Mathematical Foundations for Computer Science

MOMENTUM IN OPTIMIZATION

Group info:

Bùi Nguyễn Gia Huy (2570201)
Lê Ngọc Minh Thư (2570331)
Nguyễn Xuân Trường (2570526)
Phạm Kiều Nhật Anh (2580805)
Lê Công Minh (2212044)

Presented by: Group 11

Date: December 13, 2025

TABLE OF CONTENTS

01 INTRODUCTION

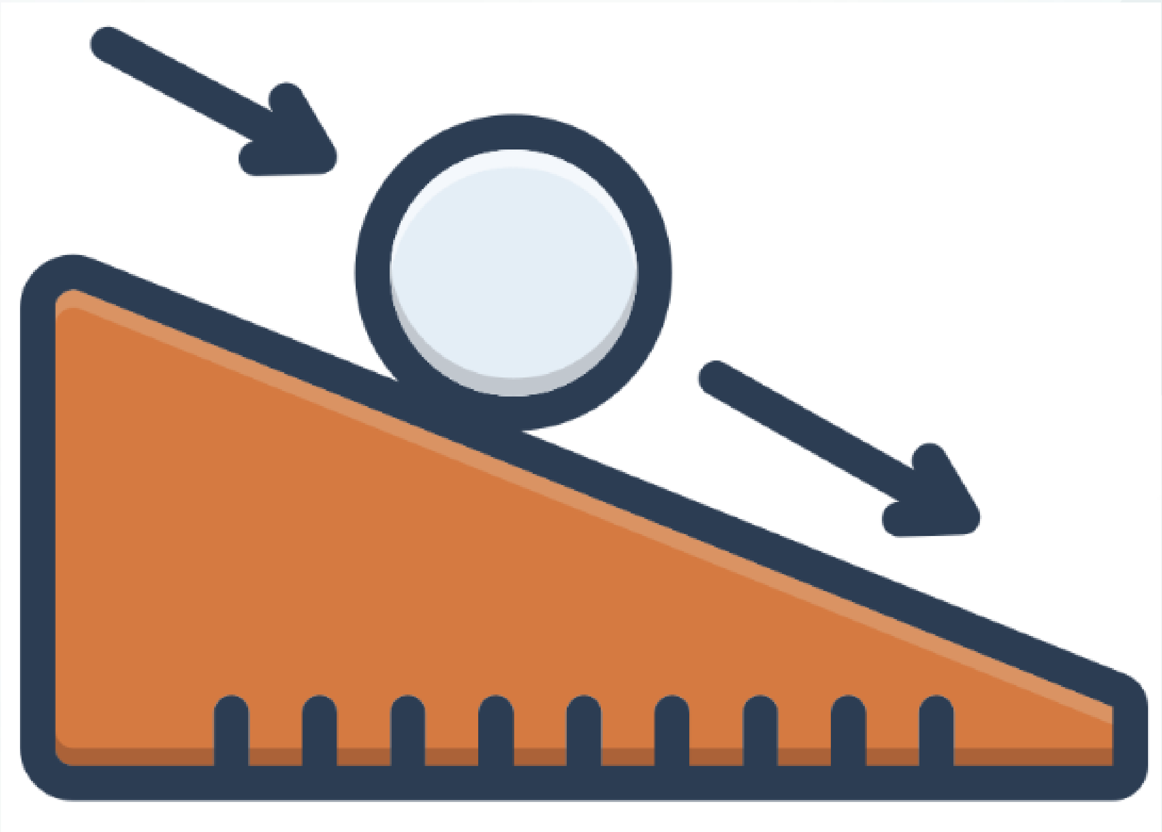
What is momentum?

What is it? And what is the usage?

Momentum introduces a velocity term that accumulates past gradients to smooth the update direction and improve the convergence speed.

Key benefits of Momentum:

- Smooth noisy updates
- Reduces zig-zagging in narrow valleys
- Speeds up convergence in flat regions
- More stable than plain SGD



02 OPTIMIZATION PROBLEM

Optimization Problem

The core task in deep learning optimization is to find the set of parameters θ that minimizes the objective (loss) function f(θ)

$$\min_{\theta \in \mathbb{R}^d} f(\theta)$$

The loss landscape in deep learning

- Highly non-convex
- Contains narrow valleys, plateaus, saddle points
- Gradient direction changes rapidly depending on curvature

Ill-Conditioning Problem

The poor performance of plain gradient descent stems from **ill-conditioning**: the loss surface exhibits dramatically different curvature in different directions.

Consider a simple quadratic function

$$f(x, y) = 0.01x^2 + 5y^2$$

The second derivatives (curvatures) are

- The curvature in the x-direction is $\frac{\partial^2 f}{\partial x^2} = 0.02$ (relatively flat)
- The curvature in the y-direction is $\frac{\partial^2 f}{\partial y^2} = 10$ (much steeper)

These form the constant Hessian matrix :

$$H = \begin{bmatrix} 0.02 & 0 \\ 0 & 10 \end{bmatrix}$$

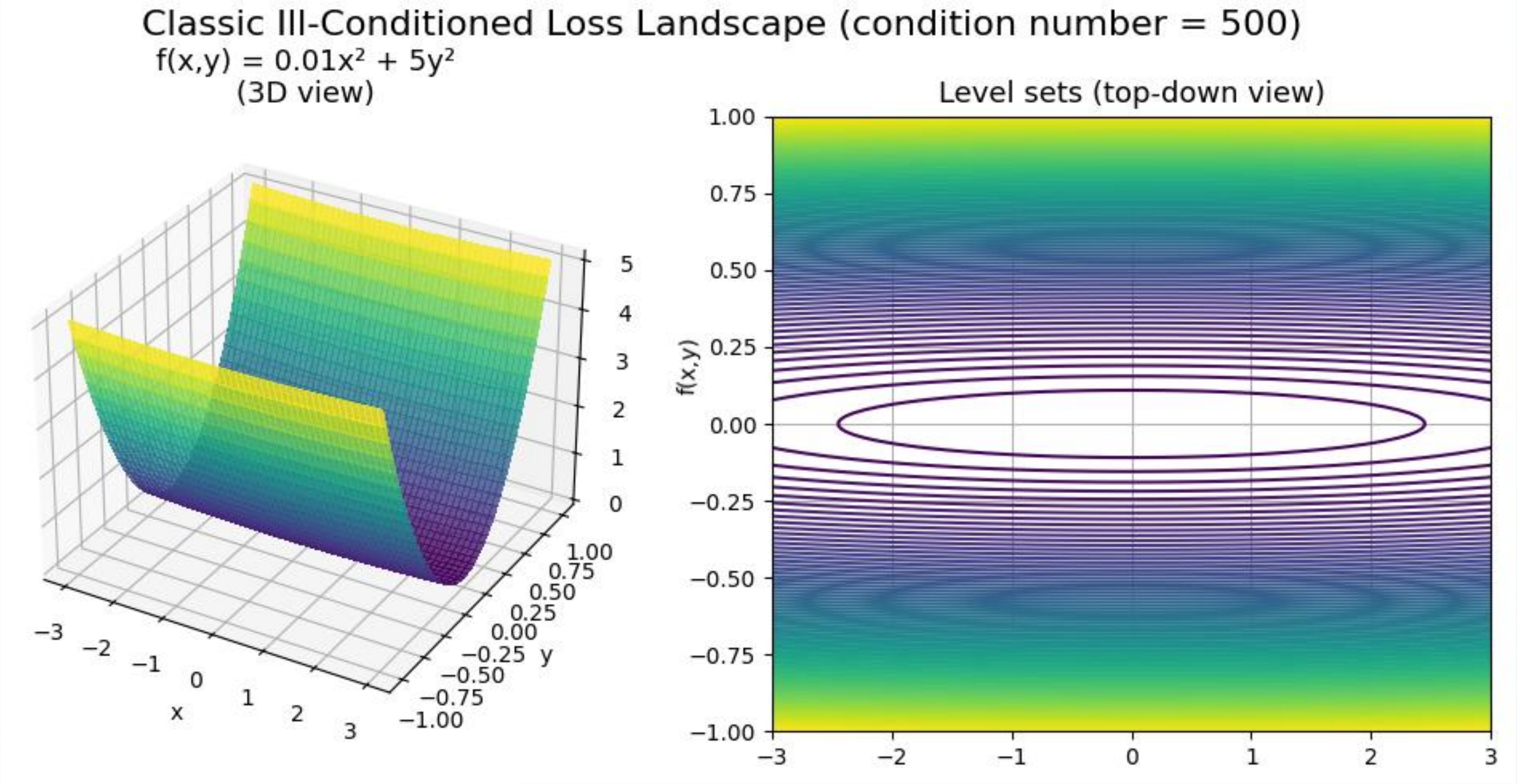
Its eigenvalues $\lambda_1 = 0.02$ and $\lambda_2 = 10$ are exactly the curvatures along the principal axes.
The **condition number** is

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{10}{0.02} = 500$$

Gradient descent dilemma:

- **Small learning rate**: stable in y direction but slow in x-direction
- **Large learning rate**: good progress in x-direction but risks divergence in y-direction

Ill-Conditioning Problem



Geometry of Curvature

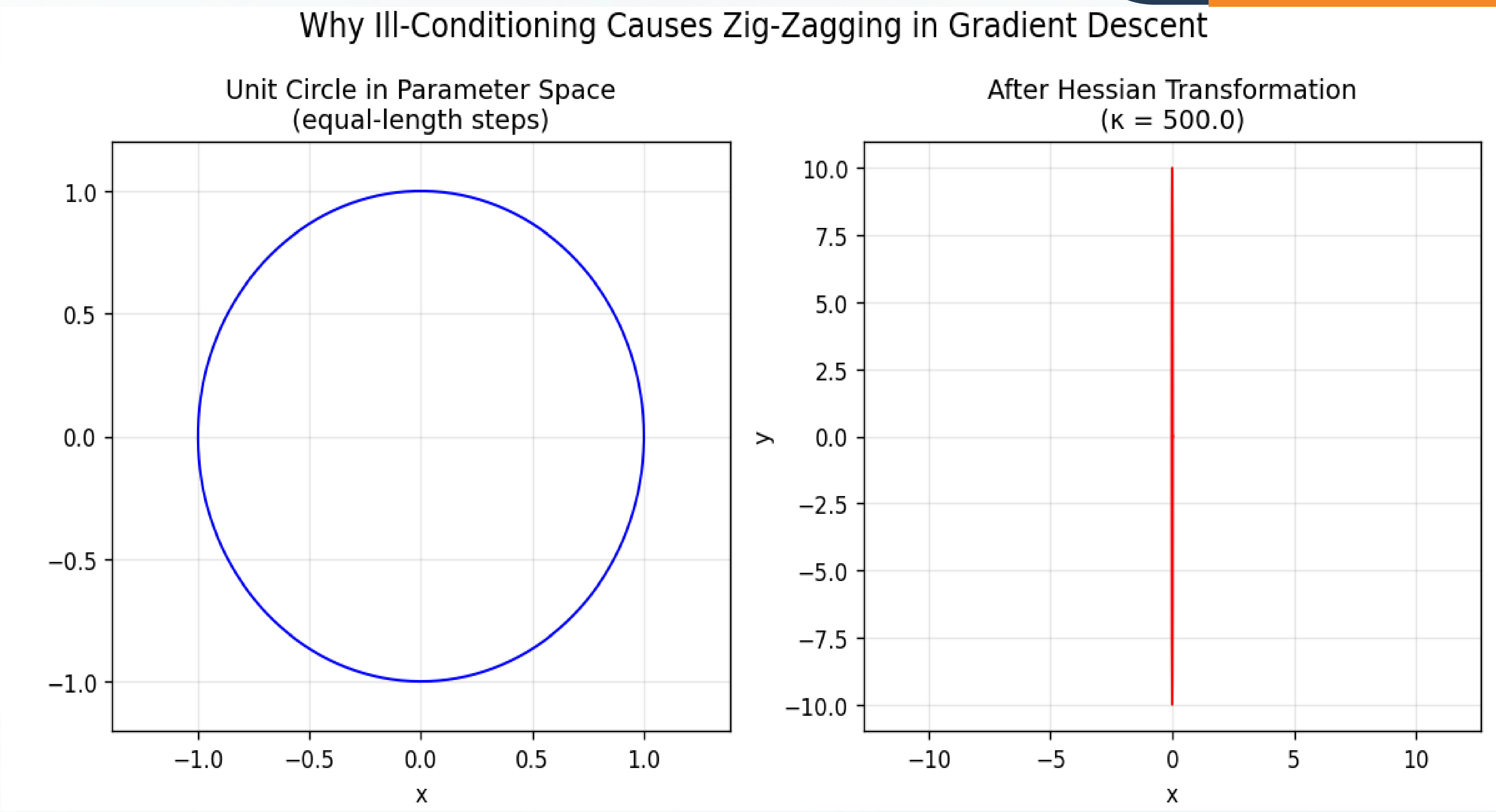
2.2 The Geometry of Curvature

The second-order Taylor expansion tells us how the function changes for a small step $\Delta\mathbf{x}$ of fixed length $\|\Delta\mathbf{x}\| = 1$:

$$f(\mathbf{x}+\Delta\mathbf{x}) \approx f(\mathbf{x}) + \nabla f^\top \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top H \Delta\mathbf{x}.$$

The term $\Delta\mathbf{x}^\top H \Delta\mathbf{x}$ is exactly the **curvature contribution**.

- Left: unit circle → all directions have identical curvature (isotropic case, $H = I$)
- Right: after linear transformation by H → becomes a highly elongated ellipse → the same unit-length step now changes the function value by a factor of up to 500x depending on direction.



Momentum Solution

Damps Oscillation

By accumulating previous updates, it reduces the detrimental effect of high curvature (ill-conditioning) and the resulting zig-zag path.

Maintains Inertia

Maintain velocity (inertia) through flat regions or saddle points, helping to "push" past areas of small gradient.

Reduces Noise Sensitivity

By averaging the gradients over several steps, it reduces the variance caused by the SGD's minibatch sampling, leading to a more stable and reliable descent path.

03 MATHEMATICAL FOUNDATIONS

When to use?

When the optimization landscape has a large disparity in curvature accross direction

Momentum modifies the update by accumulating a running average of past gradients, rather than relying solely on the current gradient. This leads to faster convergence, especially in shallow directions, by smoothing updates and reducing zig-zagging in steep directions.

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$
$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{v}_t.$$

This motivating example sets the stage for the formal **Momentum Update Rule** and its interpretation as **an exponentially weighted (leaky) average** of past gradients

Two main benefits when adding momentum into GD:

- Accelerating convergence faster
- Smoothing the descent direction by incorporating past gradient by controlling β factor

Momentum Update Rule

Momentum:

- Introduces a velocity vector that accumulates past gradients.
- Smooths noisy updates and accelerates movement in stable directions

The standard formulation used in deep learning (Sutskever et al., 2013) is:

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \eta \nabla f(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{v}_{t+1}$$

Where:

- $\beta \in [0, 1)$ is the momentum coefficient (common values: 0.9 or 0.99)
- η is the learning rate
- $v_0 = 0$

Momentum as an Exponential Weighted Average

Momentum Update Rule: $v_{t+1} = \beta v_t + \eta \nabla f(x_t)$

We can expand the recurrence:

$$\begin{aligned} v_{t+1} &= \beta\left(\beta v_{t-1} + \eta \nabla f(x_{t-1})\right) + \eta \nabla f(x_t) \\ &= \eta\Big(\nabla f(x_t) + \beta \nabla f(x_{t-1}) + \beta^2 \nabla f(x_{t-2}) + \cdots\Big) \end{aligned}$$

Expanded Sum Representation:

$$\mathbf{v}_{t+1} = \eta \sum_{k=0}^{t} \beta^k \nabla f(\mathbf{x}_{t-k})$$

This shows that:

- Gradients from recent steps have large weight
- Gradients from older steps decay exponentially
- Momentum is literally a leaky average of past gradients

Effective Memory Length

Total mass of the geometric weights

$$S_\infty = \sum_{k=0}^{\infty} \beta^k = \frac{1}{1-\beta}$$

Momentum is often said to have an effective memory length

$$\text{Effective memory} \approx \frac{1}{1-\beta}$$

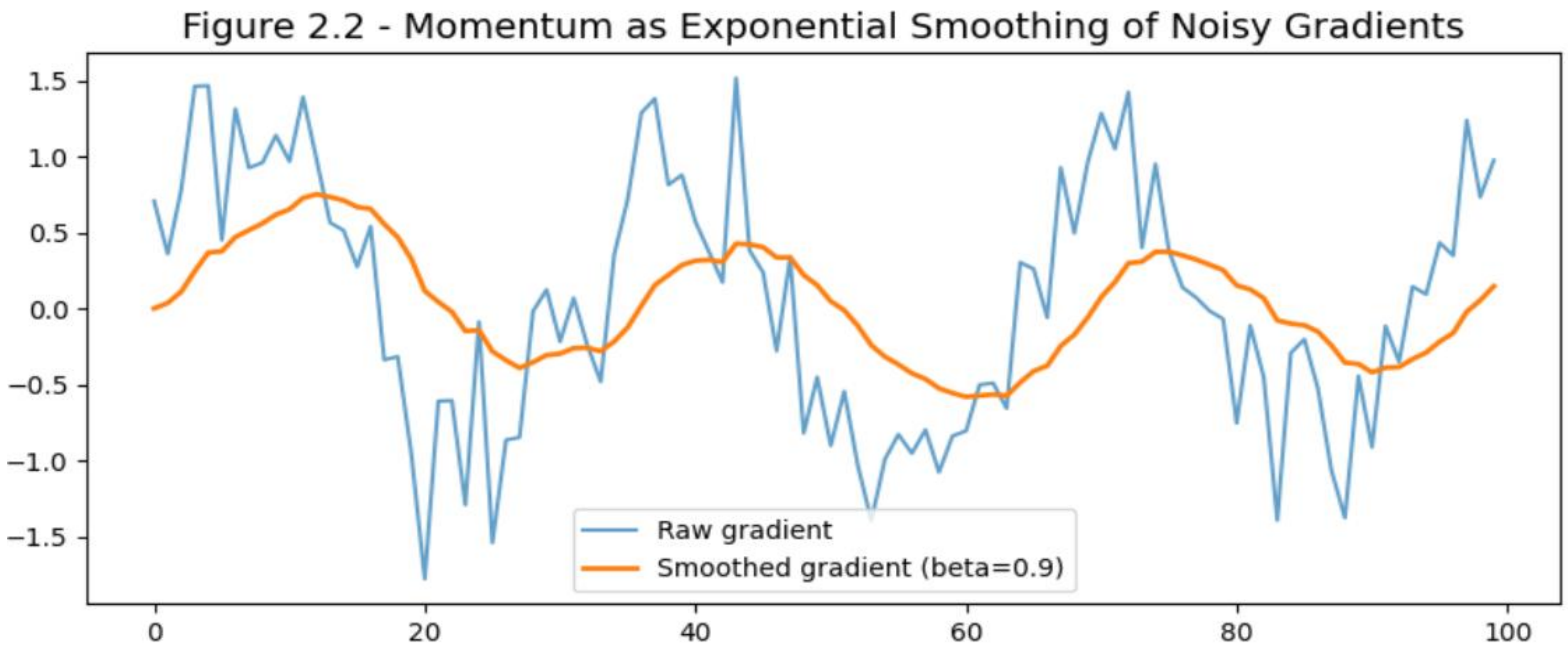
Example:

- $\beta = 0.9 \rightarrow$ remembers ~10 steps
- $\beta = 0.99 \rightarrow$ remembers ~100 steps

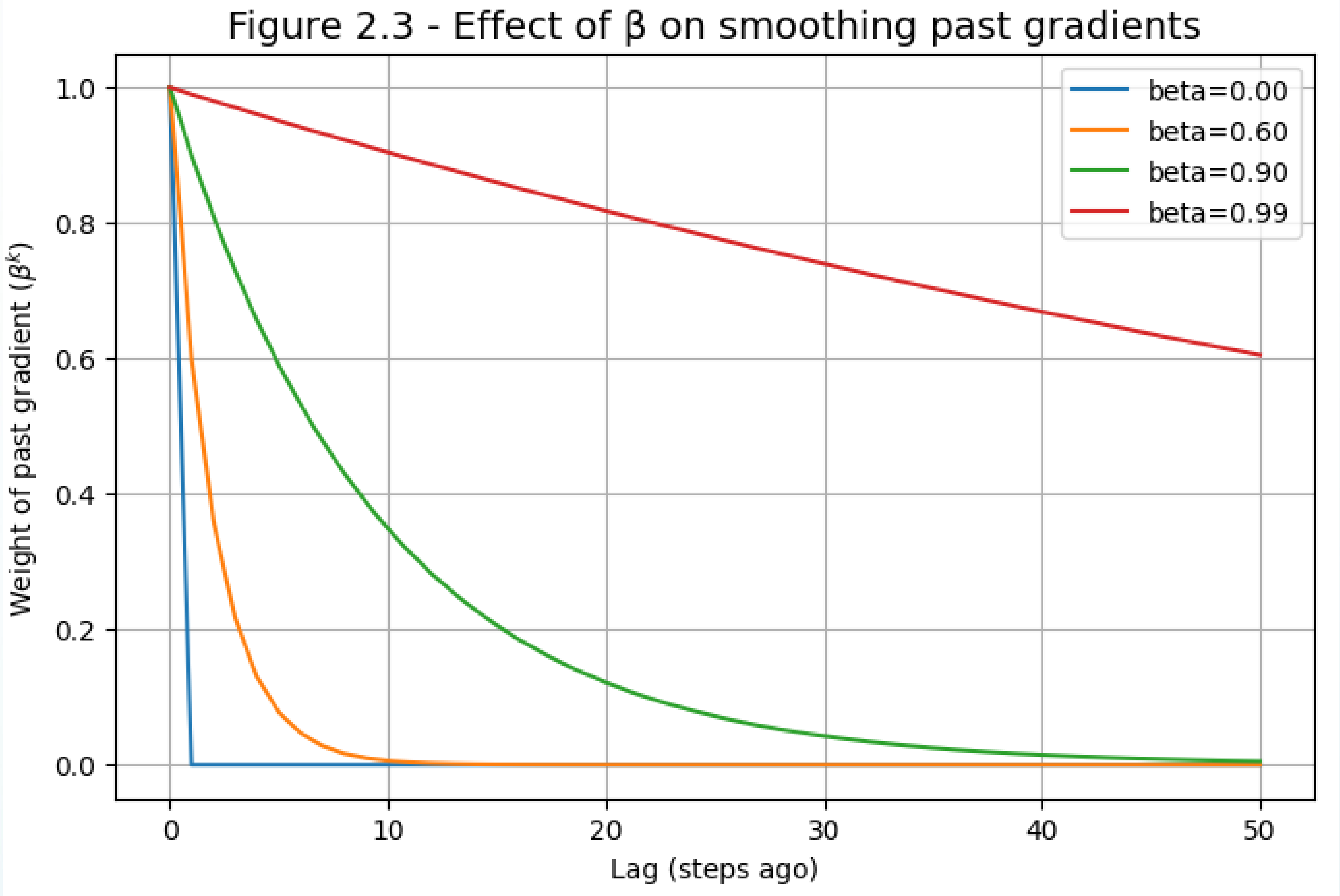
Smoothing Effect

Momentum smooths the gradient signal in two essential ways:

- **Cancels high-frequency noise**
- **Amplifies consistent directions**



How β controls smoothing



Why Momentum accelerates convergence

Recall the expanded form of the momentum velocity:

$$\mathbf{v}_t = \eta \sum_{\tau=0}^{t-1} \beta^\tau \nabla f(\mathbf{x}_{t-1-\tau})$$

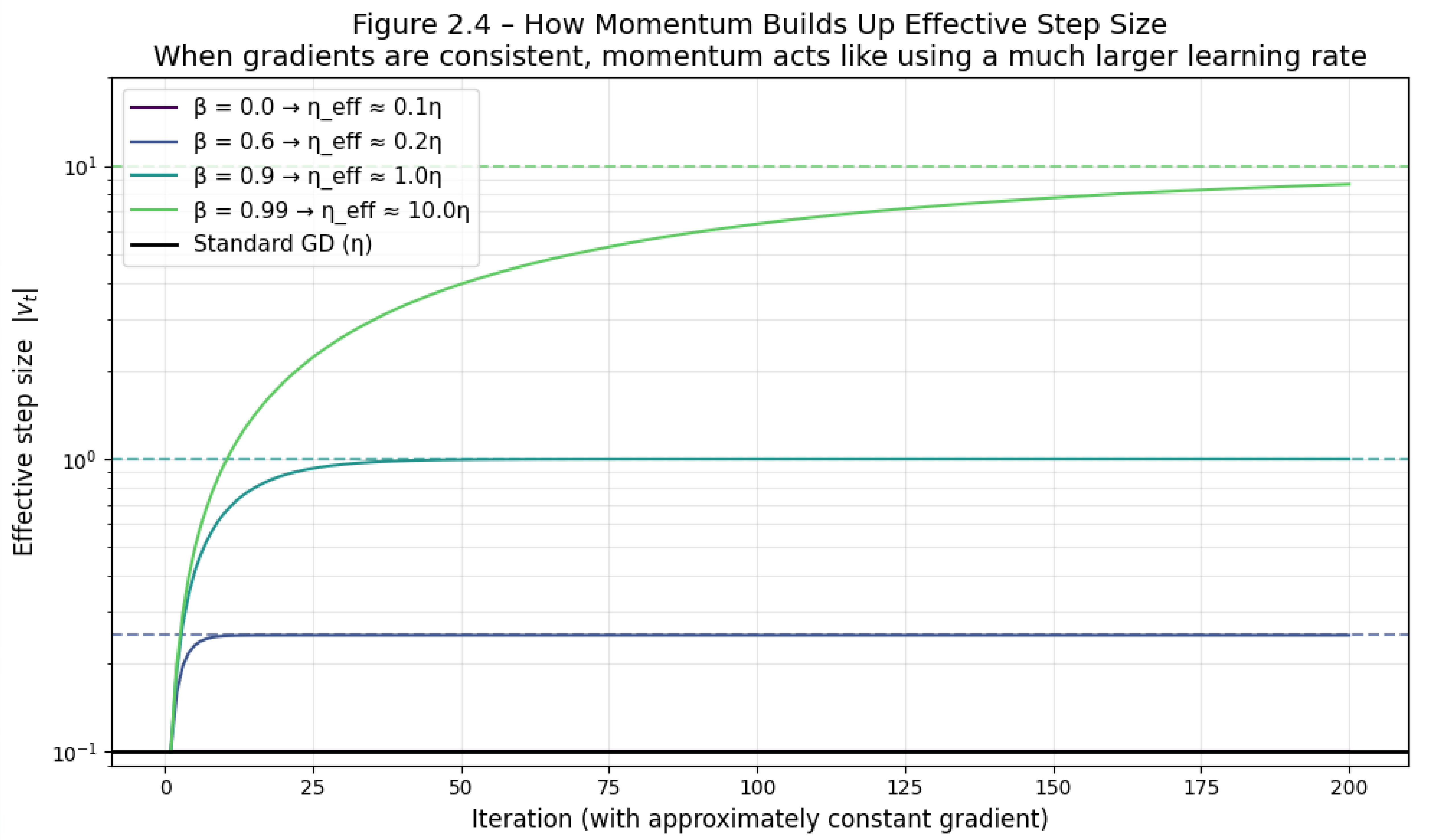
If gradients stay approximately constant, $\nabla f(\mathbf{x}_{t-1-\tau}) \approx g$ (assume g= 1.0) then:

$$v_t \approx \sum_{\tau=0}^{t-1} \beta^\tau \approx \frac{1}{1-\beta} \text{(for large t)}$$

Multiplying by the learning rate η gives an effective step size:

$$\eta_{eff} \approx \frac{\eta}{1-\beta}$$

Why Momentum accelerates convergence



04 THEORETICAL ANALYSIS

How GD & Momentum work in a mathematical way

This part will explain how and why gradient descent and momentum work in a mathematical way.

We need to prove that, in **quadratic convex function**:

- **Gradient Descent optimizes each direction independently**, scaled by how steep that direction is → That's why GD can be fast in steep direction and slow in shallow ones (Ill-conditioned problems)
- **Momentum** addresses this issue by adding velocity based on past gradient and also **acts separately in each direction**.

The Loss is Quadratic Convex

Consider the standard supervised learning setting:

- Data matrix $X \in \mathbb{R}^{n\times m}$ (full column rank), targets $y \in \mathbb{R}^n$
- Parameters $\theta \in \mathbb{R}^m$

The squared error loss is

$$\mathcal{L}(\theta) = \|X\theta - y\|^2 = (X\theta - y)^\top (X\theta - y)$$

Expanding:

$$\mathcal{L}(\theta) = \theta^\top X^\top X\theta - 2\theta^\top X^\top y + y^\top y$$

Define

$$Q = 2X^\top X, \quad \mathbf{c} = -2X^\top y, \quad b = y^\top y$$

Then

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

Why is $Q \succ 0$ (positive definite)?

- Q is symmetric by construction.
- For any $\theta \neq 0$:

$$\theta^\top Q\theta = 2\|X\theta\|^2 > 0 \quad (\text{because X has full column rank})$$

$\rightarrow$ Q is symmetric positive definite (SPD) $\rightarrow$ $\mathcal{L}$ is strongly convex $\rightarrow$ unique global minimum.

Shift Coordinates to the Minimum

Consider loss function: $$\mathcal{L}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{x}^\top \mathbf{c} + b.$$

The minimizer satisfies $\nabla \mathcal{L}(x) = 0$: $Q\mathbf{x}^* + \mathbf{c} = 0 \quad \Rightarrow \quad \boxed{\mathbf{x}^* = -Q^{-1}\mathbf{c}}.$

Define the deviation variable $\mathbf{v} = \mathbf{x} - \mathbf{x}^* \quad \Rightarrow \quad \mathbf{x} = \mathbf{x}^* + \mathbf{v}.$

Substituting $\mathbf{x} = \mathbf{x}^* + \mathbf{v}.$ into the quadratic loss function, we have:

$$\mathcal{L}(\mathbf{x}^* + \mathbf{v}) = \frac{1}{2}(\mathbf{x}^* + \mathbf{v})^\top \mathbf{Q}(\mathbf{x}^* + \mathbf{v}) + (\mathbf{x}^* + \mathbf{v})^\top \mathbf{c} + b$$

Expand:

$$\begin{aligned}
\mathcal{L}(\mathbf{x}^* + \mathbf{v}) &= \frac{1}{2}[(\mathbf{x}^*)^\top Q\mathbf{x}^* + 2(\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top Q\mathbf{v}] + (\mathbf{x}^*)^\top \mathbf{c} + \mathbf{v}^\top \mathbf{c} + b \\
&= \frac{1}{2}(\mathbf{x}^*)^\top Q\mathbf{x}^* + (\mathbf{x}^*)^\top Q\mathbf{v} + \frac{1}{2}\mathbf{v}^\top Q\mathbf{v} + (\mathbf{x}^*)^\top \mathbf{c} + \mathbf{v}^\top \mathbf{c} + b \\
&= \underbrace{(\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top \mathbf{c}}_{\text{linear in } \mathbf{v}} + \underbrace{\frac{1}{2}(\mathbf{x}^*)^\top Q\mathbf{x}^* + (\mathbf{x}^*)^\top \mathbf{c} + b}_{\text{constant}} + \underbrace{\frac{1}{2}\mathbf{v}^\top Q\mathbf{v}}_{\text{quadratic in } \mathbf{v}}
\end{aligned}$$

Shift Coordinates to the Minimum

$$\mathcal{L}(\mathbf{x}^* + \mathbf{v}) = \underbrace{(\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top \mathbf{c}}_{\text{linear in } \mathbf{v}} + \underbrace{\frac{1}{2}(\mathbf{x}^*)^\top Q\mathbf{x}^* + (\mathbf{x}^*)^\top \mathbf{c} + b}_{\text{constant}} + \underbrace{\frac{1}{2}\mathbf{v}^\top Q\mathbf{v}}_{\text{quadratic in } \mathbf{v}}$$

By the linear term, we have:

$$(\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top \mathbf{c} = (\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top(-Q\mathbf{x}^*)$$

Transpose property of a matrix product $(\mathbf{v}^\top Q\mathbf{x}^*)^\top = (\mathbf{x}^*)^\top Q^\top \mathbf{v})$

$$(\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top \mathbf{c} = (\mathbf{x}^*)^\top Q\mathbf{v} + \mathbf{v}^\top(-Q\mathbf{x}^*) = (\mathbf{x}^*)^\top Q\mathbf{v} - (\mathbf{x}^*)^\top Q\mathbf{v} = 0$$

Local quadratic expansion around the minimum

$$\mathcal{L}(\mathbf{x}^* + \mathbf{v}) = \frac{1}{2}\mathbf{v}^\top Q\mathbf{v} + \mathcal{L}(\mathbf{x}^*).$$

S

Eigen-Decomposition & Decoupling

Spectral Decomposition (for real symmetric and positive definite)

$$\mathbf{Q} = \mathbf{O}^\top \Lambda \mathbf{O}, \quad \mathbf{O}^\top \mathbf{O} = I, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m),\ \lambda_i > 0.$$

Where:

- $\mathbf{O}$ is an orthogonal matrix of eigenvectors ($\mathbf{O}^\top \mathbf{O} = I$)
- Λ is a diagonal matrix of eigenvalues ($\lambda_i > 0$), which correspond to the curvature along each principal direction.

Rotate Coordinates:

$$\mathbf{z} = O\mathbf{v} \longrightarrow \boxed{\mathbf{v} = O^\top \mathbf{z}}$$

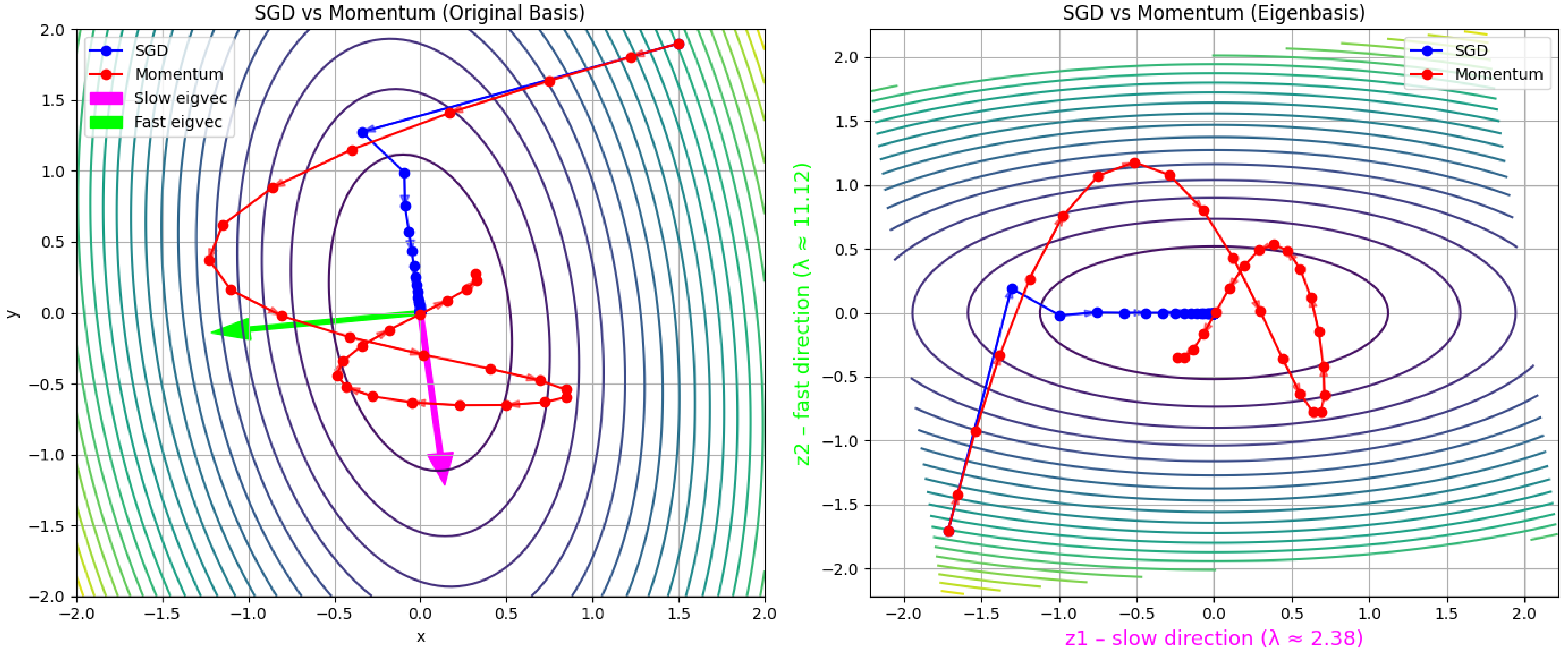
This rotation aligns the coordinate axes with eigenvectors of **Q**.

Quadratic in Rotated basis:

$$h(\mathbf{x}) = \frac{1}{2}\mathbf{v}^\top \mathbf{Q}\mathbf{v} + h(\mathbf{x}^*) = \frac{1}{2}\mathbf{z}^\top \Lambda \mathbf{z} + b^* = \sum_{i=1}^{m} \lambda_i z_i^2 + h(\mathbf{x}^*)$$

Now **Q** becomes **diagonal**, eliminating all cross terms.

Eigen-Decomposition & Decoupling




```
Eigenvalues: [ 2.3839377 11.1160623]
Slow direction (horizontal): λ = 2.3839377008567553
Fast direction (vertical):  λ = 11.116062299143245
```

Dynamics of GD & Momentum in the Eigen-basis

After rotating coordinates:

$$h(\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top \Lambda \mathbf{z}, \qquad \nabla h(\mathbf{z}) = \Lambda \mathbf{z}.$$

Since $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ch eigen-direction evolves independently.

Gradient Descent (1D per eigenvalue) $z_{t,i} = (1 - \eta\lambda_i)\, z_{t-1,i}.$

→ Flat directions (small λi) ⇒ factor ≈ 1 ⇒ extremely slow convergence.

Momentum

$$\begin{aligned} v_{t,i} &= \beta v_{t-1,i} + \eta\lambda_i z_{t-1,i}, \\ z_{t,i} &= z_{t-1,i} - v_{t,i}. \end{aligned} \longrightarrow \begin{bmatrix} z_{t,i} \\ v_{t,i} \end{bmatrix} = \underbrace{\begin{bmatrix} 1-\eta\lambda_i & -1 \\ \eta\lambda_i & \beta \end{bmatrix}}_{A_i} \begin{bmatrix} z_{t-1,i} \\ v_{t-1,i} \end{bmatrix}.$$

→ Contraction factor becomes spectral radius of Ai.

→ Massive acceleration in flat directions, unlike GD.

05 PRACTICAL IMPLEMENTATION

Ill-conditioned quadratic function

Scalar function

Benchmark on MNIST dataset

Ill-Conditioned Quadratic Function

Consider the ill-conditioned quadratic function:

$$f(x_1, x_2) = 0.01\, x_1^2 + 5\, x_2^2.$$

The gradient of this function is:

$$\frac{\partial f}{\partial x_1} = 0.02 x_1, \qquad \frac{\partial f}{\partial x_2} = 10 x_2.$$

With learning rate **η=0.15**, we examine the behavior of standard **Gradient Descent :**

- **Along the x1 direction (small curvature):** Contraction factor is (1 − **η** . 0.02)=0.997

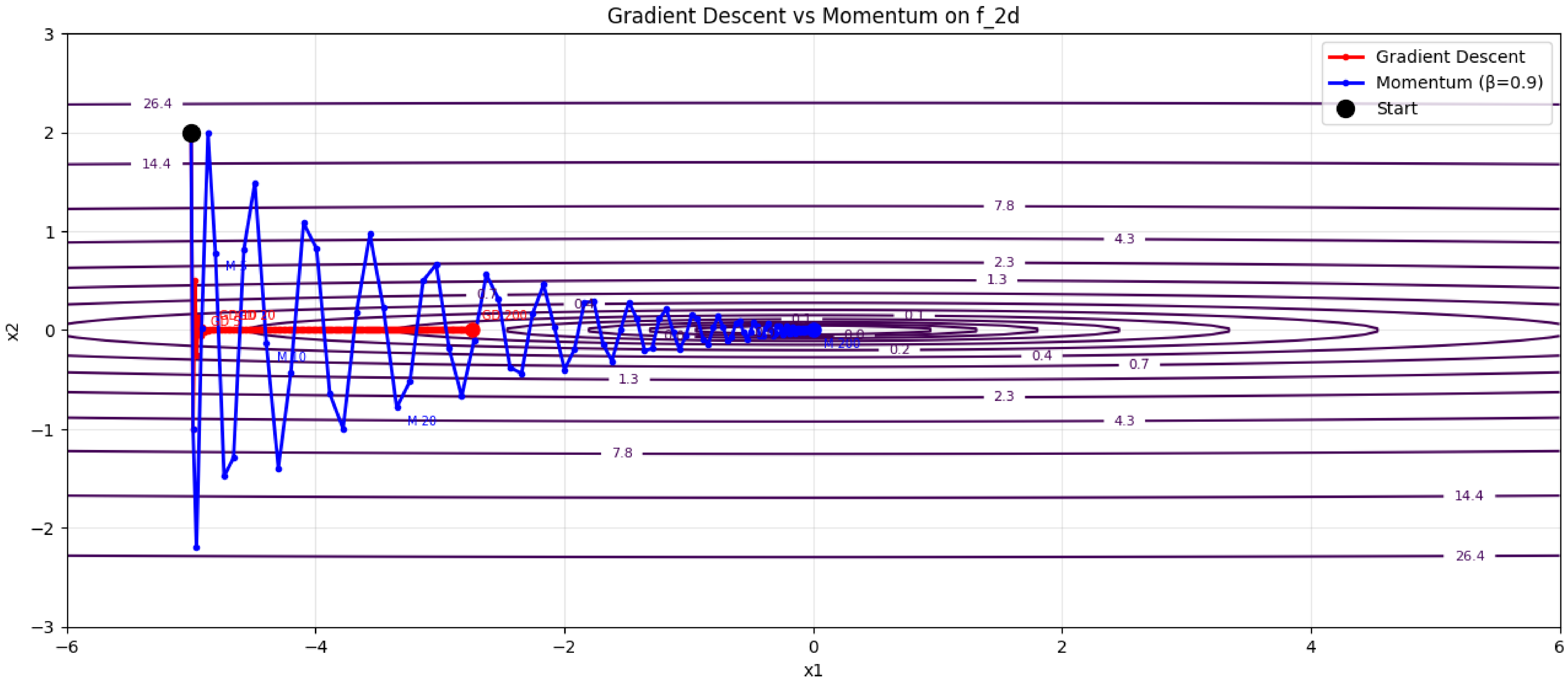
⇒ x1 decreases extremely slowly (requiring thousands of steps to reach 0).

- **Along the x2 direction (large curvature)**: Contraction factor is (1− **η** · 10)= −0.5

⇒ x2 changes sign at each step and exhibits strong oscillations (zig-zag)

To address this issue, we apply Momentum with β=0.9.

Ill-Conditioned Quadratic Function



Iteration	GD Value	Momentum Value
Iter 0	20.250000	20.250000
Iter 5	0.262132	3.251005
Iter 10	0.235439	0.288125
Iter 15	0.228452	2.223874
Iter 20	0.221690	3.176514
Iter 25	0.215128	2.328358
Iter 200	0.075163	0.000000

Table: Function Values During Optimization on f_2d

Scalar Function

Using Gradient descent

We analyze a simple scalar quadratic function:

$$f(x) = \frac{\lambda}{2} x^2$$

Using Gradient descent:

$$x_{t+1} = x_t - \eta\lambda x_t = (1 - \eta\lambda) x_t .$$

This optimization converges at exponential rate since after t steps:

$$x_t = (1 - \eta\lambda)^t x_0$$

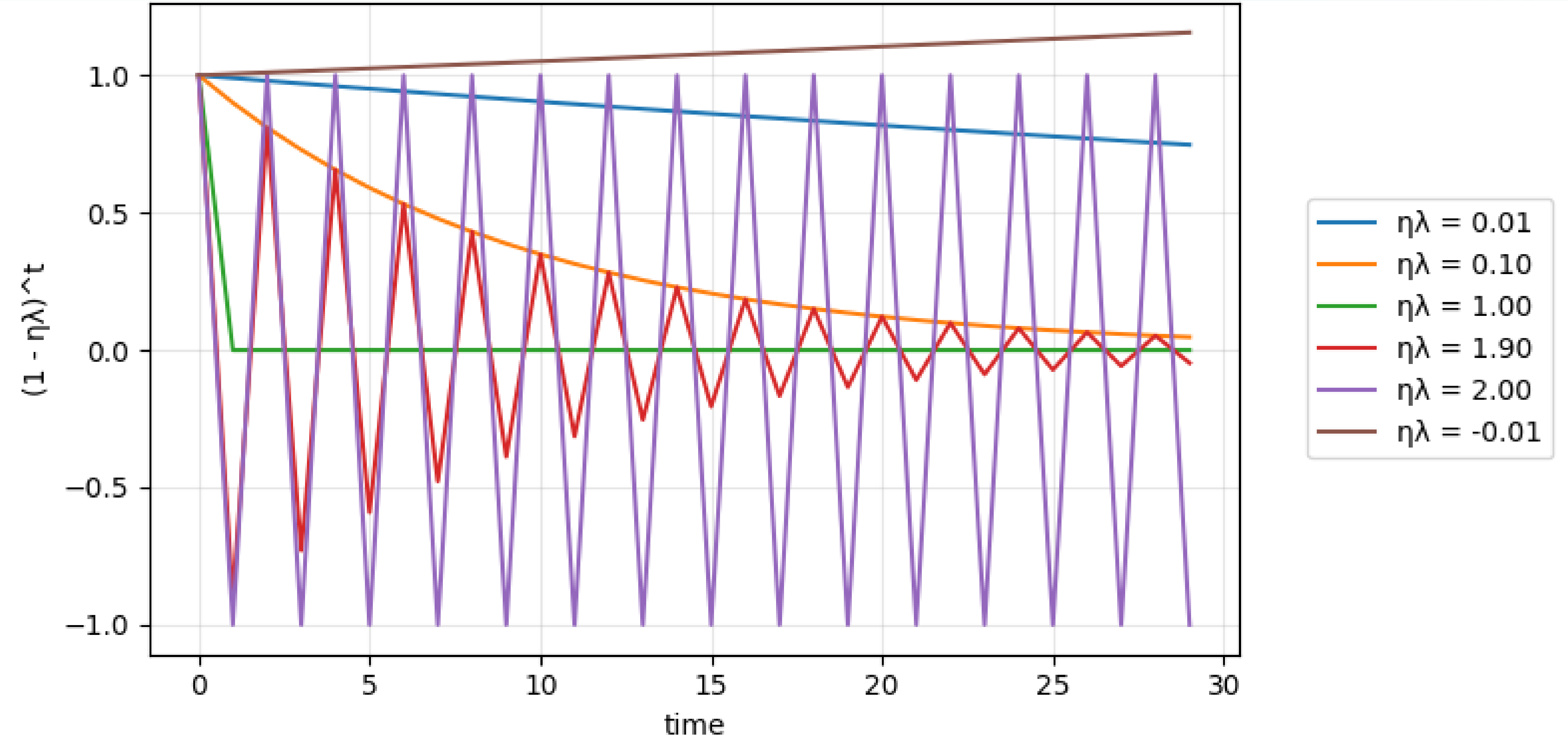
Behavior of convergence depends on the factor $(1-\eta\lambda)$:

- If $|1 - \eta\lambda| < 1 \Leftrightarrow 0 < \eta\lambda < 2$ then $x_t \rightarrow 0$: the algorithm converges
- If $|1 - \eta\lambda| > 1$ then $x \rightarrow \infty$: the algorithm diverges
- If $|1 - \eta\lambda| = 1 \rightarrow \eta\lambda = 2/\eta\lambda = 0$: never converge

Scalar Function

Using Gradient descent

The chart below showing how $\mathbf{x_t/x_0}$ change over time - how fast the variable converges to 0 under different values of $1 - \eta\lambda$.



Scalar Function

Using Momentum

$$v_{t+1} = \beta v_t + \lambda x_t$$
$$x_{t+1} = x_t - \eta v_{t+1} = x_t - \eta\beta v_t + \eta\lambda x_t = -\eta\beta v_t + (1-\eta\lambda) x_t$$

Converting to matrices:

$$\begin{bmatrix} v_{t+1} \\ x_{t+1} \end{bmatrix} = \begin{bmatrix} \beta & \lambda \\ -\eta\beta & (1-\eta\lambda) \end{bmatrix} \begin{bmatrix} v_t \\ x_t \end{bmatrix} = \mathbf{R}(\beta, \eta, \lambda) \begin{bmatrix} v_t \\ x_t \end{bmatrix}.$$

After t iterations:

$$\begin{bmatrix} v_t \\ x_t \end{bmatrix} = \mathbf{R}^t \begin{bmatrix} v_0 \\ x_0 \end{bmatrix}$$

The method converges if and only if the spectral radius is strictly less than 1:

$$\rho(\mathbf{R}) = \max(|\sigma_1|, |\sigma_2|) < 1$$

Stability region for Momentum:

$$0 < \eta\lambda < 2 + 2\beta$$

Training Neural Networks

1. Dataset:

- 60 000 training images, 10 000 testing images
- 28 x 28 grayscale

2. Runtime environment: Google Colab CPU

3. Batch size:

- Mini-batch = 64

4. Benchmark:

- Vanilla SGD: $\beta = 0.0$
- SGD + Momentum: $\beta = 0.9$
- SGD + High Momentum: $\beta = 0.99$

Input (28 x 28)

↓

Flatten → 784 features

↓

Linear 784 → 128

↓

ReLU

↓

Linear 128 → 64

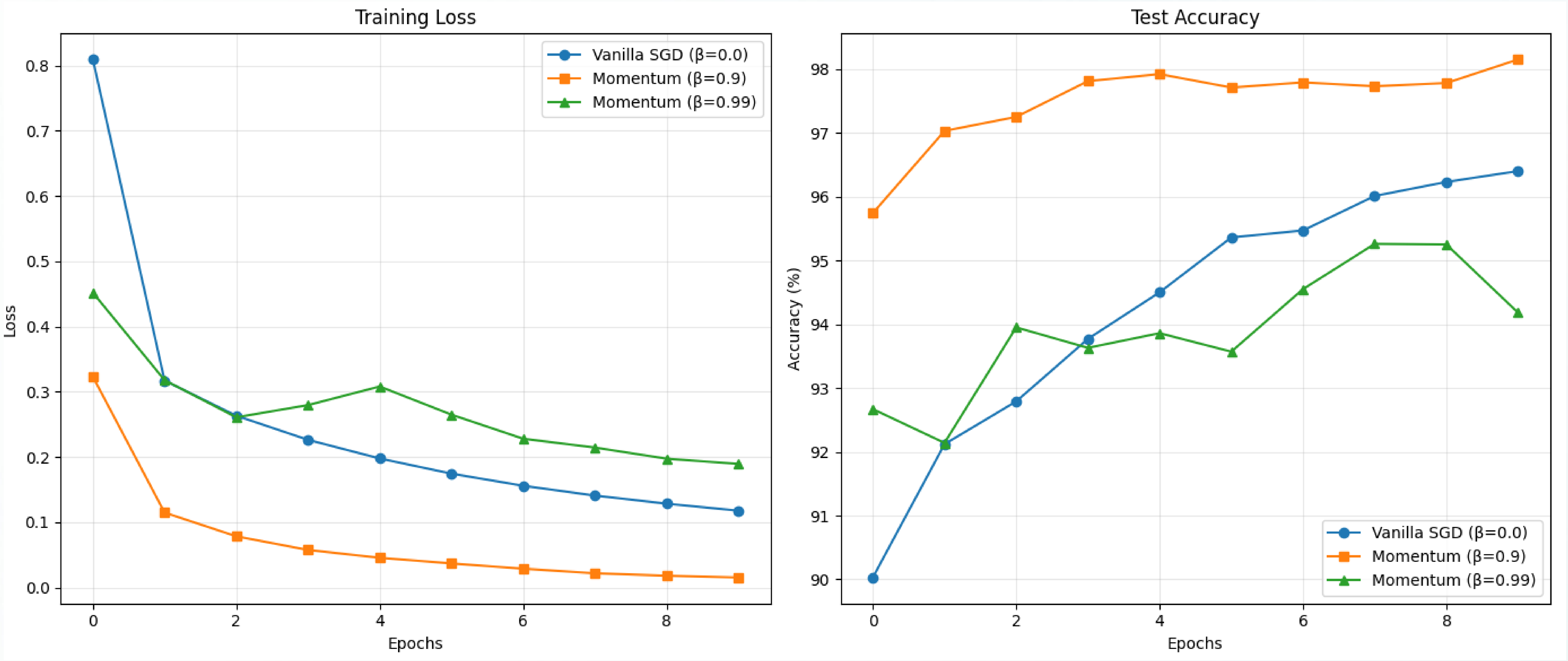
↓

ReLU

↓

Linear 64 → 10 logits

Momentum performs better of time and accuracy



06 EXERCISES

Exercise 1:

Use other combinations of momentum hyperparameters and learning rates and observe and analyze the different experimental results.

$$f(x_1, x_2) = 0.01x_1^2 + 5x_2^2.$$

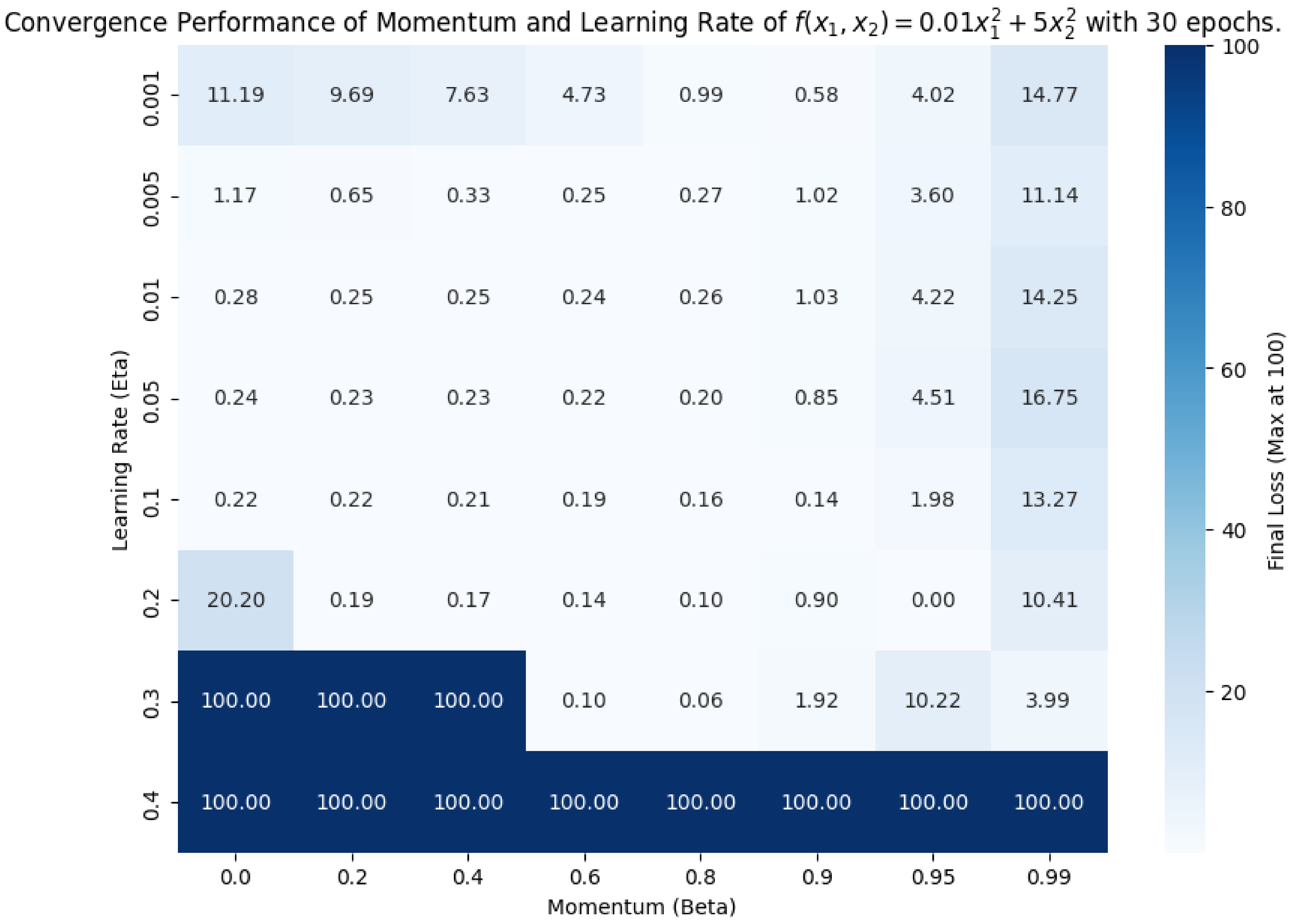
We calculate gradient descent:

$$\frac{\partial f}{\partial x_1} = 0.02x_1, \quad \frac{\partial f}{\partial x_2} = 10x_2.$$

Momentum Update:

$$\begin{cases} v_{1,t+1} = \beta v_{1,t} + 0.02x_{1,t} \\ v_{2,t+1} = \beta v_{2,t} + 10x_{2,t} \\ x_{1,t+1} = x_{1,t} - \eta v_{1,t+1} \\ x_{2,t+1} = x_{2,t} - \eta v_{2,t+1} \end{cases}$$

Exercise 1:


Convergence Performance of Momentum and Learning Rate of $f(x_1, x_2) = 0.01x_1^2 + 5x_2^2$ with 30 epochs.
Learning Rate (Eta)
Momentum (Beta)
Final Loss (Max at 100)
0.001
0.005
0.01
0.05
0.1
0.2
0.3
0.4
0.0
0.2
0.4
0.6
0.8
0.9
0.95
0.99
11.19 9.69 7.63 4.73 0.99 0.58 4.02 14.77
1.17 0.65 0.33 0.25 0.27 1.02 3.60 11.14
0.28 0.25 0.25 0.24 0.26 1.03 4.22 14.25
0.24 0.23 0.23 0.22 0.20 0.85 4.51 16.75
0.22 0.22 0.21 0.19 0.16 0.14 1.98 13.27
20.20 0.19 0.17 0.14 0.10 0.90 0.00 10.41
100.00 100.00 100.00 0.10 0.06 1.92 10.22 3.99
100.00 100.00 100.00 100.00 100.00 100.00 100.00 100.00
100
80
60
40
20

Exercise 2:

Try out gradient descent and momentum for a quadratic problem where you have multiple eigenvalues, i.e.,

$$f(x) = \frac{1}{2}\sum_i \lambda_i x_i^2, e.g. \lambda_i = 2^{-i}$$

Plot how the values of x decrease for the initialization $x_i = 1.$

We calculate gradient descent use the formula:

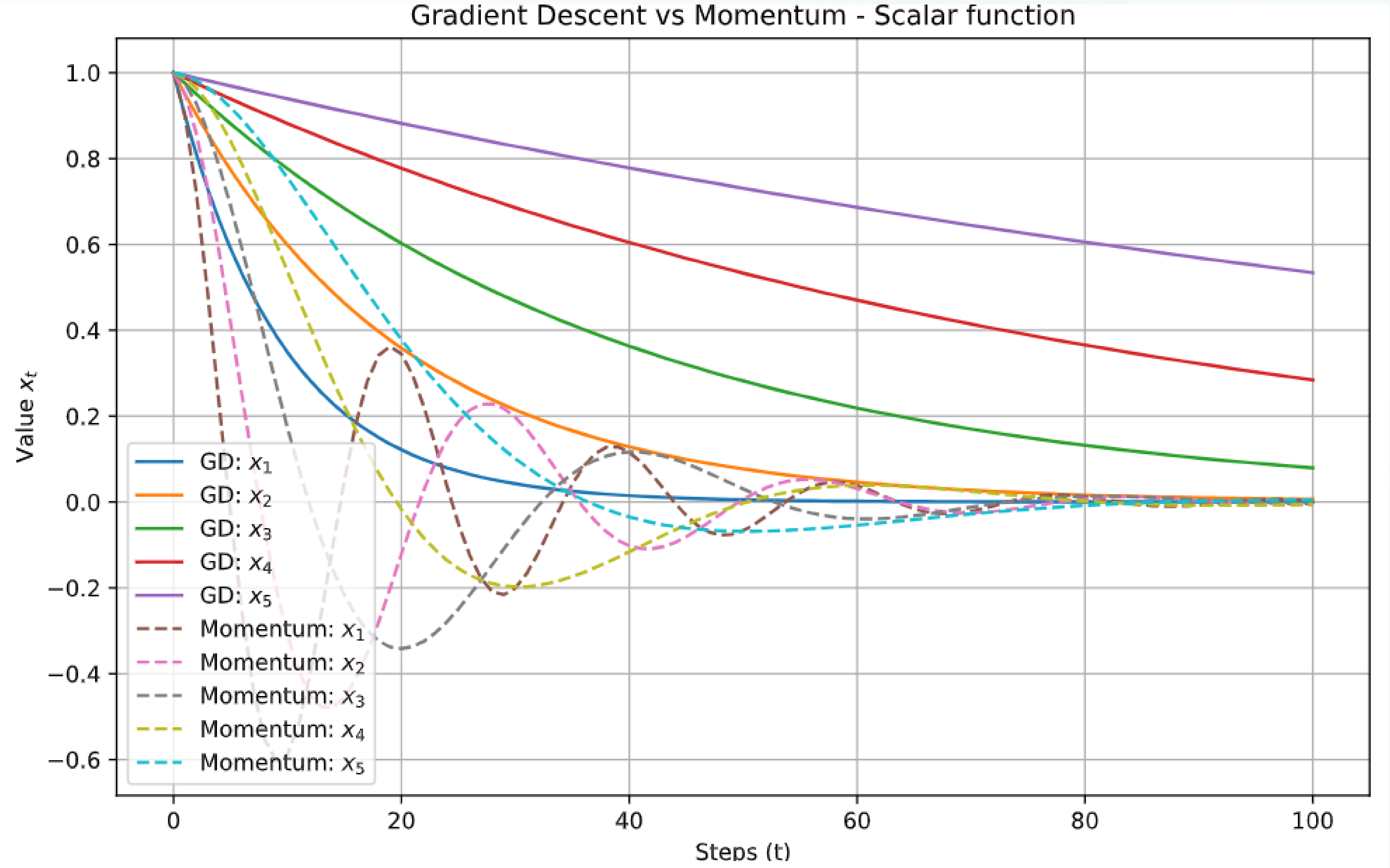
$$x_{i,t+1} = (1 - \eta\lambda_i)x_{i,t}$$

Momentum Update:

$$v_{i,t+1} = \beta v_{i,t} + \lambda_i x_{i,t}$$

$$x_{i,t+1} = x_{i,t} - \eta v_{i,t+1}$$

Exercise 2:


Gradient Descent vs Momentum - Scalar function
Value x_t
Steps (t)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100
GD: x_1
GD: x_2
GD: x_3
GD: x_4
GD: x_5
Momentum: x_1
Momentum: x_2
Momentum: x_3
Momentum: x_4
Momentum: x_5

Exercise 3:

Problem statement: Derive the minimizer and minimum value of the quadratic $h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + b, \quad \mathbf{Q} \succ 0.$

Step 1: Derive with respect to x: $\nabla h(\mathbf{x}) = \mathbf{Q}\mathbf{x} + \mathbf{c}$

Step 2: To find the minimizer, set the gradient to 0 $\mathbf{Q}\mathbf{x} + \mathbf{c} = 0 \quad \Rightarrow \quad \mathbf{x}^* = -\mathbf{Q}^{-1}\mathbf{c}.$

Step 3: Substitute x* into h(x*)

$$\begin{aligned} h(\mathbf{x}^*) &= \frac{1}{2}(\mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^* + \mathbf{c}^\top \mathbf{x}^* + b \\ &= \frac{1}{2}\mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{Q}\mathbf{Q}^{-1}\mathbf{c} - \mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c} + b \\ &= \frac{1}{2}\mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c} - \mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c} + b \\ &= b - \frac{1}{2}\mathbf{c}^\top \mathbf{Q}^{-1}\mathbf{c}. \end{aligned}$$

Exercise 3:

Let take a numerical example to understand more:

$$h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + b, \quad \mathbf{Q} \succ 0.$$

Let $\mathbf{Q} = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} -2 \\ -6 \end{bmatrix}, \quad b = 0.$

From previous result, we have:

First, compute $\det \mathbf{Q}$ (for 2x2 matrix: $ad - bc$):

$$\det \mathbf{Q} = 3 \cdot 2 - 1 \cdot 1 = 6 - 1 = 5.$$

The inverse is

$$\mathbf{Q}^{-1} = \frac{1}{5}\begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}.$$

The minimizer is

$$\mathbf{x}^* = -\mathbf{Q}^{-1}\mathbf{c} = -\frac{1}{5}\begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} -2 \\ -6 \end{bmatrix} = -\frac{1}{5}\begin{bmatrix} 2 \\ -16 \end{bmatrix} = \begin{bmatrix} -0.4 \\ 3.2 \end{bmatrix}.$$

Exercise 3:

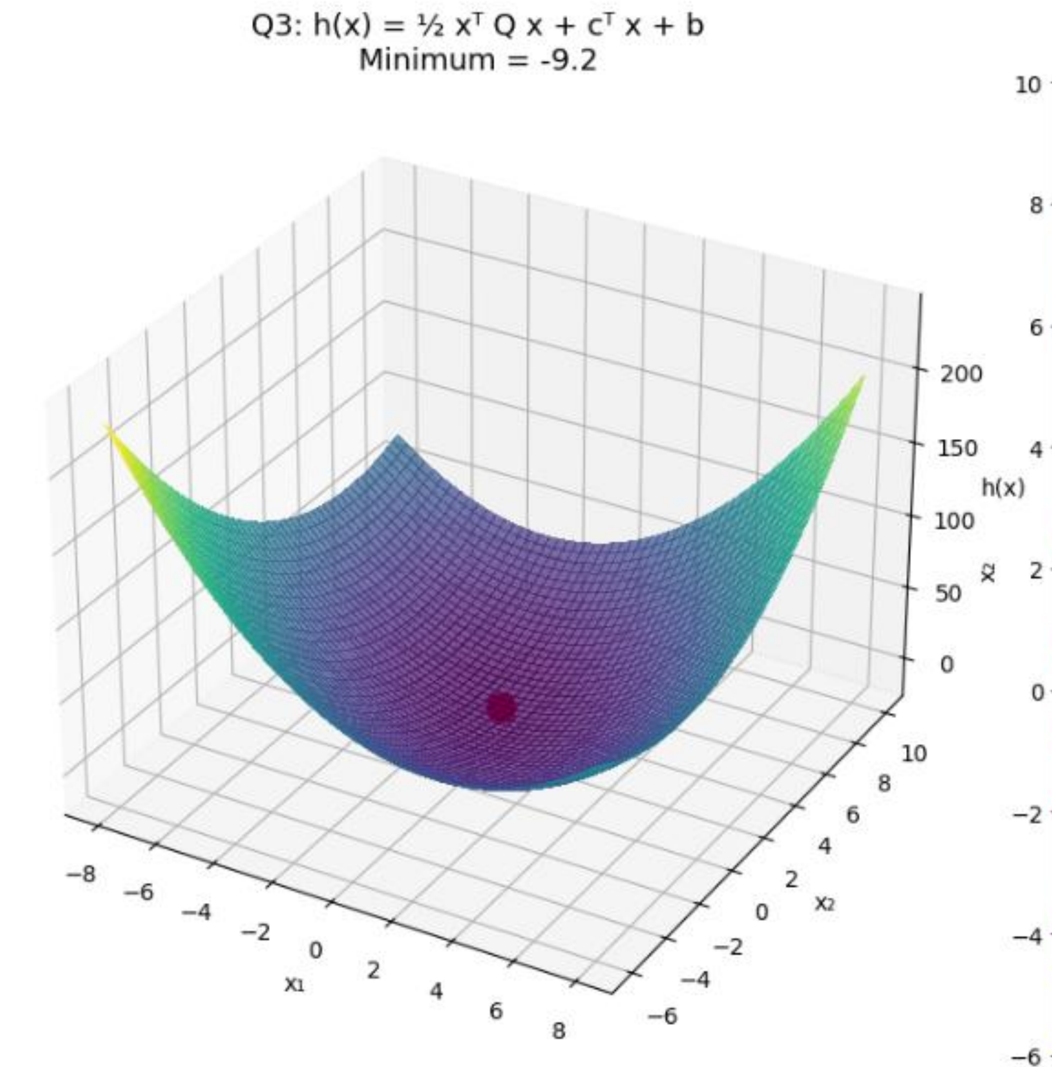
Let take a numerical example to understand more:

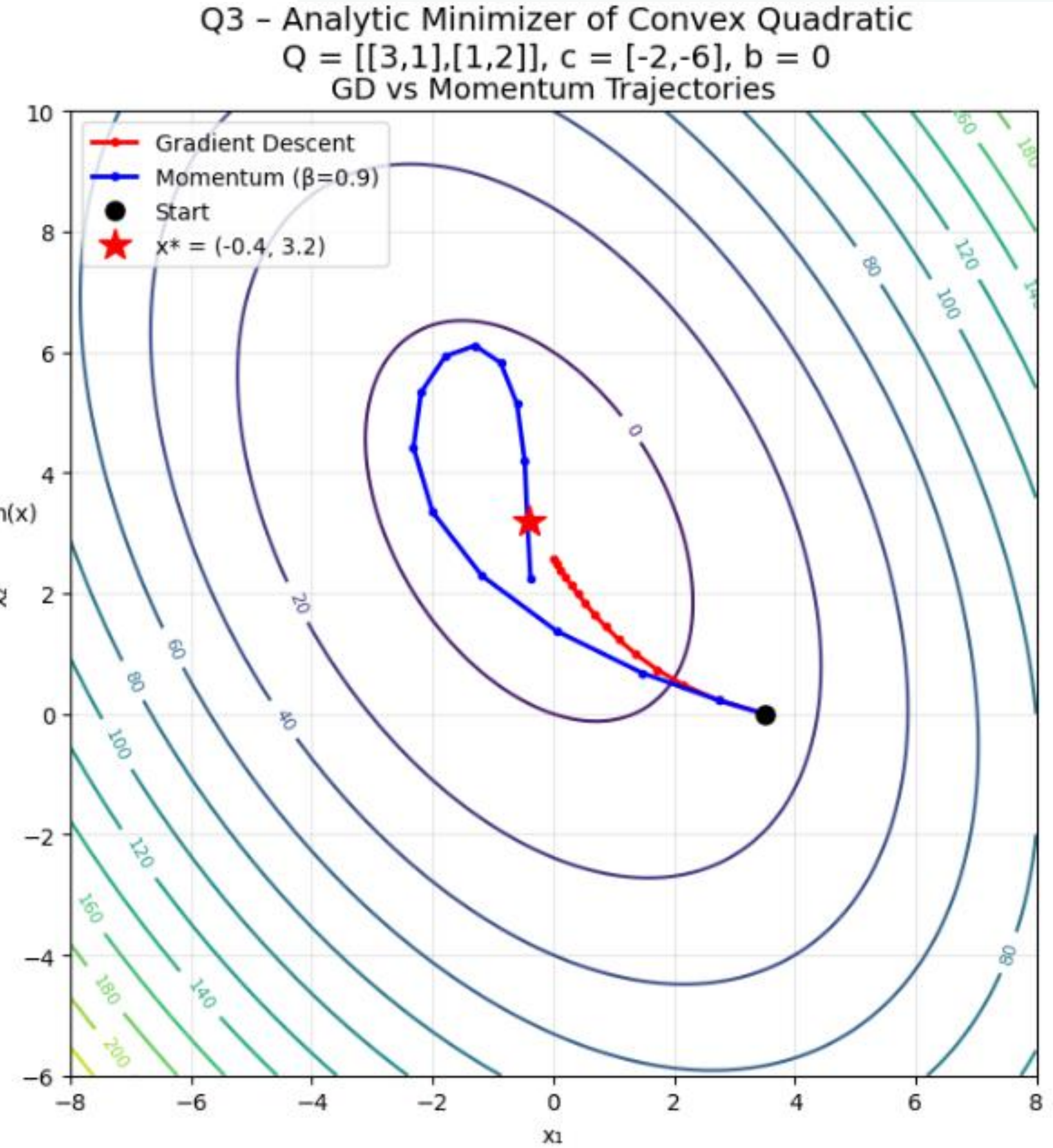
$$h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + b, \quad \mathbf{Q} \succ 0.$$

Replace the result into the origin function to find minimum, we have:

$$\mathbf{Q}^{-1}\mathbf{c} = \frac{1}{5}\begin{bmatrix} 2 \\ -16 \end{bmatrix} = \begin{bmatrix} 0.4 \\ -3.2 \end{bmatrix}, \quad \mathbf{c}^\top(\mathbf{Q}^{-1}\mathbf{c}) = (-2)(0.4) + (-6)(-3.2) = 18.4.$$

Thus, $h(\mathbf{x}^*) = -\frac{1}{2} \cdot 18.4 = -9.2.$





Exercise 4:

Problem statement: What changes when we perform stochastic gradient descent with momentum?

Stochastic Gradient Descent (SGD) updates parameters using noisy gradients from one sample (or a mini-batch).

- **Without Momentum:** $$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} f(\mathbf{x}_t)$$
- **With Momentum:** $$v_{t+1} = \beta v_t - \nabla_{\mathbf{x}} f(\mathbf{x}_t)$$ $$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta v_{t+1}$$

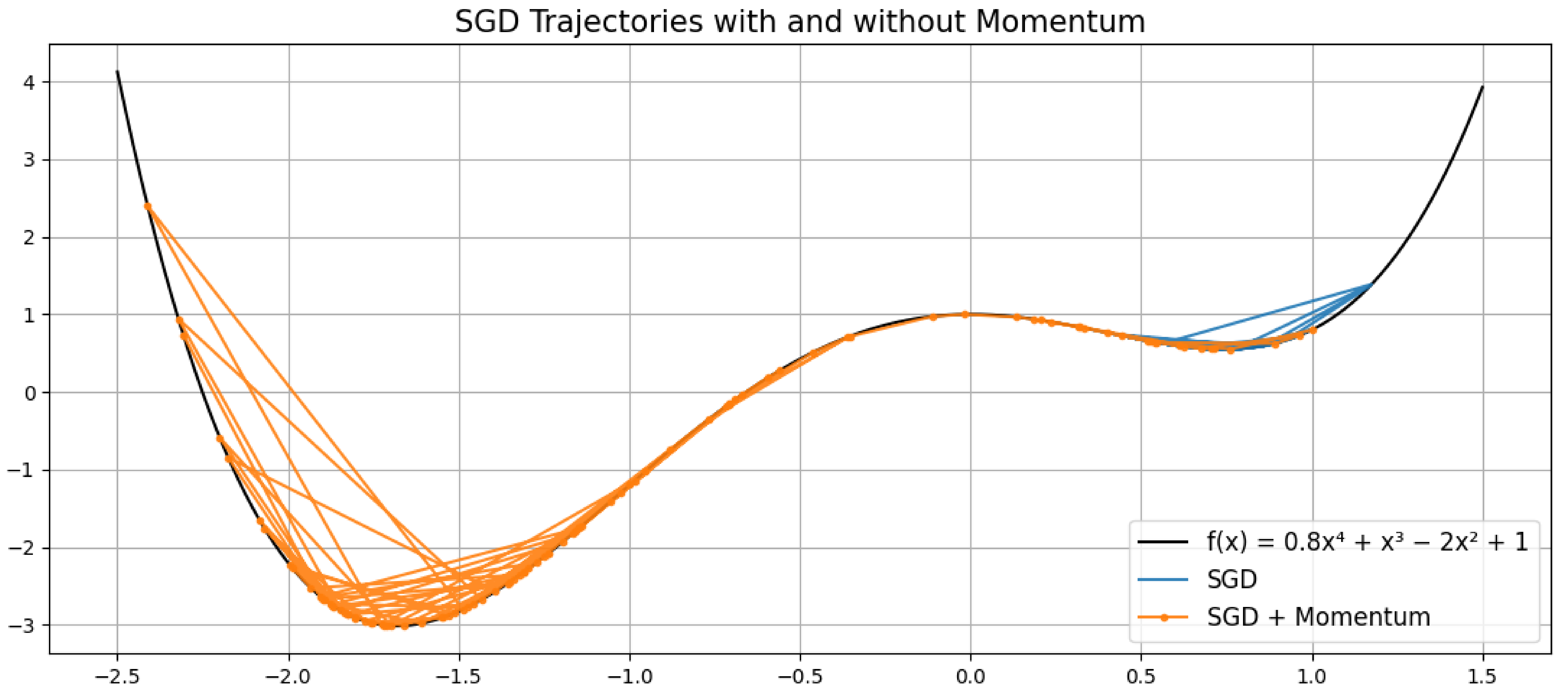
Key changes:

- Smoother updates - noise is filtered by the velocity term.
- Faster convergence, especially in shallow or flat regions.
- Reduced oscillation along steep directions.

Exercise 4:

Consider function:

$$f(x) = 0.8x^4 + x^3 - 2x^2 + 1$$



Exercise 4:

Problem statement: What happens when we use minibatch stochastic gradient descent with momentum?

- **Stochastic Gradient Descent (SGD):** updates parameters using one sample, so the gradient is very noisy.
- **Mini-batch SGD**: uses a small batch (e.g., 16 or 32 samples), reducing noise while keeping updates fast.
- **Momentum**: accumulates past gradients to smooth the trajectory and speed up convergence.

→ So when we combine them: Each step uses the average gradient over a mini-batch, and Momentum accumulates these batch-gradients over time.

This produces (with momentum):

1. Faster convergence than plain mini-batch SGD
2. Lower variance in updates
3. A smoother path through the optimization landscape

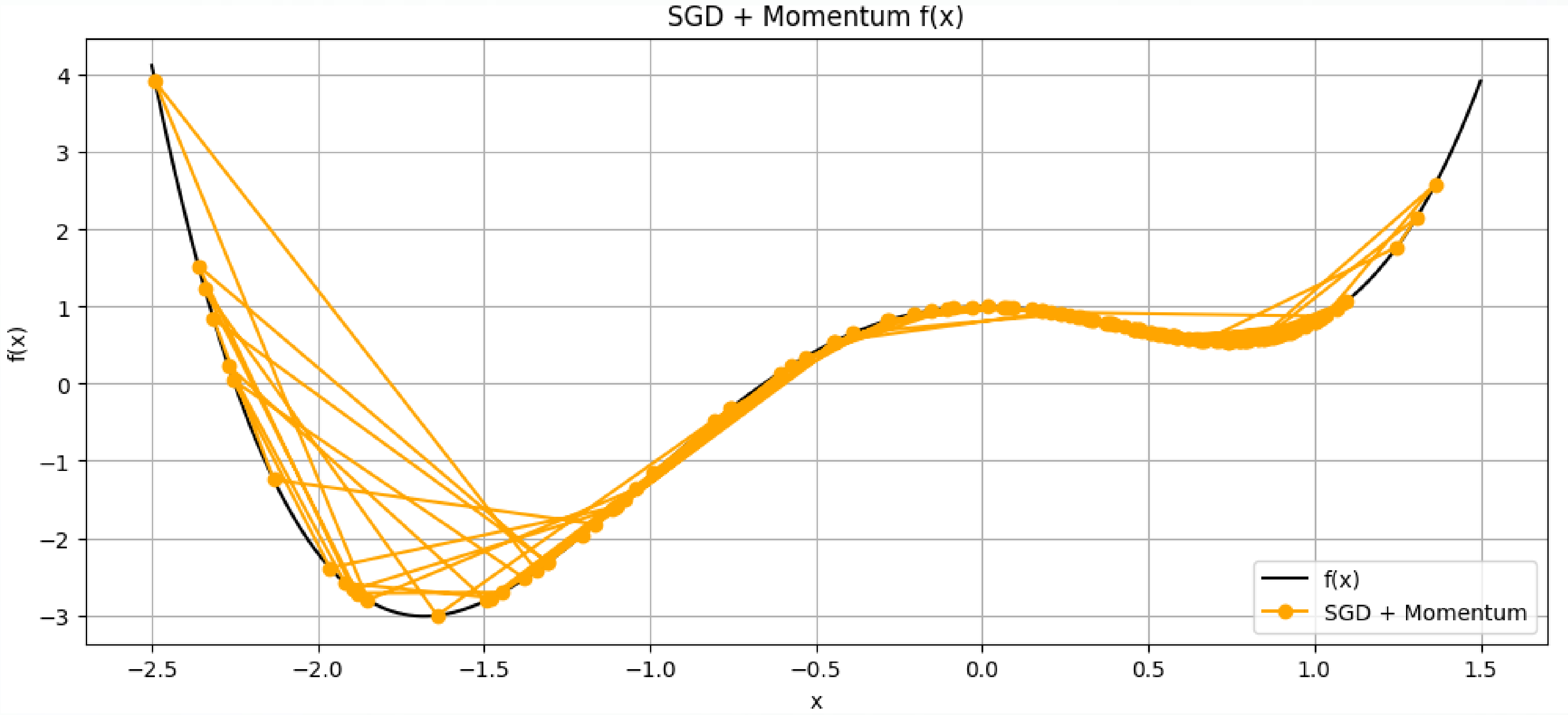
$$v_{t+1} = \beta v_t + \frac{1}{|B|}\sum_{i\in B} \nabla_{\mathbf{x}} f(\mathbf{x}_t)$$

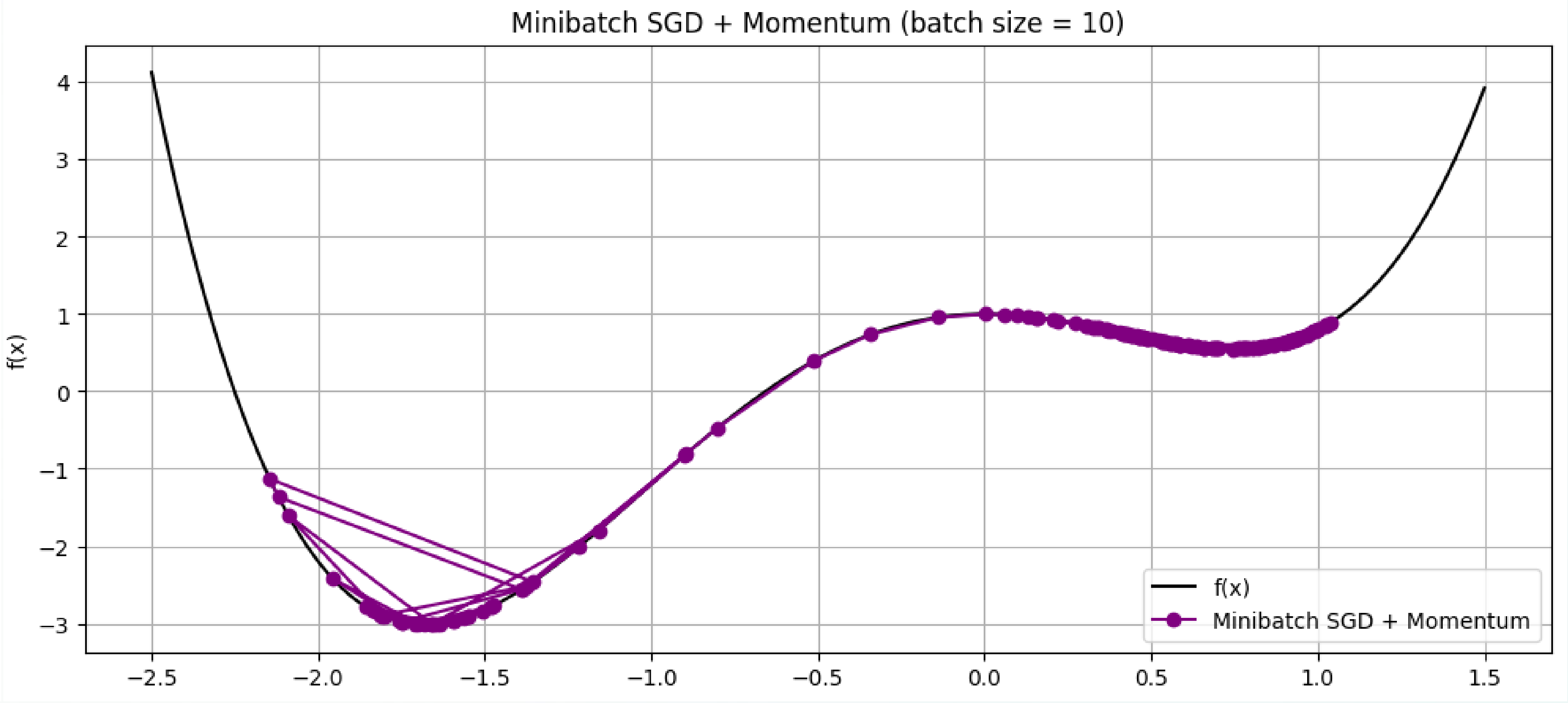
$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta v_{t+1}$$

Exercise 4:

Consider function:

$$f(x) = 0.8x^4 + x^3 - 2x^2 + 1$$





Summary

- Momentum replaces the raw gradient with a leaky average of past gradients.
- It is desirable for both noise-free gradient descent and (noisy) stochastic gradient descent.
- The effective number of past gradients that influence each update is approximately: $\frac{1}{1-\beta}$
- For convex quadratic functions, momentum can be analyzed exactly
- Momentum is easy to implement and widely supported. The only additional requirement is storing a velocity vector alongside the parameters.

Mathematical Foundations for Computer Science

THANK YOU FOR LISTENING

Date: December 13, 2025