to the target tolerance within the maximum number of iterations, while DG and SIDG still do so in about 1113 iterations, and DG- δ and SIDG- δ need only 113 iterations.

Remark 4.1 (Why DG and SIDG have the same metrics in this experiment). An important observation in the tables is that DG and SIDG have essentially identical iteration counts, final losses, and times-to-tolerance in the least squares experiments, and the same is true for DG- δ and SIDG- δ . This is not accidental. In the quadratic setting, the gradient is affine,

$$\nabla_{\theta} \mathcal{L}(\theta) = Q\theta - c,$$

so the discrete gradient and the semi-implicit linearization reduce to the same underlying linear relation. As a result, DG and SIDG generate the same metrics for this problem up to negligible numerical differences.

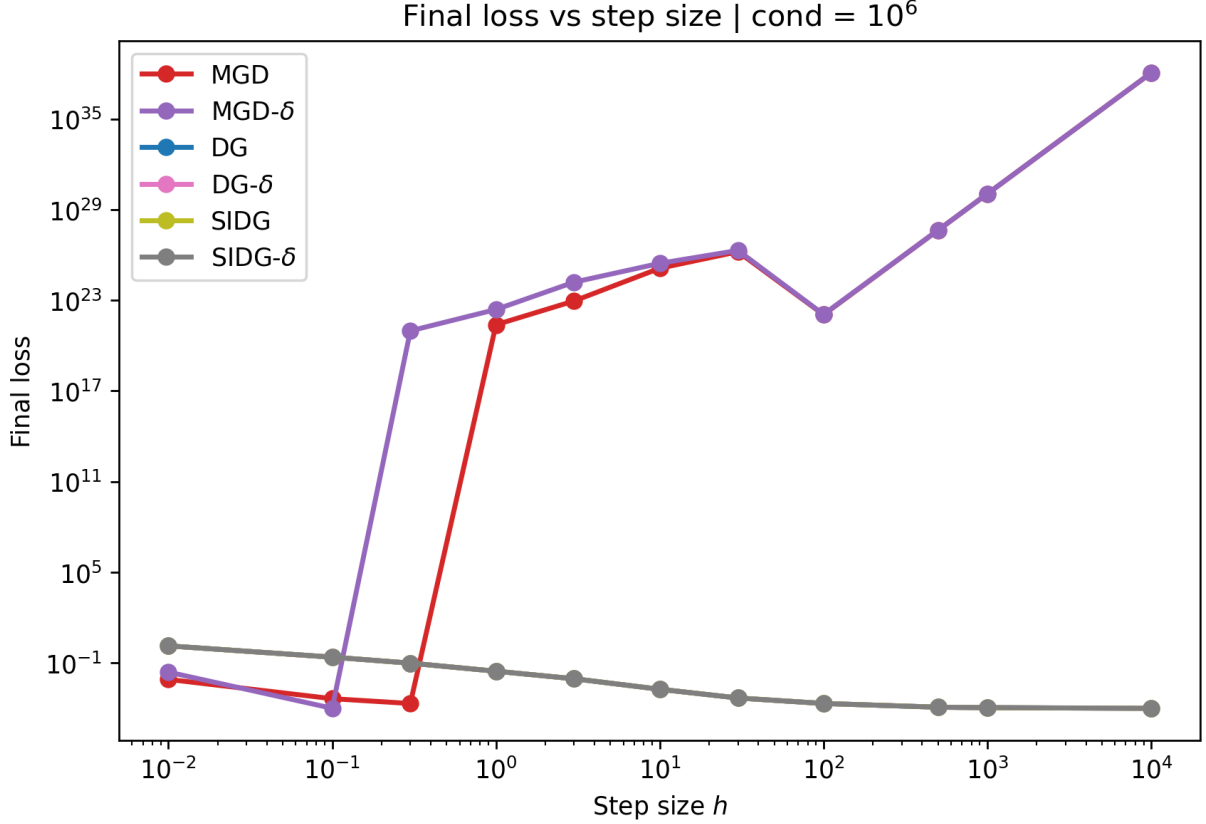


Figure 4.2: Final loss versus step size for $\kappa = 10^6$. The explicit methods become unstable as h increases, while the implicit methods remain stable over a much larger range of step sizes.

Runtime and time-to-tolerance

Although the implicit methods involve more work per iteration, their reduced iteration counts often translate into competitive or even better wall-clock performance.

For example, at $\kappa = 10^4$ (see Figure 4.3b), MGD requires 0.7342 ± 0.0974 seconds to reach the tolerance, whereas DG- δ requires only 0.0595 ± 0.0147 seconds and SIDG- δ only 0.0653 ± 0.0041 seconds. Hence, the improved iteration complexity is strong enough to outweigh the additional cost per step. At $\kappa = 10^6$, the same pattern remains: MGD- δ reaches tolerance in 0.3478 ± 0.0487 seconds, while DG- δ and SIDG- δ do so in only 0.0595 ± 0.0147 and 0.0653 ± 0.0041 seconds respectively. So in the most ill-conditioned regime, the implicit δ -methods are not only more stable and more iteration-efficient, but also substantially faster in wall-clock time.

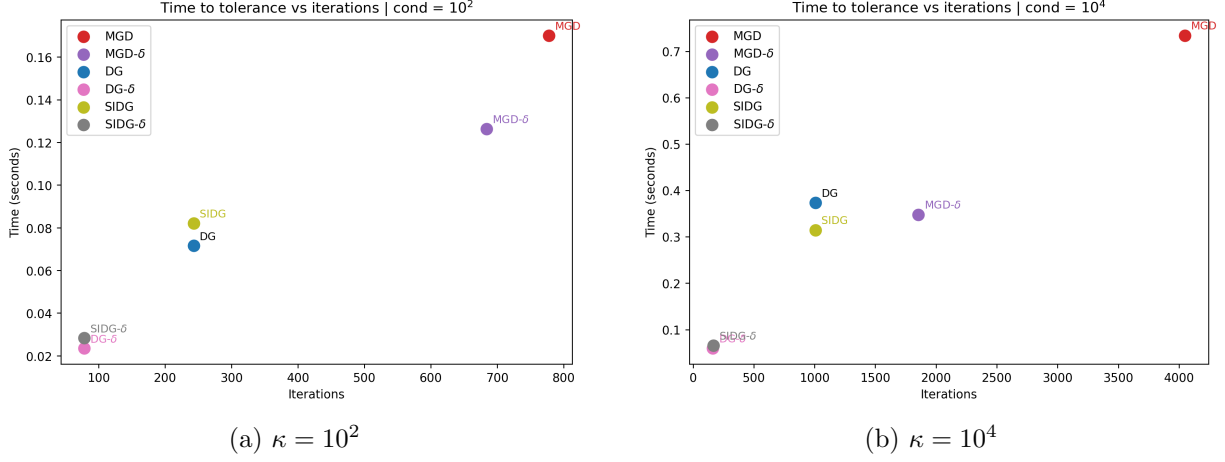


Figure 4.3: Time to tolerance versus iterations for increasing condition numbers. The δ -implicit variants achieve a significantly better tradeoff between iteration count and runtime, with the advantage becoming more pronounced as the condition number increases.

4.2 Regularized Logistic Regression

We consider ℓ_2 -regularized binary logistic regression as a benchmark to compare the proposed structure-preserving methods with classical optimization algorithms. This problem provides a simple yet informative setting in which both the conditioning of the feature matrix and the separability of the class labels can be controlled explicitly.

4.2.1 Experimental Setup

Let $A \in \mathbb{R}^{n \times d}$ denote the feature matrix with rows $a_i^\top$, and let labels $y_i \in \{-1, +1\}$ for $i = 1, \dots, n$. The goal is to learn parameters $\theta \in \mathbb{R}^n$ that predict labels via $\hat{y}_i = \text{sign}(a_i^\top \theta)$. We minimize the regularized logistic loss

$$\mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i a_i^\top \theta)) + \frac{\lambda}{2} \|\theta\|^2, \quad (4.1)$$

where $\lambda > 0$ is the ℓ_2 -regularization parameter LeCun et al. [12]. The gradient is given by

$$\nabla_{\theta} \mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^n y_i a_i \sigma(-y_i a_i^\top \theta) + \lambda \theta, \quad \sigma(t) := \frac{1}{1 + e^{-t}}. \quad (4.2)$$

The dataset is generated synthetically to allow independent control of conditioning and classification difficulty. We construct A using the singular value decomposition

$$A = U\Sigma V^\top,$$

where U and V are random orthonormal matrices and the singular values of Σ are chosen so that $\text{cond}(A) = \kappa$. We consider both well-conditioned ($\kappa \approx 1$) and ill-conditioned ($\kappa = 1000$) regimes. After construction, features are standardized to have a zero mean and unit variance.

To generate the labels, we first sample a ground-truth parameter vector

$$\theta_\star \sim \mathcal{N}(0, I),$$

where $\mathcal{N}(0, I)$ is the standard normal and scale it by a *margin parameter* $m > 0$:

$$\theta_\star \leftarrow m \theta_\star. \tag{4.3}$$

For each sample, we compute logits

$$\ell_i := a_i^\top \theta_\star,$$

and generate labels according to a probabilistic logistic model:

$$p_i := \sigma(\ell_i), \quad y_i = \begin{cases} +1, & \text{with probability } p_i, \\ -1, & \text{with probability } 1 - p_i. \end{cases} \tag{4.4}$$

The margin parameter m controls the difficulty of the classification problem:

- Small m : logits are close to zero, producing noisier labels.
- Large m : logits have larger magnitude, leading to more separable data.

To better illustrate this refer to plot (see Figure 4.4).

We study performance across a 2×2 grid of problem settings defined by:

- **Condition number:** $\kappa \in \{\kappa_{\text{small}} = 1, \kappa_{\text{large}} = 1000\}$,
- **Margin:** $m \in \{m_{\text{small}} = 1, m_{\text{large}} = 4\}$.

This yields four regimes:

1. $m_{\text{small}}, \kappa_{\text{small}}$ (noisy, well-conditioned),
2. $m_{\text{small}}, \kappa_{\text{large}}$ (noisy, ill-conditioned),

3. $m_{\text{large}}, \kappa_{\text{small}}$ (separable, well-conditioned),
4. $m_{\text{large}}, \kappa_{\text{large}}$ (separable, ill-conditioned).

Lastly, we report the training $\|\nabla_{\theta} \mathcal{L}(\theta)\|$ and test loss as well as test accuracy. Training is terminated when

$$\|\nabla_{\theta} \mathcal{L}(\theta)\| \leq 10^{-3},$$

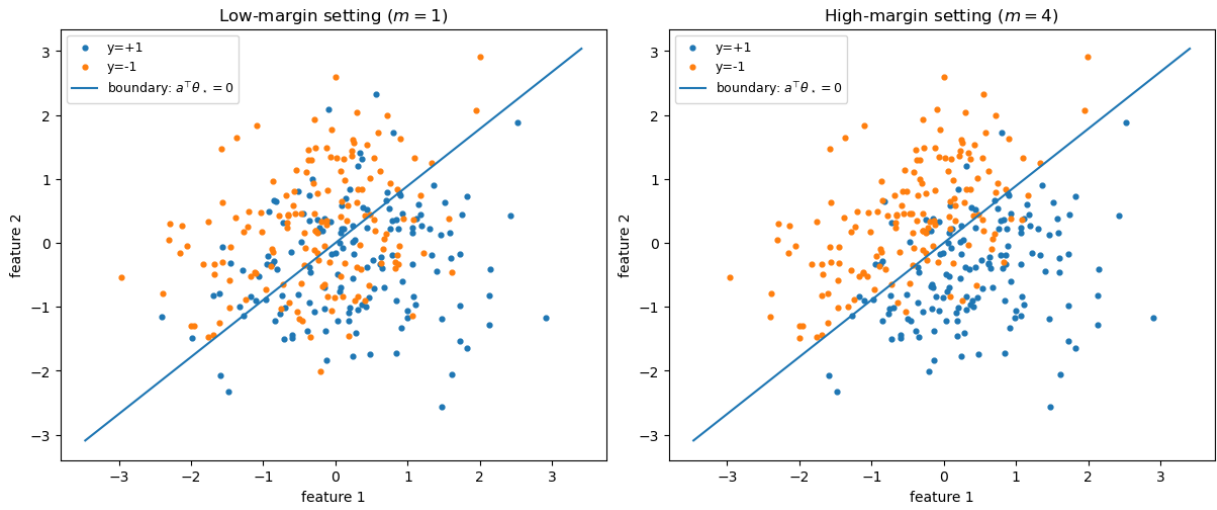


Figure 4.4: Illustration of the effect of the margin parameter on label separability for a 2-D example. For $m = 1$, the labels are less separable and lie closer to the decision boundary. For $m = 4$, the labels become more separable. The line $a^{\top} \theta_{\star} = 0$ denotes the decision boundary of the classifier.

4.2.2 Results

We focus on the ill-conditioned regime i.e, $\kappa = 1000$. The corresponding well-conditioned results when $\kappa = 1$ are reported in Appendix D.1.1. In that setting, all methods reach the stopping criterion with nearly identical losses and accuracies, and thus, the comparison is less informative. The ill-conditioned case provides a more meaningful test of optimization behavior.

The results reveal three main trends. First, all methods achieve similar final losses and accuracies across both margin regimes, indicating that they converge to comparable solutions. Second, the primary difference lies in optimization efficiency. Among the structure-preserving methods, SIDG consistently requires the fewest iterations and outperforms DG in both iteration count and runtime. Third, while the δ -modification improves explicit methods slightly, it provides little benefit for SIDG and increases computational cost. Overall, SIDG achieves the best balance between iteration efficiency, runtime, and solution quality, and performs competitively with L-BFGS without requiring line searches.

Table 4.2: Regularized logistic regression results for the ill-conditioned, high-margin regime ($m = 4, \kappa = 1000$). All methods achieve similar final losses and accuracies. However, among the structure preserving methods SIDG and SIDG- δ require far fewer iterations than first-order methods and are competitive with L-BFGS, while DG incurs higher runtime due to inner iterations. Furthermore, the SIDG methods achieve strong generalization performance, with test loss and accuracy comparable to Adam. Results are reported as mean $\pm$ standard deviation across random seeds.

Method	Train Loss	Test Loss	Test Acc	Iter	Time (s)
Adam	5.175e-02 $\pm$ 1.979e-02	6.184e-02 $\pm$ 1.789e-02	0.979 $\pm$ 0.010	215 $\pm$ 14	0.120 $\pm$ 0.011
GD	5.298e-02 $\pm$ 1.582e-02	7.553e-02 $\pm$ 1.891e-02	0.977 $\pm$ 0.012	825 $\pm$ 98	0.437 $\pm$ 0.065
MGD	5.296e-02 $\pm$ 2.003e-02	6.180e-02 $\pm$ 1.960e-02	0.975 $\pm$ 0.011	185 $\pm$ 22	0.087 $\pm$ 0.012
MGD- δ	5.276e-02 $\pm$ 2.005e-02	6.253e-02 $\pm$ 1.994e-02	0.975 $\pm$ 0.011	175 $\pm$ 18	0.091 $\pm$ 0.019
DG	5.379e-02 $\pm$ 1.982e-02	6.144e-02 $\pm$ 1.930e-02	0.974 $\pm$ 0.013	210 $\pm$ 45	0.291 $\pm$ 0.053
DG- δ	5.319e-02 $\pm$ 2.000e-02	6.183e-02 $\pm$ 1.941e-02	0.975 $\pm$ 0.011	210 $\pm$ 29	0.291 $\pm$ 0.034
SIDG	5.155e-02 $\pm$ 1.932e-02	6.050e-02 $\pm$ 1.745e-02	0.979 $\pm$ 0.010	65 $\pm$ 14	0.276 $\pm$ 0.062
SIDG- δ	5.155e-02 $\pm$ 1.932e-02	6.049e-02 $\pm$ 1.743e-02	0.979 $\pm$ 0.010	65 $\pm$ 14	0.480 $\pm$ 0.115
LBFGS	4.891e-02 $\pm$ 1.967e-02	6.435e-02 $\pm$ 1.561e-02	0.977 $\pm$ 0.011	35 $\pm$ 14	0.086 $\pm$ 0.033

4.2.3 Discussion

Table 4.2: Separable, ill-conditioned regime ($m = 4, \kappa = 1000$). We begin with the large-margin setting, where the data is nearly separable and the optimization problem is dominated by ill-conditioning rather than label noise. In this regime, all methods achieve very similar final losses and test accuracies, indicating convergence to essentially the same solution.

Table 4.3: Regularized logistic regression results for the ill-conditioned, low-margin regime ($m = 1, \kappa = 1000$), where label noise increases problem difficulty. All methods again reach similar solutions, but SIDG maintains the lowest iteration counts among structure-preserving methods and remains competitive with L-BFGS. Dashes indicate failure to converge. Results are reported as mean $\pm$ standard deviation across random seeds.

Method	Train Loss	Test Loss	Test Acc	Iter	Time (s)
Adam	$2.021\text{e-}01 \pm 3.604\text{e-}02$	$2.228\text{e-}01 \pm 2.846\text{e-}02$	0.904 ± 0.028	250 ± 18	0.140 ± 0.012
GD	-	-	-	-	-
MGD	$2.040\text{e-}01 \pm 3.610\text{e-}02$	$2.205\text{e-}01 \pm 2.830\text{e-}02$	0.904 ± 0.027	165 ± 34	0.084 ± 0.018
MGD- δ	$2.040\text{e-}01 \pm 3.606\text{e-}02$	$2.205\text{e-}01 \pm 2.838\text{e-}02$	0.904 ± 0.025	145 ± 41	0.075 ± 0.034
DG	$2.040\text{e-}01 \pm 3.610\text{e-}02$	$2.204\text{e-}01 \pm 2.827\text{e-}02$	0.905 ± 0.025	165 ± 34	0.391 ± 0.058
DG- δ	$2.040\text{e-}01 \pm 3.606\text{e-}02$	$2.205\text{e-}01 \pm 2.838\text{e-}02$	0.904 ± 0.025	145 ± 41	0.328 ± 0.065
SIDG	$2.035\text{e-}01 \pm 3.587\text{e-}02$	$2.211\text{e-}01 \pm 2.858\text{e-}02$	0.904 ± 0.024	70 ± 11	0.250 ± 0.045
SIDG- δ	$2.035\text{e-}01 \pm 3.589\text{e-}02$	$2.211\text{e-}01 \pm 2.859\text{e-}02$	0.904 ± 0.024	70 ± 11	0.471 ± 0.078
LBFGS	$2.010\text{e-}01 \pm 3.623\text{e-}02$	$2.262\text{e-}01 \pm 2.852\text{e-}02$	0.902 ± 0.029	65 ± 14	0.165 ± 0.032

The primary distinction between methods lies in optimization efficiency. Adam requires 215 ± 14 iterations to reach the tolerance, while the momentum methods (MGD and MGD- δ) reduce this to approximately 175–185 iterations and achieve the fastest wall-clock times due to their inexpensive updates.

The DG methods require over 200 iterations and incur significantly higher runtime, reflecting the cost of the inner fixed-point iterations. In contrast, SIDG and SIDG- δ converge in only 65 ± 14 iterations, substantially fewer than all first-order methods and comparable to L-BFGS, which requires 35 ± 14 iterations. However, L-BFGS has higher test loss and marginally lower test accuracy compared to the other methods.

SIDG also improves over DG in both iteration count and runtime. Despite both methods being implicit, the semi-implicit formulation of SIDG avoids costly inner iterations, resulting in more efficient updates. However, SIDG- δ has higher runtime than all methods due to its two inner conjugate gradient solves (Algorithm ??).

Table 4.3: Noisy, ill-conditioned regime ($m = 1, \kappa = 1000$). We now consider the more challenging setting which is the low-margin setting, where label noise is present in addition to ill-conditioning.

Adam, MGD, and MGD- δ follow the same qualitative behavior as in the separable case, but require more iterations to reach the tolerance. The DG methods again exhibit similar iteration counts to MGD and MGD- δ , but incur substantially higher runtime due to the cost of the fixed-point iterations. In contrast, SIDG and SIDG- δ converge in only 70 ± 11 iterations, comparable

to L-BFGS (65 ± 14), while maintaining a structured geometric update without requiring line searches. Despite its faster convergence, L-BFGS attains a higher test loss and slightly lower test accuracy than SIDG.

As in the separable case, SIDG outperforms DG in both iteration count and runtime, demonstrating that the semi-implicit formulation provides a more efficient mechanism for incorporating information about the hessian (curvature) than the fixed-point updates used in DG. The δ -variant again does not improve convergence for SIDG and increases runtime, making standard SIDG the preferable choice.

4.3 PINN Experiment

In this experiment, we evaluate the performance of the proposed optimization methods on a physics-informed neural network (PINN) Raissi et al. [21] applied to a simple linear advection equation.

4.3.1 Experimental setup

We consider the linear advection equation on the space-time domain $(t, x) \in [0, 1] \times [-1, 1]$,

$$u_t + u_x = 0, \quad (4.5)$$

with the periodic boundary conditions

$$u(t, -1) = u(t, 1), \quad (4.6)$$

and the initial condition

$$u(0, x) = g(x), \quad (4.7)$$

where

$$g(x) := \sin(\pi x) + 0.3 \cos(2\pi x) - 0.2 \sin(3\pi x).$$

We consider noisy observations of the initial condition given by

$$g_{\text{noisy}}(x) := g(x) + \varepsilon(x), \quad \varepsilon \sim \mathcal{N}(0, 0.01^2).$$

We approximate the solution using a neural network $u_\theta(t, x)$ with parameters θ . The network is trained by minimizing the physics-informed loss

$$\mathcal{L}(\theta) := \mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}} \mathcal{L}_{\text{BC}} + \lambda_{\text{IC}} \mathcal{L}_{\text{IC}}. \quad (4.8)$$

The PDE residual loss is defined as

$$\mathcal{L}_{\text{PDE}} := \frac{1}{N_F} \sum_{i=1}^{N_F} |\partial_t u_\theta(t_i, x_i) + \partial_x u_\theta(t_i, x_i)|^2, \quad (4.9)$$

where (t_i, x_i) are collocation points sampled uniformly from the domain. The periodic boundary condition loss is given by

$$\mathcal{L}_{\text{BC}} := \frac{1}{N_B} \sum_{i=1}^{N_B} |u_\theta(t_i, -1) - u_\theta(t_i, 1)|^2. \quad (4.10)$$

The initial condition loss is defined using noisy observations:

$$\mathcal{L}_{\text{IC}} := \frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} |u_{\theta}(0, x_i) - g_{\text{noisy}}(x_i)|^2. \quad (4.11)$$

The derivatives $\partial_t u_{\theta}$ and $\partial_x u_{\theta}$ are computed using automatic differentiation. Furthermore, N_F denotes the number of space-time volume points, N_B denotes the number of boundary points, and N_{IC} denotes the number of initial space-time volume points. In our experiments, we use $N_F = 256$, $N_B = 64$, and $N_{\text{IC}} = 256$. Lastly, the collocation points (t_i, x_i) are sampled uniformly from the space-time domain:

$$t_i \sim \mathcal{U}([0, 1]), \quad x_i \sim \mathcal{U}([-1, 1]).$$

Boundary points are sampled as

$$t_i \sim \mathcal{U}([0, 1]), \quad x \in \{-1, 1\},$$

and initial condition points are sampled as

$$x_i \sim \mathcal{U}([-1, 1]), \quad t = 0.$$

We evaluate performance using the relative L^2 error

$$\mathcal{L}_{\text{rel}} := \frac{\|u_{\theta} - u\|_{L^2}}{\|u\|_{L^2}}, \quad (4.12)$$

computed using adaptive Gauss–Legendre quadrature over the space-time domain. Training is terminated early when

$$\mathcal{L}_{\text{rel}} \leq 10^{-1}.$$

We use a fully connected feedforward neural network to approximate the solution $u_{\theta}(t, x)$. The network takes $(t, x) \in \mathbb{R}^2$ as input and outputs a scalar prediction $u_{\theta}(t, x) \in \mathbb{R}$. It consists of 4 hidden layers of width 64, with tanh activation functions applied after each hidden layer. The use of tanh activations is standard in PINN architectures, as they provide smooth approximations and enable stable computation of derivatives via automatic differentiation.

4.3.2 Results

The PINN experiments show that L-BFGS achieves the best overall performance, attaining the lowest $\mathcal{L}_{\text{rel}}$ error, lowest loss, and fastest convergence in both iterations and runtime, though with higher variability across runs. Among the structure-preserving methods, SIDG provides the best overall balance between accuracy, iteration efficiency, and runtime. SIDG and SIDG- δ significantly reduce iteration counts compared to explicit methods while maintaining competitive accuracy. Lastly, while the δ -modification improves MGD and DG by adding damping, it provides limited benefit for SIDG and increases computational cost due to additional linear solves. Table 4.4 summarizes these results.

Table 4.4: PINN performance across seeds (mean $\pm$ standard deviation). L-BFGS achieves the best overall performance in terms of accuracy and convergence speed, though with higher variability. Among the structure-preserving methods, SIDG and SIDG- δ provide the best balance between accuracy and efficiency, requiring significantly fewer iterations than explicit methods while maintaining competitive runtime. DG methods are substantially slower due to expensive implicit updates.

Method	Final rel L_2	Final Loss	Iter to Tol	Time (s)
LBFGS	$7.208 \times 10^{-2} \pm 2.186 \times 10^{-2}$	$2.707 \times 10^{-3} \pm 1.172 \times 10^{-3}$	120.0 ± 44.7	3.5 ± 1.3
SIDG- δ	$9.353 \times 10^{-2} \pm 4.984 \times 10^{-3}$	$3.815 \times 10^{-3} \pm 5.457 \times 10^{-4}$	560.0 ± 151.7	9.3 ± 2.3
MGD- δ	$9.510 \times 10^{-2} \pm 3.363 \times 10^{-3}$	$4.189 \times 10^{-3} \pm 4.157 \times 10^{-4}$	1240.0 ± 230.2	4.5 ± 0.8
SIDG	$9.643 \times 10^{-2} \pm 2.888 \times 10^{-3}$	$4.018 \times 10^{-3} \pm 4.896 \times 10^{-4}$	500.0 ± 122.5	6.2 ± 1.5
Adam	$9.706 \times 10^{-2} \pm 1.579 \times 10^{-3}$	$3.960 \times 10^{-3} \pm 2.924 \times 10^{-4}$	1020.0 ± 216.8	4.0 ± 0.5
DG- δ	$9.856 \times 10^{-2} \pm 8.492 \times 10^{-4}$	$4.563 \times 10^{-3} \pm 5.975 \times 10^{-4}$	2280.0 ± 887.1	18.9 ± 7.6
DG	$9.925 \times 10^{-2} \pm 7.109 \times 10^{-4}$	$4.292 \times 10^{-3} \pm 2.431 \times 10^{-4}$	2720.0 ± 1096.4	22.3 ± 9.0
MGD	$9.940 \times 10^{-2} \pm 5.273 \times 10^{-4}$	$4.361 \times 10^{-3} \pm 3.734 \times 10^{-4}$	3180.0 ± 828.9	11.6 ± 2.7
GD	$9.922 \times 10^{-1} \pm 6.509 \times 10^{-3}$	$5.149 \times 10^{-1} \pm 6.493 \times 10^{-2}$	-	-

4.3.3 Discussion

Table 4.4 shows that L-BFGS achieves the strongest overall performance, attaining the lowest final $\mathcal{L}_{\text{rel}}$ error, the lowest loss, and the fastest convergence in both iterations and wall-clock time. However, its larger standard deviation indicates greater variability across runs.

Among the structure-preserving methods, SIDG and SIDG- δ perform best overall. In particular, SIDG- δ attains the lowest $\mathcal{L}_{\text{rel}}$ error among non-L-BFGS methods, while SIDG reaches the target tolerance in fewer iterations. Both variants significantly outperform Adam, MGD, DG, and their δ -variants in iteration efficiency while maintaining competitive runtime.

Adam remains a strong baseline with good runtime performance, but requires more iterations to reach the same tolerance. The momentum-based methods MGD and MGD- δ also converge reliably, but require substantially more iterations, reflecting slower convergence of explicit momentum dynamics.

The DG and DG- δ methods exhibit highly consistent performance across runs, as indicated by their small standard deviations. However, they are considerably slower in both iterations and wall-clock time. As shown in Figure 4.5, their trajectories (green and orange curves) display visible spikes and plateaus due to the implicit updates and variability in the inner solves. While these oscillations are regular when viewed across iterations, they appear irregular in time because the computational cost per iteration is non-uniform. Each update requires multiple gradient evaluations and fixed-point solves, leading to occasional expensive steps and slower overall convergence.

The δ -modification improves performance for MGD and DG by introducing additional damping, reducing error and iteration count. However, this benefit does not extend to SIDG. In the SIDG setting, the δ -variant requires solving an additional linear system via conjugate gradients (Algorithm ??), compared to a single solve in the standard formulation (Algorithm 5), increasing computational cost without clear gains in accuracy.

Finally, GD fails to reach the desired tolerance in any run, with $\mathcal{L}_{\text{rel}}$ error remaining close to 1, confirming its inadequacy for this problem.

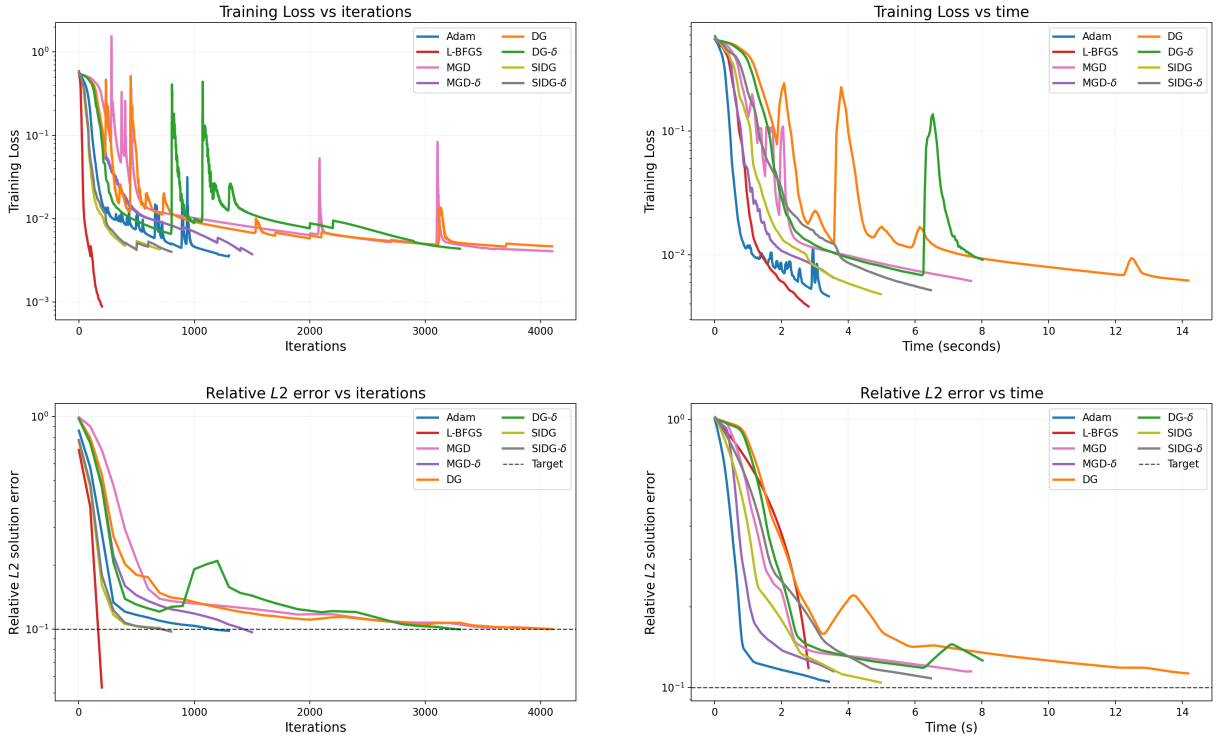


Figure 4.5: Comparison of all methods on the PINN inverse problem. Top row: training metrics. Bottom row: $\mathcal{L}_{\text{rel}}$ metrics. The dashed line indicates the target tolerance $\text{rel}L_2 \leq 10^{-1}$. We removed GD from the plots for clarity.

Figure 4.5 further illustrates these behaviors. L-BFGS decreases the loss extremely rapidly and achieves the lowest values in the shortest time. SIDG and SIDG- δ exhibit smooth and stable decay, while momentum methods show more oscillatory behavior. DG methods converge reliably but are significantly slower due to their expensive updates.

In terms of $\mathcal{L}_{\text{rel}}$ error, L-BFGS again performs best, reaching and surpassing the target tolerance quickly. Among structure-preserving methods, SIDG variants achieve the fastest convergence

in iterations with stable behavior. Adam converges reliably but more slowly, while MGD and DG methods require substantially more iterations or time.

4.4 CIFAR-10 Experiments

Our last experiment involves a classification task on the CIFAR-10 dataset. The dataset consists of 60,000 color images of size 32×32 across 10 classes, with 50,000 training samples and 10,000 test samples.

4.4.1 Experimental Setup

We train a residual neural network (ResNet) that takes an input image $x \in \mathbb{R}^{32 \times 32 \times 3}$ and outputs class probabilities over 10 classes. Let $y \in \{1, \dots, 10\}$ denote the ground-truth class label. The network produces logits $f(x) \in \mathbb{R}^{10}$, which are converted into probabilities via the softmax function:

$$p_k(x) := \frac{e^{f_k(x)}}{\sum_{j=1}^{10} e^{f_j(x)}}, \quad k = 1, \dots, 10.$$

The cross-entropy loss for a single example is defined as

$$\ell(x, y) := -\log p_y(x).$$

Substituting the softmax expression, this becomes

$$\ell(x, y) = -f_y(x) + \log \sum_{j=1}^{10} e^{f_j(x)}.$$

Given a mini-batch $\mathcal{B} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{B}|}$ of size $|\mathcal{B}| = 256$, the empirical loss is

$$\mathcal{L}(\theta) := \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \ell(x_i, y_i) = -\frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left[f_{y_i}(x_i) - \log \sum_{j=1}^{10} e^{f_j(x_i)} \right],$$

Training is performed using mini-batches of size 256 for 20 epochs. The data is then augmented and normalized. Hyperparameters, including the step size, damping parameters, and memory size for quasi-Newton updates, are tuned manually for each optimization method. All reported results correspond to the best-performing hyperparameter configurations identified during tuning.

4.4.2 Results

The CIFAR-10 experiments reveal that DG- δ achieves the strongest generalization performance, attaining the highest test accuracy and lowest validation loss. Adam remains a strong baseline

with competitive generalization and faster runtime. And the SIDG methods achieve very low training loss and competitive validation performance, providing a favorable trade-off between optimization quality and runtime. Tables 4.5-4.6 summarize these results.

For each run, the model selected for reporting is the one achieving the best validation accuracy during training, and the corresponding test metrics are evaluated using that saved model.

Table 4.5: CIFAR-10 generalization performance at the best validation epoch (mean $\pm$ standard deviation across seeds, sorted by test accuracy). DG- δ achieves the strongest overall generalization performance, attaining the highest test accuracy and lowest validation loss. Adam also performs strongly with competitive accuracy and lower runtime, while SIDG methods achieve comparable performance with slightly lower accuracy but better training optimization.

Optimizer	Best Epoch	Test Acc	Test Loss	Val Acc	Val Loss
DG-δ	19.8 ± 0.4	0.8342 ± 0.0064	0.4988 ± 0.0264	0.8345 ± 0.0061	0.4816 ± 0.0264
Adam	19.4 ± 0.9	0.8295 ± 0.0110	0.5125 ± 0.0200	0.8322 ± 0.0069	0.4909 ± 0.0200
SIDG	18.8 ± 1.1	0.8201 ± 0.0138	0.5471 ± 0.0320	0.8229 ± 0.0099	0.5176 ± 0.0320
SIDG- δ	18.8 ± 1.1	0.8201 ± 0.0032	0.5444 ± 0.0174	0.8230 ± 0.0028	0.5193 ± 0.0174
DG	19.8 ± 0.4	0.8193 ± 0.0041	0.5327 ± 0.0111	0.8156 ± 0.0048	0.5280 ± 0.0111
MGD- δ	18.8 ± 1.3	0.8184 ± 0.0075	0.5511 ± 0.0144	0.8208 ± 0.0043	0.5241 ± 0.0144
MGD	19.4 ± 0.5	0.8132 ± 0.0095	0.5597 ± 0.0117	0.8188 ± 0.0057	0.5433 ± 0.0117
SGD	19.0 ± 0.7	0.8103 ± 0.0094	0.5715 ± 0.0361	0.8104 ± 0.0103	0.5544 ± 0.0361
L-BFGS	12.2 ± 5.6	0.4711 ± 0.0070	2.1619 ± 0.7586	0.4575 ± 0.0068	2.1551 ± 0.7586

Tables 4.5-4.6 show that DG- δ achieves the strongest overall generalization performance in this experiment, attaining the highest test accuracy, lowest mean loss, the highest validation accuracy, and the lowest validation loss. Adam does comparatively well and has a better runtime. The SIDG-based methods also perform strongly on validation and test metrics, while having better run time than the DG methods.

4.4.3 Discussion

To better understand the training dynamics, Figure 4.6 compares all methods in terms of both training and validation performance over wall-clock time.

The bottom-right figure shows the validation accuracy. Adam (blue curve) and DG- δ (green curve) provide the strongest overall performance, combining rapid initial improvement with the highest final validation accuracy. The SIDG-based methods improve more gradually, but still reach competitive validation accuracy by the end of training.

The top-left figure shows the training loss and highlights a different aspect of the optimization behavior. In particular, the SIDG methods (yellow and grey curves) attain the lowest training

Table 4.6: CIFAR-10 training and runtime performance (mean $\pm$ standard deviation across seeds). SIDG methods achieve the lowest training loss, while DG methods incur higher runtime due to more expensive updates. SGD attains the highest training accuracy, but doesn't have competitive generalization metrics.

Optimizer	Train Acc	Train Loss	Time (s)
DG- δ	0.85824 $\pm$ 0.00473	0.40833 $\pm$ 0.01245	386.93 $\pm$ 1.74
Adam	0.88640 $\pm$ 0.00177	0.39600 $\pm$ 0.00319	322.86 $\pm$ 1.50
SIDG	0.95461 $\pm$ 0.00253	0.18395 $\pm$ 0.00907	349.41 $\pm$ 0.84
SIDG- δ	0.95331 $\pm$ 0.00215	0.18616 $\pm$ 0.00672	365.23 $\pm$ 0.58
DG	0.84494 $\pm$ 0.00138	0.44635 $\pm$ 0.00356	381.45 $\pm$ 1.36
MGD- δ	0.84388 $\pm$ 0.00134	0.44747 $\pm$ 0.00522	322.40 $\pm$ 1.12
MGD	0.88474 $\pm$ 0.00223	0.43318 $\pm$ 0.00530	318.71 $\pm$ 7.09
SGD	0.96396 $\pm$ 0.00175	0.41526 $\pm$ 0.00578	325.84 $\pm$ 11.85
L-BFGS	1.00000 $\pm$ 0.00000	0.00221 $\pm$ 0.00158	389.46 $\pm$ 4.05

losses among all practical methods, indicating that they are highly effective at minimizing the training objective.

For L-BFGS (red curve) while its training loss decreases extremely rapidly (top-left), its validation accuracy remains substantially worse than that of the other methods (bottom-right), and its validation loss increases over time (bottom-left). This behavior suggests severe overfitting.

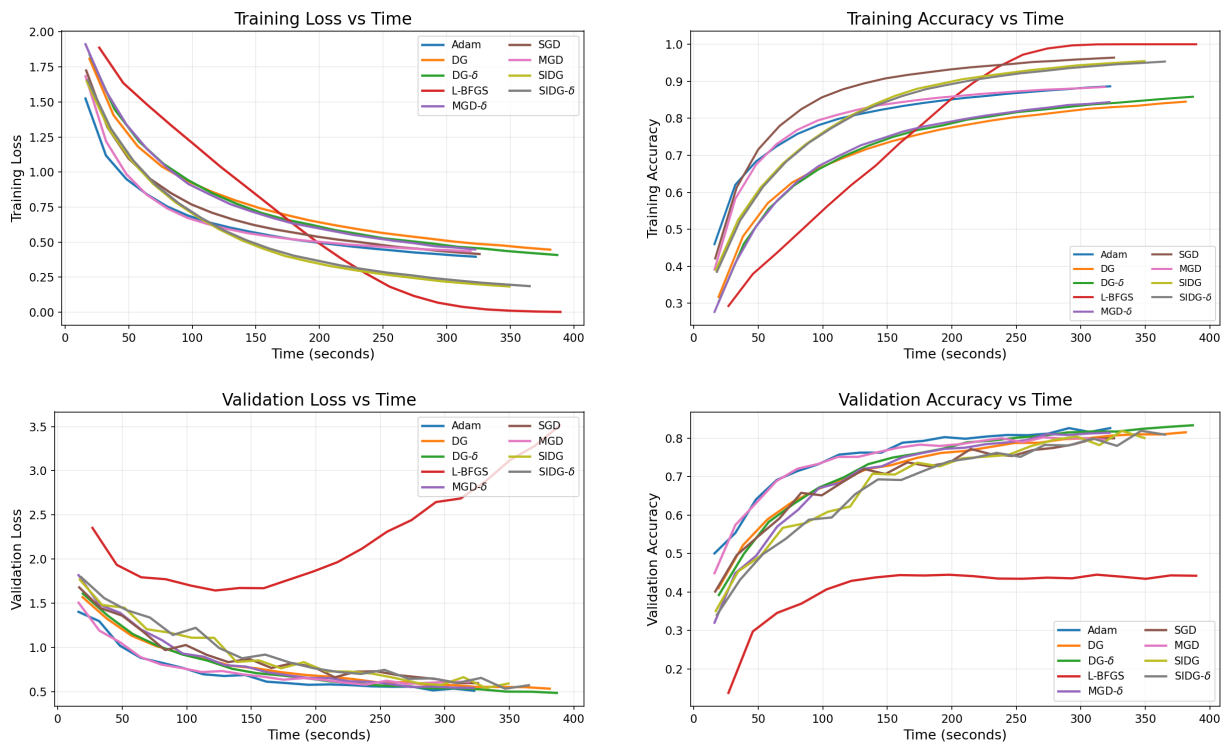


Figure 4.6: Comparison of all methods on CIFAR-10. Top row: training metrics. Bottom row: validation metrics

Chapter 5

Conclusion

5.1 Conclusion

This thesis developed and studied structure-preserving optimization methods by viewing first-order optimization through the framework of dissipative Hamiltonian systems. The starting point was the continuous-time formulation in Section 3, where the Hamiltonian (3.1) combines the objective value and kinetic energy, and the resulting dynamics satisfy the dissipation law (3.12). This perspective motivates optimization methods that do not merely descend the objective, but are designed to inherit the underlying energy-dissipative structure of the continuous system.

From this viewpoint, the classical momentum gradient descent method was derived as a semi-implicit discretization of the dissipative Hamiltonian dynamics (see (3.20), (3.21), and (3.22)). Under the standard assumptions of L -smoothness and compact sub-level sets stated in Section 2.4, Theorem 3.3 gives the nonconvex best-iterate bound

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 = \mathcal{O}\left(\frac{1}{K}\right),$$

with the associated step-size restriction $h < 2\gamma/L$.

The main theoretical development of the thesis was the derivation and analysis of discrete-gradient methods for dissipative Hamiltonian optimization. In Section 3.4, the discrete gradient construction was introduced through the defining properties (3.28)-(3.29), and the resulting DG updates were derived in (3.38)-(3.39). These updates preserve a discrete analog of the continuous dissipation mechanism by construction. The well-posedness analysis showed that the implicit step can be reformulated as a fixed-point problem, and under the contraction condition of Lemma 3.9, the DG update is uniquely defined and the inner fixed-point iteration converges linearly to the solution. This provides a mathematically well-founded, implicit, structure-preserving optimizer.

The convergence analysis of DG established that the method achieves nonconvex rates comparable to those of MGD. In particular, Theorem 3.13 and Corollary 3.14 show that both the discrete gradient and the true gradient residuals satisfy best-iterate $\mathcal{O}(1/K)$ bounds. Moreover, Corollary 3.18 and Theorem 3.19 show that bounded DG iterates admit convergent subsequences and that every limit point is a stationary point of $\mathcal{L}$.

A second contribution of the thesis was the introduction of additional damping in the parameter dynamics. This led to the MGD- δ method in Section 3.3 and the DG- δ method in Section 3.4.3. We prove explicit nonconvex best-iterate bounds for DG- δ , see Theorem 3.16 and Corollary 3.17. However, rigorous convergence rates for MGD- δ are left for future work.

The third major contribution was the development of the semi-implicit discrete-gradient method (SIDG), designed to reduce DG’s computational cost while retaining its key dissipative features. Section 3.5 derived SIDG by linearizing the discrete gradient and replacing the exact Hessian with a sequence of symmetric positive definite matrix approximations leading to the updates (3.90)-(3.91) and, in the analysis, the equivalent form (3.92)-(3.96). Under the spectral bounds on the Hessian approximations, Theorem 3.25 proved the discrete energy estimate (3.105), showing that the energy is non-increasing whenever the step-size condition is satisfied. Corollary 3.26 then provided the explicit admissible range

$$0 < h < \frac{2\gamma}{L - m_B}$$

when $L > m_B$, and no restriction when $L \leq m_B$. This is less restrictive than the MGD condition in Remark B.5, highlighting an important theoretical advantage of SIDG. Beyond energy decay, the thesis established a nonconvex gradient-rate result for SIDG. Theorem 3.30 gives the best-iterate estimate (3.114), showing that SIDG also attains an $\mathcal{O}(1/K)$ stationarity guarantee under the assumptions of Theorem 3.25. Moreover, SIDG is substantially cheaper per iteration than the fully implicit DG method, as demonstrated in the numerical results (see Chapter 4).

The numerical experiments in Chapter 4 demonstrate that these theoretical features translate into useful practical behavior. In the least-squares experiment of Section 4.1, where conditioning could be controlled explicitly, the implicit structure-preserving methods were markedly more robust to ill-conditioning than MGD, and the δ -variants required fewer iterations to reach the prescribed tolerance (see Table 4.1). In the ill-conditioned logistic-regression experiments of Section 4.2, SIDG achieved the best balance between iteration count, runtime, and solution quality among the structure-preserving methods, while remaining competitive with L-BFGS. In the PINN experiment of Section 4.3, SIDG and SIDG- δ were the strongest structure-preserving methods, achieving substantially better iteration efficiency than MGD and DG. However, for this experiment, L-BFGS remained the best-performing method overall. Finally, in the CIFAR-10 experiment of Section 4.4, DG- δ achieved the strongest generalization performance, while SIDG methods attained the lowest training losses and a favorable runtime-performance trade-off. Taken

together, these experiments show that the structure-preserving methods achieve competitive performance on modern machine learning problems.

5.2 Future Work

Several natural directions follow from the present work.

First, a rigorous convergence analysis of MGD- δ remains open. In Section 3.3, the method is introduced through a modified dissipative Hamiltonian system with discrete updates (3.26)-(3.27), but no convergence theorem analogous to Theorem 3.3 is established. Since the numerical results in Section 4.1 show that MGD- δ can substantially outperform MGD in ill-conditioned regimes, particularly when MGD fails to reach the target tolerance, it would be valuable to establish rigorous stationarity guarantees.

A similar analysis is also required for SIDG- δ . In Section 3.5.2, the semi-implicit δ -scheme is derived via (3.120)-(3.122), and Lemma 3.32 establishes well-posedness of the updates. However, no analogue of Theorem 3.25 or Theorem 3.30 is currently available.

Second, all theoretical results in this thesis are deterministic. In practice, however, modern machine learning relies on stochastic gradients computed from mini-batches, as in SGD and Adam, rather than full gradients. A natural direction for future work is to extend the analysis to such stochastic settings by replacing $\nabla_{\theta}\mathcal{L}(\theta_k)$ with a stochastic gradient g_k satisfying conditions such as

$$\mathbb{E}[g_k \mid \theta_k] = \nabla_{\theta}\mathcal{L}(\theta_k), \quad \mathbb{E}\|g_k - \nabla_{\theta}\mathcal{L}(\theta_k)\|^2 \leq \sigma^2,$$

meaning that the stochastic gradient is an unbiased estimator of the true gradient and has bounded variance. Within this setting, it would be important to determine whether the dissipative Hamiltonian framework, the DG method, and the SIDG method admit expected-energy dissipation inequalities of a suitable form, whether one can establish convergence rates in expectation for $\min_{0 \leq k \leq K} \mathbb{E}\|\nabla_{\theta}\mathcal{L}(\theta_k)\|^2$, and how gradient noise interacts with the damping parameters γ and δ . Since the CIFAR-10 experiment in Section 4.4 already employs mini-batch gradients in practice, developing a stochastic theory would bridge the gap between the analysis and the implementation.

Third, the effect of the parameter δ requires more theoretical analysis. In the present work, δ appears through a nonzero R_{11} block in the dissipation matrix, modifying the parameter dynamics directly (see Sections 3.3, 3.4.3 and 3.5.2). Empirically, δ often reduces oscillations and improves robustness in ill-conditioned regimes, especially in least squares. A more complete theory should quantify how δ alters damping, spectral behavior, discrete energy decay, and stability regions, especially on quadratics where more precise derivations may be possible.

Fourth, it would be interesting to study more general dissipation structures with nonzero off-diagonal blocks R_{12} and R_{21} . In Section 3.1, the general symmetric block form

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \quad R = R^\top \succeq 0$$

is noted, but the thesis only develops in detail the cases corresponding to momentum damping alone and additional damping in the parameter dynamics. Allowing R_{12} and R_{21} to be nonzero could produce new couplings between parameter and momentum dissipation, potentially leading to richer classes of optimizers.

Fifth, stronger rates under additional assumptions should be developed for all of the proposed methods. The current analysis establishes nonconvex best-iterate $\mathcal{O}(1/K)$ stationarity guarantees for MGD, DG, DG- δ , and SIDG as seen in Theorem 3.3, Theorem 3.13, Corollary 3.14, Theorem 3.16, Corollary 3.17, and Theorem 3.30. A natural next step is to study convex (see Definition E.3) and μ -strongly convex (see Definition E.4) L -smooth objectives. In these settings, one may expect improved convergence rates, such as linear convergence of iterates or energies in the strongly convex case.

Sixth, there are many choices of inner solves for the DG methods. In the current implementation, the DG and DG- δ updates are computed through fixed-point iterations justified by the contraction arguments in Lemma 3.9 and Lemma 3.15. Although this is mathematically convenient, the experiments show that the cost of these inner solves is the main computational bottleneck for DG. Future work should therefore examine alternative solvers such as relaxed fixed-point schemes Ehrhardt et al. [7]. A key question is whether one can preserve the discrete energy structure while reducing the number of inner iterations and runtime substantially.

Seventh, the SIDG implementation could be improved beyond the current conjugate-gradient-based approach. The present analysis in Section 3.5 assumes that the SIDG linear system is solved exactly, together with L-BFGS based curvature approximations. This raises several open computational questions, including when to update curvature pairs, how to better safeguard the Hessian approximations, and whether alternative approximate solvers can further reduce runtime. Since SIDG already appears to provide the best practical balance between robustness and efficiency across several experiments, improving it could make the method substantially more competitive on larger-scale problems.

Lastly, the current theoretical results assume that the DG fixed-point problem and the SIDG linear system are solved exactly at each outer iteration. In practice, however, both are solved approximately. Extending the analysis to include residual errors from fixed-point iterations or conjugate-gradient iterations would make the theory more faithful to the algorithms actually implemented.

References

- [1] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
- [2] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d’équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.
- [3] Elena Celledoni and Eirik Hoel Høiseth. Energy-preserving and passivity-consistent numerical discretization of port-hamiltonian systems. *arXiv preprint arXiv:1706.08621*, 2017.
- [4] Elena Celledoni, Sølve Eidnes, Brynjulf Owren, and Torbjørn Ringholm. Dissipative numerical schemes on Riemannian manifolds with applications to gradient flows. *SIAM Journal on Scientific Computing*, 40(6):A3789–A3806, 2018. doi: 10.1137/17M1126992.
- [5] Hadi Daneshmand, Jonas Kohler, Aurelien Lucchi, and Thomas Hofmann. Escaping saddles with stochastic gradients. In *International Conference on Machine Learning*, pages 1155–1164. PMLR, 2018.
- [6] Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Póczos. Gradient descent can take exponential time to escape saddle points. *Advances in neural information processing systems*, 30, 2017.
- [7] Matthias J Ehrhardt, Erlend S Riis, Torbjørn Ringholm, and Carola-Bibiane Schönlieb. A geometric integration approach to smooth optimization: foundations of the discrete gradient method. *IMA Journal of Numerical Analysis*, 45(3):1269–1299, 2025.
- [8] Volker Grimm, Robert I McLachlan, David I McLaren, GRW Quispel, and Carola-Bibiane Schönlieb. Discrete gradient methods for solving variational image regularisation models. *Journal of Physics A: Mathematical and Theoretical*, 50(29):295201, 2017.
- [9] Calypso Herrera, Florian Krach, and Josef Teichmann. Local lipschitz bounds of deep neural networks. *arXiv preprint arXiv:2004.13135*, 2020.

- [10] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pages 1724–1732. PMLR, 2017.
- [11] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Conference on learning theory*, pages 1042–1085. PMLR, 2018.
- [12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- [13] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257. PMLR, 2016.
- [14] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.
- [15] Stefano Massaroli, Michael Poli, Federico Califano, Angela Faragasso, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Port-hamiltonian approach to neural network training. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 6799–6806. IEEE, 2019.
- [16] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
- [17] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.
- [18] Peter Ochs, Yunjin Chen, Thomas Brox, and Thomas Pock. ipiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.
- [19] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.
- [20] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.
- [21] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [22] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

- [23] Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling nesterov’s accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17(153):1–43, 2016.
- [24] Gerald Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.
- [25] Arjan Van der Schaft. *L2-gain and passivity techniques in nonlinear control*. Springer, 2000.
- [26] Ashia C Wilson, Benjamin Recht, and Michael I Jordan. A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

Appendices

Appendix A

Background on Lyapunov stability

This section recalls standard results from Lyapunov theory and dynamical systems. See Teschl [24] for more details.

Consider the continuous dynamical system

$$\dot{x} = f(x), \quad x \in D \subset \mathbb{R}^n, \quad (\text{A.1})$$

where $f \in C(D, \mathbb{R}^n)$. For each initial condition $x_0 \in D$ we denote by $x(t)$ the solution satisfying $x(0) = x_0$.

Definition A.1 (Invariant set). A set $S \subset D$ is called **invariant** if

$$x_0 \in S \Rightarrow x(t) \in S \quad \forall t \in \mathbb{R}.$$

It is called **positively invariant** if $x_0 \in S \Rightarrow x(t) \in S, \quad \forall t \geq 0$.

Definition A.2 (Stable equilibrium). Let x^* be an equilibrium of the system $\dot{x} = f(x)$. The equilibrium x^* is called **stable** in the sense of Lyapunov if, for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$\|x(0) - x^*\| < \delta \Rightarrow \|x(t) - x^*\| < \varepsilon \quad \forall t \geq 0.$$

Definition A.3 (Asymptotically stable equilibrium). An equilibrium x^* is called **locally asymptotically stable** if it is Lyapunov stable and there exists $\delta_0 > 0$ such that

$$\|x(0) - x^*\| < \delta_0 \Rightarrow x(t) \rightarrow x^* \quad \text{as } t \rightarrow \infty.$$

Definition A.4 (Lyapunov function). Let x^* be an equilibrium of $\dot{x} = f(x)$. A continuously differentiable function $V : D \rightarrow \mathbb{R}$ is called a **Lyapunov function** if

1. $V(x^*) = 0$ and $V(x) > 0$ for all $x \neq x^*$,
2. $\dot{V}(x) = \nabla V(x)^\top f(x) \leq 0$ for all $x \in D$.

Theorem A.5 (Lyapunov stability theorem). *Let V be a Lyapunov function for the system $\dot{x} = f(x)$. If $\dot{V}(x) \leq 0$ for all $x \in D$, then the equilibrium x^* is stable. If in addition $\dot{V}(x) < 0$ for all $x \neq x^*$, then x^* is locally asymptotically stable.*

In many nonlinear systems the derivative $\dot{V}$ is only negative semidefinite. In such cases convergence can still be established using LaSalle's invariance principle.

Theorem A.6 (LaSalle invariance principle). *Consider $\dot{x} = f(x)$ on a domain $D \subset \mathbb{R}^n$ and let $\Omega \subset D$ be a compact positively invariant set. Suppose $V : D \rightarrow \mathbb{R}$ is a C^1 function satisfying*

$$\dot{V}(x) = \nabla V(x)^\top f(x) \leq 0, \quad \forall x \in \Omega.$$

Define

$$E = \{x \in \Omega : \dot{V}(x) = 0\},$$

and let M be the largest invariant subset of E . Then every solution with initial condition in Ω satisfies

$$\text{dist}(x(t), M) \rightarrow 0, \quad \text{as } t \rightarrow \infty.$$

Appendix B

Convergence analysis of Momentum Gradient Descent

In this section we analyze the convergence of the Momentum Gradient Descent (MGD) iteration introduced in Chapter 3. The following proofs in the MGD method are adapted from Ochs et al. [18], but we present them here in a simplified form that is consistent with the notation and assumptions used throughout this thesis.

Recall that MGD is given by

$$\theta_{k+1} = \theta_k + \beta(\theta_k - \theta_{k-1}) - \alpha \nabla \mathcal{L}(\theta_k), \quad (\text{B.1})$$

where $\alpha = h^2 > 0$ and $0 \leq \beta < 1$ where $\beta = 1 - h\gamma$. For convenience, define

$$d_k := \theta_k - \theta_{k-1}, \quad k \geq 0.$$

Then (B.1) can be written as

$$d_{k+1} = \beta d_k - \alpha \nabla \mathcal{L}(\theta_k). \quad (\text{B.2})$$

Lemma B.1. *Under Assumption 2.5, the iterates generated by (B.1) satisfy*

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \frac{\beta}{\alpha} \langle d_k, d_{k+1} \rangle - \left(\frac{1}{\alpha} - \frac{L}{2} \right) \|d_{k+1}\|^2. \quad (\text{B.3})$$

Proof. Since $\mathcal{L}$ is L -smooth, by the descent lemma 2.8,

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \langle \nabla \mathcal{L}(\theta_k), \theta_{k+1} - \theta_k \rangle + \frac{L}{2} \|\theta_{k+1} - \theta_k\|^2.$$

Since $\theta_{k+1} - \theta_k = d_{k+1}$, this becomes

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \langle \nabla \mathcal{L}(\theta_k), d_{k+1} \rangle + \frac{L}{2} \|d_{k+1}\|^2.$$

From (B.2),

$$\alpha \nabla \mathcal{L}(\theta_k) = \beta d_k - d_{k+1}.$$

Taking the inner product with d_{k+1} gives

$$\langle \nabla \mathcal{L}(\theta_k), d_{k+1} \rangle = \frac{\beta}{\alpha} \langle d_k, d_{k+1} \rangle - \frac{1}{\alpha} \|d_{k+1}\|^2.$$

Substituting this into the smoothness inequality yields (B.3). $\square$

We now define a Lyapunov function and show monotone decay.

Lemma B.2. *Assume that $0 \leq \beta < 1$ and*

$$\alpha < \frac{2(1 - \beta)}{L}. \quad (\text{B.4})$$

Define

$$H_k := \mathcal{L}(\theta_k) + c \|d_k\|^2, \quad c := \frac{1}{2} \left(\frac{1}{\alpha} - \frac{L}{2} \right). \quad (\text{B.5})$$

Then $c > 0$, and

$$\mathcal{E}_{k+1} \leq H_k - \rho (\|d_{k+1}\|^2 + \|d_k\|^2), \quad (\text{B.6})$$

where

$$\rho = c - \frac{\beta}{2\alpha} = \frac{1}{2\alpha} (1 - \beta) - \frac{L}{4} > 0. \quad (\text{B.7})$$

Hence, $\{H_k\}_{k \geq 0}$ is nonincreasing.

Proof. By (B.3),

$$\begin{aligned} \mathcal{E}_{k+1} - H_k &= \mathcal{L}(\theta_{k+1}) - \mathcal{L}(\theta_k) + c(\|d_{k+1}\|^2 - \|d_k\|^2) \\ &\leq - \left(\frac{1}{\alpha} - \frac{L}{2} - c \right) \|d_{k+1}\|^2 + \frac{\beta}{\alpha} \langle d_k, d_{k+1} \rangle - c \|d_k\|^2. \end{aligned}$$

By the definition of c ,

$$\frac{1}{\alpha} - \frac{L}{2} - c = c.$$

Therefore,

$$\mathcal{E}_{k+1} - H_k \leq -c(\|d_{k+1}\|^2 + \|d_k\|^2) + \frac{\beta}{\alpha} \langle d_k, d_{k+1} \rangle.$$

Using the inequality

$$2\langle a, b \rangle \leq \|a\|^2 + \|b\|^2,$$

we obtain

$$\mathcal{E}_{k+1} - H_k \leq - \left(c - \frac{\beta}{2\alpha} \right) (\|d_{k+1}\|^2 + \|d_k\|^2).$$

This proves (B.6). Finally,

$$c - \frac{\beta}{2\alpha} = \frac{1}{2} \left(\frac{1}{\alpha} - \frac{L}{2} \right) - \frac{\beta}{2\alpha} = \frac{1}{2\alpha} (1 - \beta) - \frac{L}{4},$$

which is positive exactly when (B.4) holds. $\square$

Corollary B.3. *Under the assumption that $\mathcal{L}$ is L -smooth and has compact sub-level sets, and assuming (B.4), we have*

$$\sum_{k=0}^{\infty} \|d_k\|^2 < \infty, \quad d_k \rightarrow 0.$$

Moreover, for every $K \geq 0$,

$$\frac{1}{K+1} \sum_{k=0}^K \|d_k\|^2 \leq \frac{\mathcal{E}_0 - \mathcal{L}_{\inf}}{\rho(K+1)}, \quad (\text{B.8})$$

and

$$\min_{0 \leq k \leq K} \|d_k\|^2 \leq \frac{\mathcal{E}_0 - \mathcal{L}_{\inf}}{\rho(K+1)}. \quad (\text{B.9})$$

In particular,

$$\min_{0 \leq k \leq K} \|\theta_k - \theta_{k-1}\|^2 = \mathcal{O}(1/K).$$

Proof. Summing (B.6) from $k = 0$ to K gives

$$\mathcal{E}_{K+1} - \mathcal{E}_0 \leq -\rho \sum_{k=0}^K (\|d_{k+1}\|^2 + \|d_k\|^2).$$

Since

$$\mathcal{E}_{K+1} = \mathcal{L}(\theta_{K+1}) + c\|d_{K+1}\|^2 \geq \mathcal{L}_{\inf},$$

we obtain

$$\rho \sum_{k=0}^K (\|d_{k+1}\|^2 + \|d_k\|^2) \leq \mathcal{E}_0 - \mathcal{L}_{\inf}.$$

Dropping one of the two nonnegative sums yields

$$\rho \sum_{k=0}^K \|d_k\|^2 \leq \mathcal{E}_0 - \mathcal{L}_{\inf}.$$

Dividing by $K + 1$ gives (B.8). Since the series is summable, $d_k \rightarrow 0$. Finally,

$$\min_{0 \leq k \leq K} \|d_k\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|d_k\|^2,$$

which proves (B.9). □

Theorem B.4. *Suppose that $0 \leq \beta < 1$ and*

$$\alpha < \frac{2(1-\beta)}{L}.$$

Let

$$\rho := \frac{1}{2\alpha}(1-\beta) - \frac{L}{4} > 0.$$

Then the iterates $\{(\theta_k, \omega_k)\}_{k \geq 0}$ generated by MGD satisfy

$$\sum_{k=0}^K \|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{2}{\alpha^2} \frac{H_0 - \underline{\mathcal{L}}}{\rho}, \quad (\text{B.10})$$

and therefore

$$\min_{0 \leq k \leq K} \|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{2}{\alpha^2} \frac{H_0 - \underline{\mathcal{L}}}{\rho(K+1)} = \mathcal{O}\left(\frac{1}{K}\right). \quad (\text{B.11})$$

Proof. From the MGD update,

$$d_{k+1} = -\alpha \nabla \mathcal{L}(\theta_k) + \beta d_k,$$

so

$$\alpha \nabla \mathcal{L}(\theta_k) = \beta d_k - d_{k+1}.$$

Hence,

$$\|\nabla \mathcal{L}(\theta_k)\|^2 = \left\| \frac{\beta}{\alpha} d_k - \frac{1}{\alpha} d_{k+1} \right\|^2.$$

Using $\|a - b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we obtain

$$\|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{2}{\alpha^2} (\beta^2 \|d_k\|^2 + \|d_{k+1}\|^2) \leq \frac{2}{\alpha^2} (\|d_k\|^2 + \|d_{k+1}\|^2),$$

since $0 \leq \beta < 1$. Summing from $k = 0$ to K gives

$$\sum_{k=0}^K \|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{2}{\alpha^2} \sum_{k=0}^K (\|d_k\|^2 + \|d_{k+1}\|^2).$$

From the previous Lyapunov estimate,

$$\rho \sum_{k=0}^K (\|d_k\|^2 + \|d_{k+1}\|^2) \leq H_0 - \underline{\mathcal{L}},$$

hence,

$$\sum_{k=0}^K \|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{2}{\alpha^2} \frac{H_0 - \underline{\mathcal{L}}}{\rho}.$$

Finally,

$$\min_{0 \leq k \leq K} \|\nabla \mathcal{L}(\theta_k)\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\nabla \mathcal{L}(\theta_k)\|^2,$$

which yields the result. □

Remark B.5. With $h > 0$, $\alpha = h^2$ and $\beta = 1 - h\gamma$ the step size restriction is

$$\alpha < \frac{2(1 - \beta)}{L} \implies h < \frac{2\gamma}{L}.$$

Appendix C

Convergence analysis of DG- δ

In this section, we provide the proofs of the convergence rates of the DG- δ scheme in Chapter 3.

Theorem C.1 (*Best-iterate rate for DG- δ*). *Under the assumption that $\mathcal{L}$ is L -smooth and has compact sub-level sets, let $\{(\theta_k, \omega_k)\}_{k \geq 0}$ be generated by the DG- δ method with parameters $\delta > 0$, $\gamma > 0$, and $h > 0$. Then, for every $K \geq 0$,*

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\delta h(K+1)}. \quad (\text{C.1})$$

In particular,

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}(\theta_k, \theta_{k+1})\|^2 = \mathcal{O}(1/K).$$

Proof. For the DG- δ method, the dissipation matrix is

$$R = \text{diag}(\delta I_n, \gamma I_n), \quad \delta > 0, \gamma > 0.$$

Hence, by Proposition 3.6,

$$\mathcal{H}_{k+1} - \mathcal{H}_k = -h \bar{\nabla} \mathcal{H}(\xi_k, \xi_{k+1})^\top R \bar{\nabla} \mathcal{H}(\xi_k, \xi_{k+1}).$$

Since

$$\bar{\nabla} \mathcal{H}(\xi_k, \xi_{k+1}) = \begin{bmatrix} \bar{\nabla}_{\theta} \mathcal{L}_k \\ m_k \end{bmatrix}, \quad m_k := \frac{\omega_k + \omega_{k+1}}{2}, \quad (\text{C.2})$$

it follows that

$$\mathcal{H}_{k+1} - \mathcal{H}_k = -h\delta \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 - h\gamma \|m_k\|^2 \leq 0. \quad (\text{C.3})$$

Summing (C.3) from $k = 0$ to K yields

$$\delta h \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 + \gamma h \sum_{k=0}^K \|m_k\|^2 = \mathcal{H}_0 - \mathcal{H}_{K+1}.$$

By lower boundedness we have,

$$\mathcal{H}_{K+1} = \mathcal{L}(\theta_{K+1}) + \frac{1}{2} \|\omega_{K+1}\|^2 \geq \mathcal{L}_{\inf}.$$

Therefore,

$$\delta h \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 \leq \mathcal{H}_0 - \mathcal{L}_{\inf}.$$

Also,

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2,$$

and so

$$\min_{0 \leq k \leq K} \|\bar{\nabla}_{\theta} \mathcal{L}_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\delta h(K+1)}.$$

This proves the result. $\square$

Corollary C.2 (*Best-iterate rate for the true gradient for DG- δ*). Under the assumptions of Theorem 3.16, for every $K \geq 0$,

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 \leq \frac{C(\gamma, \delta, L, h)}{K+1}, \quad (\text{C.4})$$

where

$$C(\gamma, \delta, L, h) := \left(\frac{2 \left(1 + \frac{L\delta h}{2}\right)^2}{\delta h} + \frac{L^2 h}{2\gamma} \right) (\mathcal{H}_0 - \mathcal{L}_{\inf}). \quad (\text{C.5})$$

In particular,

$$\min_{0 \leq k \leq K} \|\nabla_{\theta} \mathcal{L}(\theta_k)\|^2 = \mathcal{O}(1/K).$$

Proof. By Lemma 3.8,

$$\|\bar{\nabla}_{\theta} \mathcal{L}_k - \nabla_{\theta} \mathcal{L}(\theta_k)\| \leq \frac{L}{2} \|\theta_{k+1} - \theta_k\|.$$

From the DG- δ update, recall (C.2)

$$\theta_{k+1} - \theta_k = h(m_k - \delta \bar{\nabla}_{\theta} \mathcal{L}_k),$$

we obtain

$$\|\theta_{k+1} - \theta_k\| \leq h\|m_k\| + \delta h\|\bar{\nabla}_\theta \mathcal{L}_k\|.$$

Hence,

$$\begin{aligned} \|\nabla_\theta \mathcal{L}(\theta_k)\| &\leq \|\bar{\nabla}_\theta \mathcal{L}_k\| + \|\bar{\nabla}_\theta \mathcal{L}_k - \nabla_\theta \mathcal{L}(\theta_k)\| \\ &\leq \|\bar{\nabla}_\theta \mathcal{L}_k\| + \frac{L}{2}\|\theta_{k+1} - \theta_k\| \\ &\leq \left(1 + \frac{L\delta h}{2}\right) \|\bar{\nabla}_\theta \mathcal{L}_k\| + \frac{Lh}{2}\|m_k\|. \end{aligned}$$

Using $(a+b)^2 \leq 2a^2 + 2b^2$, we obtain

$$\|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq 2 \left(1 + \frac{L\delta h}{2}\right)^2 \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 + \frac{L^2 h^2}{2} \|m_k\|^2.$$

Using

$$\min_{0 \leq k \leq K} \|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq \frac{1}{K+1} \sum_{k=0}^K \|\nabla_\theta \mathcal{L}(\theta_k)\|^2,$$

we obtain

$$\min_{0 \leq k \leq K} \|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq \frac{2 \left(1 + \frac{L\delta h}{2}\right)^2}{K+1} \sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 + \frac{L^2 h^2}{2(K+1)} \sum_{k=0}^K \|m_k\|^2.$$

From (C.3), summing from $k = 0$ to K gives

$$\delta h \sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 + \gamma h \sum_{k=0}^K \|m_k\|^2 \leq \mathcal{H}_0 - \mathcal{L}_{\inf}.$$

Therefore,

$$\sum_{k=0}^K \|\bar{\nabla}_\theta \mathcal{L}_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\delta h}, \quad \sum_{k=0}^K \|m_k\|^2 \leq \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{\gamma h}.$$

Substituting these bounds into the previous estimate yields

$$\min_{0 \leq k \leq K} \|\nabla_\theta \mathcal{L}(\theta_k)\|^2 \leq \left(\frac{2 \left(1 + \frac{L\delta h}{2}\right)^2}{\delta h} + \frac{L^2 h}{2\gamma} \right) \frac{\mathcal{H}_0 - \mathcal{L}_{\inf}}{K+1}.$$

This proves (C.4) with $C(\gamma, \delta, L, h)$ given by (C.5). $\square$

The arguments used in the analysis of the DG method extend directly to the DG- δ scheme. In particular, under Assumptions (2.5)-(2.6), one can establish analogous estimates and bounds 3.10-3.12 for the sequences $\{\theta_k\}_{k \geq 0}$ and $\{\omega_k\}_{k \geq 0}$. The proofs follow similarly from the DG proofs and are omitted.

Appendix D

Experiments

The plots in these sections are omitted from the main text to avoid visual clutter and to emphasize the key performance comparisons summarized in the respective sections in Chapter 4

D.1 Regularized Logistic Regression additional plots

D.1.1 Tables for well-conditioned case

We additionally report results for the well-conditioned regime ($\kappa = 1$). In contrast to the ill-conditioned setting discussed in the main text, this regime is numerically stable and does not strongly differentiate between optimization methods.

Table D.1: Regularized logistic regression results for the well-conditioned high-margin case ($m = 4, \kappa = 1$). Results are reported as mean $\pm$ standard deviation across random seeds.

Method	Train Loss	Test Loss	Test Acc	Iter	Time (s)
Adam	5.159e-02 $\pm$ 4.919e-03	1.382e-01 $\pm$ 7.444e-02	0.969 $\pm$ 0.021	225 $\pm$ 0	0.130 $\pm$ 0.004
LBFGS	3.792e-02 $\pm$ 8.131e-03	7.515e-02 $\pm$ 3.993e-02	0.970 $\pm$ 0.019	25 $\pm$ 0	0.024 $\pm$ 0.002
MGD	4.100e-02 $\pm$ 7.055e-03	6.733e-02 $\pm$ 2.984e-02	0.968 $\pm$ 0.017	25 $\pm$ 0	0.012 $\pm$ 0.001
MGD- δ	3.988e-02 $\pm$ 7.240e-03	6.834e-02 $\pm$ 3.238e-02	0.968 $\pm$ 0.017	25 $\pm$ 0	0.013 $\pm$ 0.001
DG	3.920e-02 $\pm$ 8.911e-03	8.954e-02 $\pm$ 5.700e-02	0.969 $\pm$ 0.021	25 $\pm$ 0	0.064 $\pm$ 0.003
DG- δ	4.146e-02 $\pm$ 9.692e-03	1.024e-01 $\pm$ 6.770e-02	0.968 $\pm$ 0.023	25 $\pm$ 0	0.062 $\pm$ 0.003
SIDG	3.994e-02 $\pm$ 7.435e-03	6.771e-02 $\pm$ 3.108e-02	0.968 $\pm$ 0.017	25 $\pm$ 0	0.066 $\pm$ 0.001
SIDG- δ	3.994e-02 $\pm$ 7.435e-03	6.771e-02 $\pm$ 3.108e-02	0.968 $\pm$ 0.017	25 $\pm$ 0	0.112 $\pm$ 0.002

Table D.2: Regularized logistic regression results for the well-conditioned low-margin case ($m = 1, \kappa = 1$). Results are reported as mean $\pm$ standard deviation across random seeds.

Method	Train Loss	Test Loss	Test Acc	Iter	Time (s)
Adam	1.668e-01 $\pm$ 3.021e-02	1.980e-01 $\pm$ 6.251e-02	0.920 $\pm$ 0.027	220 $\pm$ 11	0.118 $\pm$ 0.010
LBFGS	1.667e-01 $\pm$ 3.021e-02	1.980e-01 $\pm$ 6.180e-02	0.919 $\pm$ 0.027	25 $\pm$ 0	0.018 $\pm$ 0.002
MGD	1.669e-01 $\pm$ 3.009e-02	1.998e-01 $\pm$ 6.335e-02	0.919 $\pm$ 0.027	115 $\pm$ 14	0.059 $\pm$ 0.008
MGD- δ	1.669e-01 $\pm$ 3.005e-02	1.990e-01 $\pm$ 6.195e-02	0.920 $\pm$ 0.027	95 $\pm$ 11	0.045 $\pm$ 0.007
DG	1.668e-01 $\pm$ 3.018e-02	1.983e-01 $\pm$ 6.184e-02	0.920 $\pm$ 0.027	60 $\pm$ 14	0.085 $\pm$ 0.015
DG- δ	1.668e-01 $\pm$ 3.015e-02	1.985e-01 $\pm$ 6.149e-02	0.920 $\pm$ 0.027	50 $\pm$ 0	0.077 $\pm$ 0.005
SIDG	1.668e-01 $\pm$ 3.012e-02	1.975e-01 $\pm$ 6.186e-02	0.920 $\pm$ 0.027	25 $\pm$ 0	0.061 $\pm$ 0.002
SIDG- δ	1.668e-01 $\pm$ 3.012e-02	1.975e-01 $\pm$ 6.186e-02	0.920 $\pm$ 0.027	25 $\pm$ 0	0.096 $\pm$ 0.003

In both well-conditioned settings, all methods converge to nearly identical solutions in terms of loss and accuracy, and most methods reach the stopping criterion in a similar number of iterations.

D.2 CIFAR-10 additional plots

For main results, refer to Tables 4.5, 4.6 and Figure 4.6.

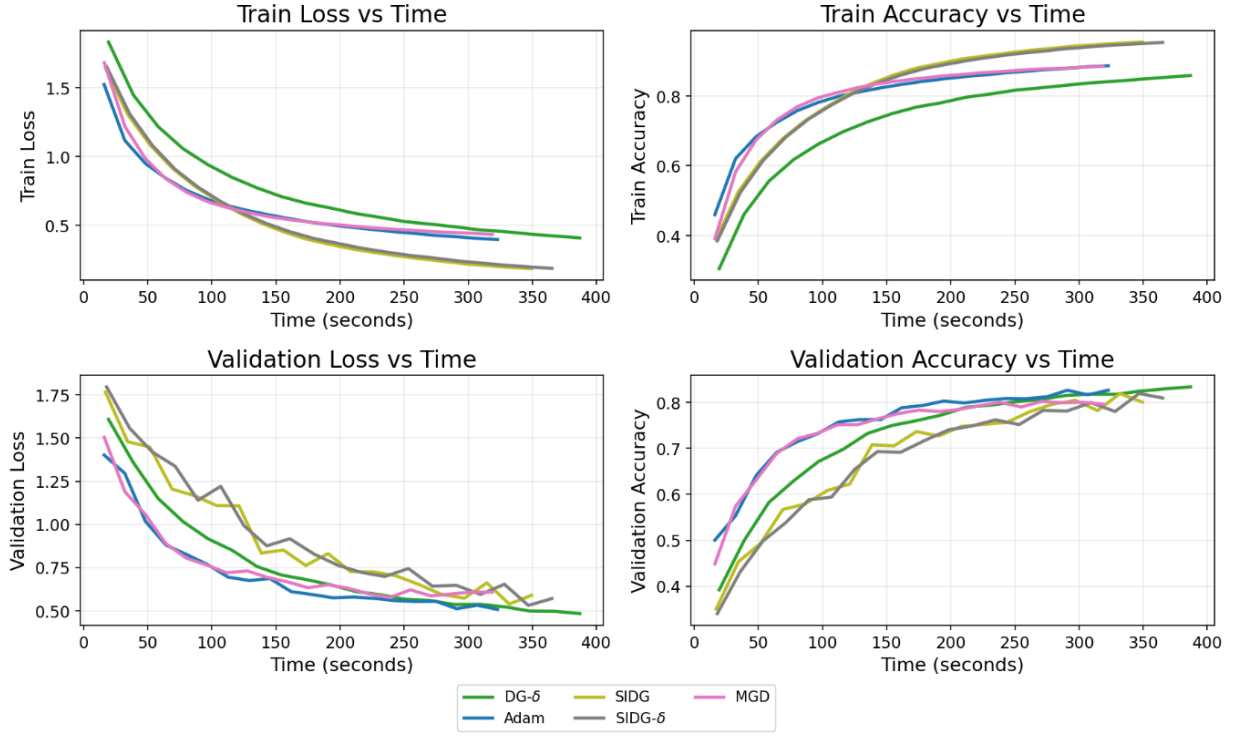


Figure D.1: Comparison of the best methods on CIFAR-10. Top row: training metrics. Bottom row: validation metrics

Overall, these detailed trajectories provide additional evidence supporting the conclusions drawn in Section 4.4.

Appendix E

Additional Material

E.1 Inner Products and Norms

Definition E.1 (Inner Product). An inner product on a vector space $\mathbb{R}^n$ is a mapping $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ such that for all $x, y, z \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$:

1. **Linearity:** $\langle \alpha x + y, z \rangle = \alpha \langle x, z \rangle + \langle y, z \rangle$,
2. **Symmetry:** $\langle x, y \rangle = \langle y, x \rangle$,
3. **Positive definiteness:** $\langle x, x \rangle \geq 0$, with equality if and only if $x = 0$.

In $\mathbb{R}^n$, the standard inner product is given by

$$\langle x, y \rangle = x^\top y = \sum_{i=1}^n x_i y_i.$$

Definition E.2 (Norm). A norm on $\mathbb{R}^n$ is a function $\| \cdot \| : \mathbb{R}^n \rightarrow \mathbb{R}$ such that for all $x, y \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$:

1. **Non-negativity:** $\|x\| \geq 0$, with equality if and only if $x = 0$,
2. **Absolute scalability:** $\|\alpha x\| = |\alpha| \|x\|$,
3. **Triangle inequality:** $\|x + y\| \leq \|x\| + \|y\|$.

The norm induced by an inner product is given by

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

In particular, the Euclidean norm is

$$\|x\|_2 = \sqrt{x^\top x}.$$

E.2 Convexity and Smoothness Definitions

Definition E.3 (Convexity). A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \quad \forall x, y \in \mathbb{R}^n.$$

Definition E.4 (Strong convexity). A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is μ -strongly convex if

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2} \|y - x\|^2, \quad \forall x, y \in \mathbb{R}^n.$$

Definition E.5 (Global Lipschitz continuity). A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called globally L -Lipschitz continuous if there exists a constant $L > 0$ such that

$$\|f(x) - f(y)\| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Definition E.6 (Local Lipschitz continuity on a set). A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called locally Lipschitz continuous on a set $K \subset \mathbb{R}^n$ if there exists a constant $L_K > 0$ such that

$$\|f(x) - f(y)\| \leq L_K \|x - y\|, \quad \forall x, y \in K.$$

Definition E.7 (Global L -smoothness). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable. We say that f is globally L -smooth if its gradient is globally Lipschitz continuous, that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Definition E.8 (Local L -smoothness on a set). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be differentiable. We say that f is locally L -smooth on a set $K \subset \mathbb{R}^n$ if there exists a constant $L_K > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L_K \|x - y\|, \quad \forall x, y \in K.$$

E.3 Proof of the Descent Lemma

This proof is taken from Nesterov [16].

Lemma E.9. *If $\mathcal{L}$ is L -smooth, then for all $\theta, \tilde{\theta} \in \mathbb{R}^n$*

$$\mathcal{L}(\tilde{\theta}) \leq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^\top (\tilde{\theta} - \theta) + \frac{L}{2} \|\tilde{\theta} - \theta\|^2. \quad (\text{E.1})$$

Proof. Let $\theta, \tilde{\theta} \in \mathbb{R}^n$ and define

$$\phi(t) := \mathcal{L}(\theta + t(\tilde{\theta} - \theta)), \quad t \in [0, 1].$$

By the chain rule,

$$\phi'(t) = \nabla_{\theta} \mathcal{L}(\theta + t(\tilde{\theta} - \theta))^{\top} (\tilde{\theta} - \theta).$$

Using the Fundamental Theorem of Calculus,

$$\mathcal{L}(\tilde{\theta}) - \mathcal{L}(\theta) = \int_0^1 \nabla_{\theta} \mathcal{L}(\theta + t(\tilde{\theta} - \theta))^{\top} (\tilde{\theta} - \theta) dt.$$

Adding and subtracting $\nabla_{\theta} \mathcal{L}(\theta)^{\top} (\tilde{\theta} - \theta)$ on the right-hand side of the above equation, we obtain

$$\mathcal{L}(\tilde{\theta}) - \mathcal{L}(\theta) = \nabla_{\theta} \mathcal{L}(\theta)^{\top} (\tilde{\theta} - \theta) + \int_0^1 \left(\nabla_{\theta} \mathcal{L}(\theta + t(\tilde{\theta} - \theta)) - \nabla_{\theta} \mathcal{L}(\theta) \right)^{\top} (\tilde{\theta} - \theta) dt.$$

By Cauchy-Schwarz and L -smoothness,

$$\mathcal{L}(\tilde{\theta}) - \mathcal{L}(\theta) \leq \nabla_{\theta} \mathcal{L}(\theta)^{\top} (\tilde{\theta} - \theta) + \int_0^1 Lt \|\tilde{\theta} - \theta\|^2 dt = \nabla_{\theta} \mathcal{L}(\theta)^{\top} (\tilde{\theta} - \theta) + \frac{L}{2} \|\tilde{\theta} - \theta\|^2.$$

which proves the claim. $\square$

E.4 Well-posedness and Fixed-Point Theory

Definition E.10 (Well-posedness). In this thesis, a one-step implicit update is said to be *well-posed* if, for each current iterate, the next iterate exists uniquely.

Definition E.11 (Fixed point). Let X be a set and let $\Phi : X \rightarrow X$ be a mapping. A point $x^* \in X$ is called a *fixed point* of Φ if

$$\Phi(x^*) = x^*.$$

Definition E.12 (Contraction mapping). Let $(X, \|\cdot\|)$ be a normed vector space. A mapping $\Phi : X \rightarrow X$ is called a *contraction* if there exists a constant $0 \leq \lambda < 1$ such that

$$\|\Phi(x) - \Phi(y)\| \leq \lambda \|x - y\| \quad \forall x, y \in X.$$

Theorem E.13 (Banach Fixed-Point Theorem). *Let $(X, \|\cdot\|)$, be a complete normed vector space, and let $\Phi : X \rightarrow X$ be a contraction mapping. Then:*

1. *There exists a unique fixed point $x^* \in X$ such that*

$$\Phi(x^*) = x^*.$$

2. For any initial point $x^{(0)} \in X$, the iterates defined by

$$x^{(k+1)} = \Phi(x^{(k)}),$$

converge to x^* .

3. The convergence is linear, with

$$\|x^{(k)} - x^*\| \leq \lambda^k \|x^{(0)} - x^*\|.$$

Remark E.14. The Banach fixed-point theorem provides a standard tool for establishing well-posedness of implicit numerical schemes. In particular, if an update can be written as a fixed-point equation

$$x = \Phi(x),$$

and Φ is a contraction, then the next iterate exists, is unique, and can be computed via fixed-point iteration. This is precisely the argument used to establish the well-posedness of the discrete-gradient updates in Chapter 3.