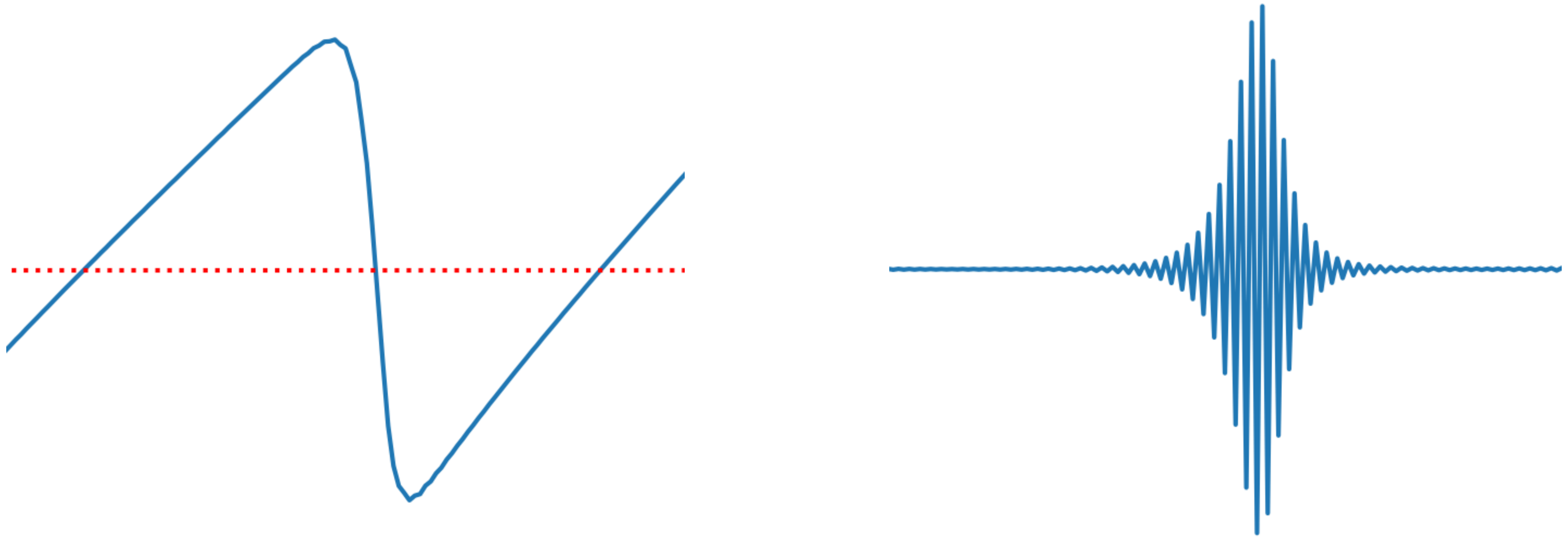
How Does Gradient Descent Work?



Jeremy Cohen · Peking University · Apr 10, 2025

This talk

- Neural networks are trained using optimization algorithms
- Yet, optimization theory is not used in deep learning. Why?
- Thesis of this talk:
    1. Existing optimization theory does not apply in deep learning …
    2. … but a different kind of theory is possible.
- Goal: convince you to help build the theory of optimization in deep learning

Gradient descent

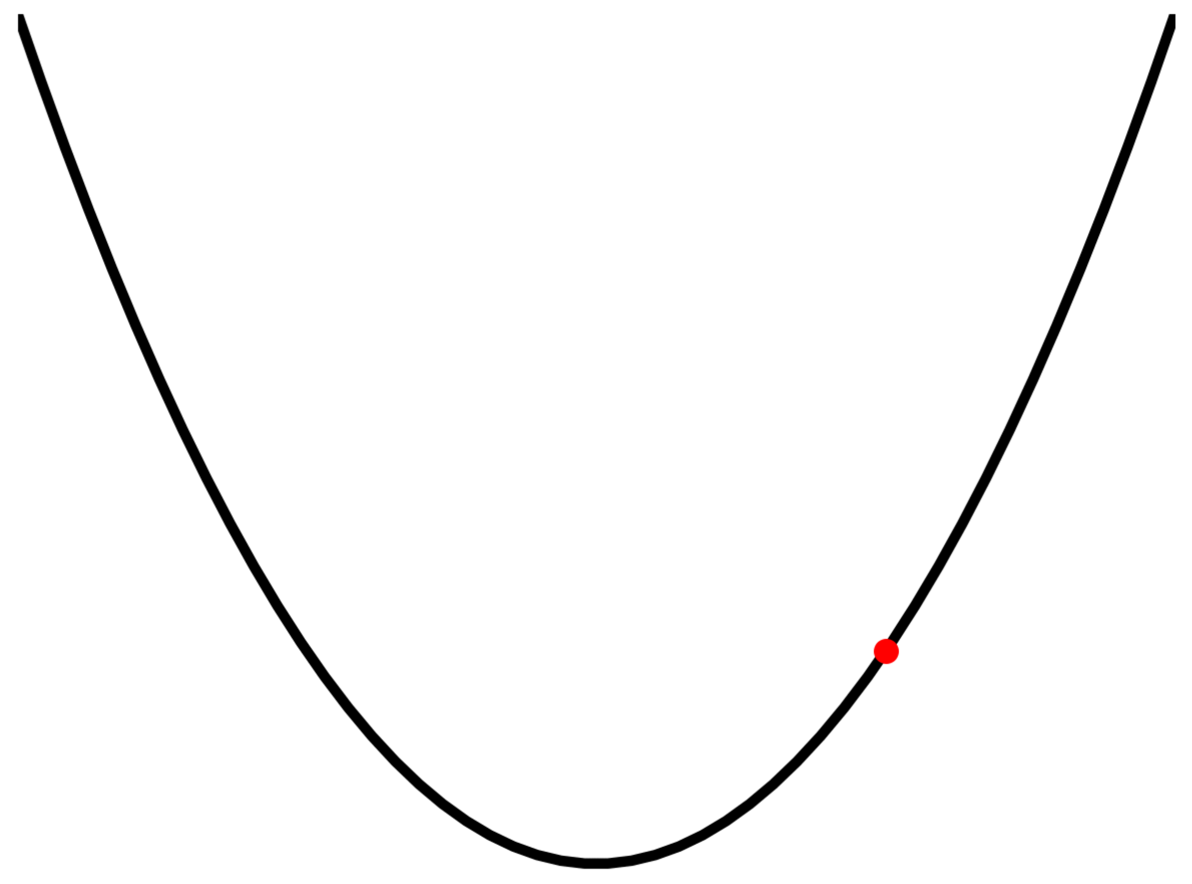
- The simplest optimizer is deterministic gradient descent (GD):

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

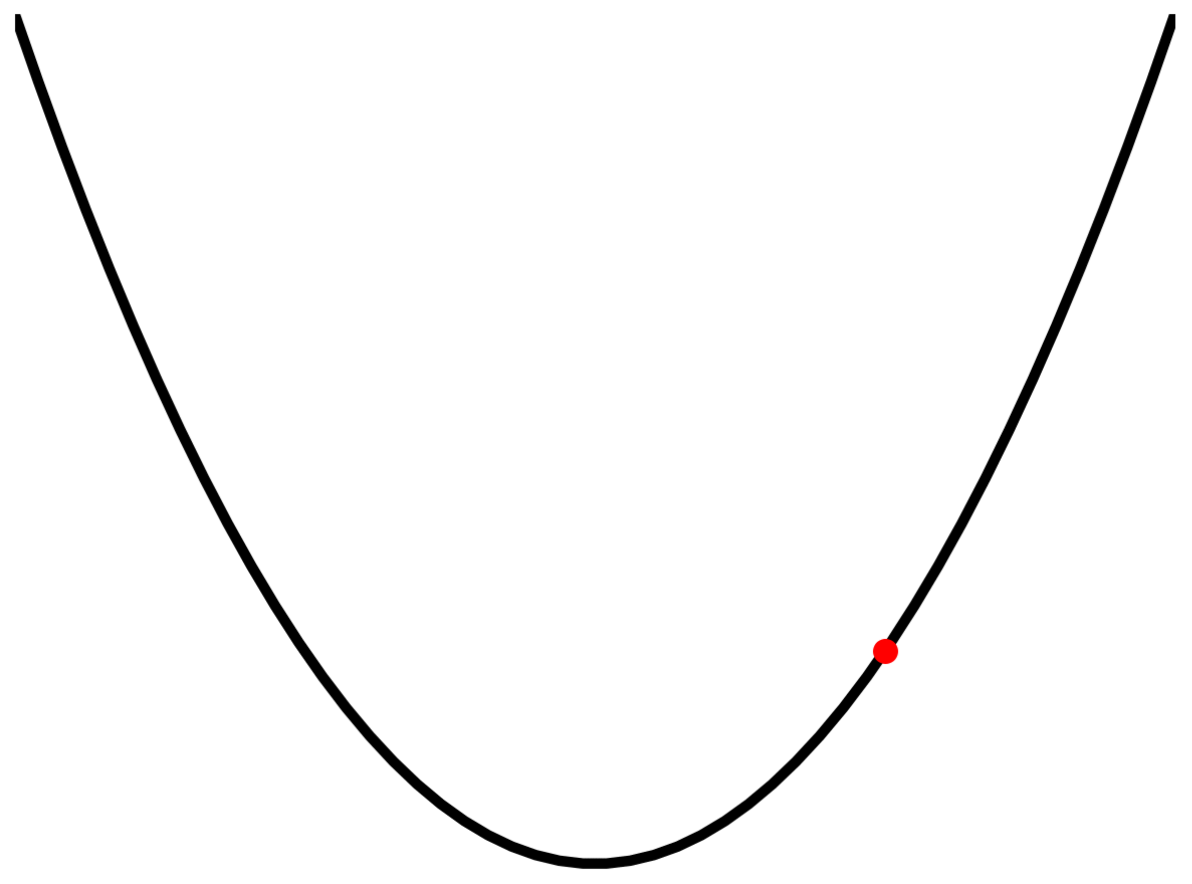
- Existing theory can’t explain the convergence of even this algorithm
- We must understand GD before we can understand more complex methods

Warm-up: quadratic objective functions

- On quadratics, GD oscillates if the *curvature* (2nd derivative) is too high
- Consider a 1d quadratic function $L(x) = \frac{1}{2}Sx^2$, with curvature $L''(x) = S$



S $< 2/\eta$



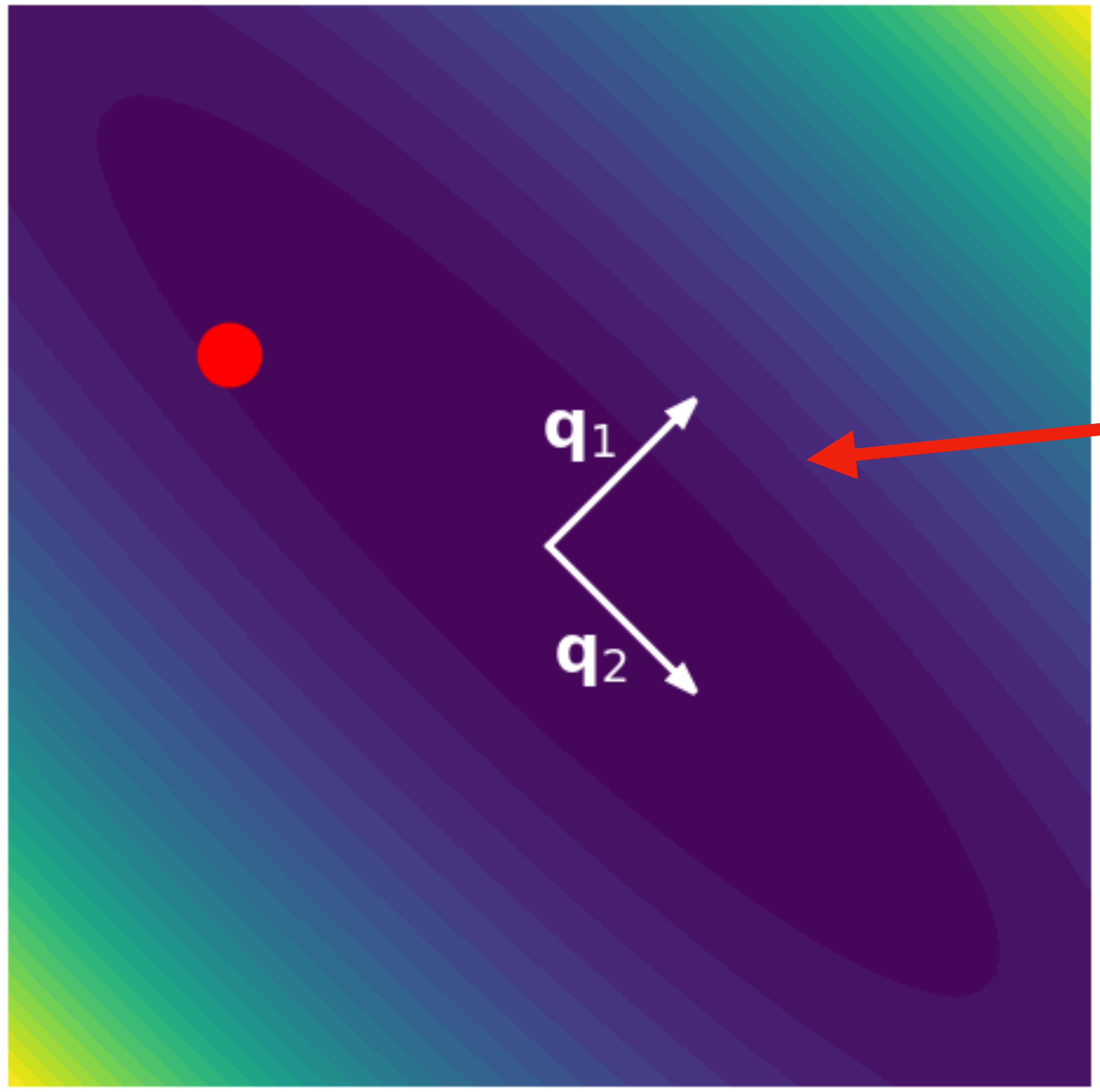
S $> 2/\eta$

Warm-up: quadratic objective functions

- For a quadratic in *multiple* dimensions, curvature is quantified by Hessian
- GD oscillates along Hessian eigenvectors with eigenvalues greater than $2/\eta$

Warm-up: quadratic objective functions

- For a quadratic in *multiple* dimensions, curvature is quantified by Hessian
- GD oscillates along Hessian eigenvectors with eigenvalues greater than $2/\eta$



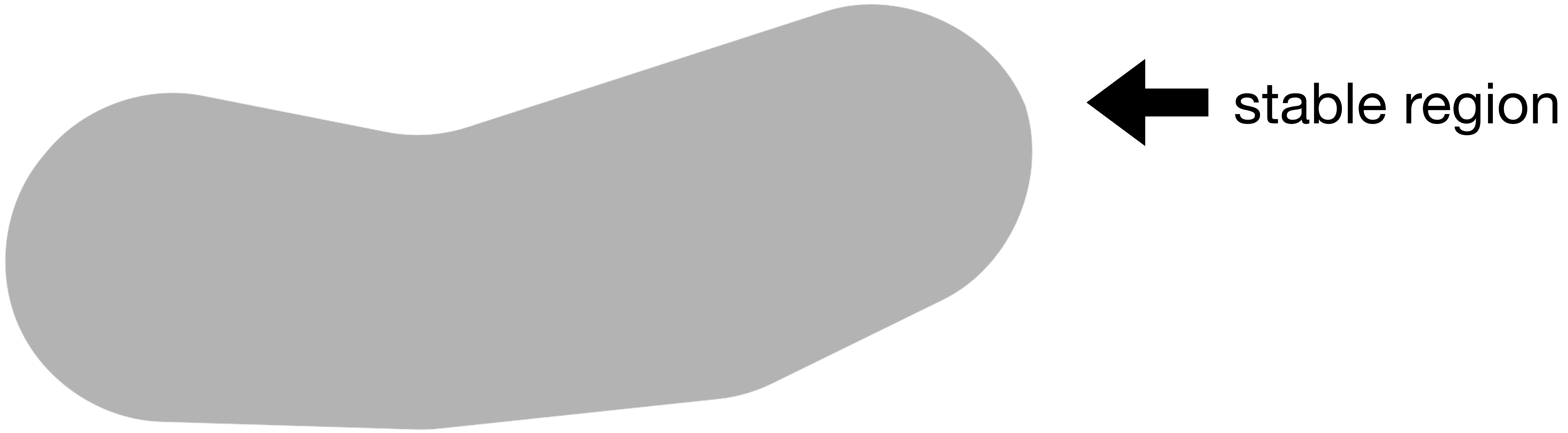
Eigenvalue is $> 2/\eta$

What about deep learning?

- For DL objectives, can take quadratic Taylor approximation around any w
- Dynamics of GD on this quadratic depend on the top eigenvalue of the Hessian $H(w)$, i.e. the *sharpness* $S(w) := \lambda_1(H(w))$
- If sharpness $S(w) > 2/\eta$, GD would diverge on the quadratic Taylor approximation
- This suggests that GD doesn't function properly if sharpness $S(w) > 2/\eta$

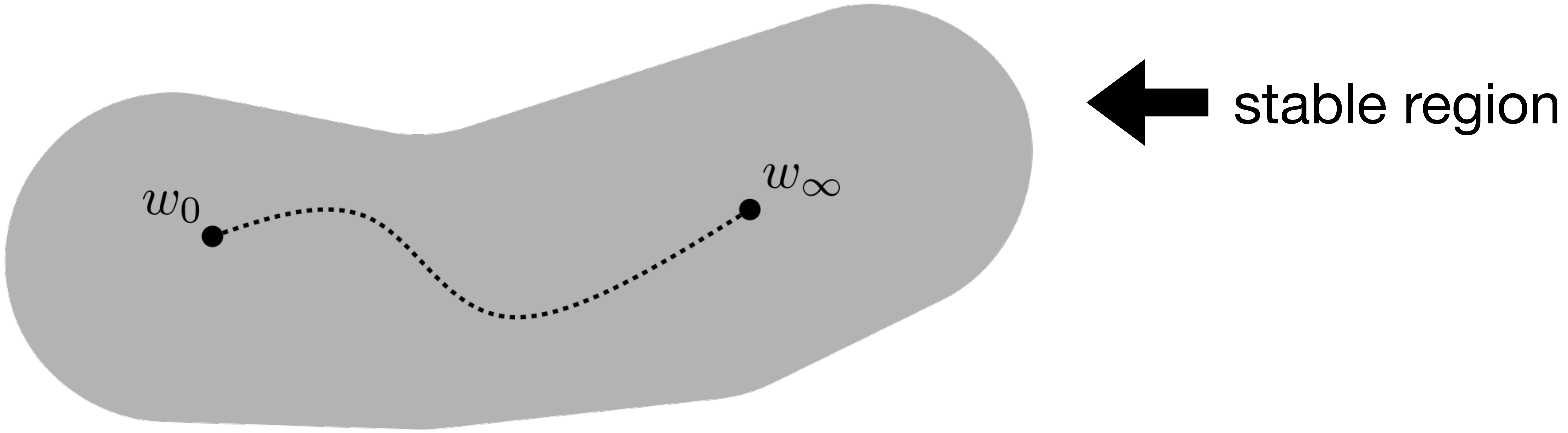
Gradient descent in deep learning

- Why does gradient descent converge in deep learning?
- Natural idea: sharpness $S(w)$ remains below $2/\eta$ throughout training
  - i.e. GD stays inside the "stable region" $\{w : S(w) \leq 2/\eta\}$



Gradient descent in deep learning

- Why does gradient descent converge in deep learning?
- Natural idea: sharpness $S(w)$ remains below $2/\eta$ throughout training
    - i.e. GD stays inside the "stable region" $\{w : S(w) \leq 2/\eta\}$



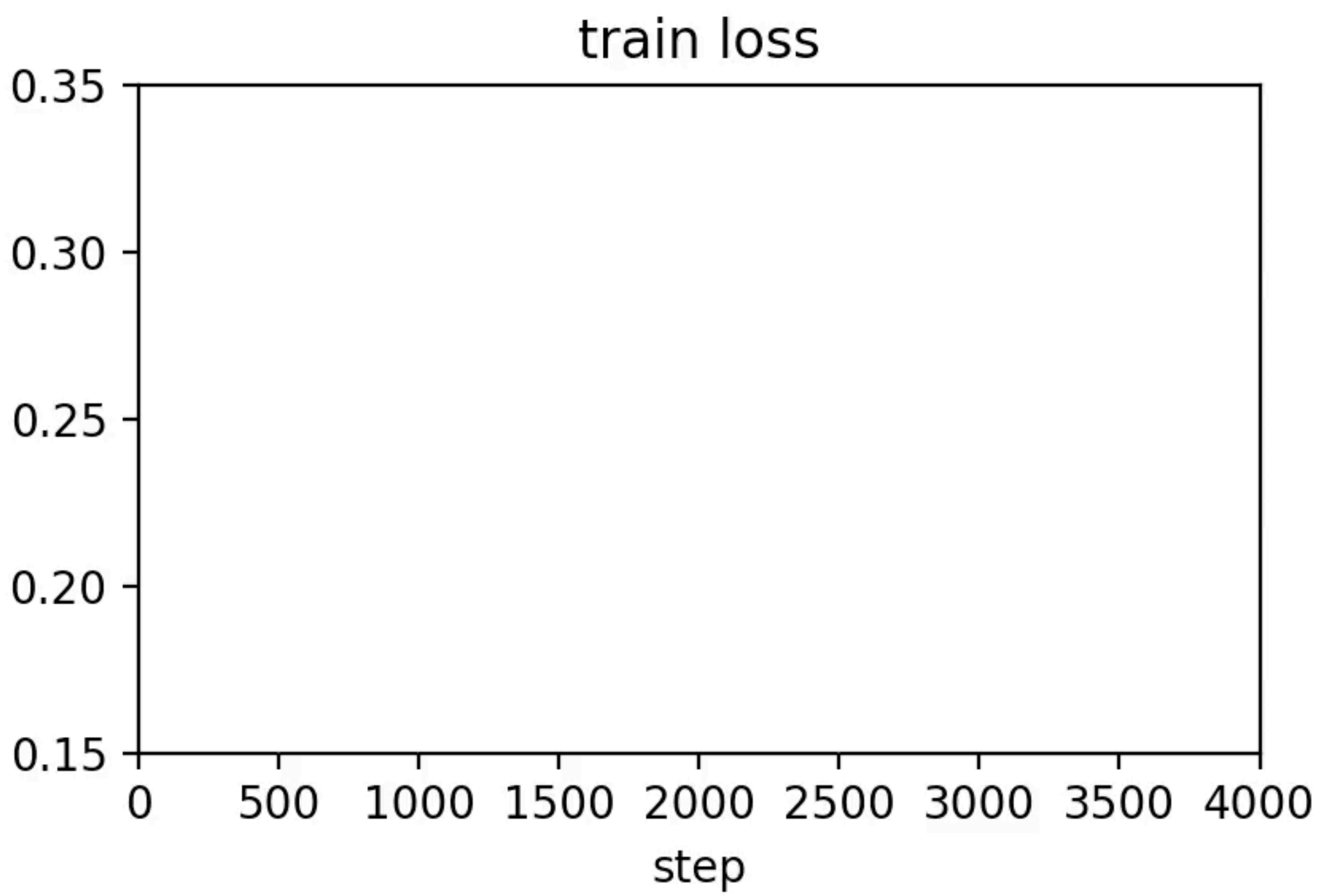
- This is the picture suggested by traditional optimization theory ("L-smoothness")

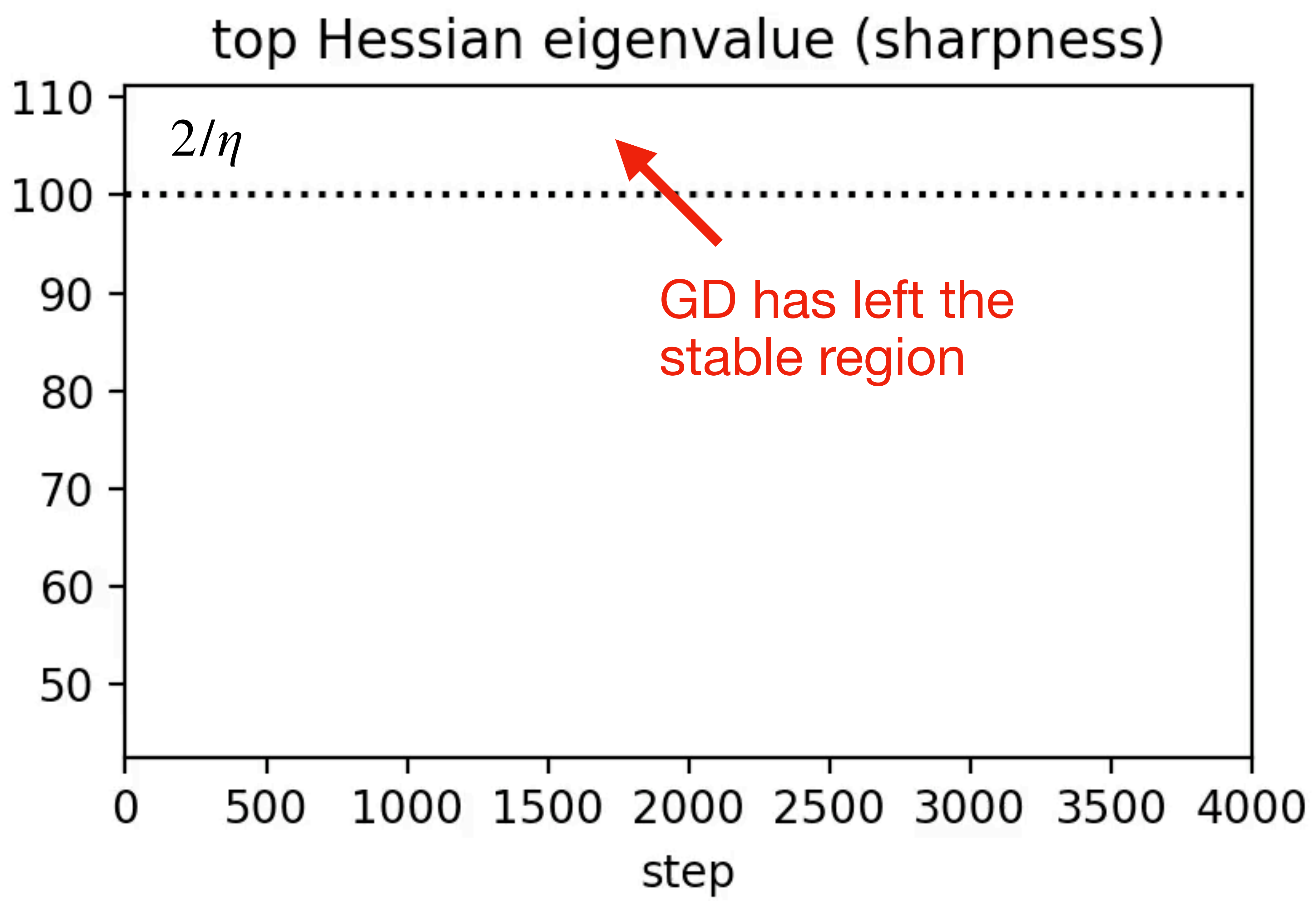
Deep learning reality

- Train neural network using GD with $\eta = 0.02$ (ViT on CIFAR-10):

Deep learning reality

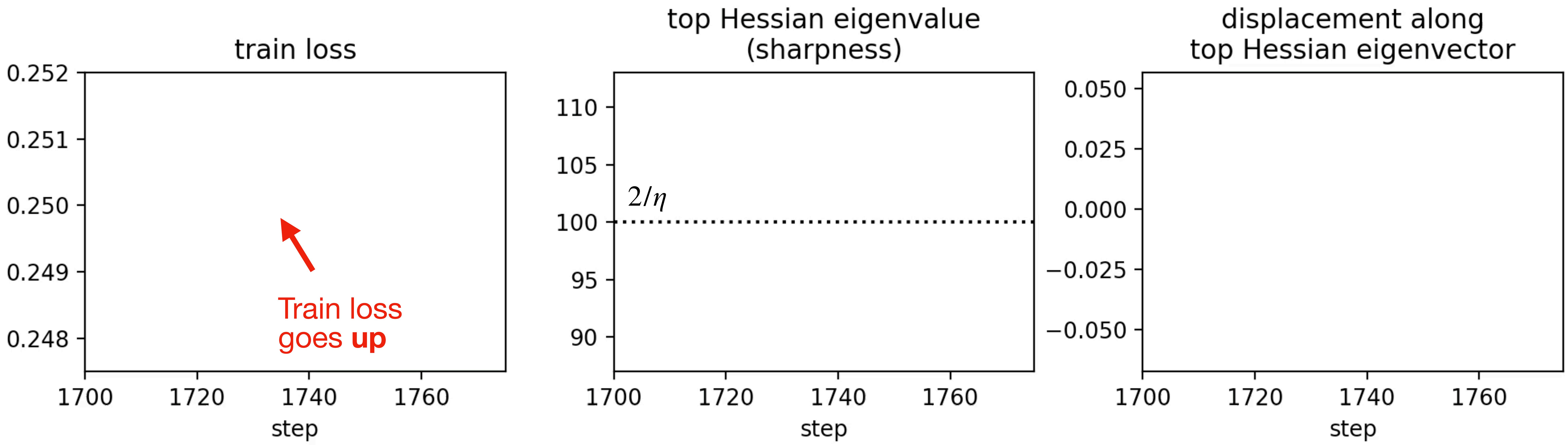
- Train neural network using GD with $\eta = 0.02$ (ViT on CIFAR-10):





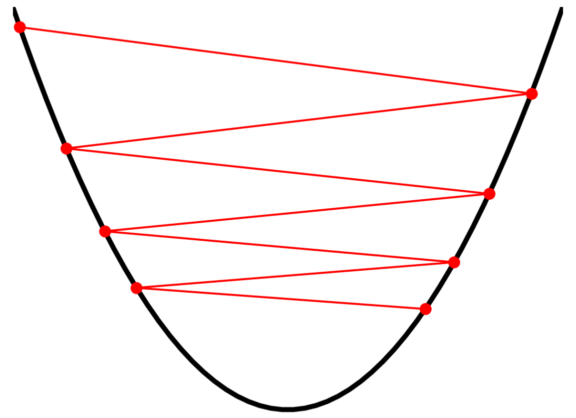
Quadratic Taylor approximation predicts growing oscillations along top Hessian eigenvector

What happens next?

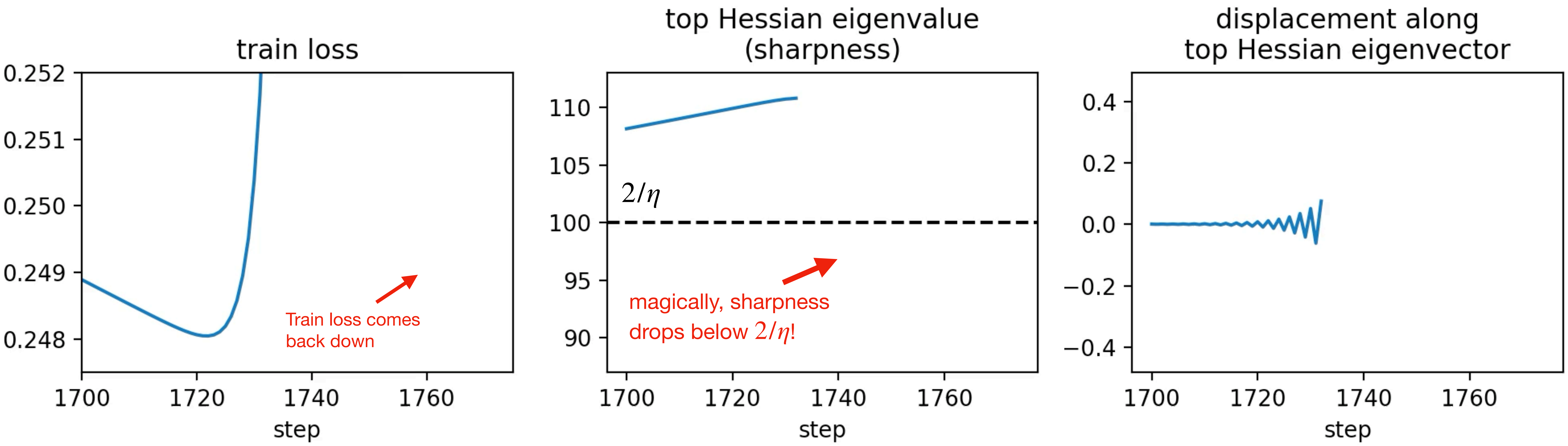


What happens next?

This quantity is predicted to oscillate

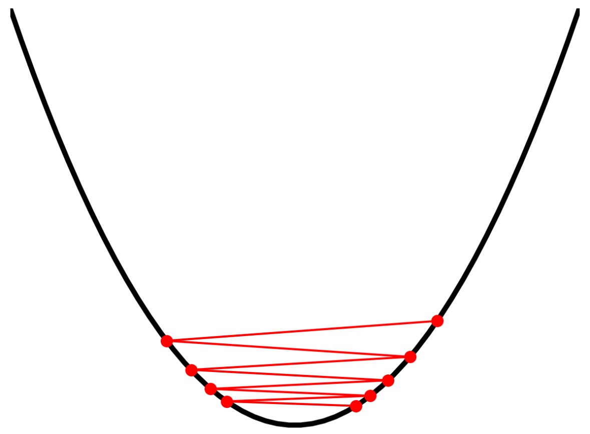


What happens next?



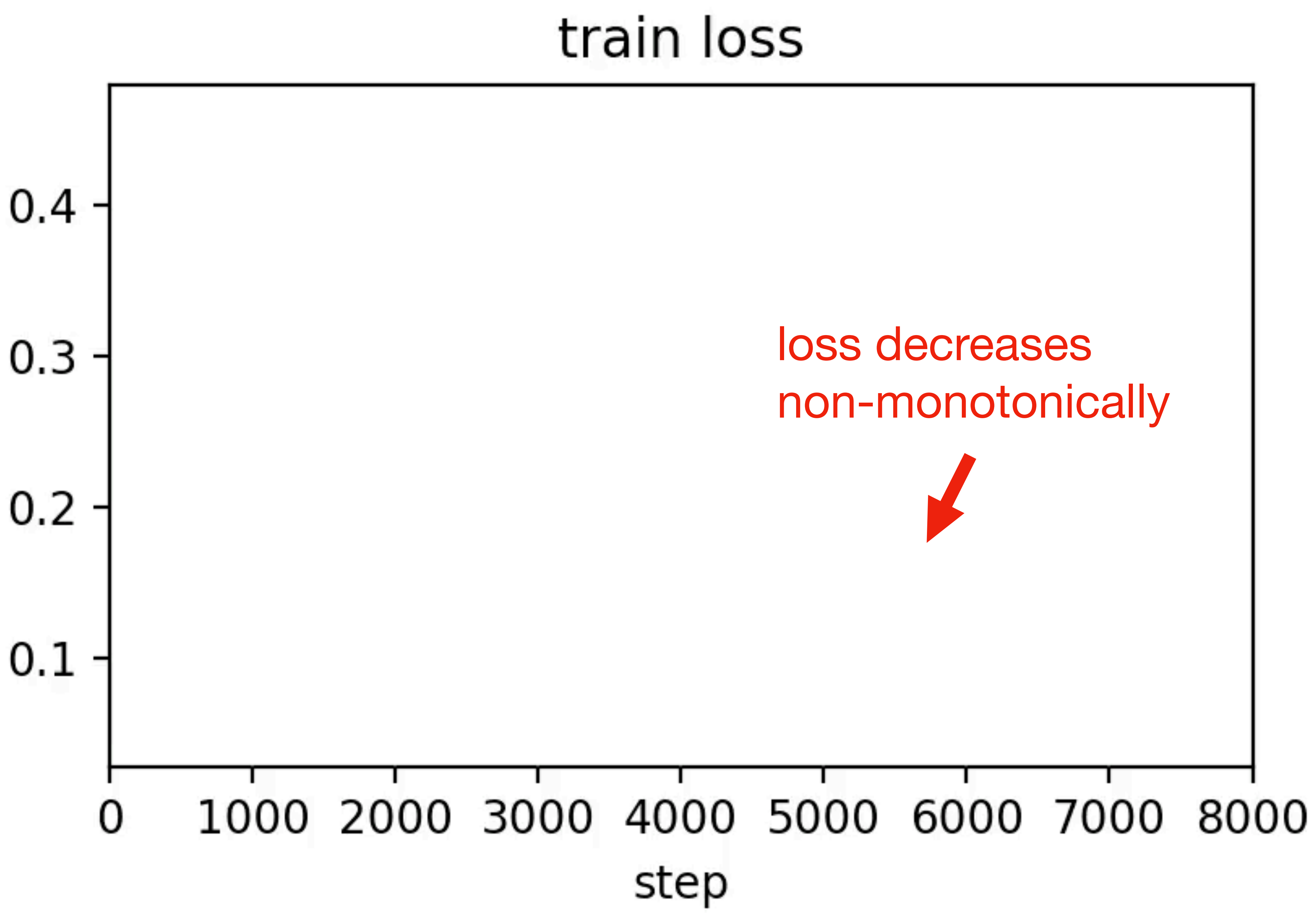
Mystery: why did the sharpness drop?

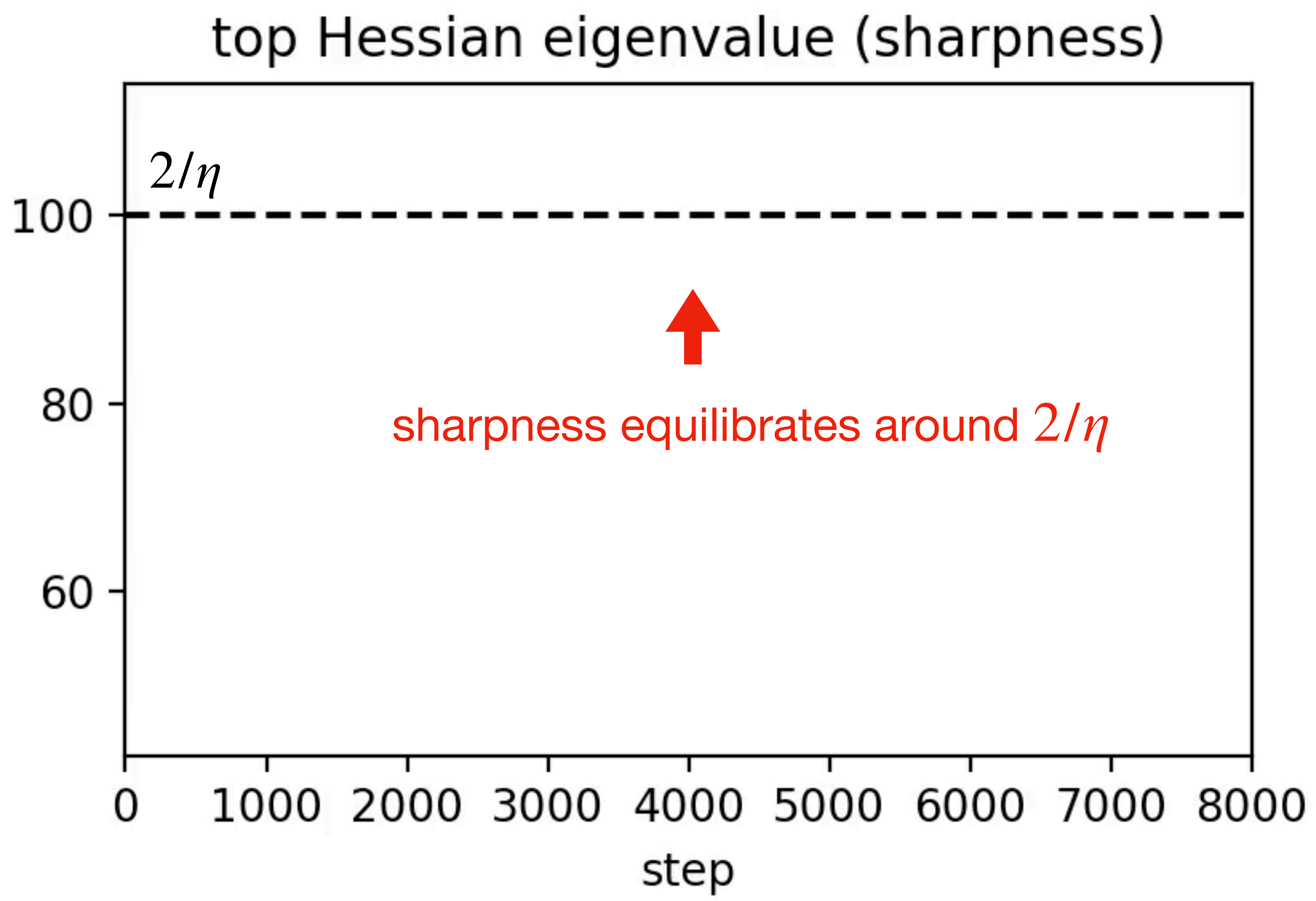
Hint: look here



oscillations shrink

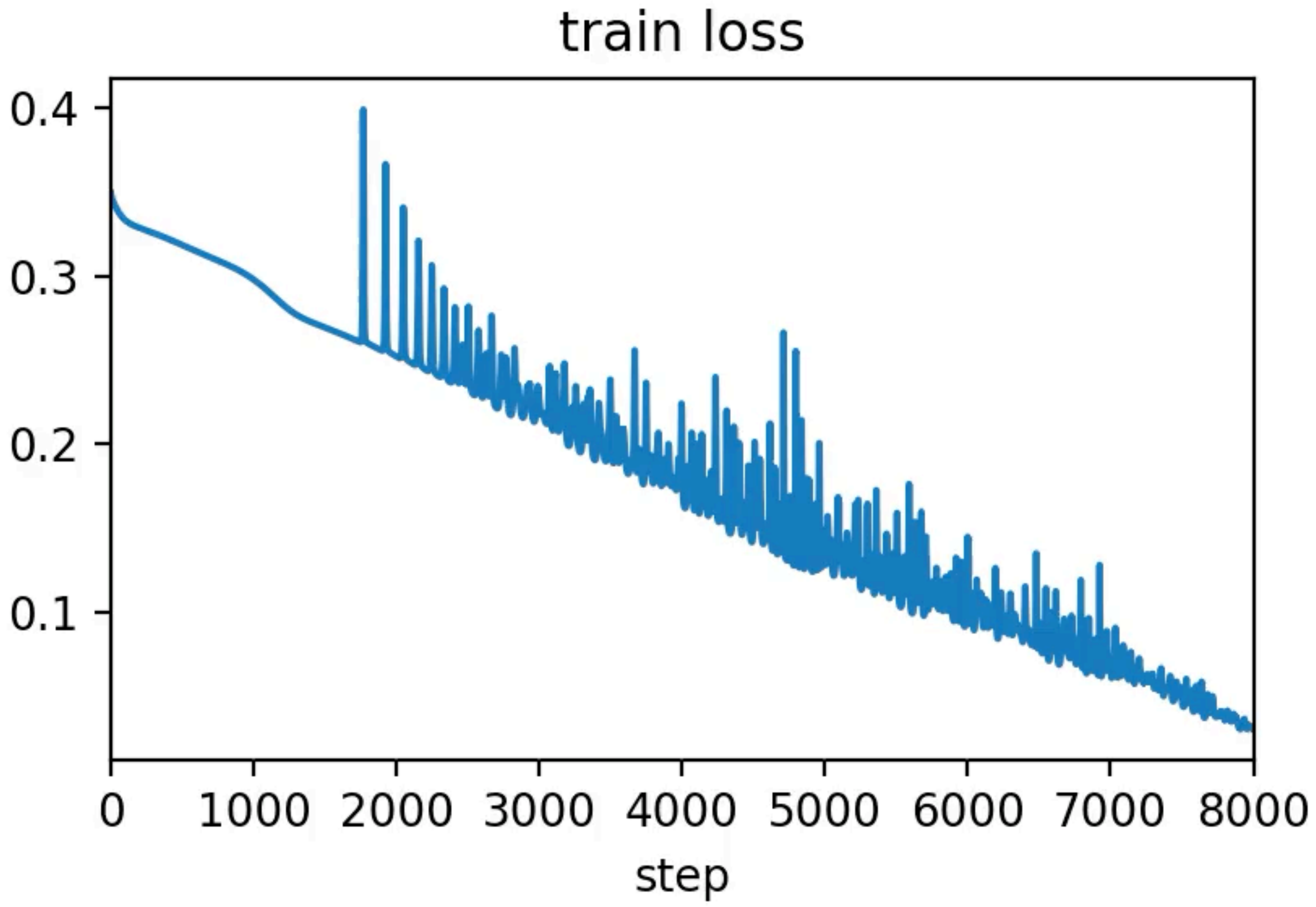
Full gradient descent trajectory

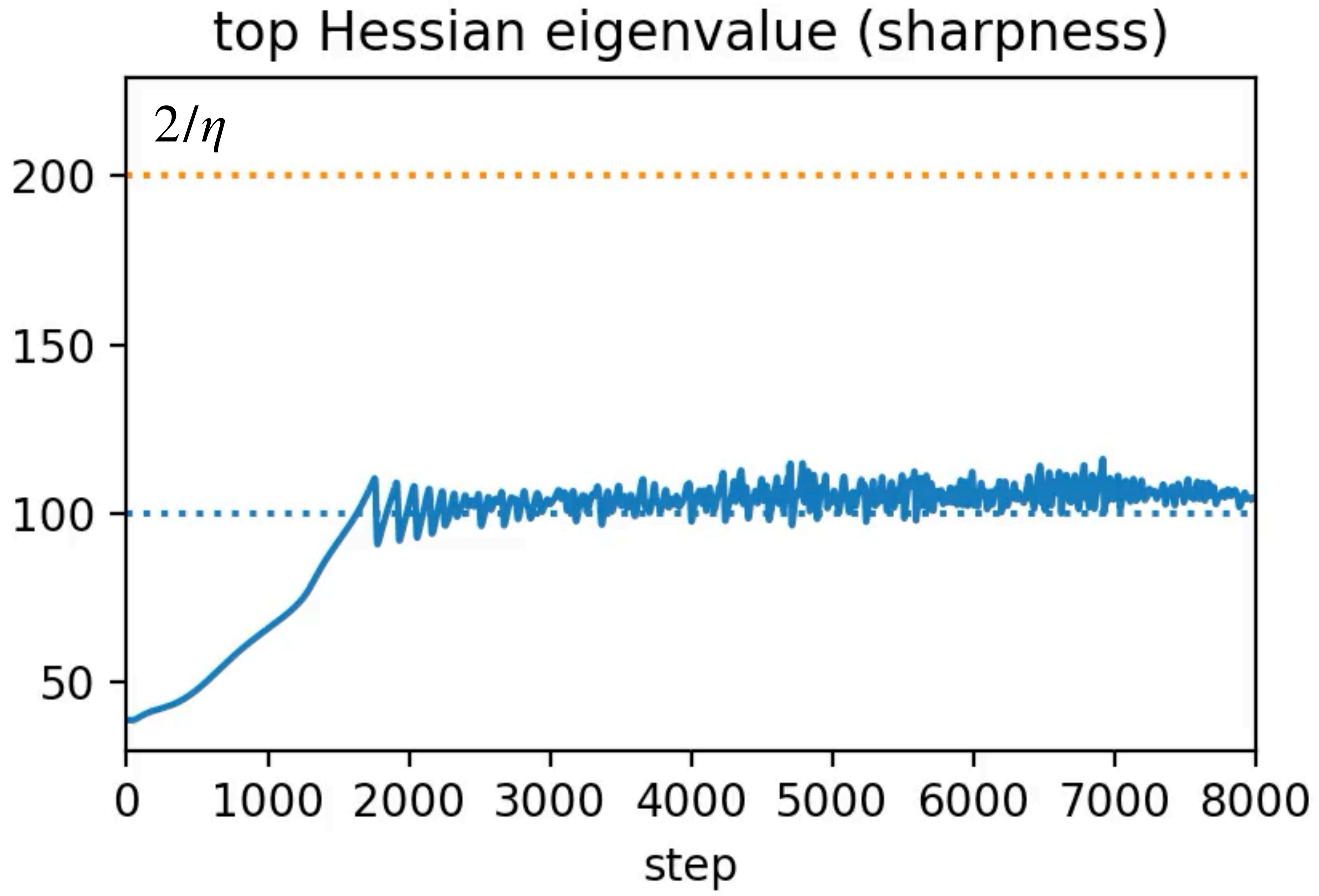




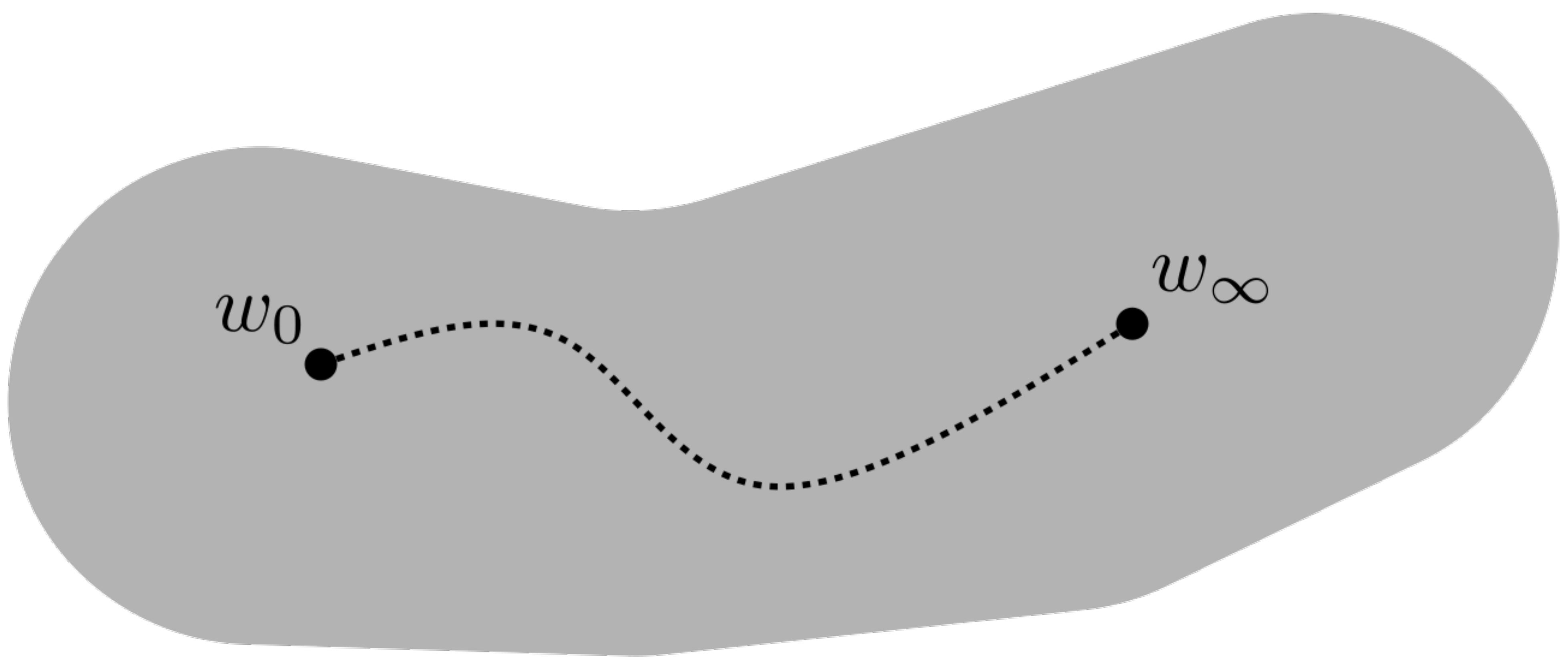
What if we train at a different learning rate?

- Train same network with smaller learning rate $\eta = 0.01$ (orange):

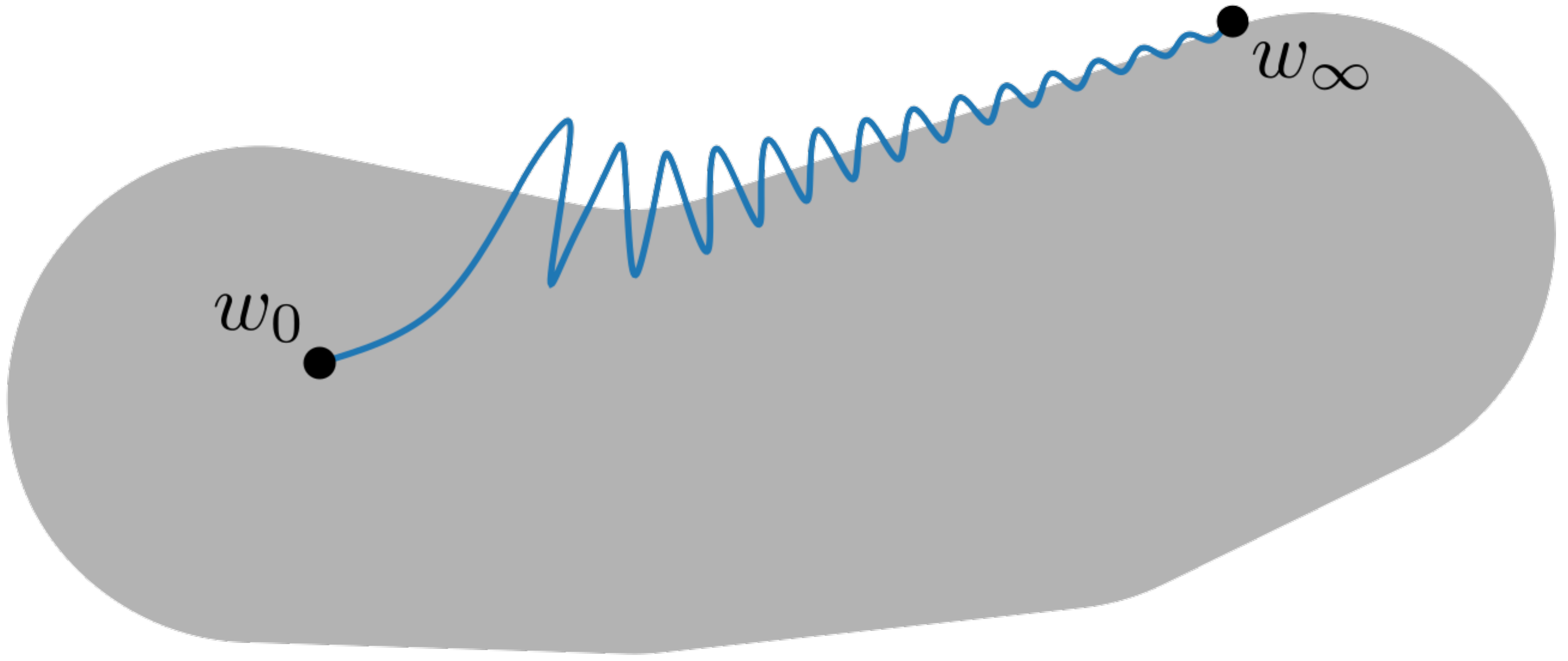




Expectation vs. reality



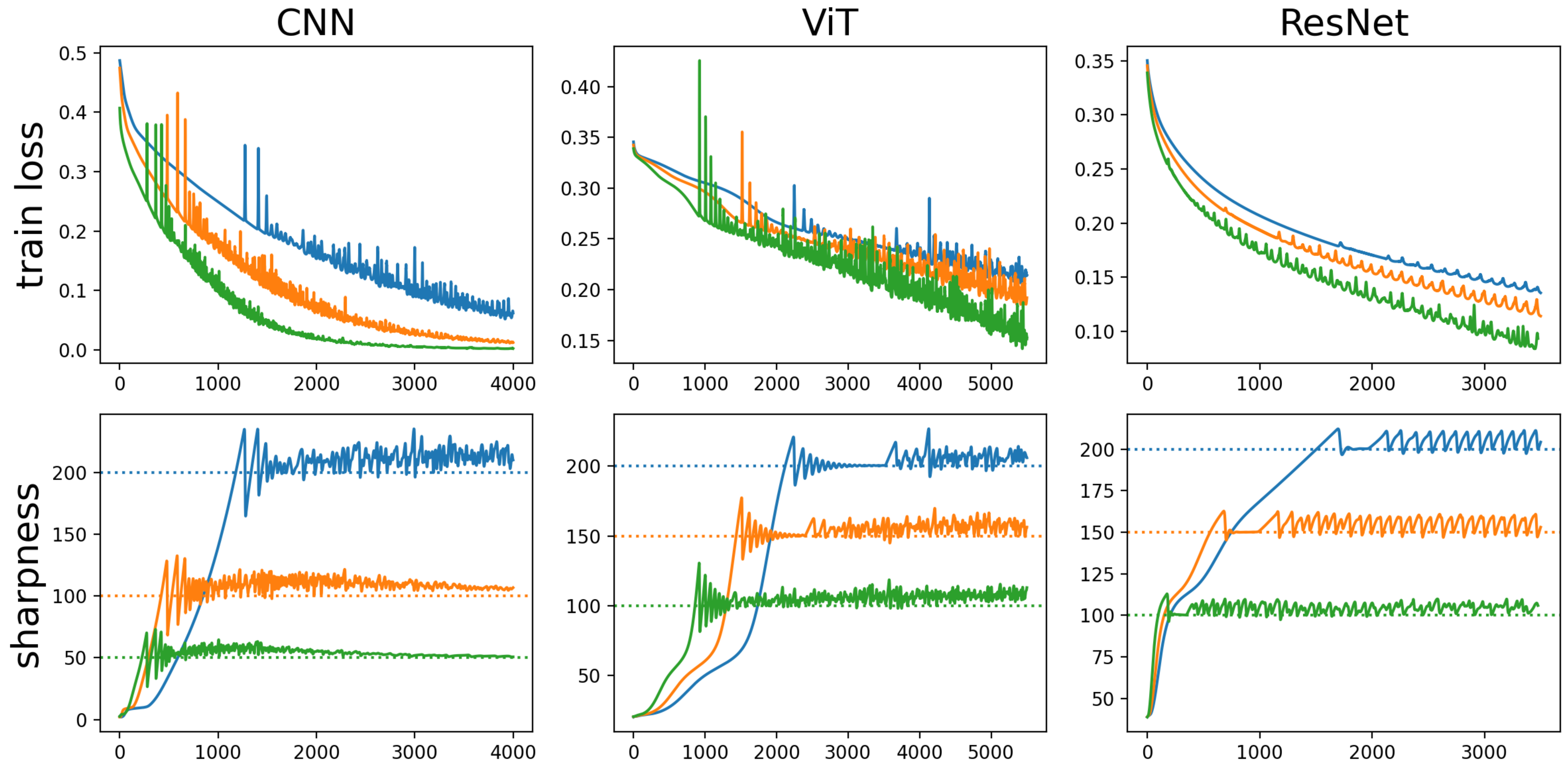
Expectation



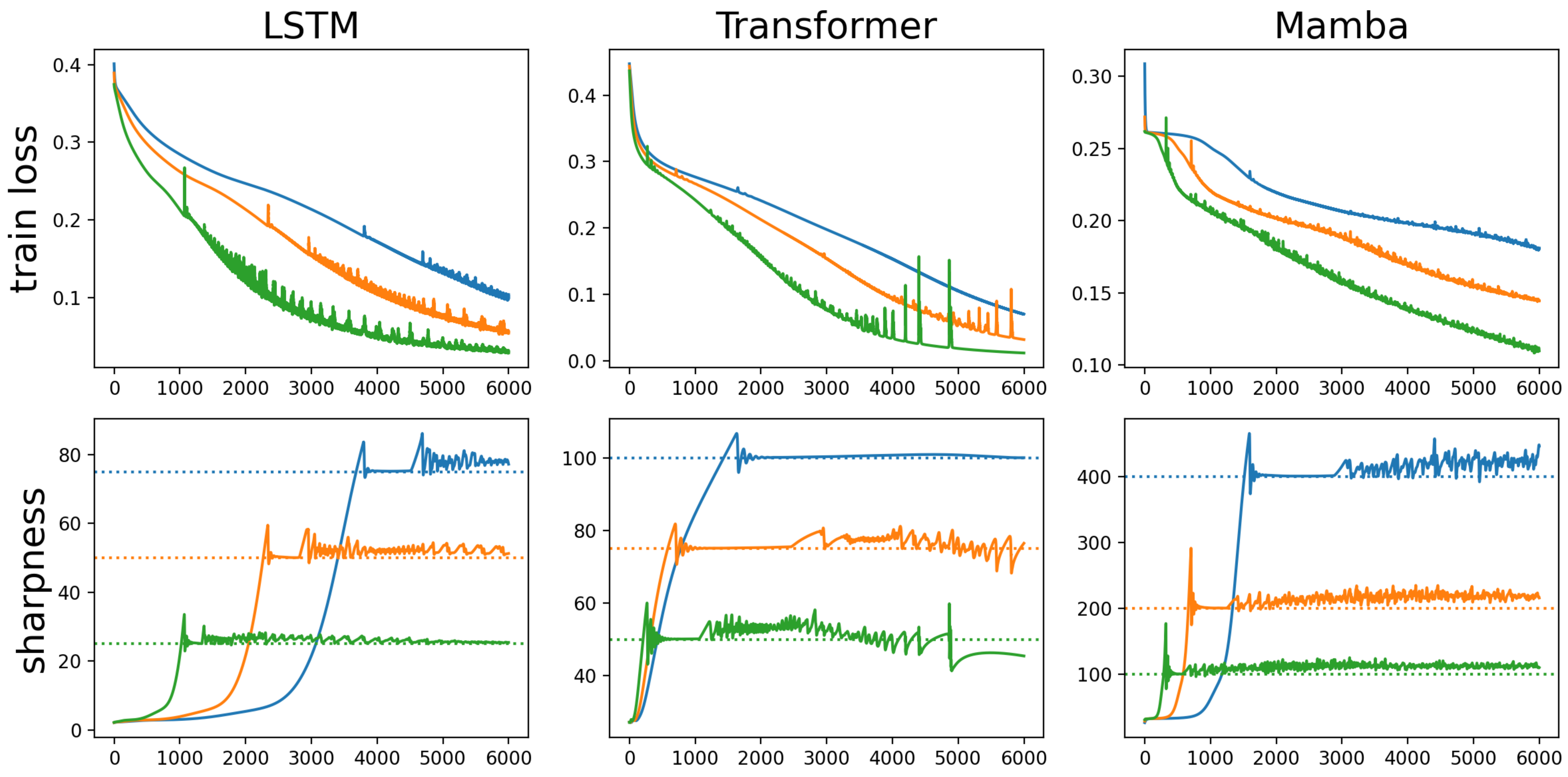
Reality

Gradient descent trains at the **edge of stability**

This behavior is generic across DL settings



This behavior is generic across DL settings



- This is not a weird edge case, it's the **typical** behavior of GD in DL

Same phenomenon

Wu, Ma, E. *How SGD Selects the Global Minima in Over-parameterized Learning: A Dynamical Stability Perspective*. NeurIPS ’18.

η	0.01	0.05	0.1	0.5	1	5
FashionMNIST	53.5 ± 4.3	39.3 ± 0.5	19.6 ± 0.15	3.9 ± 0.0	1.9 ± 0.0	0.4 ± 0.0
CIFAR10	198.9 ± 0.6	39.8 ± 0.2	19.8 ± 0.1	3.6 ± 0.4	-	-
prediction $2/\eta$	200	40	20	4	2	0.4

Observation: sharpness at end of training is $\approx 2/\eta$

What's going on?

Cohen, Kaur, Li, Kolter, Talwalkar. *Gradient descent on neural networks typically occurs at the edge of stability*. ICLR '21.

Why does gradient descent work in deep learning?

The answer

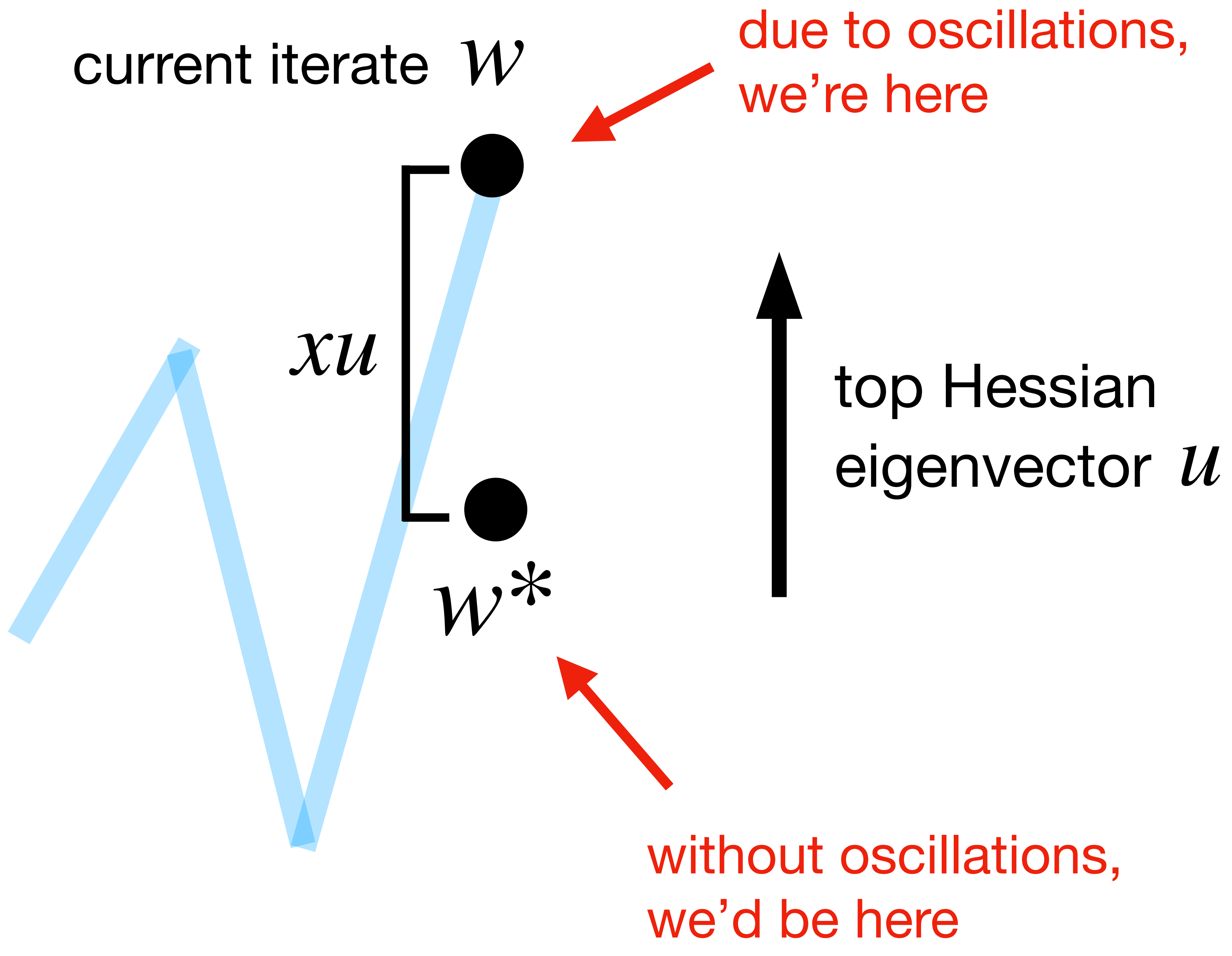


Damian*, Nichani*, Lee. *Self-stabilization: the implicit bias of gradient descent at the edge of stability.* ICLR '23.

- To understand dynamics of GD, need to Taylor expand to *third-order*.
- This expansion reveals the key ingredient missing from traditional theory:

Oscillations along the top Hessian eigenvector automatically reduce the top Hessian eigenvalue.

Informal sketch



cartoon of weight-space dynamics

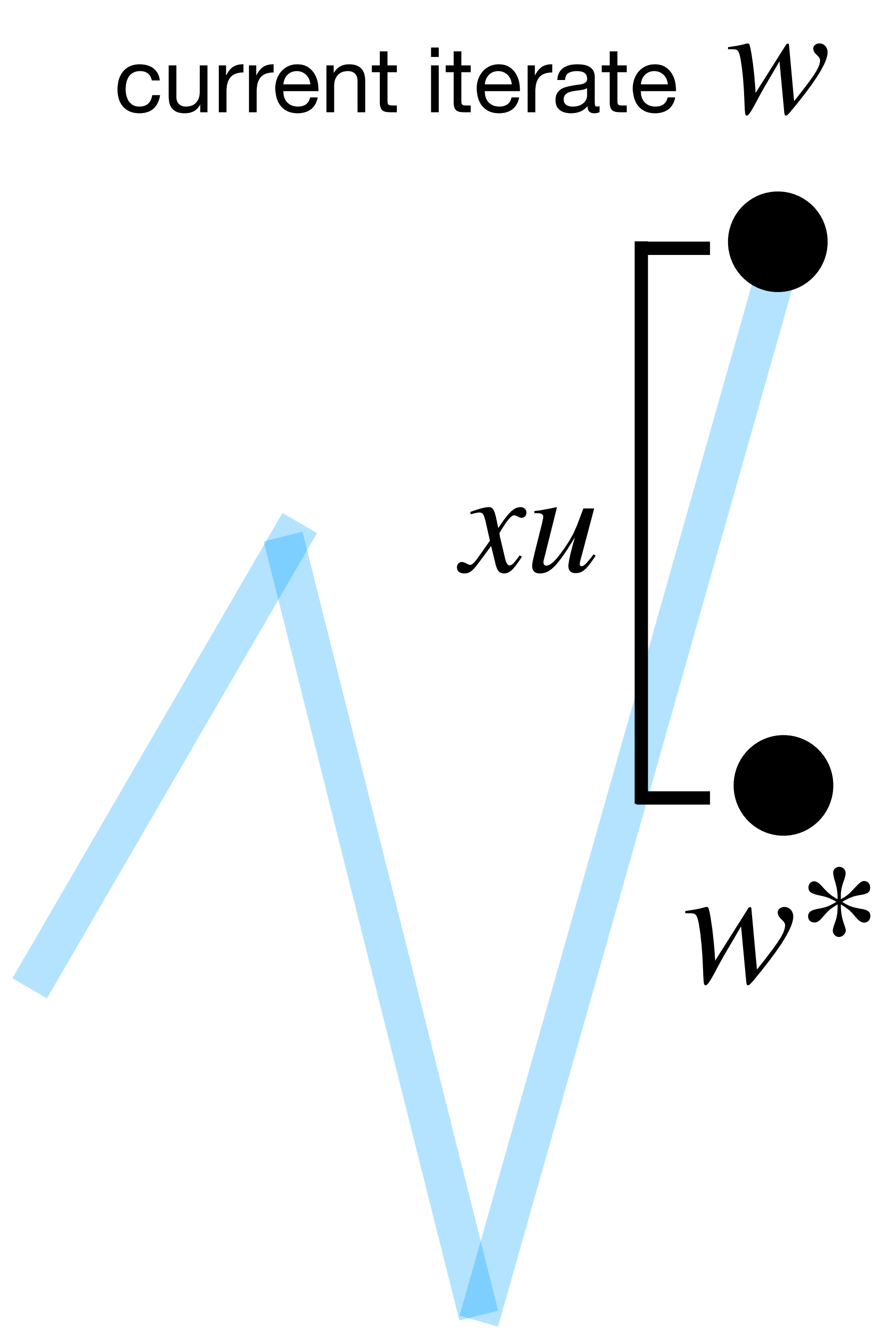
Suppose that GD is oscillating along the top Hessian eigenvector u

How does the gradient ∇L at

$$w = w^* + xu$$

relate to the gradient at w^*?

Informal sketch

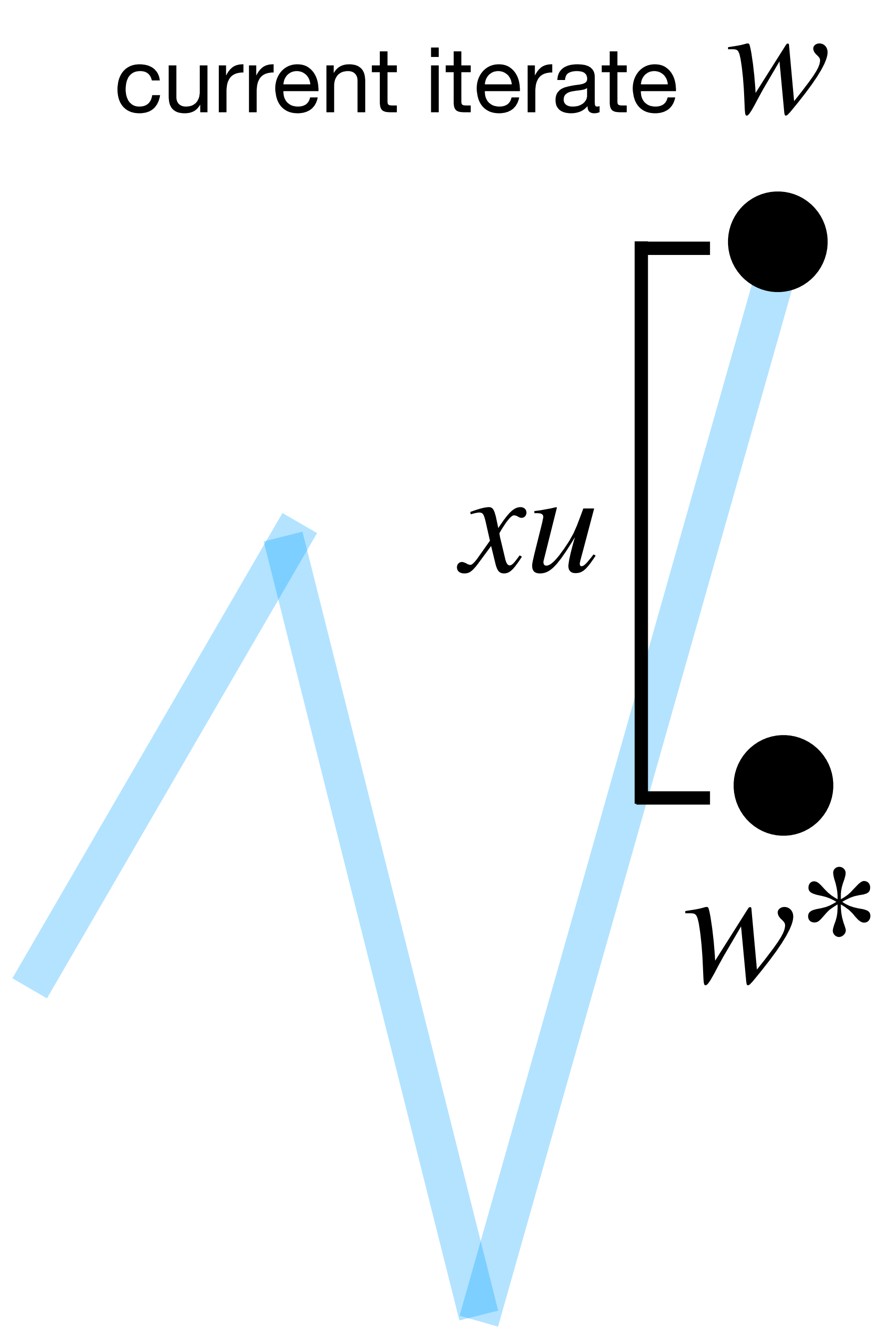


By Taylor expansion around w^*:

$$\nabla L(w^* + xu) =$$

gradient at w

Informal sketch



By Taylor expansion around w^*:

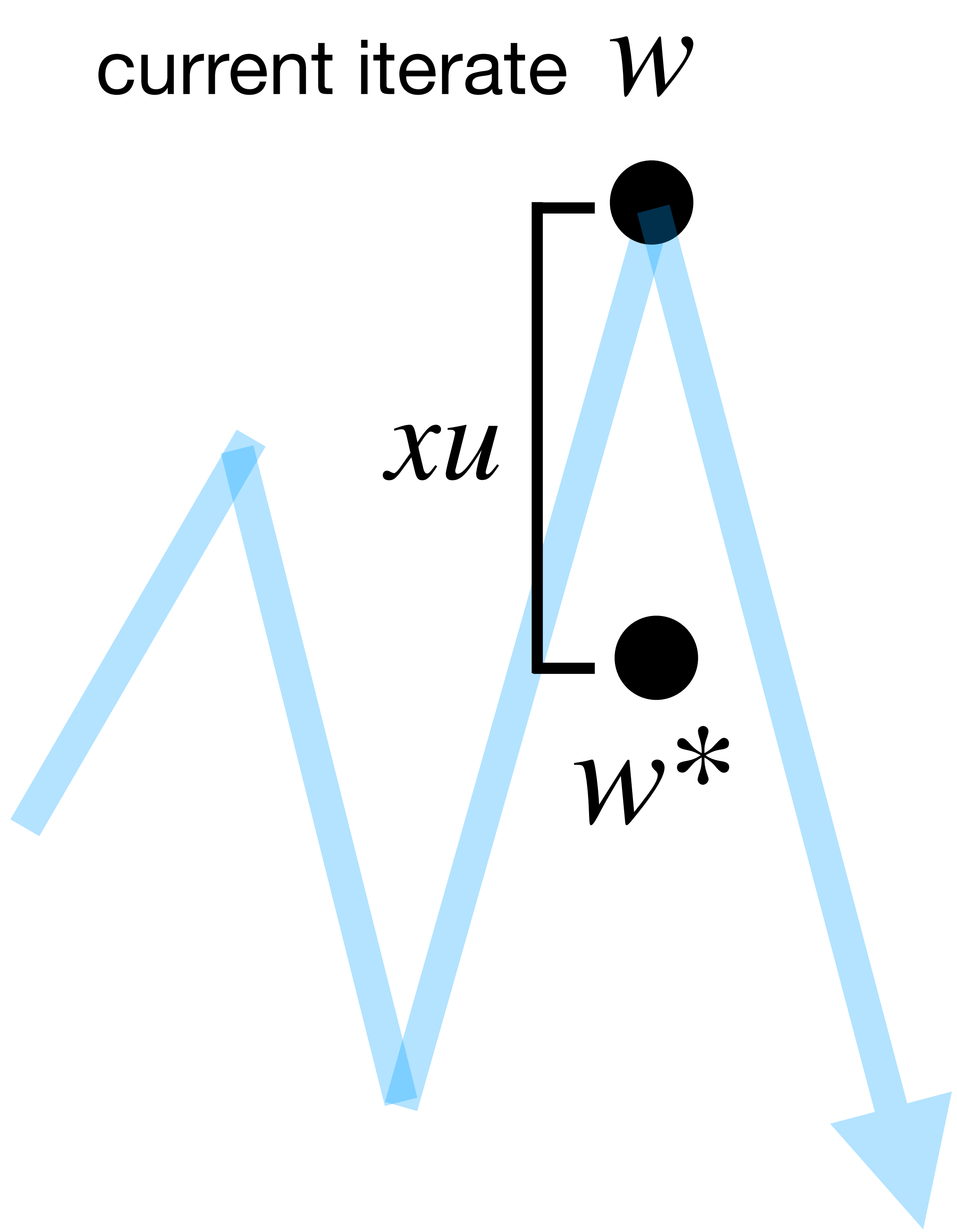
$$\nabla L(w^* + xu) = \nabla L(w^*) + O(x)$$

gradient at w — gradient at w^*

Informal sketch

Since u is a Hessian eigenvector

$$H(w^*)\,u = S(w^*)u$$



By Taylor expansion around w^*:

$$\underbrace{\nabla L(w^* + xu)}_{\text{gradient at } w} = \underbrace{\nabla L(w^*)}_{\text{gradient at } w^*} + \underbrace{H(w^*)[xu]}_{\text{oscillation}} + O(x^2)$$

$H(w^*)[xu] = S(w^*)\,x\,u$

- This term sends a negative gradient step computed at $w^* + xu$ towards the $-u$ direction.
- This term is causing us to oscillate
- The “magic” comes from the *next* term in the Taylor expansion…

Informal sketch

- The next term in the Taylor expansion is:

curvature in u direction $= S(w^*)$

$$\nabla L(w^* + xu) = \nabla L(w^*) + H(w^*)[xu] + \frac{1}{2} x^2 \nabla_{w^*} [u^T H(w^*) u] + O(x^3)$$

gradient at w

gradient at w^*

oscillation

gradient of curvature in u direction $= \nabla S(w^*)$

Informal sketch

- The next term in the Taylor expansion is:

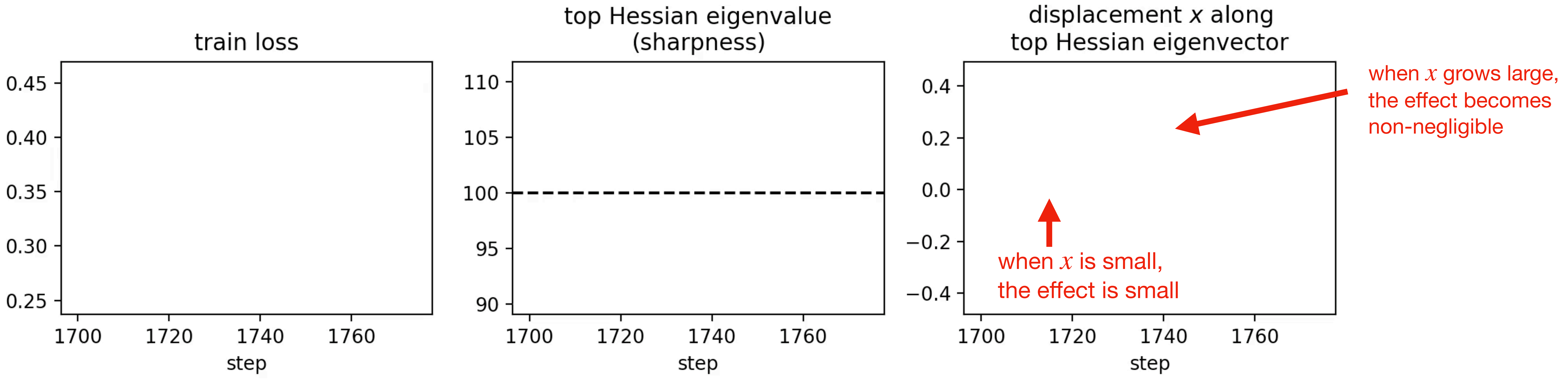
$$\underbrace{\nabla L(w^* + xu)}_{\text{gradient at } w} = \underbrace{\nabla L(w^*)}_{\text{gradient at } w^*} + \underbrace{H(w^*)[xu]}_{\text{oscillation}} + \underbrace{\frac{1}{2} x^2 \nabla S(w^*)}_{\text{gradient of sharpness}} + O(x^3)$$

- Thus, a negative gradient step computed at $w^* + xu$ automatically takes a negative gradient step *on the sharpness* with step size $\frac{1}{2}\eta x^2$.
- i.e. **oscillations automatically trigger reduction of sharpness**
  - the size of this effect is proportional to the squared magnitude of oscillation
- This is the crucial ingredient missing from the traditional theory.

Let’s revisit the behavior of GD

- When GD exits the stable region:
  - it oscillates along the top Hessian eigenvector (as expected)
  - these oscillations implicitly perform gradient descent *on the sharpness* (top Hessian eigenvalue)
  - this reduces sharpness, thereby steering GD back into the stable region

Let's revisit the behavior of GD



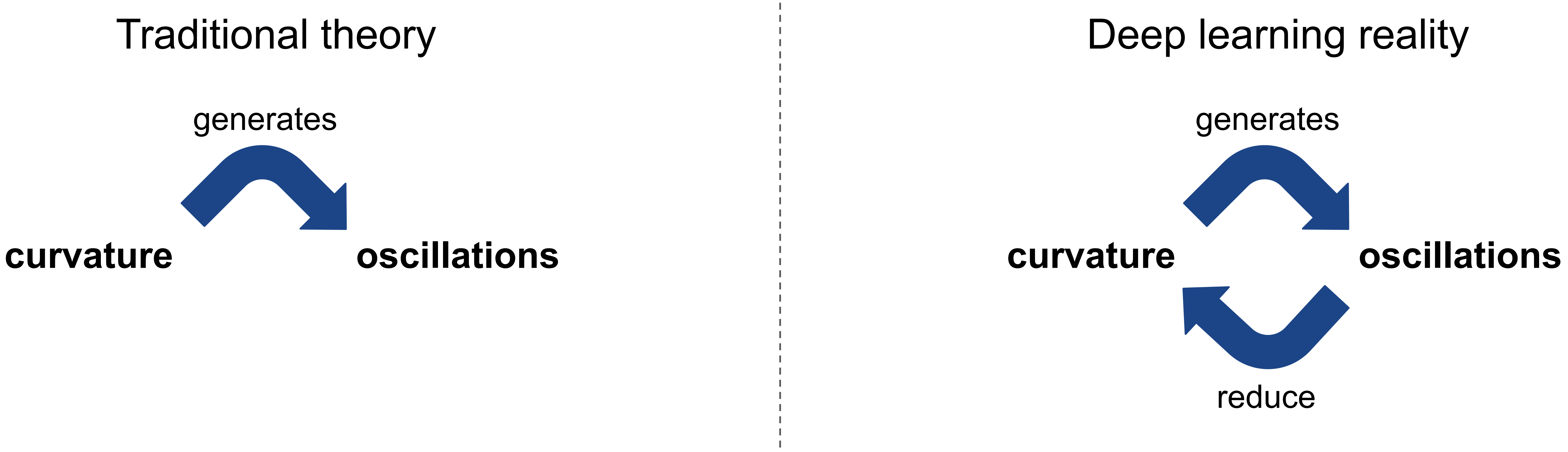
Cause and effect

Traditional theory

generates

curvature **oscillations**

Cause and effect



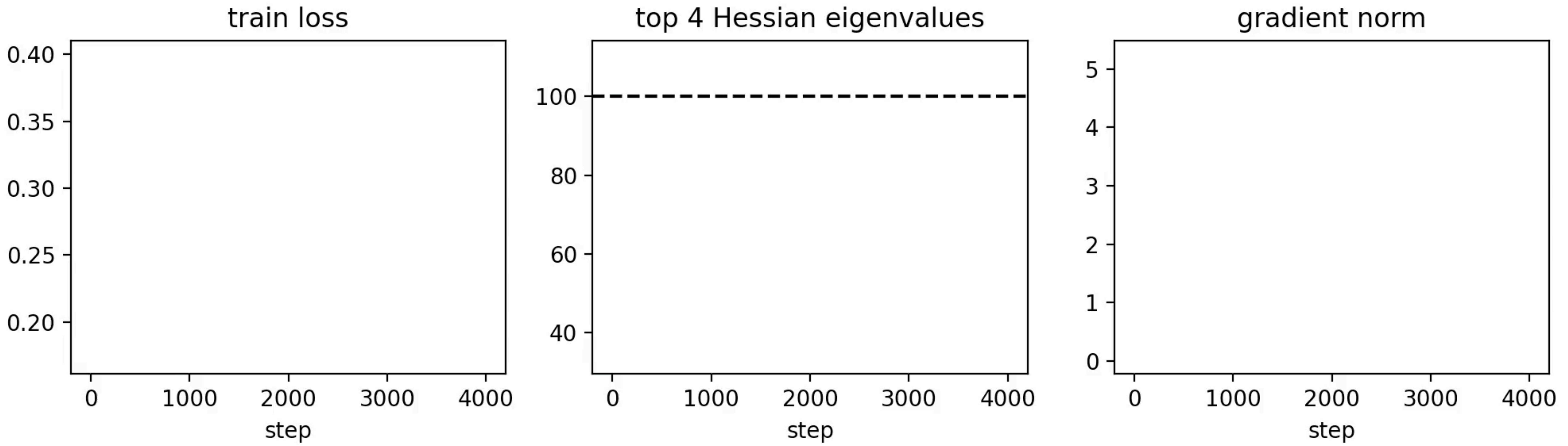
- Traditional optimization theory fails to capture the **causal structure** of the optimization process
- GD doesn't converge because the curvature is "a priori" small — it converges due to an **automatic negative feedback mechanism** that *keeps* the curvature small.

How can we analyze gradient descent?

- Unfortunately, EOS dynamics are challenging to analyze in fine-grained detail
- Need to track the mutual interactions between oscillations and curvature
- There are frequently *multiple* unstable eigenvalues => chaotic dynamics

How can we analyze gradient descent?

- Unfortunately, EOS dynamics are challenging to analyze in fine-grained detail
- Need to track the mutual interactions between oscillations and curvature
- There are frequently *multiple* unstable eigenvalues => chaotic dynamics



How can we analyze gradient descent?

Cohen*, Damian*, Talwalkar, Kolter, Lee. *Understanding Optimization in Deep Learning with Central Flows.* ICLR '25.


Alex Damian

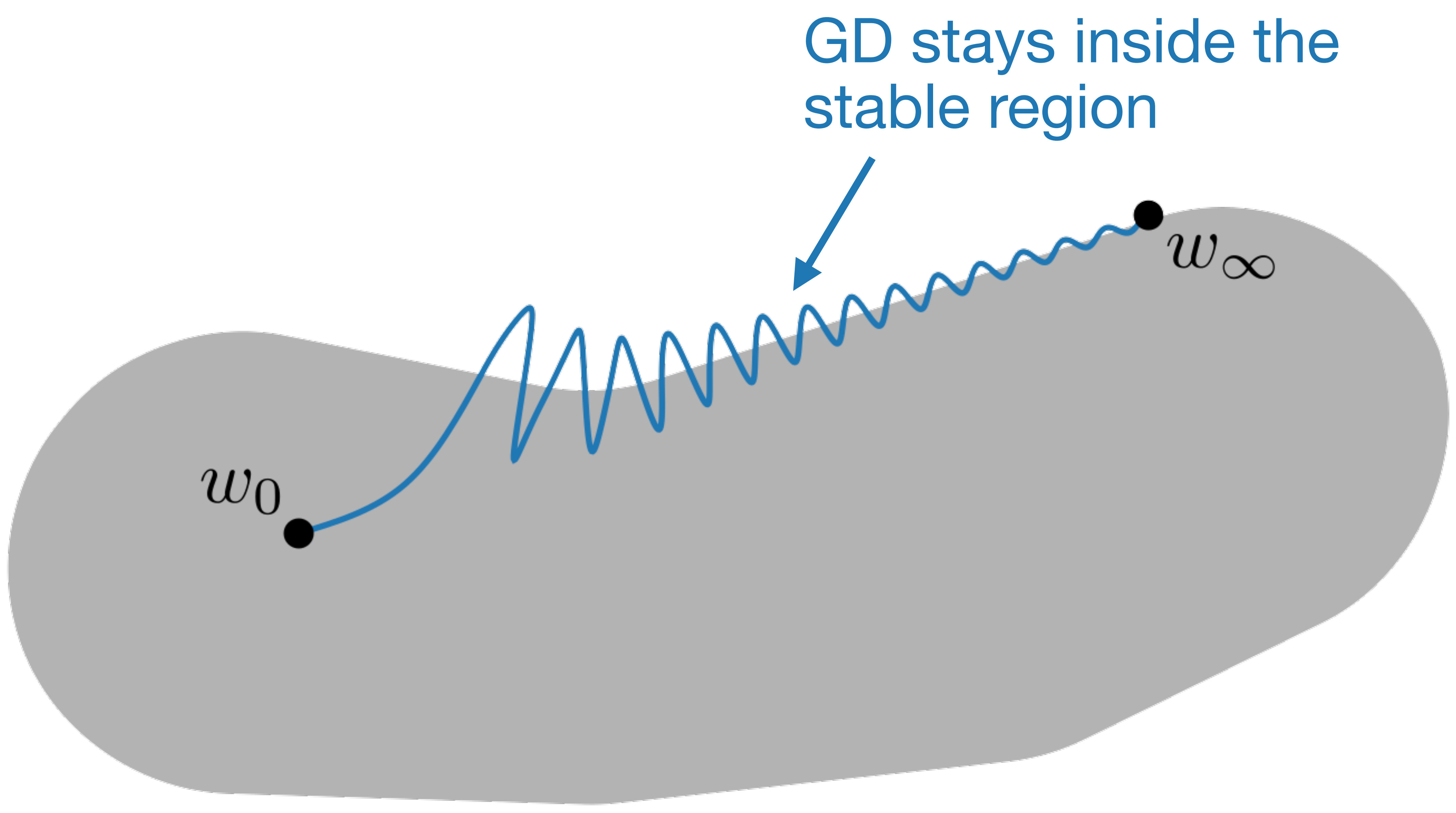
- We argue that the exact oscillatory GD trajectory doesn't matter
- Rather, what matters is the *macroscopic* path that GD takes
- This macroscopic path turns out to be much easier to understand
  - We only need to understand the oscillations in a *statistical* sense

What path does gradient descent take?

- The standard continuous-time approximation to GD is *gradient flow:*

$$\frac{dw}{dt} = -\eta \nabla L(w)$$

- GD follows gradient flow *before* EOS, but then takes a different path



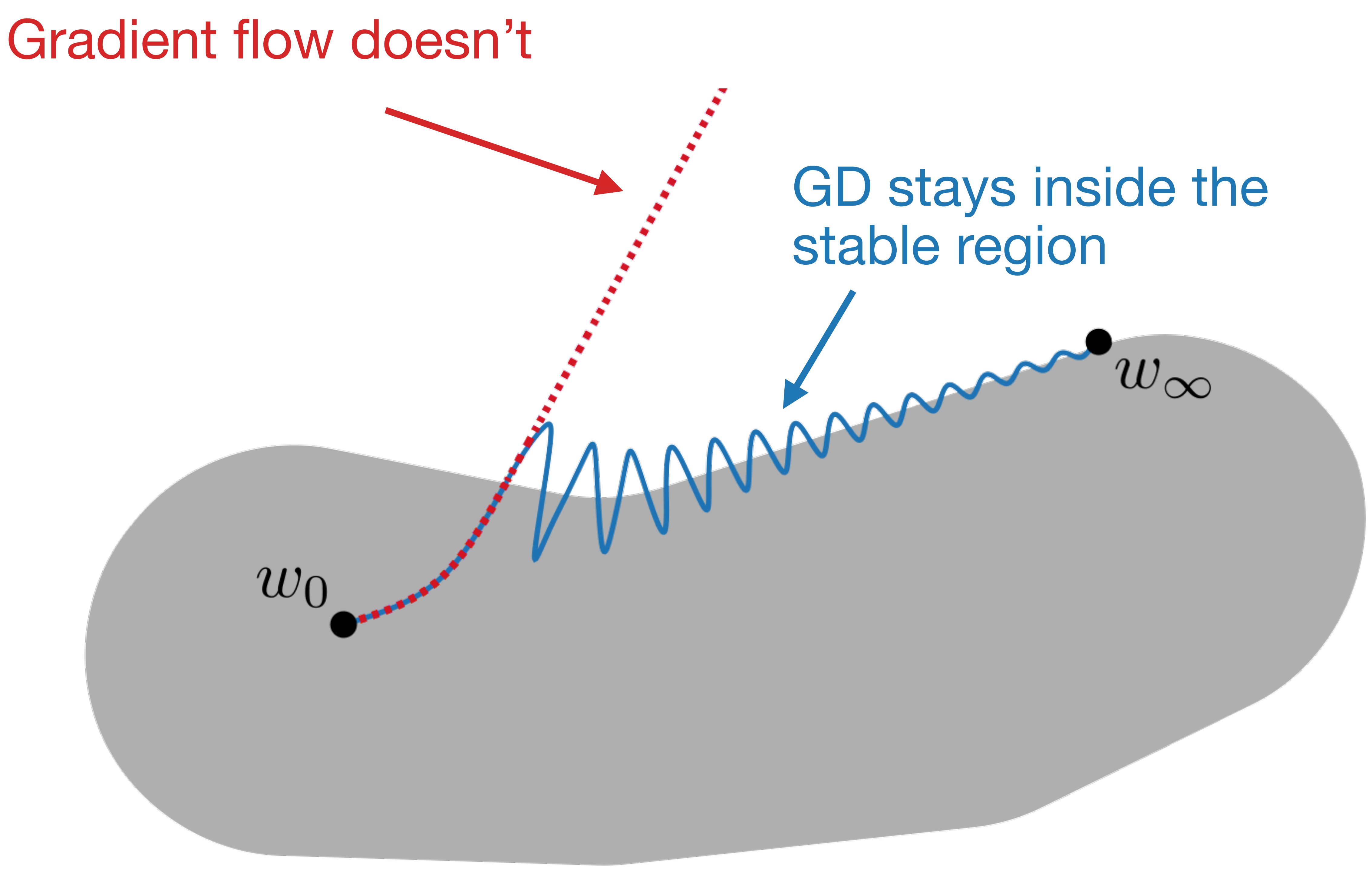
What path does gradient descent take?

- The standard continuous-time approximation to GD is *gradient flow:*

$$\frac{dw}{dt} = -\eta \nabla L(w)$$

- GD follows gradient flow *before* EOS, but then takes a different path

- Our *central flow* matches the trajectory of GD even at EOS.



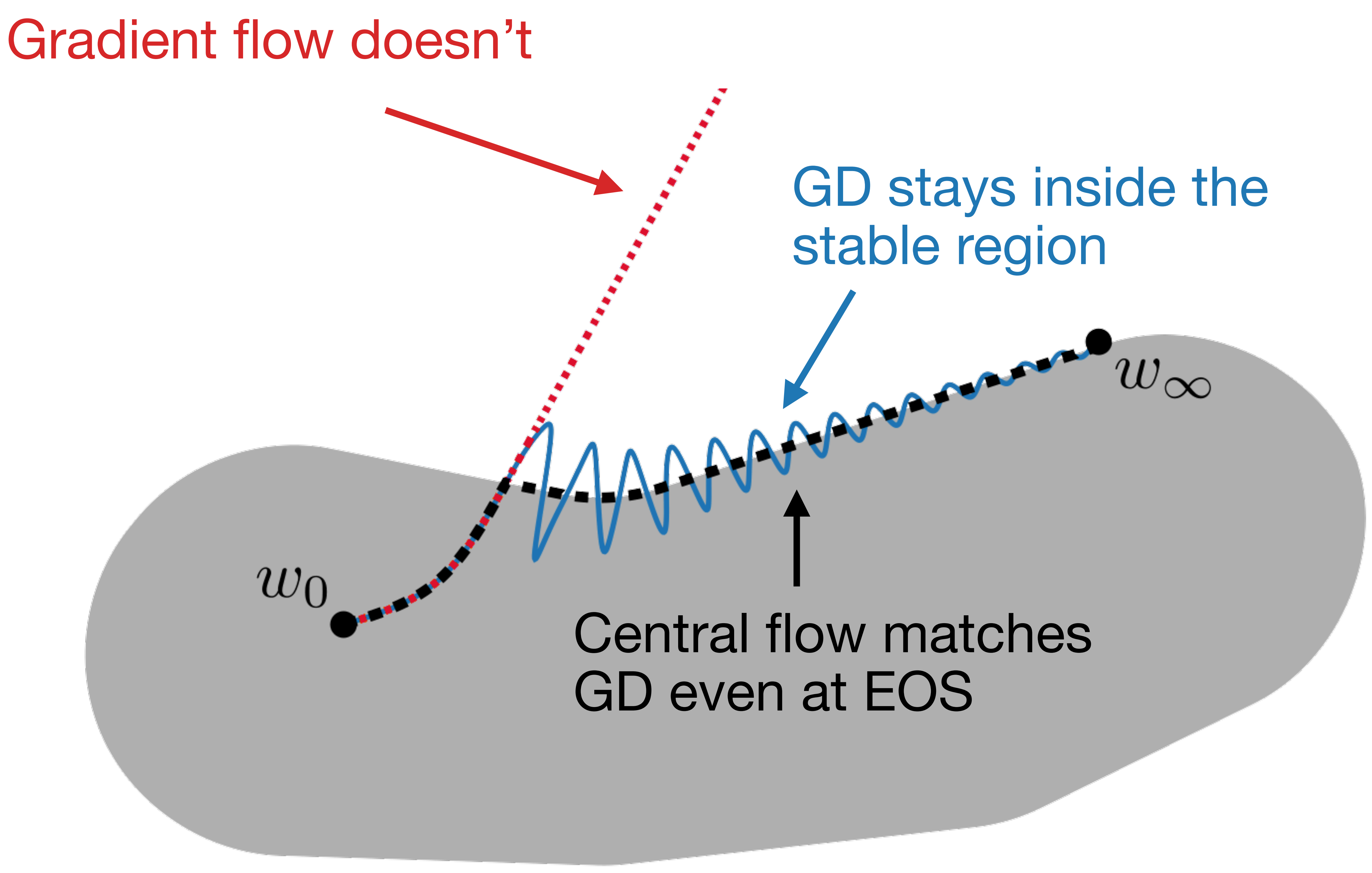
What path does gradient descent take?

- The standard continuous-time approximation to GD is *gradient flow:*

$$\frac{dw}{dt} = -\eta \nabla L(w)$$

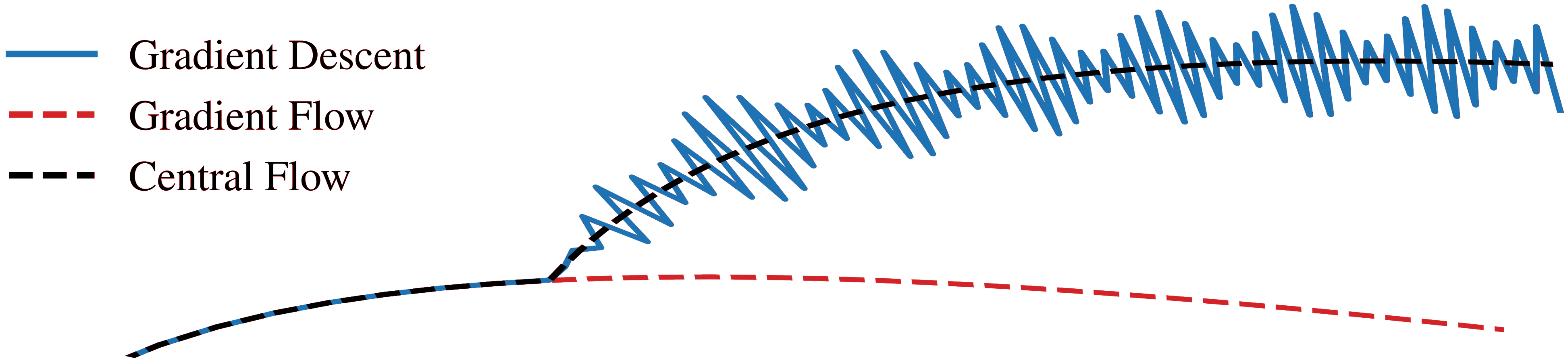
- GD follows gradient flow *before* EOS, but then takes a different path

- Our *central flow* matches the trajectory of GD even at EOS.



Central flow

- The central flow models the *time-averaged* (i.e. smoothed) GD trajectory



Deriving the central flow

- We derive the central flow using informal mathematical reasoning, and we show empirically that this flow matches the real GD trajectory
- In particular:
    - We suppose that the time-averaged trajectory can be described by a flow
    - We argue that only one flow makes sense (the central flow)
    - We show empirically that this flow matches GD in a variety of DL settings

Example: special case of 1 unstable eigenvalue

- We model the iterate as:

time-averaged iterate — top Hessian eigenvector

$$w_t = w(t) + x_t \, u_t$$

iterate — magnitude of oscillation

- Then the gradient is:

gradient at time-averaged iterate — sharpness reduction

$$\nabla L(w_t) \approx \nabla L(w(t)) + x_t \, S(w(t)) \, u_t \; + \; \frac{1}{2} x^2 \, \nabla S(w(t))$$

gradient at iterate — oscillation

- So the "time-averaged" gradient is:

variance of oscillations

gradient at time-averaged iterate — sharpness reduction

$$\mathbb{E}[\, \nabla L(w_t)] \approx \nabla L(w(t)) + \mathbb{E}[x_t] \, S(w(t)) \, u_t \; + \; \frac{1}{2} \mathbb{E}[x^2] \, \nabla S(w(t))$$

time-averaged gradient — oscillation

Example: special case of 1 unstable eigenvalue

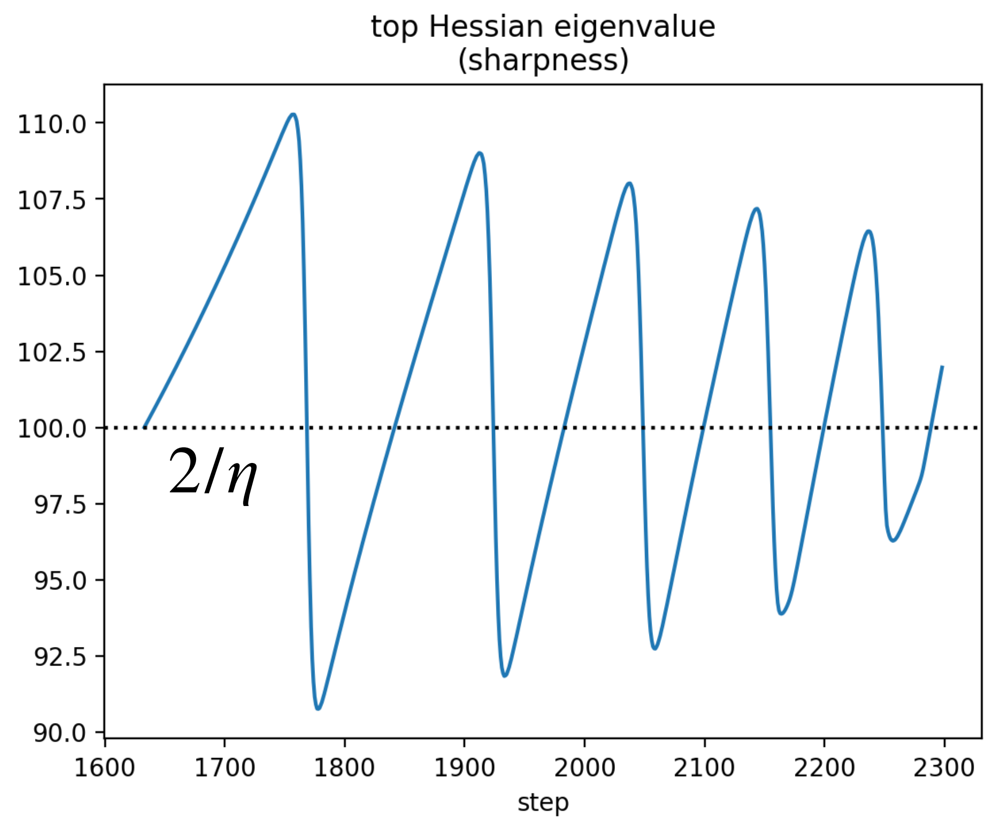
- We suppose that the time-averaged GD trajectory follows an ODE of the form:

"instantaneous variance" of the oscillations
(i.e. local time average of x^2)

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2} \sigma^2(t) \nabla S(w) \right]$$

gradient flow — sharpness penalty

- This flow averages out the oscillations, but keeps their *effect* on the trajectory.
- To determine $\sigma^2(t)$, we argue that only one value is possible.
  - Empirically, the sharpness equilibrates at $2/\eta$.
  - Therefore, we enforce that along the central flow, $\frac{dS}{dt} = 0$.



Example: special case of 1 unstable eigenvalue

- The time derivative of the sharpness under our flow is:

$$\frac{dS}{dt} = \langle \nabla S(w), \frac{dw}{dt} \rangle \quad \text{chain rule}$$

$$= \left\langle \nabla S(w), -\eta \left[\nabla L(w) \; + \; \tfrac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \right\rangle \quad \text{substitute in our flow}$$

$$= \langle \nabla S(w), -\eta \nabla L(w) \rangle \; - \; \frac{1}{2} \eta \, \sigma^2(t) \, \| \nabla S(w) \|^2 \quad \text{simplify}$$

time derivative of sharpness under gradient flow

sharpness-reduction effect of oscillations

- We see that $\frac{dS}{dt}$ is **affine** in $\sigma^2(t)$. In order for $\frac{dS}{dt} = 0$, $\sigma^2(t)$ must be:

$$\sigma^2(t) = \frac{2 \langle -\nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$

Central flow in action

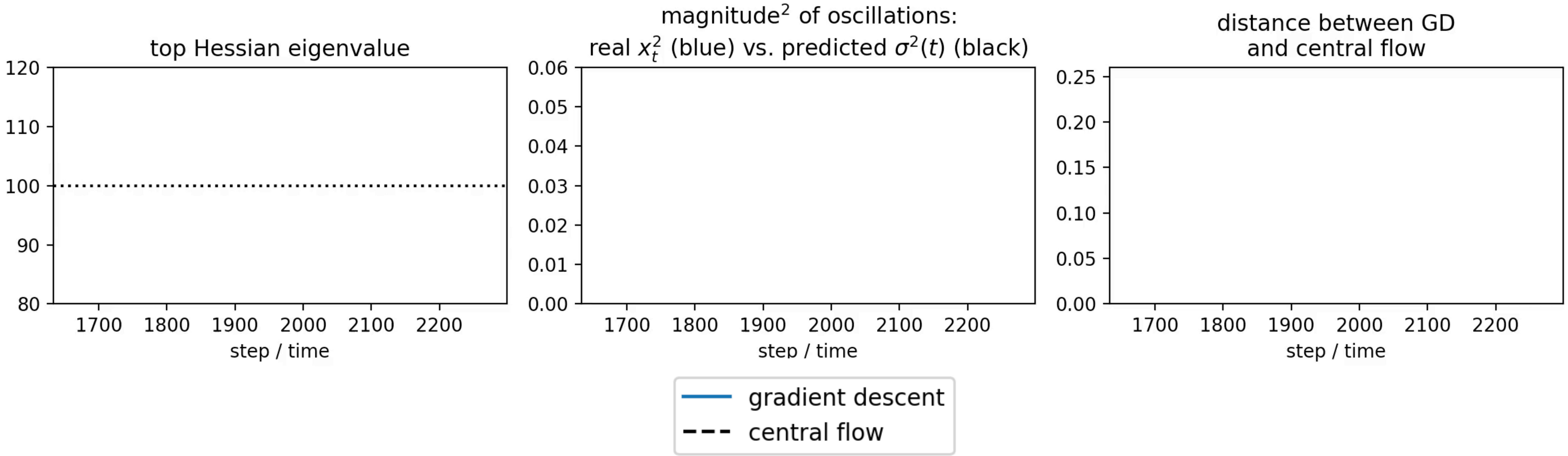
- The central flow for a single unstable eigenvalue is:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$

Central flow in action

- The central flow for a single unstable eigenvalue is:

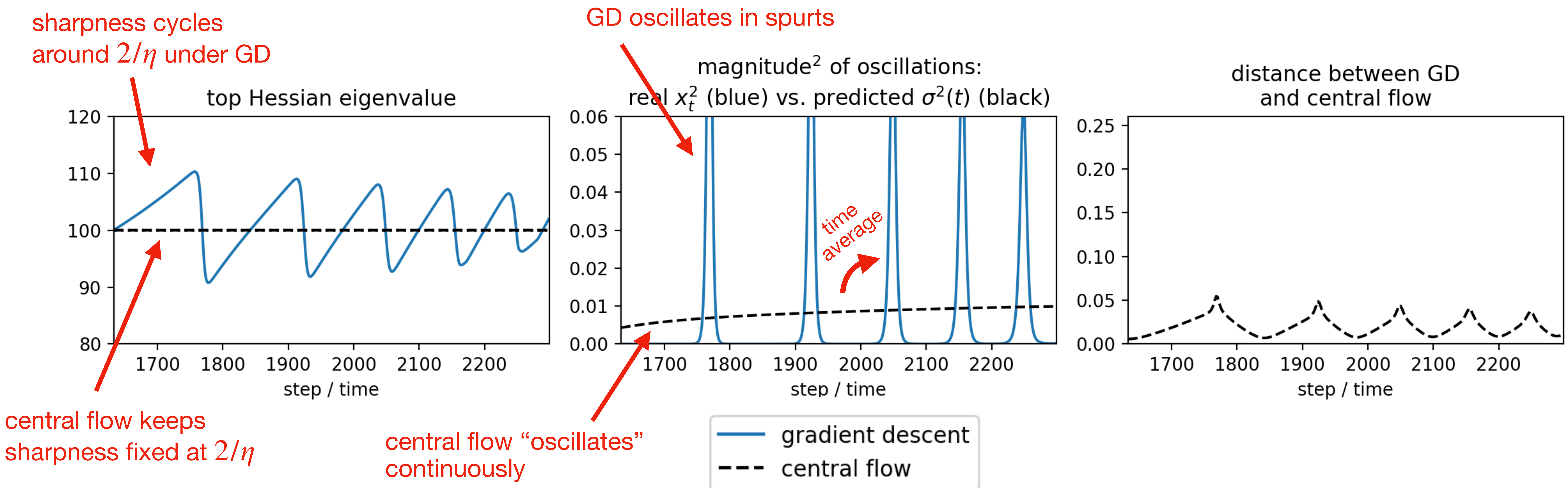
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2}\sigma^2(t)\, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2\nabla L(w), \nabla S(w) \rangle}{\|\nabla S(w)\|^2}$$



Central flow in action

- The central flow for a single unstable eigenvalue is:

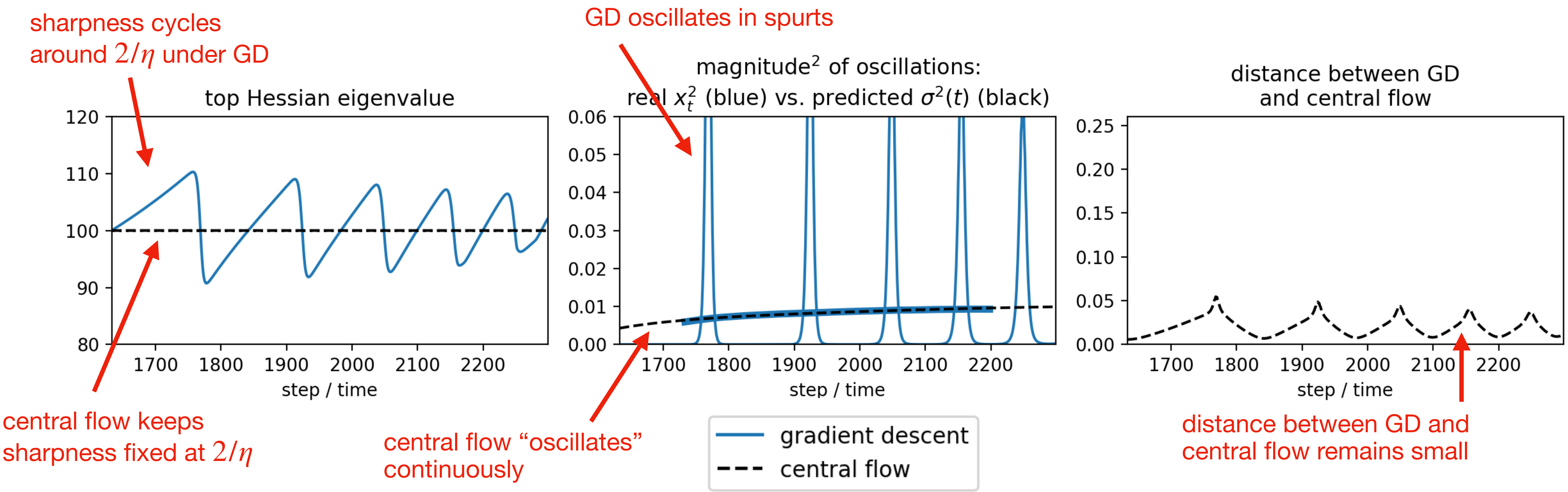
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2}\sigma^2(t)\,\nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2\nabla L(w), \nabla S(w)\rangle}{\|\nabla S(w)\|^2}$$



Central flow in action

- The central flow for a single unstable eigenvalue is:

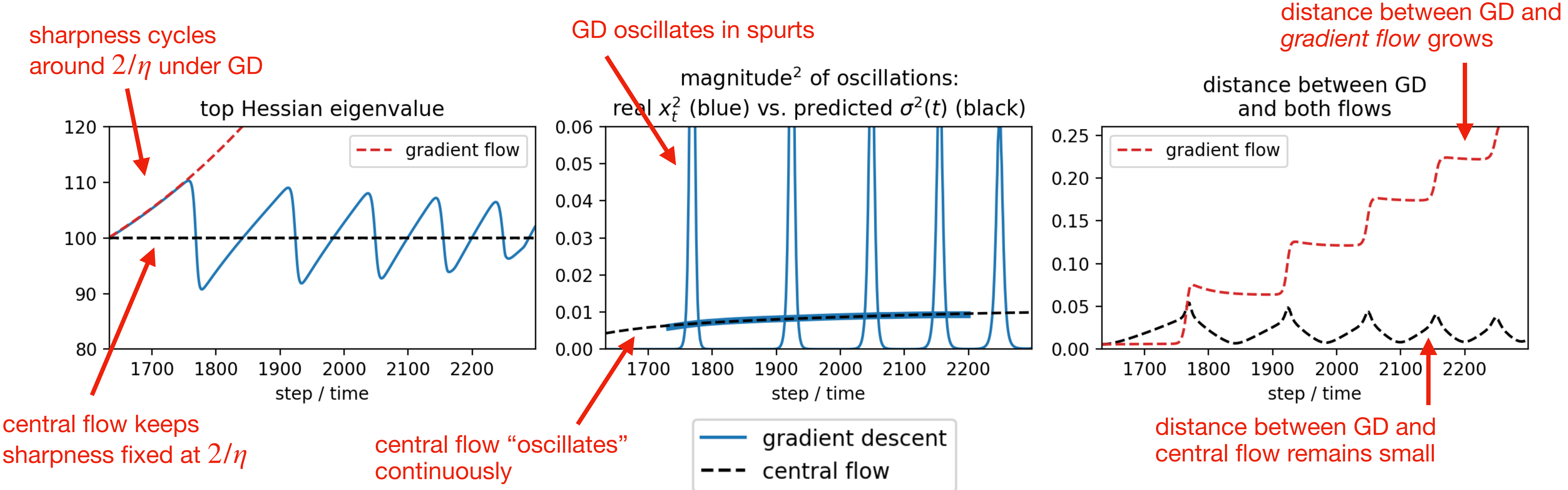
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2} \sigma^2(t) \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\|\nabla S(w)\|^2}$$



Central flow in action

- The central flow for a single unstable eigenvalue is:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2}\sigma^2(t)\, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2\nabla L(w), \nabla S(w) \rangle}{\|\nabla S(w)\|^2}$$



Takeaways

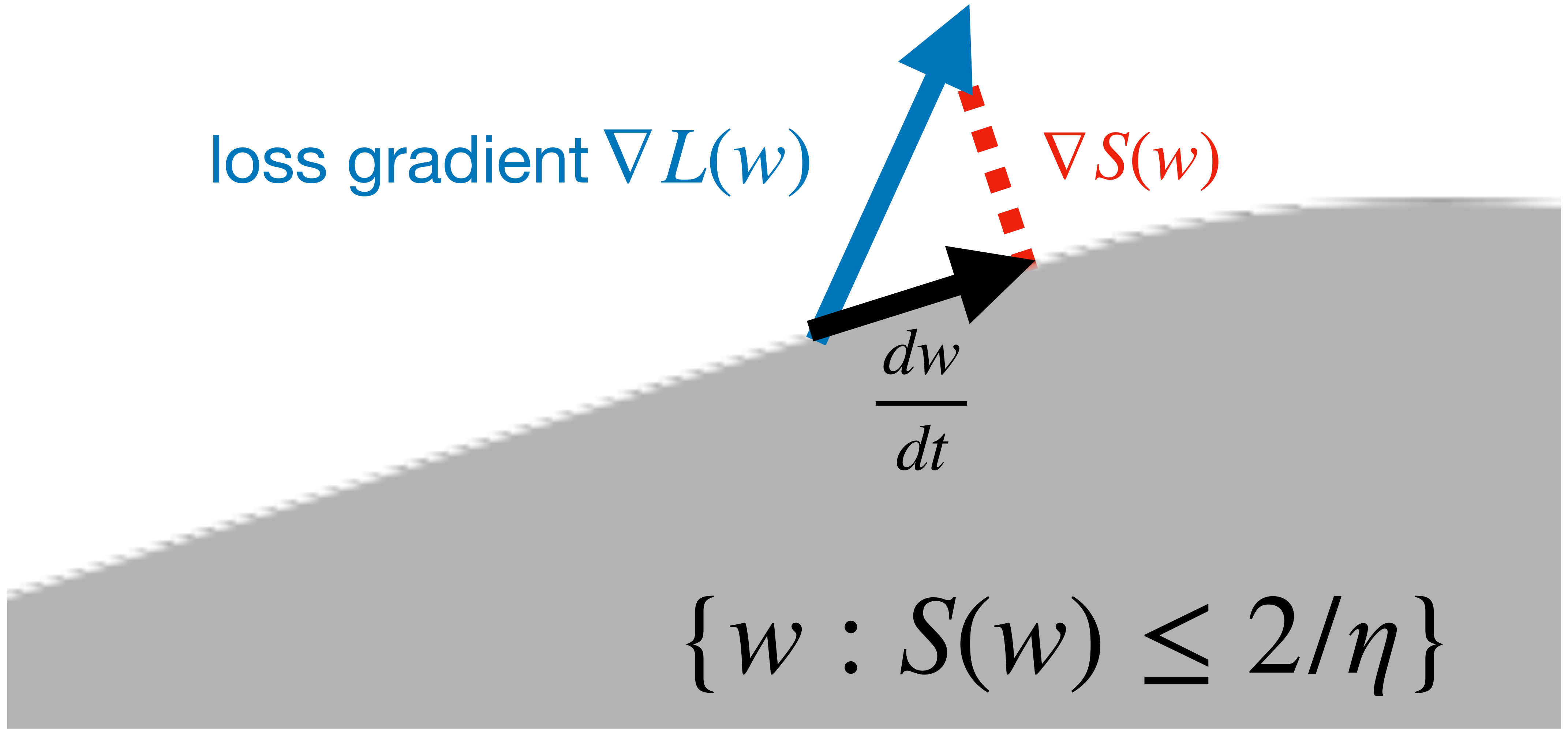
- It’s challenging to understand the oscillations in fine-grained detail
- But the *macroscopic* trajectory only depends on the *variance* of the oscillations
- This variance is easy to obtain
  - There is only one value that is compatible with the edge of stability equilibrium

Interpretation as projection

- The central flow can be equivalently interpreted as a *projected* gradient flow:

$$\frac{dw}{dt} = -\eta \left[I - \frac{\nabla S(w) \nabla S(w)^T}{\|\nabla S(w)\|^2} \right] \nabla L(w)$$

project out $\nabla S(w)$ direction from $\nabla L(w)$
to keep sharpness $S(w)$ fixed in place



Complete central flow

- Similar to before, we make the ansatz that the time-averaged iterates follow:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2} \nabla_w \langle H(w), \Sigma(t) \rangle \right]$$

implicit curvature penalty

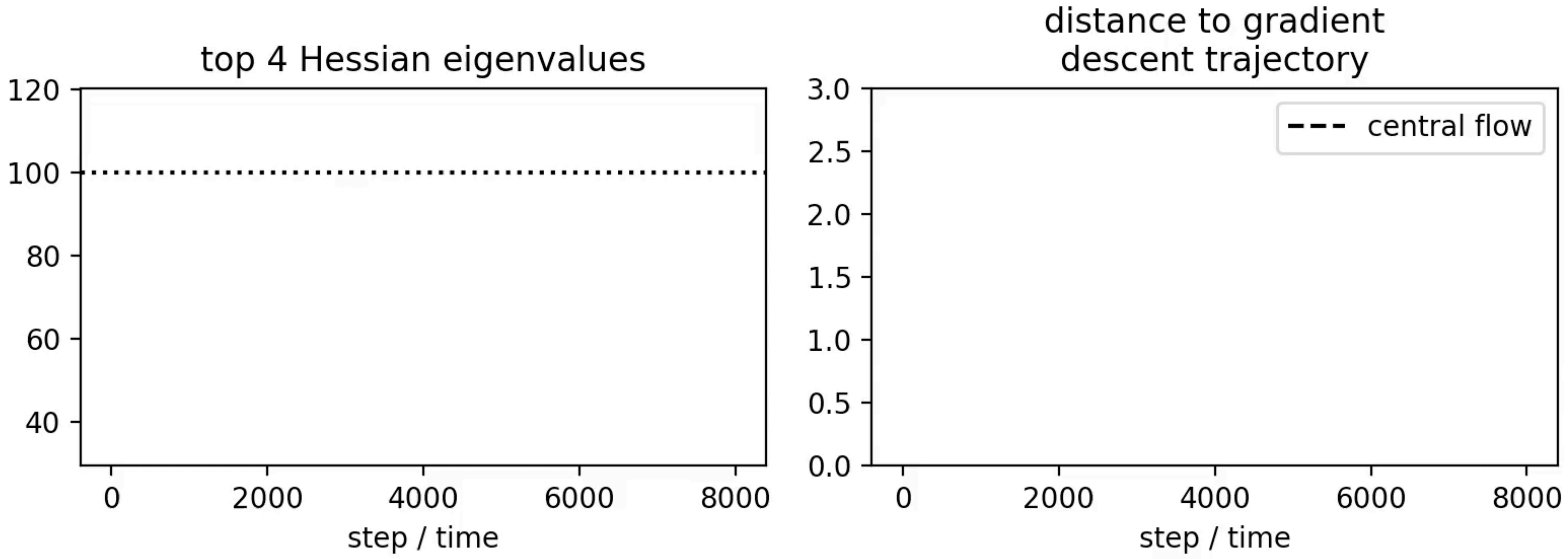
where $\Sigma(t)$ models the $\mathbb{E}[\delta_t \delta_t^T]$, the covariance of the oscillations.

- We argue that only one value of $\Sigma(t)$ is possible.

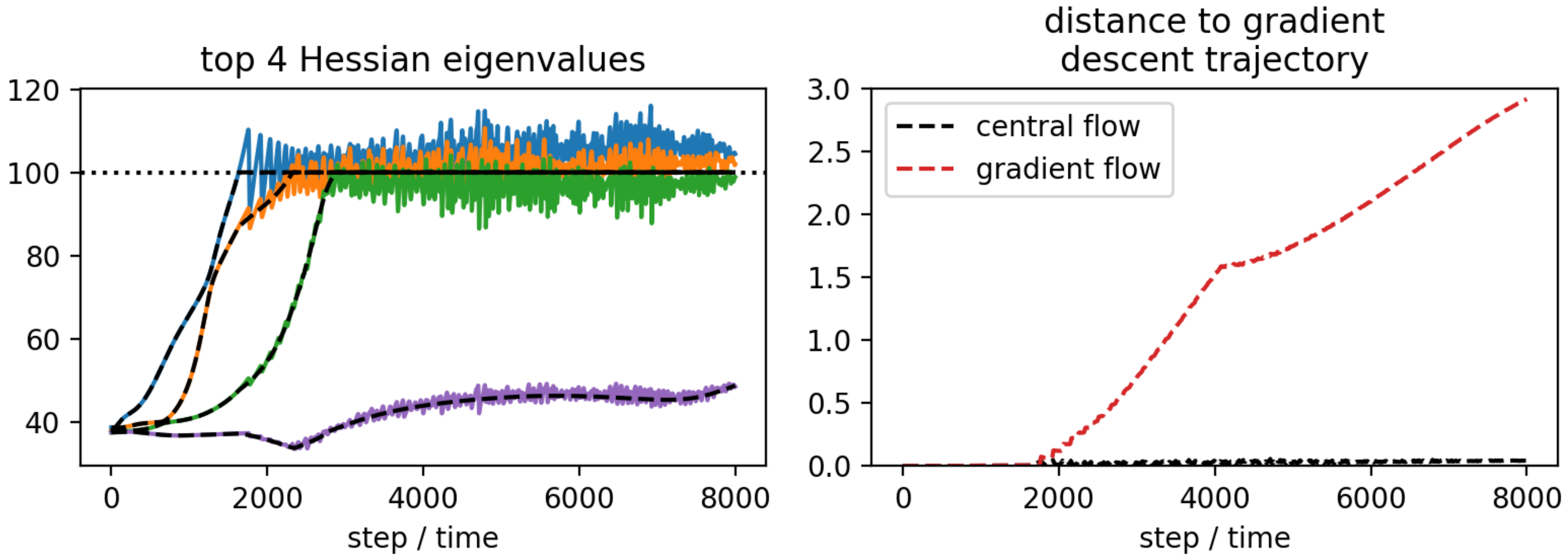
Complete central flow

- We impose three conditions on the central flow:
    1. The flow should not increase any Hessian eigenvalues above $2/\eta$
    2. $\Sigma(t)$ should be supported within the Hessian's $2/\eta$ eigenspace
    3. $\Sigma(t)$ should be positive semidefinite
- These three conditions imply that $\Sigma(t)$ must be the solution to a certain *cone complementarity problem.*
- The central flow is defined with this $\Sigma(t)$.

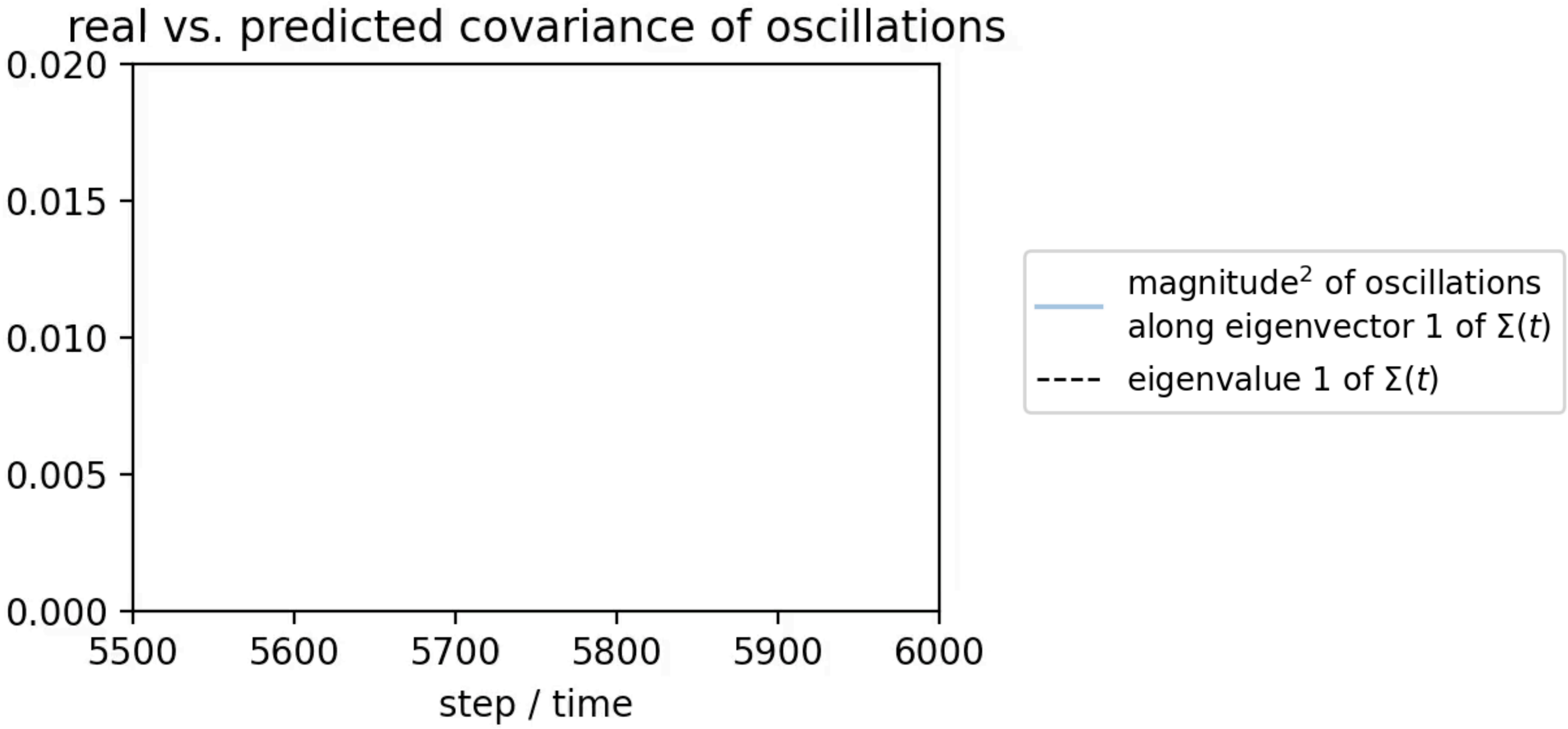
Central flow in action



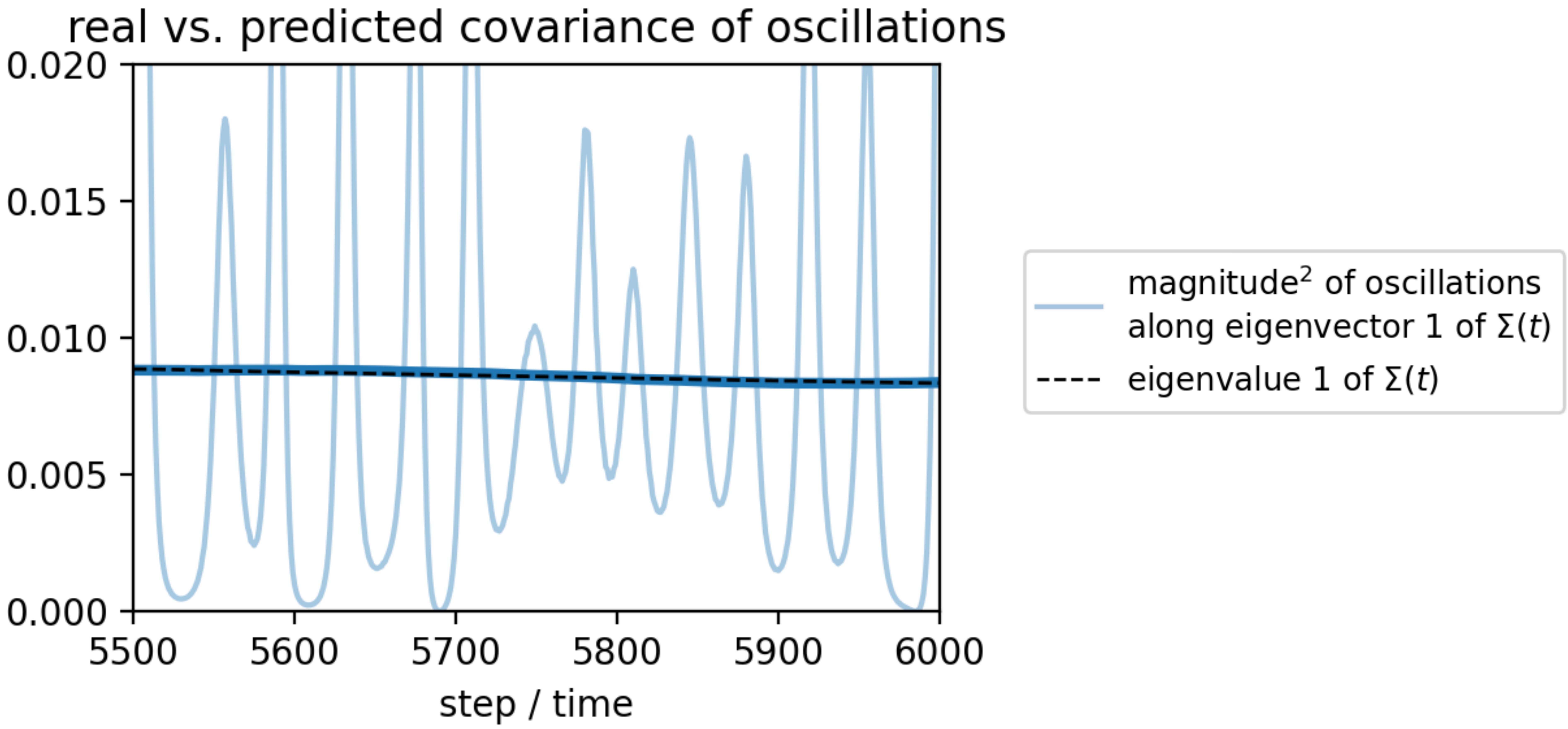
Central flow in action



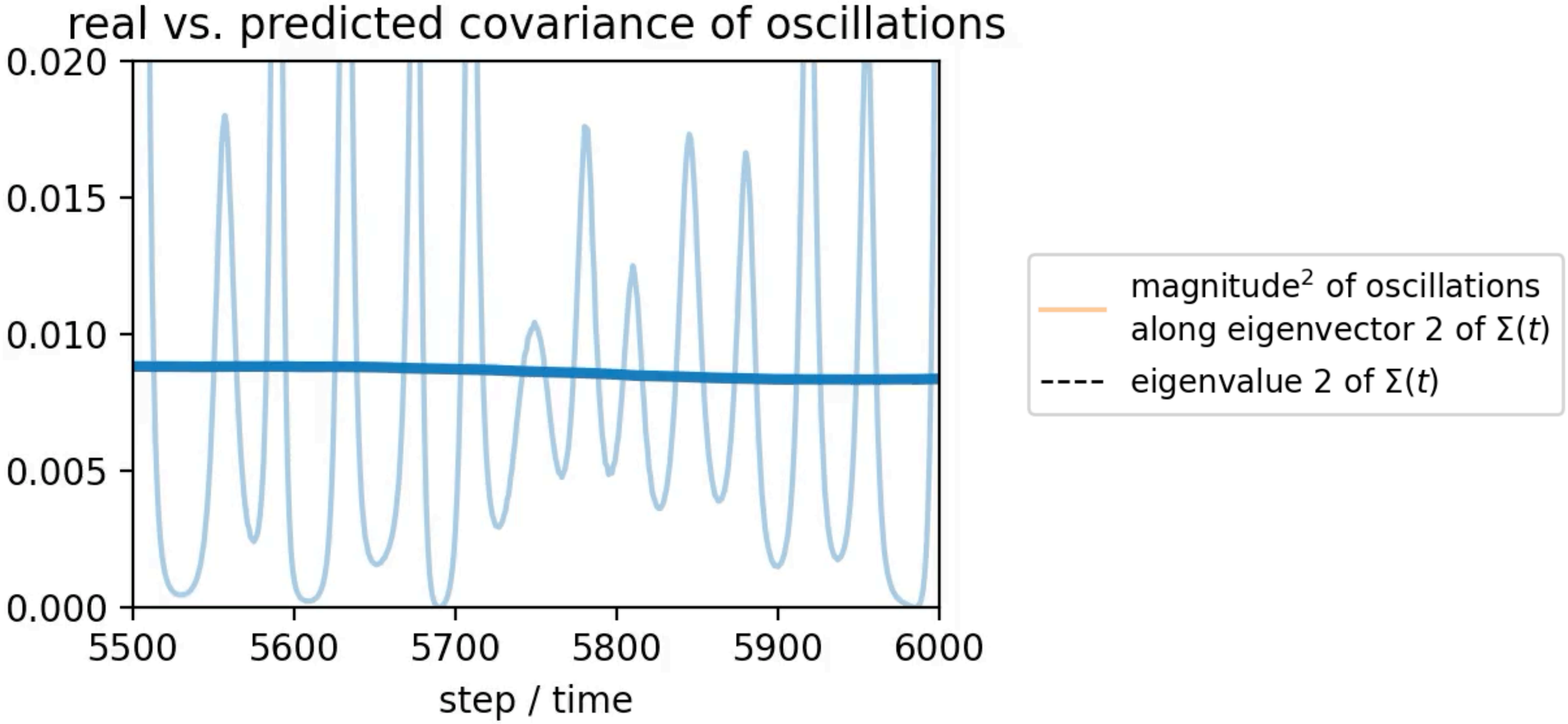
Central flow can predict oscillation covariance



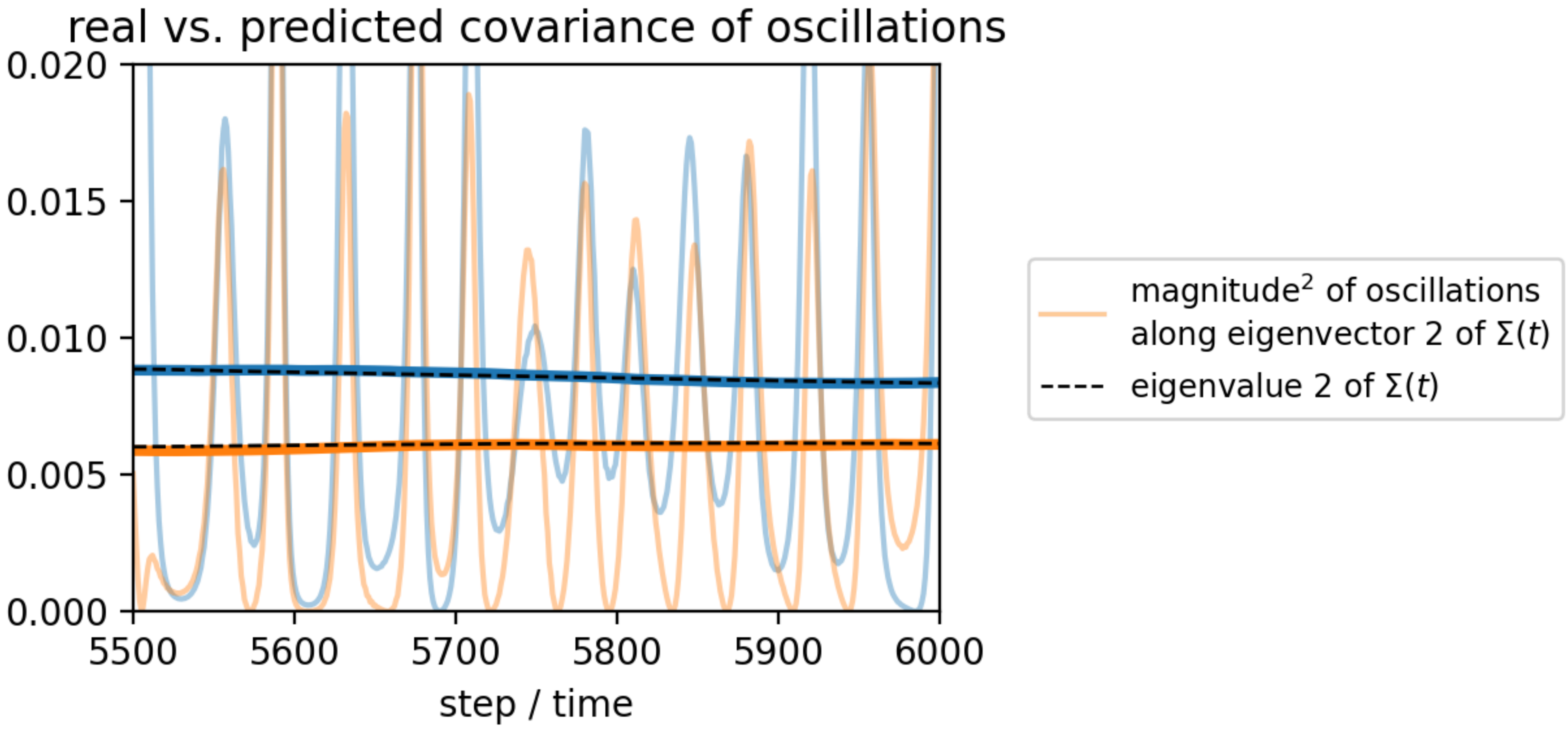
Central flow can predict oscillation covariance



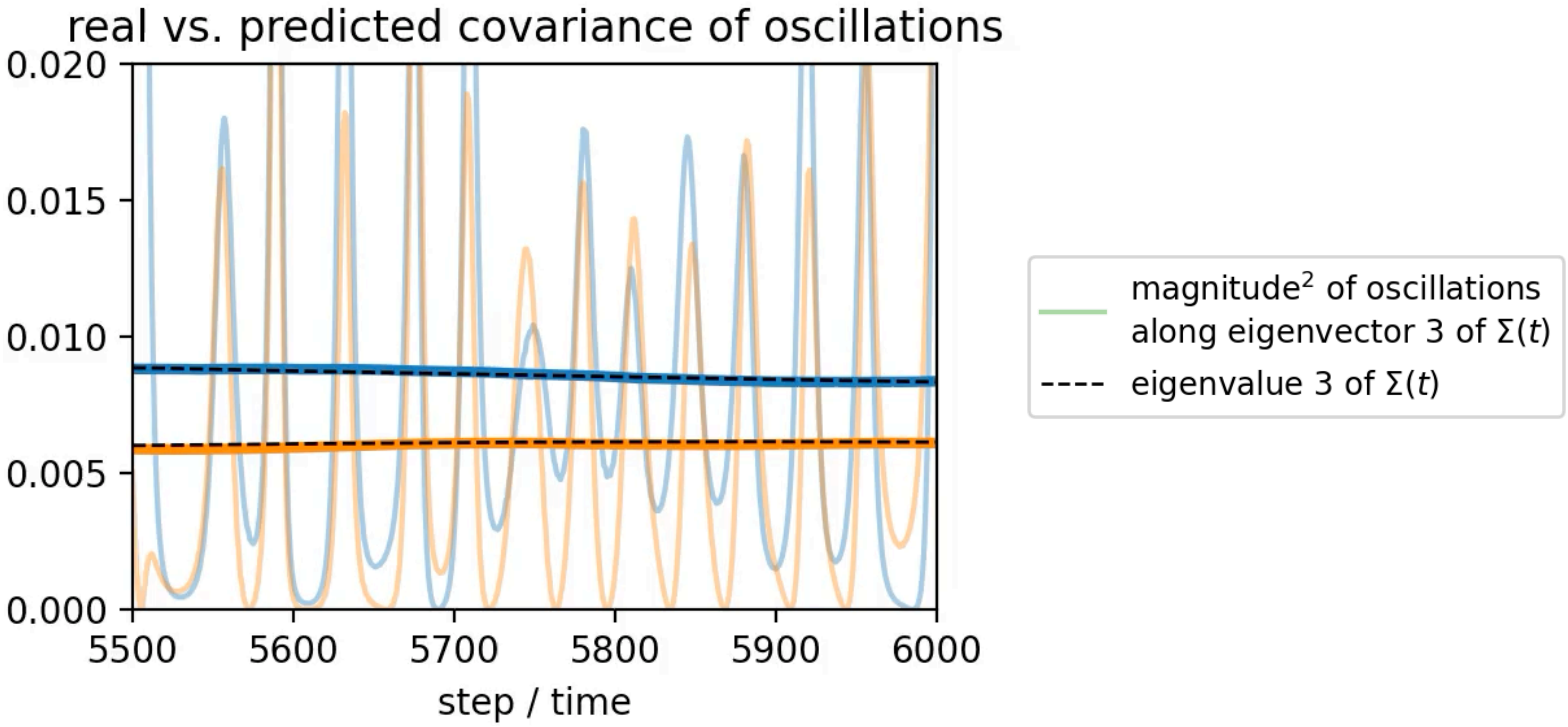
Central flow can predict oscillation covariance



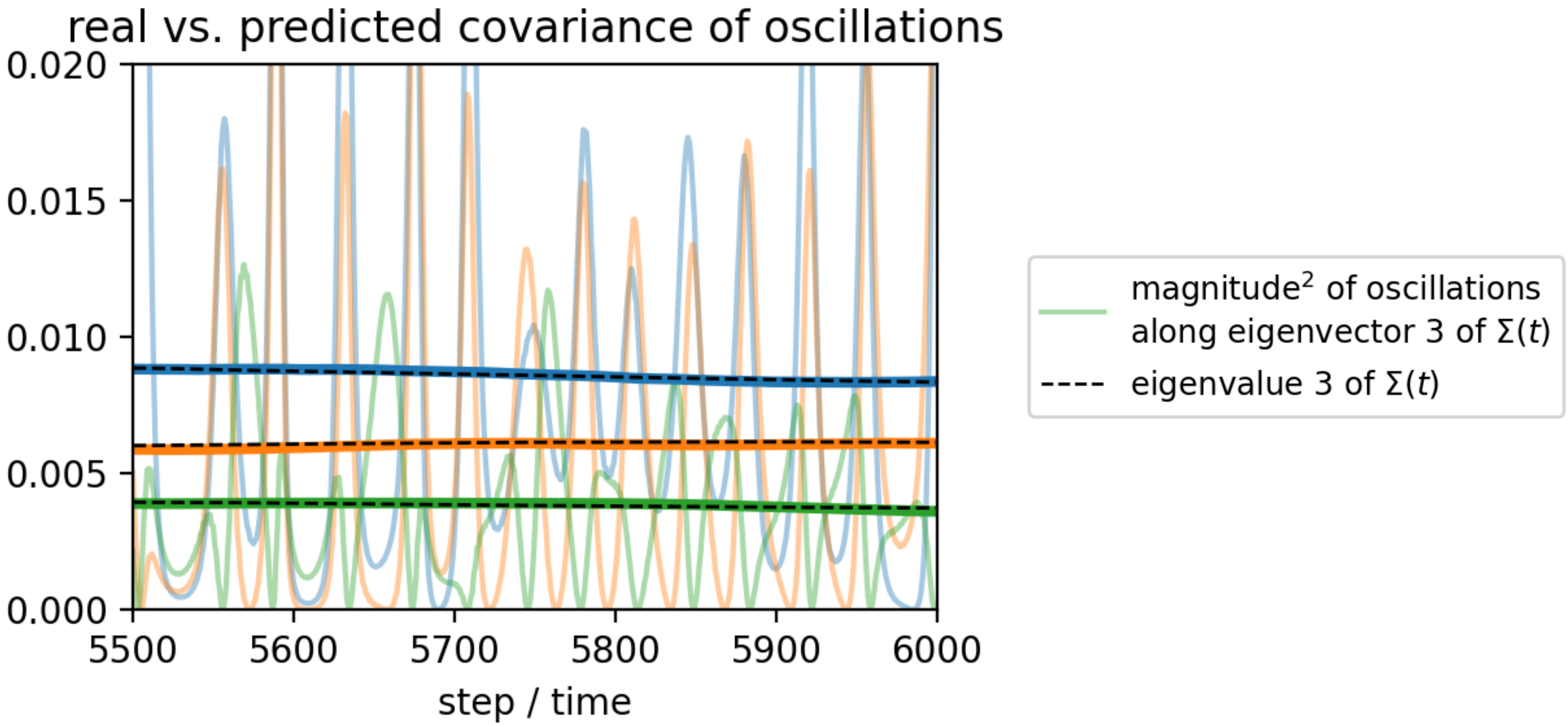
Central flow can predict oscillation covariance



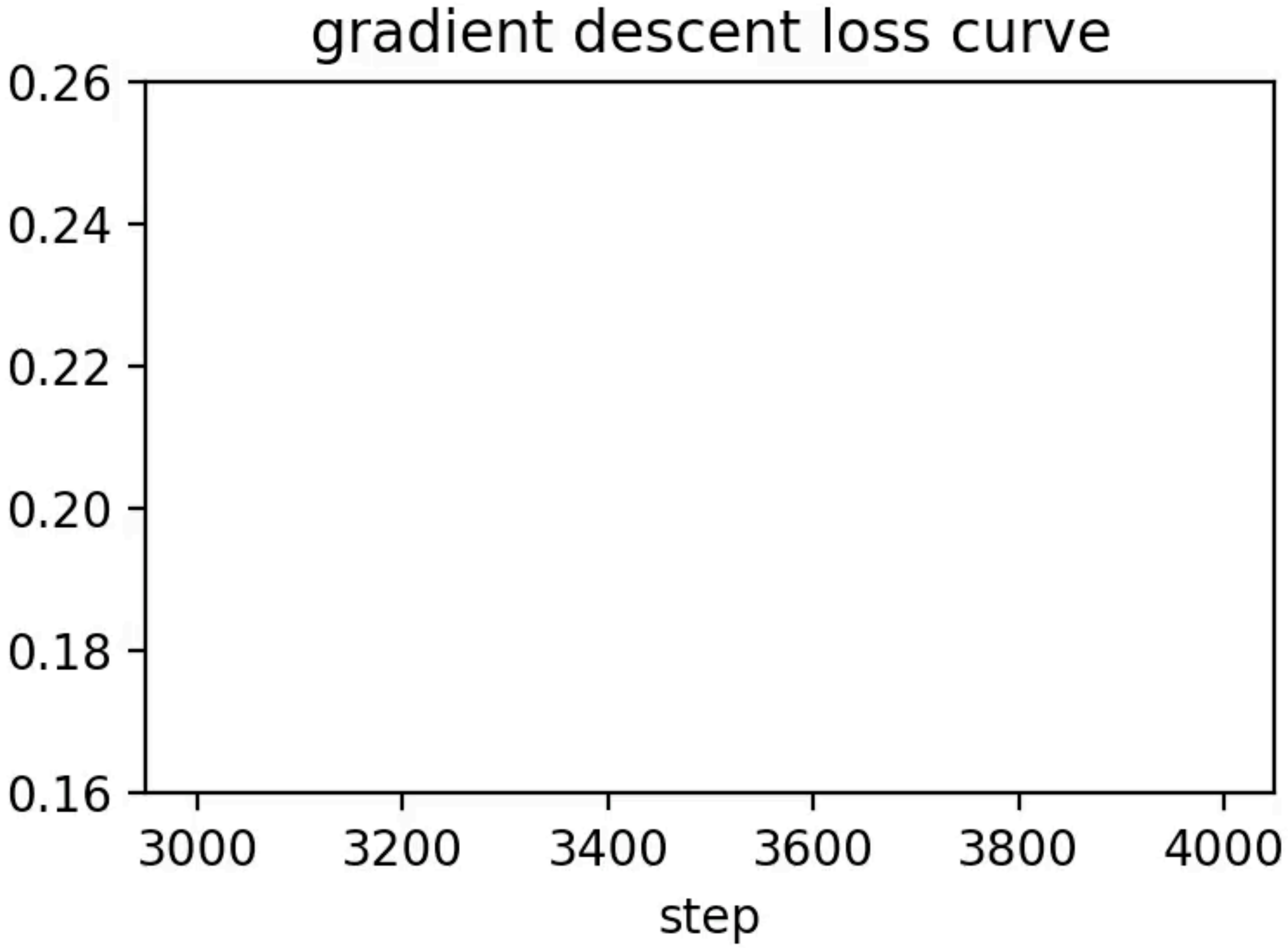
Central flow can predict oscillation covariance



Central flow can predict oscillation covariance

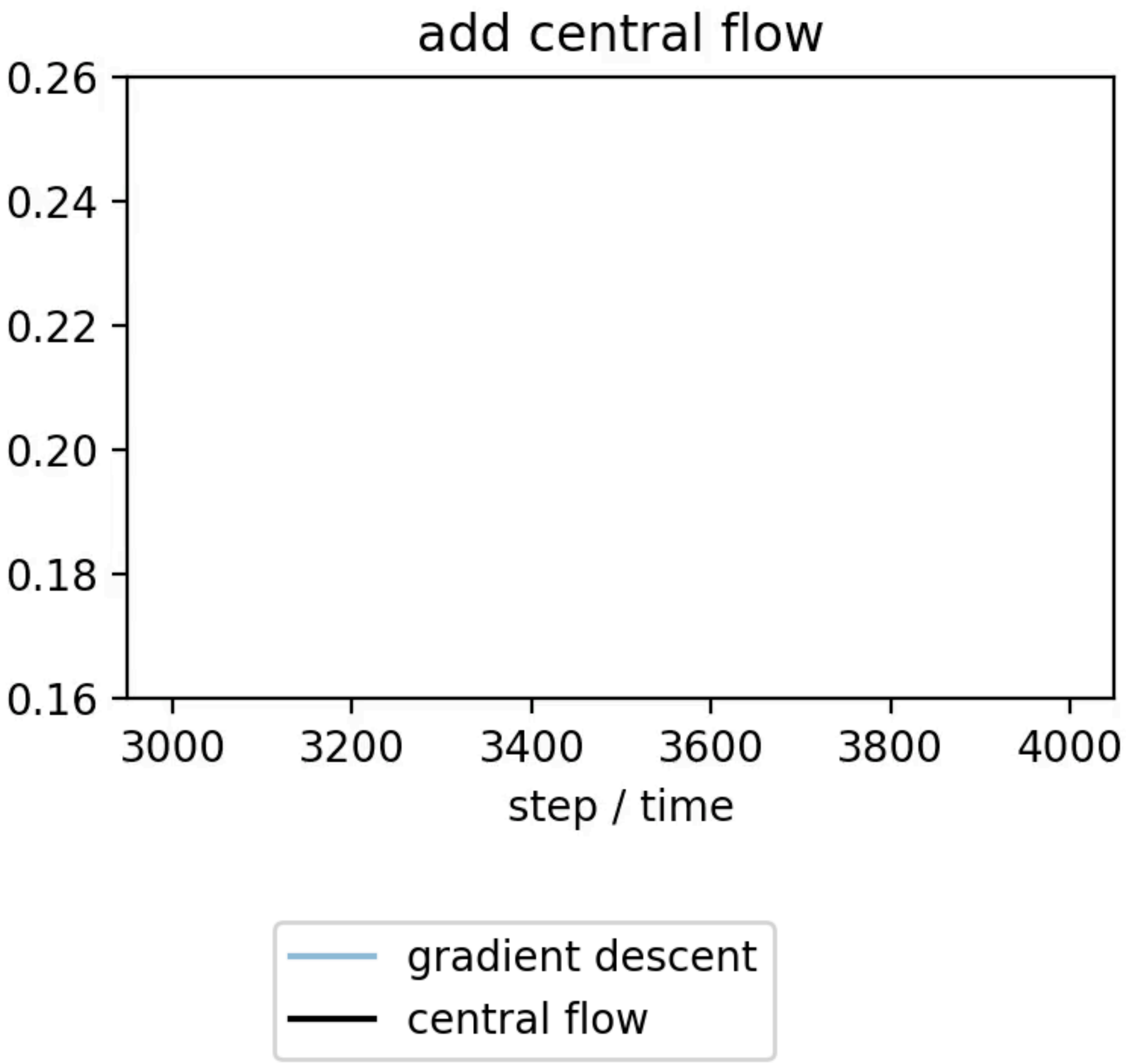


Application: reasoning about loss curves



- The gradient descent loss curve is non-monotonic…

Application: reasoning about loss curves

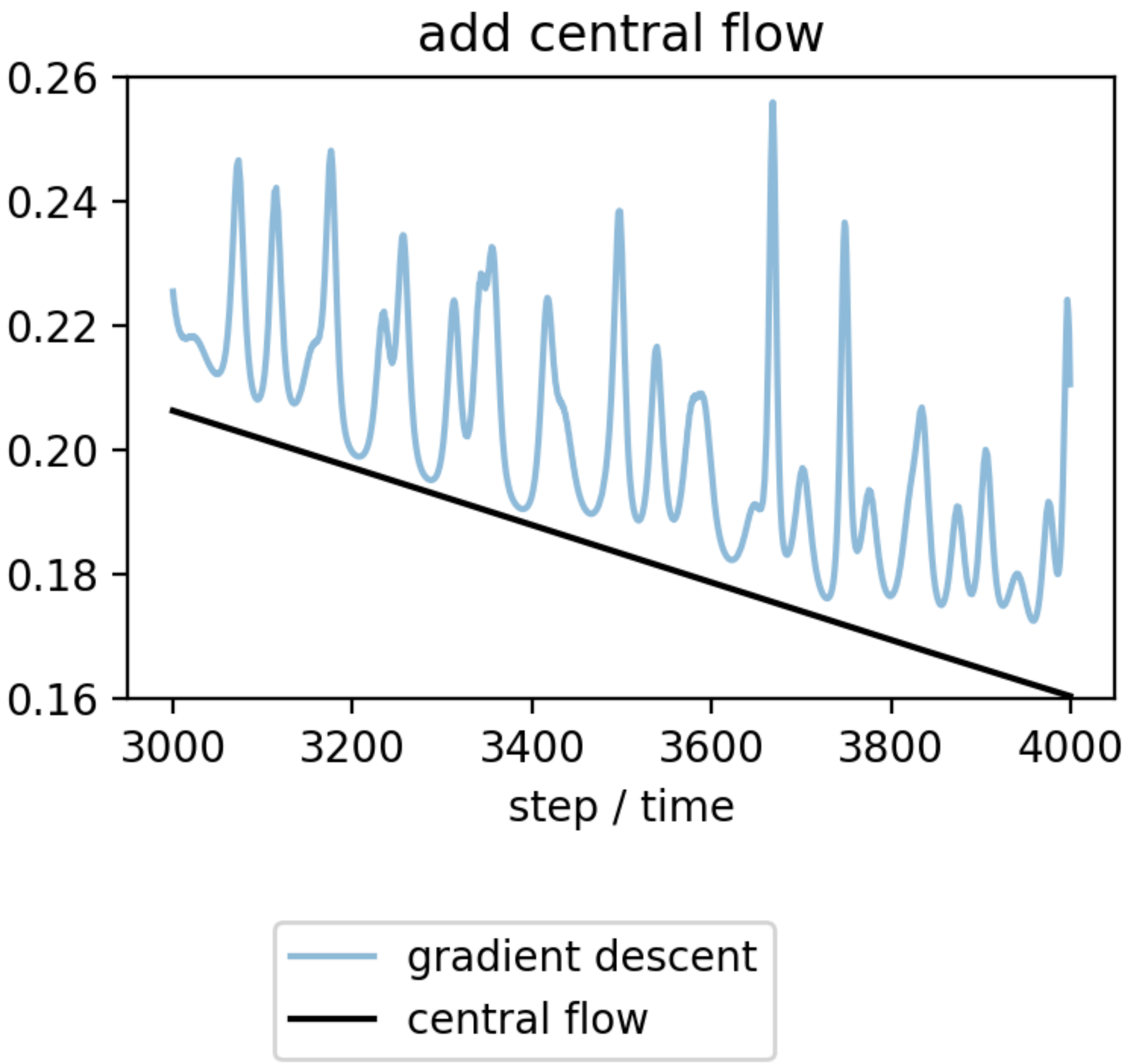


- The gradient descent loss curve is non-monotonic…
- … but the *central flow* loss monotonically decreases:

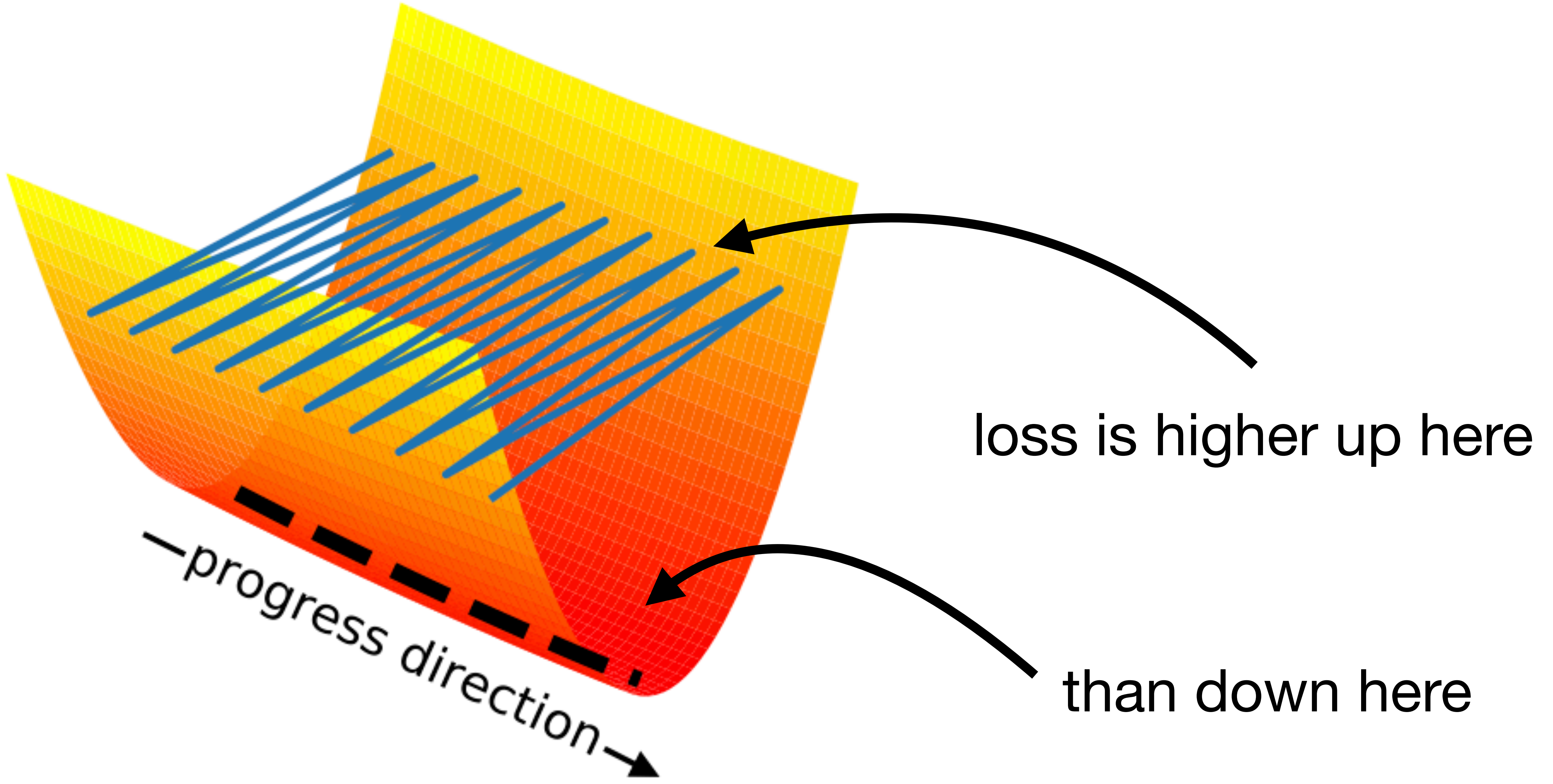
$$\frac{dL(w(t))}{dt} \leq 0$$

- The central flow loss $L(w(t))$ is a **potential function** for the optimization process.
- Its slope quantifies the speed of optimization.

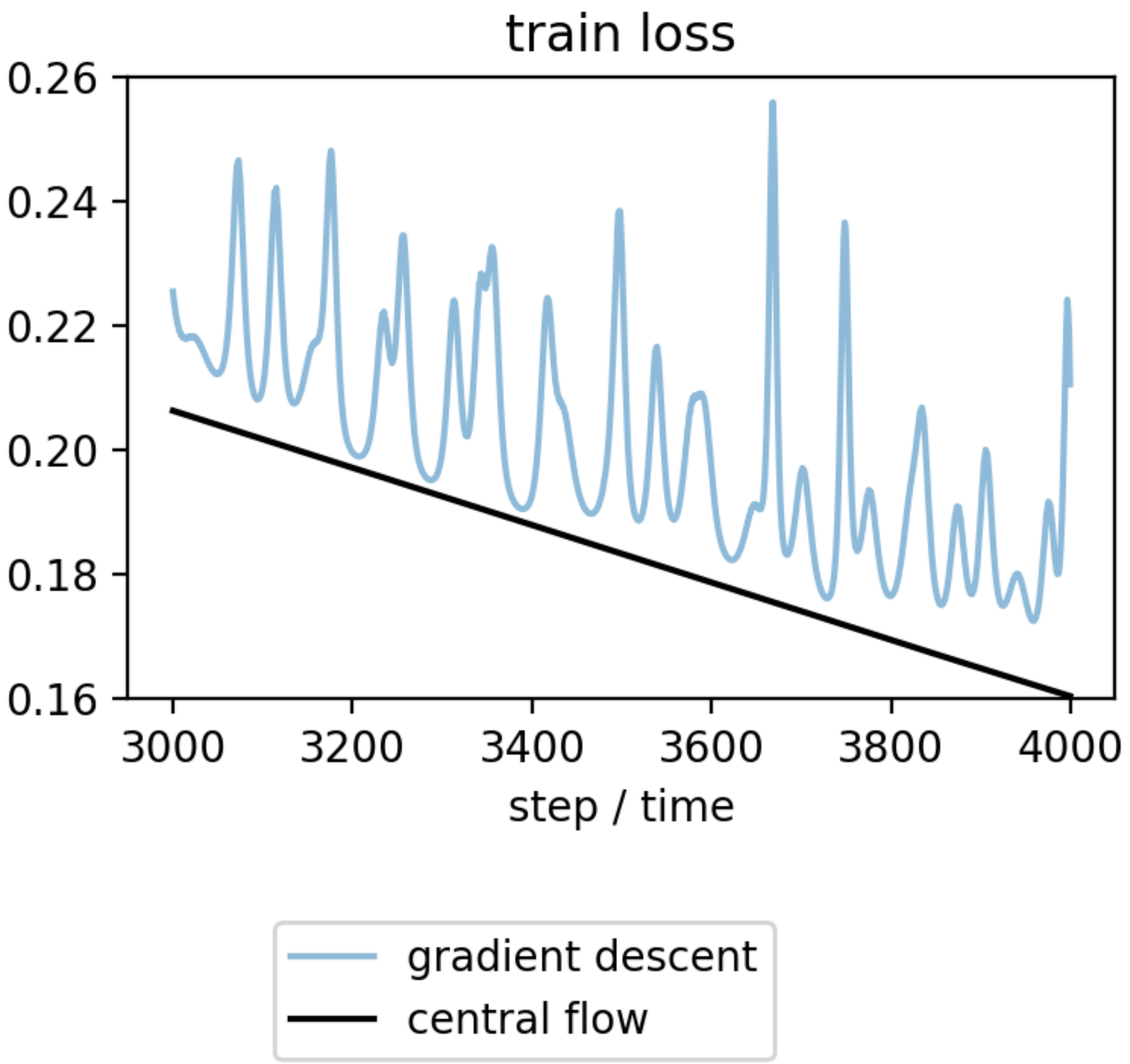
Application: reasoning about loss curves



- Loss is **higher** for GD than for central flow.
- Intuition: GD bounces between “valley walls”; central flow runs along “valley floor”



Application: reasoning about loss curves



- The central flow models *both* the mean trajectory *and* the covariance of oscillations:

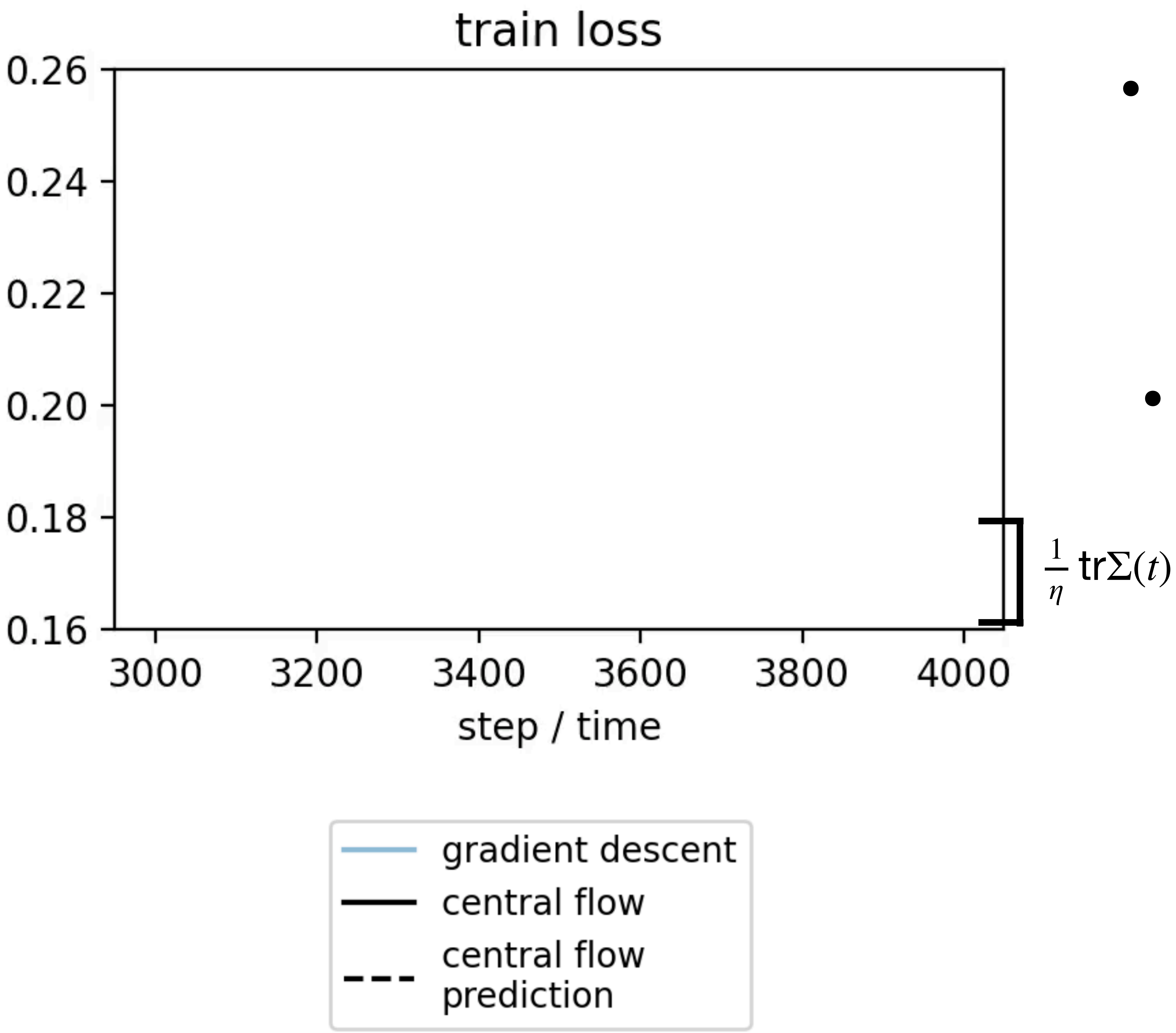
$$w_t \approx w(t) + \delta_t \quad \text{where} \quad \mathbb{E}[\delta_t] = 0, \;\; \mathbb{E}[\delta_t \delta_t^T] = \Sigma(t)$$

- Thus, it can predict the *time-averaged* train loss of gradient descent:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \frac{1}{\eta} \operatorname{tr} \Sigma(t)$$

time-averaged GD loss | loss along central flow | contribution from oscillations

Application: reasoning about loss curves



- The central flow models *both* the mean trajectory *and* the covariance of oscillations:

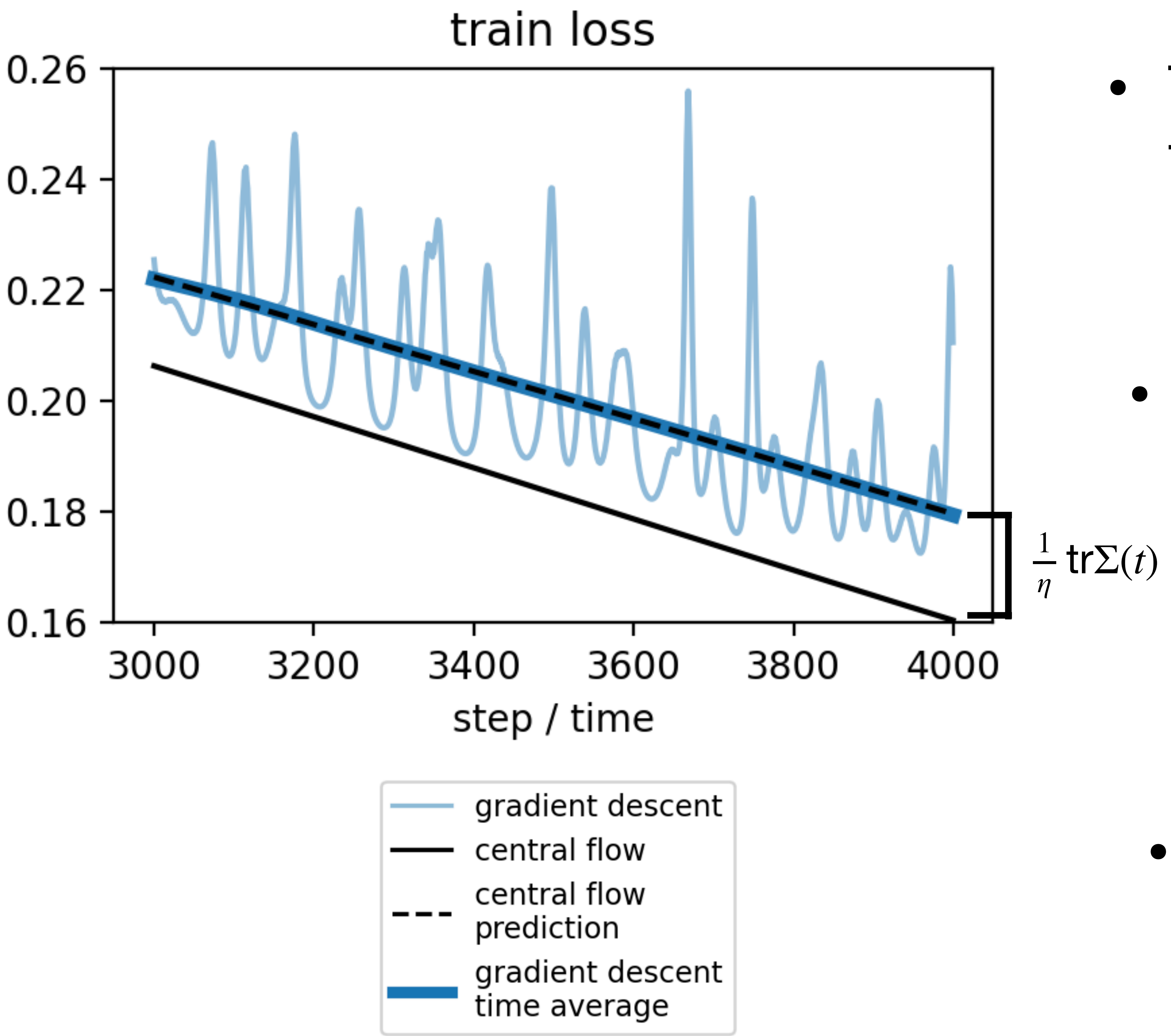
$$w_t \approx w(t) + \delta_t \quad \text{where} \quad \mathbb{E}[\delta_t] = 0, \;\; \mathbb{E}[\delta_t \delta_t^T] = \Sigma(t)$$

- Thus, it can predict the *time-averaged* train loss of gradient descent:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \frac{1}{\eta}\,\mathrm{tr}\,\Sigma(t)$$

time-averaged GD loss — loss along central flow — contribution from oscillations

Application: reasoning about loss curves



- The central flow models *both* the mean trajectory *and* the covariance of oscillations:

$$w_t \approx w(t) + \delta_t \quad \text{where} \quad \mathbb{E}[\delta_t] = 0, \;\; \mathbb{E}[\delta_t \delta_t^T] = \Sigma(t)$$

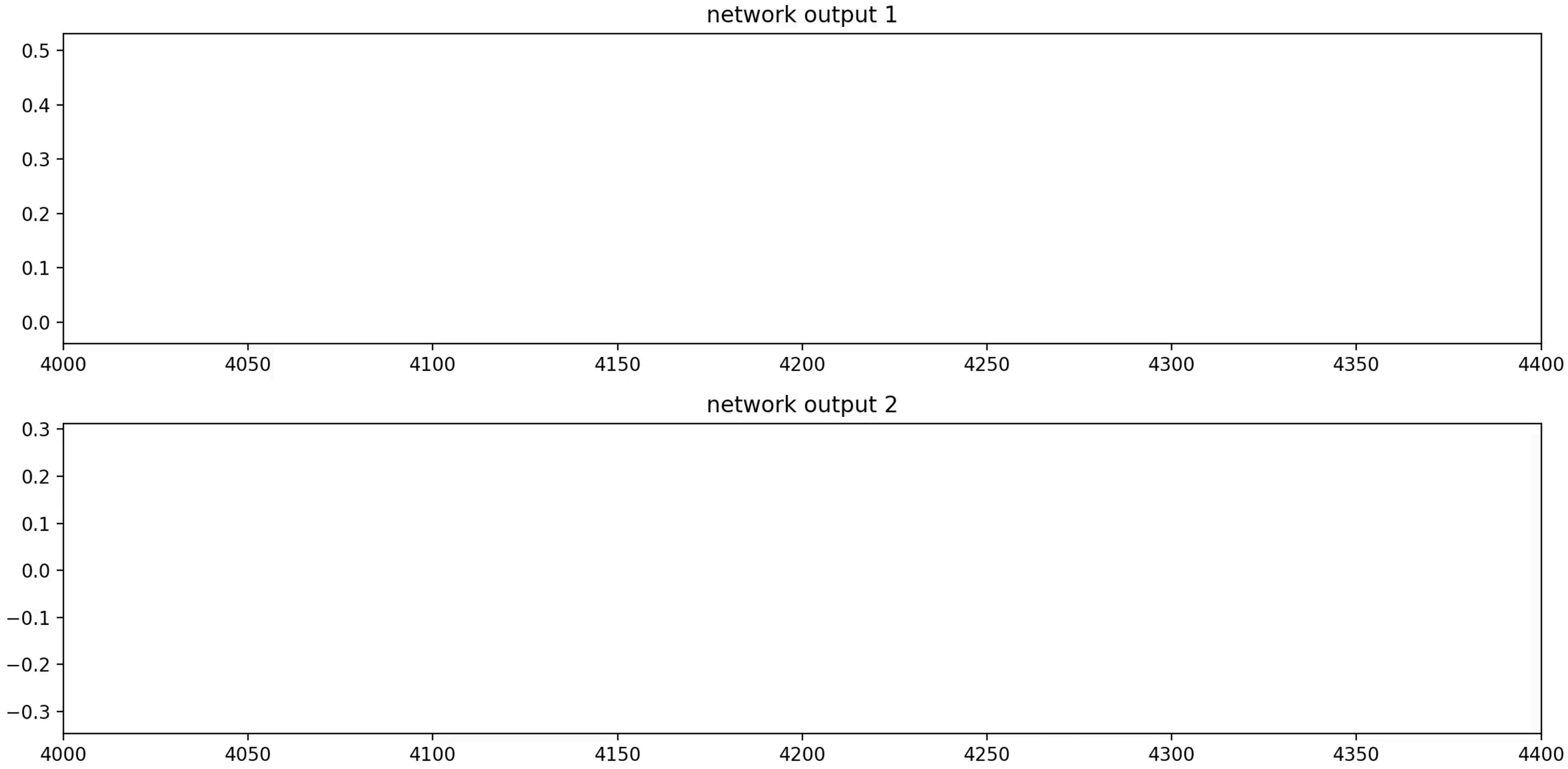
- Thus, it can predict the *time-averaged* train loss of gradient descent:

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \frac{1}{\eta}\,\mathrm{tr}\,\Sigma(t)$$

time-averaged GD loss — loss along central flow — contribution from oscillations

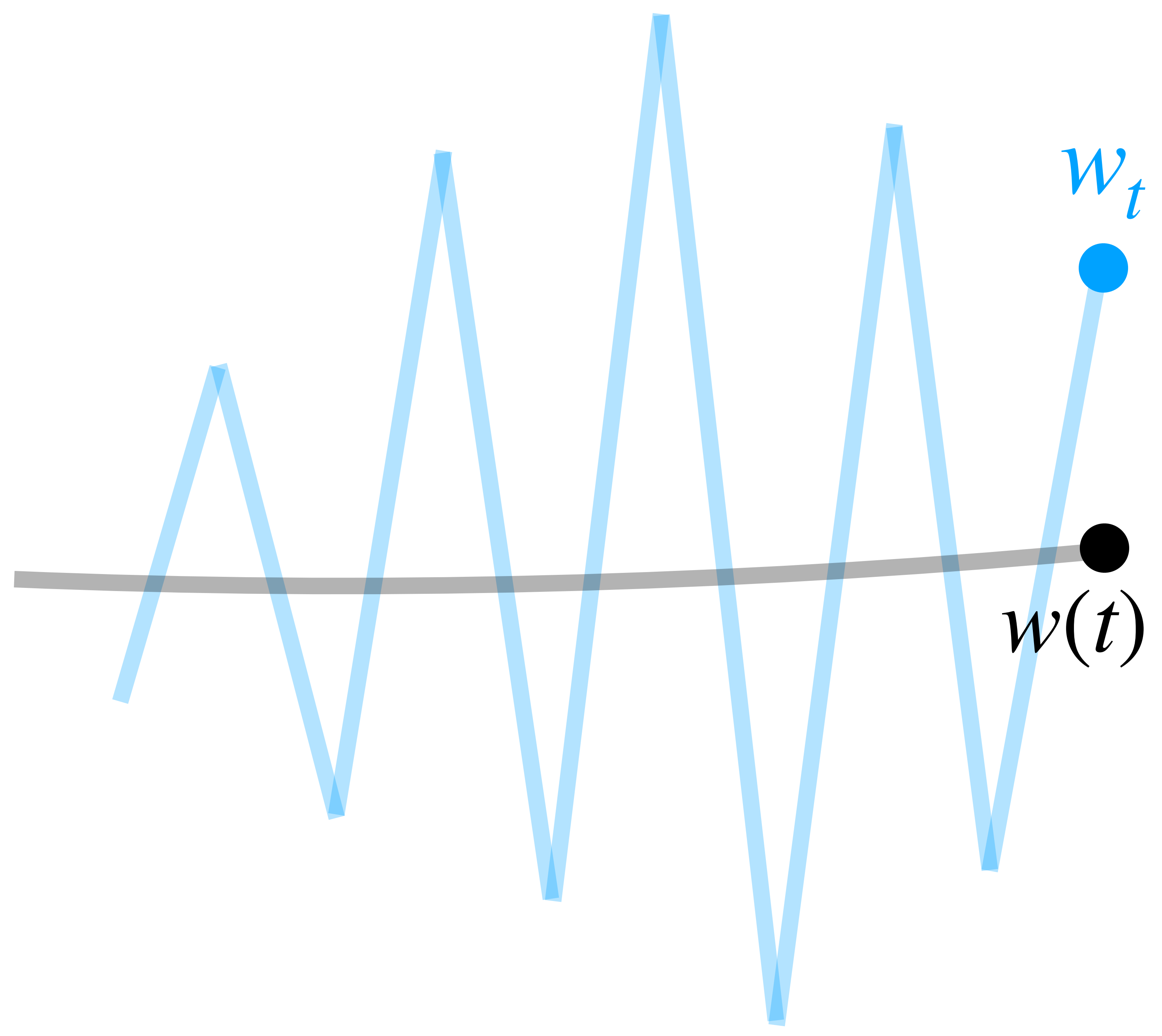
- Both $L(w(t))$ and $\mathbb{E}[L(w_t)]$ are meaningful quantities to DL practitioners

Central flow is the “true” training process



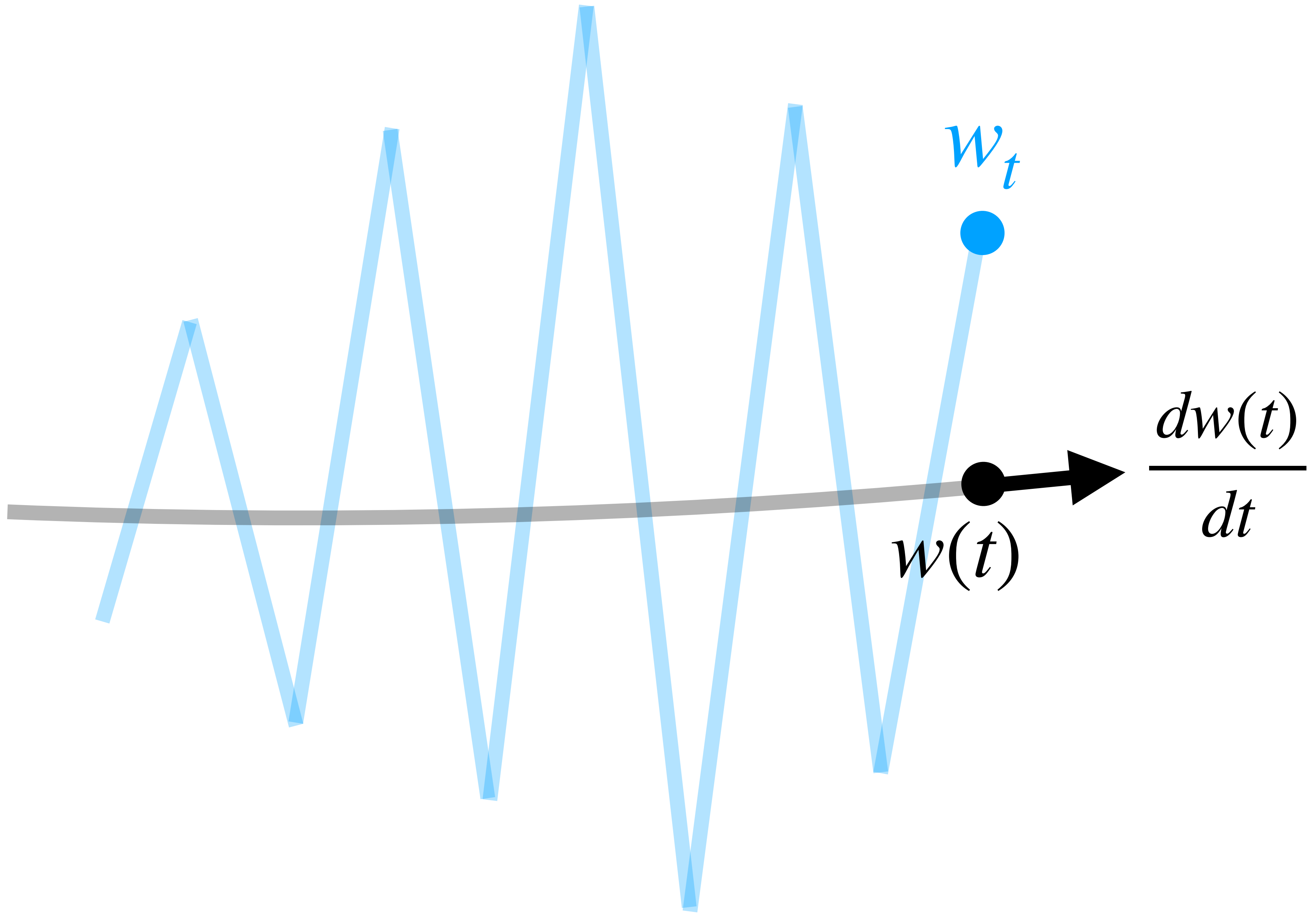
A smooth curve is a simple object

- As a smooth curve, the central flow is a simple object.



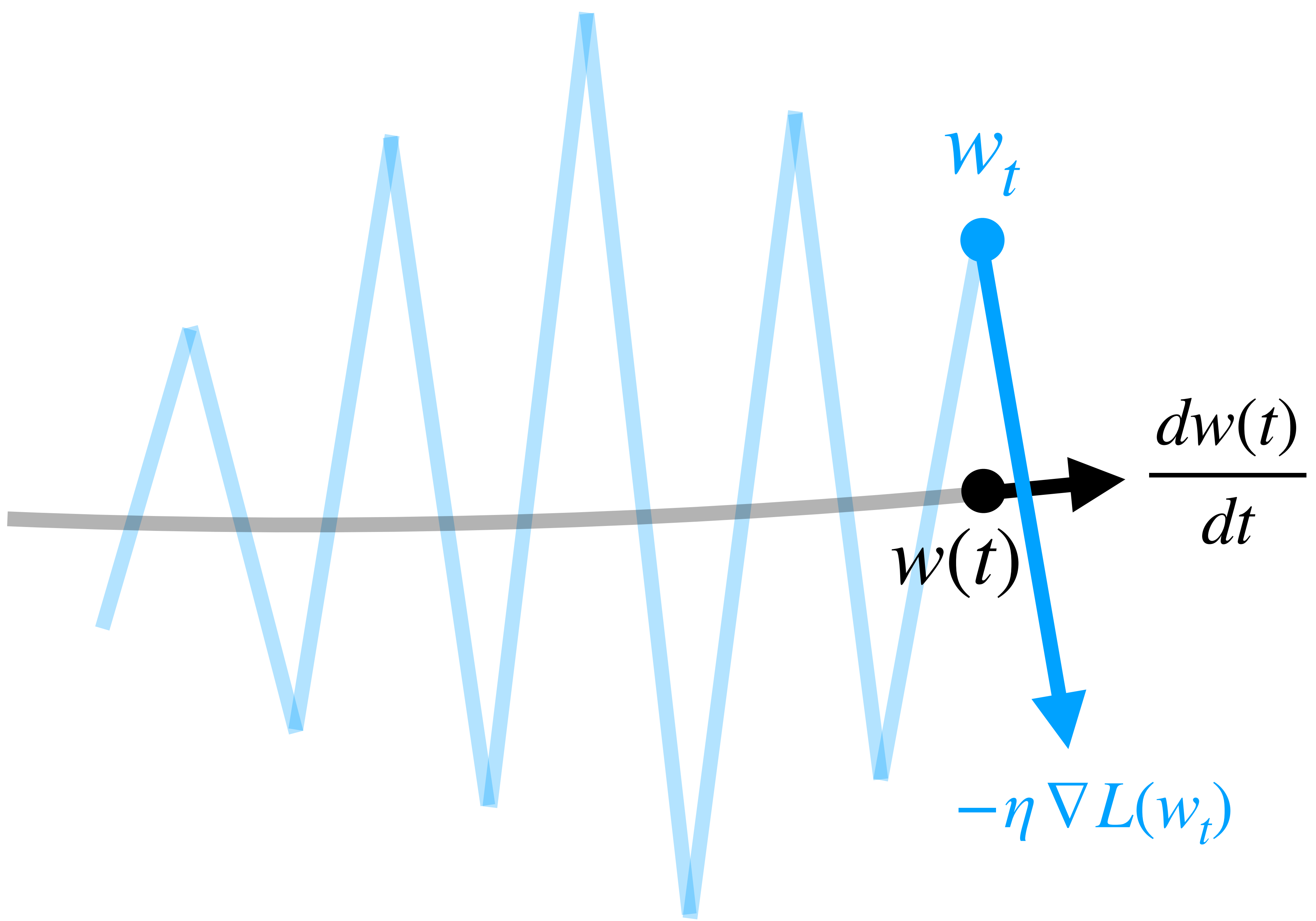
A smooth curve is a simple object

- As a smooth curve, the central flow is a simple object.
- The central flow update direction $\frac{dw}{dt}$ reflects the near-term direction of motion.

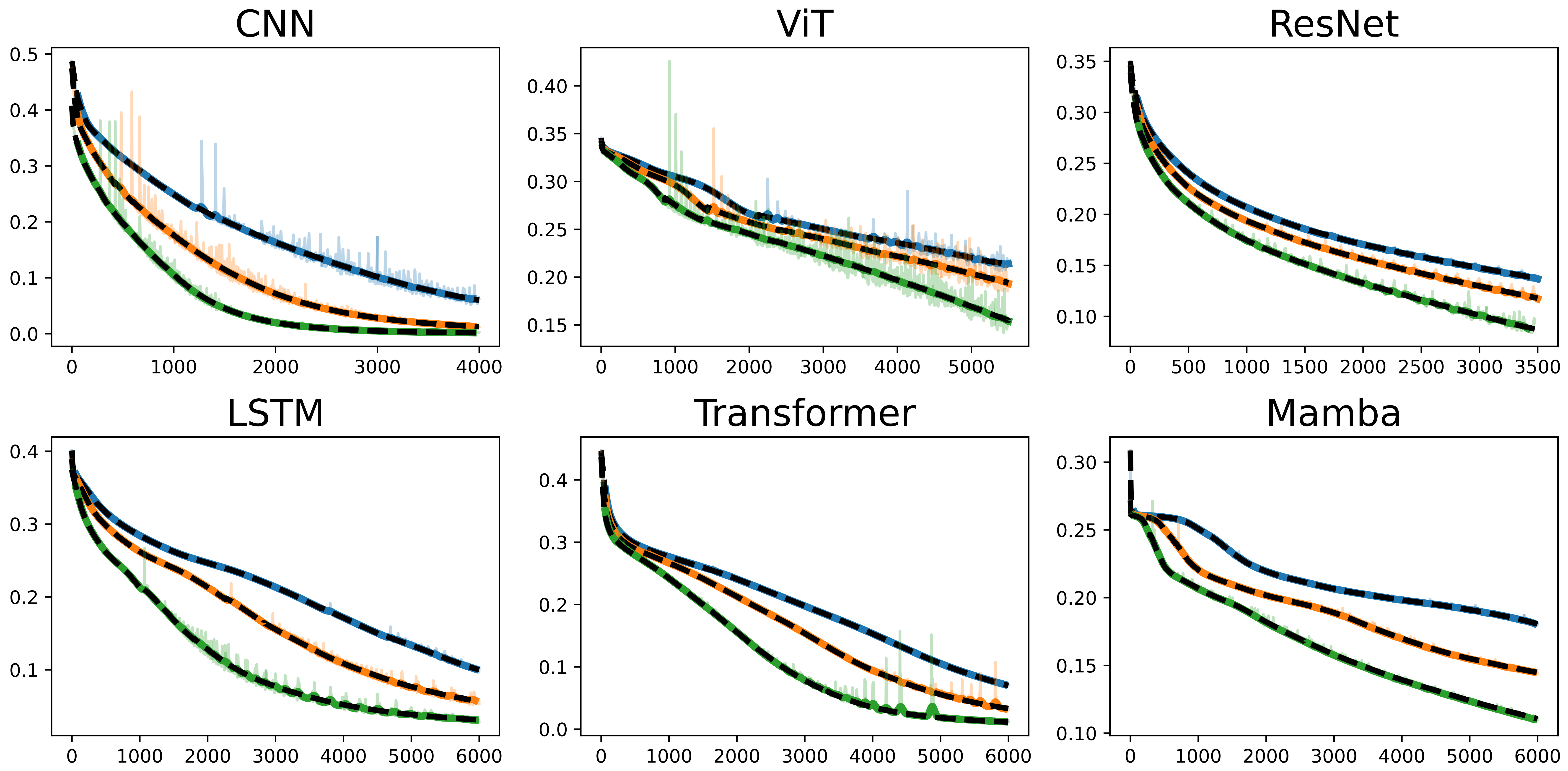


A smooth curve is a simple object

- As a smooth curve, the central flow is a simple object.
- The central flow update direction $\frac{dw}{dt}$ reflects the near-term direction of motion.
- By contrast, the GD update $-\eta \nabla L(w_t)$ is dominated by oscillations.

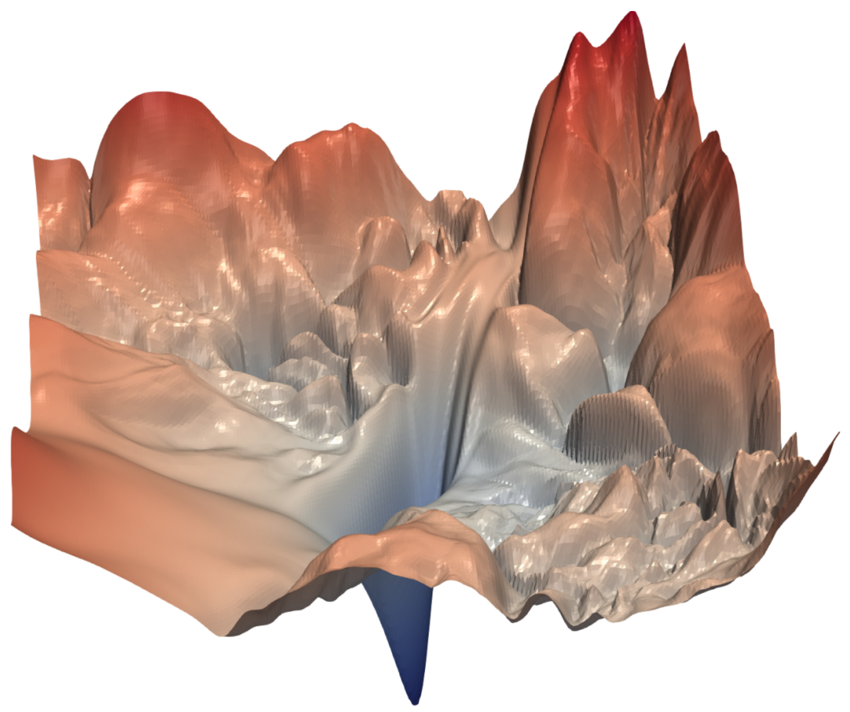


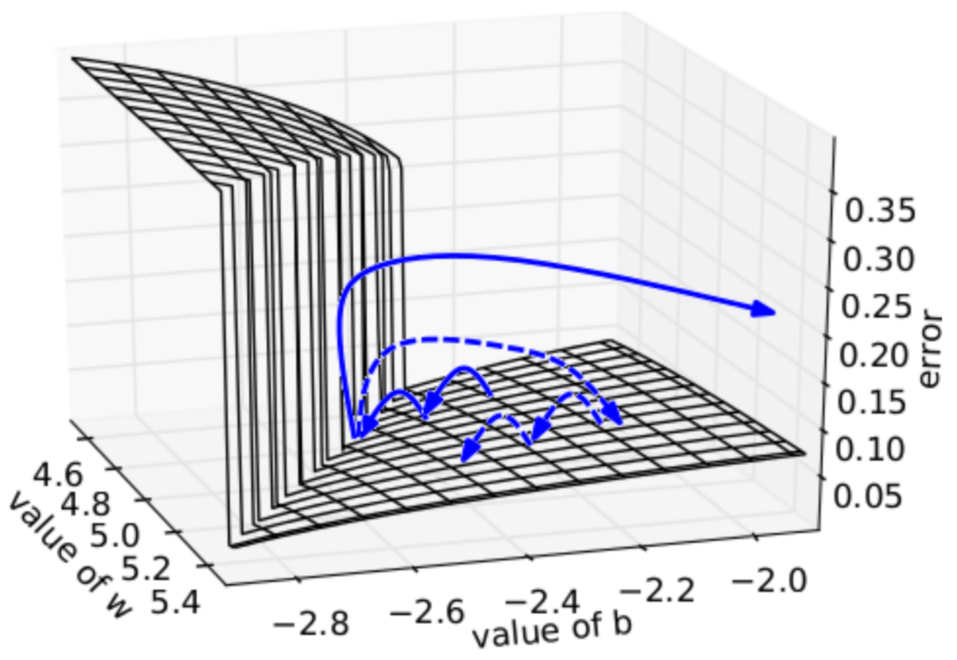
Our analysis applies to generic neural nets



Review

- Existing optimization theory does not apply in deep learning
  - Doesn't capture *cause and effect* for *deterministic gradient descent*
- But a different theory is possible
  - Deep learning objectives aren't that scary
  - Our analysis, while not rigorous, delivers accurate numerical predictions
  - Deep learning may call for a different approach than classical optimization





What is the *goal* of optimization theory?

- Classically, a common goal is to characterize global rates of convergence.
  - But this might never be possible in deep learning
- Another goal is to characterize the local rate of convergence once near a minimum
  - But deep learning optimization doesn’t occur near a minimum
- Our goal: characterize the *local dynamics throughout training*
  - These dynamics are (1) interesting, (2) important, and (3) generic.

What is the *purpose* of an optimization paper?

- ML reviewers' favorite kind of paper: theoretical analysis + new SOTA algorithm
- But we are likely still in the theory-building stage
- *Basic* research now will enable SOTA algorithm design in the future

What *methods* are acceptable?

- Optimization historically operates at a 100% level of mathematical rigor
- This standard may not be appropriate for deep learning
- People make assumptions that aren't true, so that they can leverage known proof techniques, rather than investigating what really happens
- The field should be comfortable with works at varying levels of rigor
- The right mathematical tools will develop gradually to fit the needs of the field

A good field to work on

- Deep learning is one of the defining technologies of this century
- Optimization lies at the heart of deep learning
- There is room for an entire field on the theory of optimization in deep learning
- Applied mathematicians can help turn deep learning from alchemy to science

Thanks to my collaborator Alex

Alex Damian

Cohen*, Damian*, Talwalkar, Kolter, Lee. *Understanding Optimization in Deep Learning with Central Flows.* ICLR '25.

OpenReview:

ArXiv: there's a draft on arXiv, but we're still putting the finishing touches on the final version

Email me for code: jcohen@flatironinstitute.org

PS: we also analyze Adam with $\beta_1 = 0$ (i.e. RMSProp)

- This algorithm doesn't make much sense according to traditional understandings, but works well in practice
  - How can we beat Adam if we don't understand it it
- We show that understanding how Adam sets its dynamic preconditioner requires understanding its oscillatory EOS dynamics
- We also show that Adam's efficacy relies on its ability to implicitly steer itself towards lower-curvature regions in which it can take lager steps
- Part II of this talk: "How does Adam work?"
- Thanks for listening!