Extending the Noisy Quadratic Models using Linear Matrix Inequalities

Xuchan Bao *jennybao@cs.toronto.edu*
Department of Computer Science
University of Toronto, Vector Institute

John W. Simpson-Porco *jwsimpson@ece.utoronto.ca*
Department of Electrical and Computer Engineering
University of Toronto

Roger Grosse *rgrosse@cs.toronto.edu*
Department of Computer Science
University of Toronto, Vector Institute

1 Introduction

When training neural networks, we often rely on empirical wisdom to make optimization related decisions, such as the optimizer algorithm and important hyperparameter values. There have been efforts in deriving practical insights from theoretical models. One successful example of such theoretical models is the noisy quadratic model (NQM) (Zhang et al., 2019). By running simple simulations of the NQM, one can derive important insights on how factors such as momentum, moving average and preconditioning affect the benefit of scaling up batch sizes.

However, the NQM has several limitations. One important limitation is that the noise modeling of NQM does not truthfully reflect the noise from minibatch training. The NQM applies an additive noise to the gradient, with a constant magnitude throughout the optimization process. However, the minibatch noise is multiplicative in nature, as it can be considered as a Bernoulli random variable mutliplied to the data points. Intuitively, the magnitude of minibatch noise should shrink as the optimized variable gets close to a local optimum. Second, the NQM assumes a quadratic loss function on the network parameters, which corresponds to a completely linear training dynamics. Although modern neural networks exhibits a surprisingly linear training dynamics (Jacot et al., 2018; Lee et al., 2019), especially early in the training process, nonlinearities are still necessary for feature learning and state-of-the-art performance (Chizat et al., 2019; Yang & Hu, 2021; Vyas et al., 2022; Fort et al., 2020). The NQM lacks a way for modeling the nonlinearities in training.

We adopt a flexible framework for modeling stochastic optimization of neural networks. We view the optimization as a dynamical system, and use tools from robust control theory to express the convergence of the dynamical system as the feasibility problem of a linear matrix inequality (LMI). If the LMI is feasible, then we have certified the convergence of the optimization problem. This methodology is first proposed by Lessard et al. (2016). In this work, make important adaptations in order to analyzing important elements in stochastic optimization of neural networks: 1) we derive a truthful model for minibatch noise, and modify the LMI to work with the average-case performance with respect for minibatch noise; 2) we model the nonlinearity of neural network training as an uncertainty of the network Jacobian throughout the optimization process. The amount of uncertainty is adjustable, enabling sensitivity analysis on nonlinarity in training.

A notable advantage of our framework is its flexibility. It works for all types of first-order optimizers, and can certify the worst-case asymptotic convergence rate and steady-state risk. The NQM analysis relies on the analytic solution of the dynamics of the classic (heavy-ball) momentum optimizer. This makes the NQM incompatible with other optimizers such as the Nesterov accelerated gradient (NAG) (Nesterov, 1983). In contrast, our framework seamlessly accommodates all optimizers with a linear state-space representation (i.e. all first-order optimizers). In fact, our framework predicts important differences between heavy-ball and NAG (Section 4.1), which are insights that match practical observations.

A limitation of our framework is that it only deals with asymptotic performance. In practical neural network training, we often apply early-stopping. Usually, at early stages of training, we observe rapid decrease in training loss. Asymptotic performance might be overly conservative in many scenarios. Future work should focus on finite-step performance, potentially following the line of work on Performance Estimation Problem (PEP) (Drori & Teboulle, 2014; Taylor et al., 2017; 2018; Taylor & Bach, 2019).

2 Background

2.1 Problem setup

Consider the ML model that represents the function $f : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^d$:

$$y = f_\theta(x).$$

In this paper, we focus on single-output models ($\mathcal{Y} \subset \mathbb{R}$) and assume the mean squared-error loss $\ell(x, t; \theta) = (f_\theta(x) - t)^2$. Given a training set $(\boldsymbol{x}, \boldsymbol{t}) := \{(x^{(i)}, t^{(i)})\}_{i=1}^N$, we have,

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta) = \frac{1}{2N} \sum_{i=1}^{N} \ell(x^{(i)}, t^{(i)}; \theta).$$

We update the model parameters θ using gradient-based optimization. Let θ_k be the parameter at timestep k. We denote the gradient at θ_k as:

$$g_k := \nabla_\theta \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) \tag{1}$$

2.2 First-order optimizers as state-space models

First order optimizers can be expressed as state-space models. Denote $\xi \in \mathbb{R}^n$ as the state, which have the state-space representation:

$$\begin{aligned} \xi_{k+1} &= \bar{A}\xi_k + \bar{B}g_k \\ \theta_k &= \bar{C}\xi_k + \bar{D}g_k \end{aligned}$$

For first-order optimizers, we have $\bar{D} = 0$.

Example 1: heavy-ball momentum For heavy-ball momentum, we have:

$$\bar{A} = \begin{bmatrix} (1+\beta)I & -\beta I \\ I & 0 \end{bmatrix},\ \bar{B} = \begin{bmatrix} -\eta I \\ 0 \end{bmatrix},\ \bar{C} = \begin{bmatrix} I & 0 \end{bmatrix},\ \bar{D} = 0$$

Example 2: Nesterov accelerated gradient For Nesterov accelerated gradient, we have:

$$\bar{A} = \begin{bmatrix} (1+\beta)I & -\beta I \\ I & 0 \end{bmatrix},\ \bar{B} = \begin{bmatrix} -\eta I \\ 0 \end{bmatrix},\ \bar{C} = \begin{bmatrix} (1+\beta)I & -\beta I \end{bmatrix},\ \bar{D} = 0$$

The heavy-ball momentum and NAG have very different dynamics and intuitive interpretations, despite their updates being similar. For a detailed discussion on these two momentum methods, please refer to Section 5.

2.2.1 Output-space & linearization

We would like to add gradient noise to the system. In particular, we would like the gradient noise to have covariance $C \approx \nabla^2_{\theta\theta}\mathcal{L}$.

$$\tilde{g}_k = g_k + \omega_k,\ \omega_k \sim \mathcal{N}(0, C) \tag{2}$$

In order to model the gradient noise with covariance C, we write the optimizer in the output-space. The optimizer now takes in the output-space gradient $\nabla_{\boldsymbol{y}}\mathcal{L}$ and outputs the model output $y = f_\theta(x)$.

Output-space gradient $\nabla_{\boldsymbol{y}}\mathcal{L}$ Using the chain rule, we have:

$$\nabla_\theta \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta) = J_{y\theta,k}^\top \nabla_{\boldsymbol{y}} \mathcal{L}, \quad J_{y\theta,k} := \frac{\partial \boldsymbol{y}_k}{\partial \theta_k} \in \mathbb{R}^{N \times d}$$

To simply notations, we refer to $J_{y\theta,k}$ as J_k from now on. Substitute into the state-space equation:

$$\xi_{k+1} = \bar{A}\xi_k + \bar{B} J_k^\top \nabla_{\boldsymbol{y}} \mathcal{L}$$

Linearized model with output $\boldsymbol{y}$ To completely put the optimizer in the output-space, we also need to output $\boldsymbol{y}$ instead of θ. To do so, we linearize the model around θ^*:

$$f_\theta(\boldsymbol{x}) \approx f_{\theta^*}(\boldsymbol{x}) + \frac{\partial f}{\partial \theta}\Big|_{\theta=\theta^*} (\theta - \theta^*)$$

Since θ^* is a constant, we can "center" the linearization by replacing all quantities with the difference from the optimal, i.e. by defining $\bar{\theta} = \theta - \theta^*$, $\bar{y} = y - y^*$ etc. Again, to avoid notational clutter, we drop the terms involving θ^*. The linearized model is:

$$y_k \approx J(\theta^*)\theta_k$$

Putting them together, the optimizer in output-space has the following state-space representation:

$$\begin{aligned} \xi_{k+1} &= \bar{A}\xi_k + \bar{B} J_k^\top \nabla_{\boldsymbol{y}} \mathcal{L}_k \\ y_k &= J(\theta^*)\bar{C}\xi_k \end{aligned}$$

2.3 Certifying convergence by solving the SDP problem

We use tools from robust control to certify the convergence of an optimization problem. On a high level, the convergence certification is based on Lyapunov stability theory. For an autonomous system (i.e. without external input), the system is stable if we can find a Lyapunov function $V(\xi) = \xi^\top P \xi$ ($P \succ 0$) such that it is non-increasing along the trajectory: $V(\xi_{k+1}) \leq V(\xi_k)$.

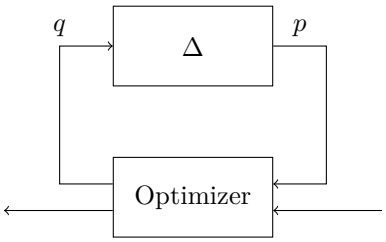


Figure 1: Diagram for the interconnected system with the optimizer and the Jacobian uncertainty.

Now, consider the interconnected system in Figure 1. We use *dissipativity*, a version of Lyapunov stability theory that is generalized to systems with inputs and outputs, to show that the interconnected system is stable. The dissipativity of a system with input p and output q is defined with respect to a certain supply rate $s(q, p)$. Intuitively, the supply rate represents the rate at which energy is injected into the system, and a system is dissipative if energy of the state (measured by the storage function $V(\xi)$) increases at a slower rate than $s(q, p)$. In particular, we let the storage function to be quadratic, $V(\xi) = \xi^\top P \xi$, in order to be compatible with numerical optimization. Mathematically, the interconnected system is dissipative with supply rate $s(q, p)$ if we can

$$\text{find } P \succ 0 \text{ such that } V(\xi) = \xi^\top P \xi, \text{ and } V(\xi_{k+1}) - V(\xi_k) \leq s(q_k, p_k). \tag{3}$$

Dissipativity has a directly link to the notions of stability we use in this paper. Eventually, equation 3 will be expressed in terms of a linear matrix inequality (LMI), which we will solve numerically using convex optimization tools.

In practice, the supply rate often encapsulates parts of the system that are difficult to model: they might be nonlinear, with uncertainty, or unknown. Therefore, instead of modeling those parts directly, it is common to specify the supply rate in terms of bounds. In particular, we focus on bounds in the quadratic form $\begin{bmatrix} q \\ p \end{bmatrix}^\top \Pi \begin{bmatrix} q \\ p \end{bmatrix}$, because they fit well with the LMI. The key part of applying this framework is to find appropriate quadratic constraints for the optimization problem we are concerned with.

2.4 NQM

We analyze the noisy quadratic model (NQM) using this framework.

The state-space representation of the optimizer in the output-space is:

$$\xi_{k+1} = \bar{A}\xi_k + \bar{B}J^\top(\nabla_{\boldsymbol{y}}\mathcal{L}_k + \epsilon_k), \quad \epsilon_k \sim \mathcal{N}(0, I) \tag{4}$$

$$y_k = J(\theta^*)\bar{C}\xi_k \tag{5}$$

Since we have a quadratic model, $\nabla_{\boldsymbol{y}}\mathcal{L}_k = y_k$, and $J_k = J(\theta^*) := J$ for all k. Substitute into the state-space representation, we have:

$$\xi_{k+1} = \underbrace{(\bar{A} + \bar{B}J^\top J\bar{C})}_{A}\xi_k + \underbrace{\bar{B}J^\top}_{B_w}\epsilon_k \tag{6}$$

and

$$y_k = \underbrace{J\bar{C}}_{C}\xi_k \tag{7}$$

Theorem 1 (NQM asymptotic convergence rate)**.** *Consider the optimization on a noisy quadratic model characterized by (A, B_w) (as defined in equation 6). If for some $0 < \rho < 1$, there exists a $P \succ 0$ such that:*

$$A^\top PA - \rho^2 P \prec 0 \tag{8}$$

Then the loss asymptotically converges with rate ρ^2.

Intuitively, we aim to find a Lyapunov function (analogous to the potential energy of the system) $V(\xi_k) = \xi_k^\top P\xi_k$ such that it decreases at rate at least ρ^2 following the dynamics defined in equation 6. Note that the convergence rate is not affected by the label noise.

Theorem 2 (NQM steady-state risk)**.** *For the optimization on a noisy quadratic model (A, B_w) (as defined in equation 6), for some $0 < \gamma < 1$, if there exists a $P \succ 0$ such that:*

$$APA^\top - P + B_w B_w^\top \preceq 0 \tag{9}$$

and

$$Tr(CPC^\top) \leq \gamma^2 \tag{10}$$

then the steady-state risk of the optimizer on the noisy-quadratic model is upper-bounded by γ^2.

Proofs of Theorem 1 and Theorem 2 can be found in Appendix A.1.

3 Framework for certifying convergence under minibatch and nonlinearity

3.1 Dimensionality reduction: weight-space projection and Jacobian binning

Before introducing the model for minibatch noise, it is worth revisiting the dimensions of the quantities in Equations (6) and (7). The optimizer-related matrices $\bar{A}, \bar{B}, \bar{C}$ are in the weight space, and their dimensions

are multiples of d (e.g. for gradient descent, $\bar{A}, \bar{B}, \bar{C} \in \mathbb{R}^{d\times d}$, and for heavy-ball and NAG, $\bar{A} \in \mathbb{R}^{2d\times 2d}$, $\bar{B} \in \mathbb{R}^{2d\times d}$, $\bar{C} \in \mathbb{R}^{d\times 2d}$). The Jacobian J has dimension $N \times d$, where N is the number of data points optimized on.

For any reasonable-sized optimization problem, both N and d are large enough that naively solving the LMIs using Equations (6) and (7) would be infeasible. We propose two dimensionality reduction techniques: weight-space projection and Jacobian binning.

Weight-space projection We do not want the LMI size to grow with the size of training data, so we would like to remove the data dimension N from the LMI. Conveniently, the data dimension is summed over. In the state equations, this is reflected by the fact that any quantity with dimension N is always multiplied with $J^\top$. Assuming $N \geq d$, we have:

$$J = USV^\top, \quad U \in \mathbb{R}^{N\times d},\ S, V \in \mathbb{R}^{d\times d}$$

Let $v \in \mathbb{R}^N$ be a vector quantity with dimension N. We can project v onto the row space of U:

$$v' = U^\top v$$

Also define $J' = U^\top J$, we have:

$$J^\top v = {J'}^\top v'.$$

This way, all quantities are equivalently represented in the weight space, and the LMI size is independent of N. For notation simplicity, we will drop the primes in the rest of the paper. Unless otherwise specified (e.g. the derivation in Section 3.2), all quantities are in the weight space. Also, since the optimizers we consider are invariant to the choice of basis, we can assume without loss of generality that V is identity, and that J' is diagonal.[1]

Jacobian binning To further reduce the size of the LMI, we group the Jacobian singular values into bins. Let the singular values of J be $s_1, \ldots, s_d$. It is observed that the Hessian spectrum of neural networks often follow the power law (Zhang et al., 2019; Martin & Mahoney, 2021). Therefore, we partition the singular values into m bins ($m \ll d$), with values $\bar{s}_1, \ldots, \bar{s}_m$ arranged in logspace. We group s_i into the j-th bin, if $\bar{s}_j \leq s_i < \bar{s}_{j-1}$ ($\bar{s}_j \leq s_i$ if $j = 1$), and let the size of the i-th bin be n_i. Since we have assumed without loss of generality that J' is diagonal, we denote the reduced Jacobian as $J_{\text{bin}} = \text{diag}(\bar{s}_1, \ldots, \bar{s}_m)$.

For certifying the convergence rate, we can simply replace J with J_{bin}, as the convergence rate is asymptotic across all dimensions. For steady-state risk, in addition to replacing J with J_{bin}, we also need to scale the output y with the bin sizes, i.e. replace C with $C_{\text{bin}} = \text{diag}(\sqrt{n_1}, \ldots, \sqrt{n_m})C$. Again, to avoid notation clutter, we will drop the bin subscript in the rest of the paper, as we always apply binning to the Jacobian.

3.2 A model for minibatch noise

The noisy quadratic model uses an additive term to approximate the minibatch noise (Equation (2)). However, this model is not accurate, because the minibatch noise is multiplicative in nature. As θ converges to the optimal soluiton θ^*, the minibatch noise should also converge to zero.

A better approximation is to model the minibatch noise as i.i.d. Bernoulli random variables multiplied to the training data points at every optimization step. Let $\mathcal{B}$ be the batch size, the minibatch gradient at step can be approximated as:

$$\tilde{g}_k^{\mathcal{B}} = \nabla_\theta \frac{1}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} \ell(x^{(j)}, t^{(j)}; \theta_k), \quad b^{(j)} \overset{\text{i.i.d.}}{\sim} \text{Ber}(\frac{\mathcal{B}}{N}). \tag{11}$$

[1] It doesn't mean that neural network training is fully decoupled in the weight space. When introducing minibatch noise in Section 3.2, we will see that the noise introduces coupling between weights.

Let $y(\theta) = f(x;\theta)$ be the output of the model. Expressing the gradient in the output-space, we have:

$$\begin{aligned}
\tilde{g}_k^{\mathcal{B}} &= \frac{1}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} \nabla_y \ell(x^{(j)}, t^{(j)}; \theta_k) \nabla_\theta f(x^{(j)}; \theta_k) \\
&= \frac{N}{\mathcal{B}} J_{y\theta}^\top \begin{bmatrix} b^{(1)} & & \\ & \ddots & \\ & & b^{(N)} \end{bmatrix} \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) \\
&= J_{y\theta}^\top \underbrace{\frac{N}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} \mathbb{1}_{jj} \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)}_{\text{Output-space minibatch grad } \nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)}
\end{aligned}$$

We define $\mathbb{1}_{jj} \in \mathbb{R}^{N \times N}$ such that all elements are 0, except that the (j,j)-th element is 1.

As in Section 3.1, decompose the Jacobian as $J_{y\theta} = USV^\top$, and project the minibatch output-space gradient onto the weight space:

$$U^\top \nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) = \frac{N}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} U^\top \mathbb{1}_{jj} \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)$$

If we assume that $\nabla_y \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) \in \text{col}(U)$ (which is reasonable, as we only care about the output-space gradient that will have an effect on updating θ), then we have:

$$U^\top \nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) = \frac{N}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} U^\top \mathbb{1}_{jj} U U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) = \frac{N}{\mathcal{B}} \sum_{j=1}^{N} b^{(j)} u_j u_j^\top U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)$$

where u_j is the j-th column of U. On the right hand side, we have our usual output-space gradient projected onto the weight space $U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)$. The minibatch gradient can therefore integrated into our LMI.

For convenience, we transform the Bernoulli random variable such that it's centered at 0. Define $p^{(j)} = \frac{\mathcal{B}}{N} - b^{(j)}$, we have:

$$\begin{aligned}
U^\top \nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) &= \frac{N}{\mathcal{B}} \sum_{j=1}^{N} (\frac{\mathcal{B}}{N} - p^{(j)}) u_j u_j^\top U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k), \qquad p^{(j)} = \begin{cases} \frac{\mathcal{B}}{N} - 1, & \text{w. prob } \frac{\mathcal{B}}{N} \\ \frac{\mathcal{B}}{N}, & \text{w. prob } 1 - \frac{\mathcal{B}}{N} \end{cases} \\
&= U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) - \frac{N}{\mathcal{B}} \sum_{j=1}^{N} \left(p^{(j)} u_j u_j^\top\right) U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)
\end{aligned}$$

In state equation Equation (4), replace the output-space gradient $\nabla_{\boldsymbol{y}} \mathcal{L}_k$ with the minibatch version $\nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)$ and noting $J_{y\theta} = U^\top J$, we have:

$$\begin{aligned}
\xi_{k+1} &= \bar{A}\xi_k + \bar{B} J_{y\theta}^\top (\nabla_{\boldsymbol{y}}^{\mathcal{B}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) + \epsilon_k) \\
&= \bar{A}\xi_k + \bar{B} J^\top \left(U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) - \frac{N}{\mathcal{B}} \sum_{j=1}^{N} \left(p^{(j)} u_j u_j^\top\right) U^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) \right) + \bar{B} J^\top U^\top \epsilon_k
\end{aligned}$$

As before, we adopt a slight abuse of notation, and drop the projection $U^\top$ for the output-space signals ($\nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k)$ and ϵ_k).

$$\xi_{k+1} = \bar{A}\xi_k + \bar{B} J^\top \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) - \sum_{j=1}^{N} \left(p^{(j)} \frac{N}{\mathcal{B}} \bar{B} J^\top u_j u_j^\top\right) \nabla_{\boldsymbol{y}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{t}; \theta_k) + \bar{B} J^\top \epsilon_k$$

Substituting in Equation (7), we have:

$$\xi_{k+1} = \bar{A}\xi_k + \bar{B}J^\top J\bar{C}\xi_k - \sum_{j=1}^{N} \big(\underbrace{\frac{N}{\mathcal{B}}\bar{B}J^\top u_j u_j^\top J\bar{C}}_{-A_j} \big)\xi_k \cdot p^{(j)} + \bar{B}J^\top \epsilon_k$$

Define $A_j = \frac{N}{\mathcal{B}}\bar{B}J^\top u_j u_j^\top J\bar{C}$, we have our state equation with minibatch noise.

State Equation for Minibatch Noise

$$\xi_{k+1} = A\xi_k + \sum_{j=1}^{N} A_j \xi_k \cdot p_k^{(j)} + B_w w_k \tag{12}$$

where $p_k^{(j)}$ are i.i.d. zero-centered Bernoulli random variables, and

$$A = \bar{A} + \bar{B}J^\top J\bar{C}, \tag{13a}$$

$$A_j = -\frac{N}{\mathcal{B}}\bar{B}J^\top u_j u_j^\top J\bar{C}, \text{ for } j = 1, \ldots, N \tag{13b}$$

$$B_w = \bar{B}J^\top \tag{13c}$$

We show the LMI conditions for certifying the asymptotic convergence rate and the steady-state risk of the NQM with minibatch noise.

Theorem 3 (Asymptotic convergence rate of NQM with minibatch noise)**.** *Consider the system in Equation* (12)*, where the noise signals w_k and $p_k^{(j)}$ satisfy:*

$$\mathbb{E}[w_k] = 0, \qquad \mathbb{E}[p_k^{(j)}] = 0,$$
$$\mathbb{E}[w_k w_\tau^\top] = 0 \ \forall k \neq \tau, \qquad p_k^{(j)} \text{ are i.i.d.},$$
$$\mathbb{E}[w_k w_k^\top] = \Sigma \succ 0, \qquad \mathbb{E}[p_k^{(j)} p_k^{(j)}] = \sigma_j^2.$$

If there exists $P \succ 0$ and $0 < \rho < 1$ such that:

$$A^\top P A - \rho^2 P + \sum_{j=1}^{N} \sigma_j^2 A_j^\top P A_j \preceq 0 \tag{14}$$

Then the expected loss $\frac{1}{2}\mathbb{E}[y_k^\top y_k]$ asymptotically converges with rate ρ.

The proof can be found in Appendix A.1. The proof techniques are similar to the proof of Theorem 1.

Theorem 4 (Steady-state error of NQM with minibatch noise)**.** *Assume the following system is well-posed:*

$$\xi_{k+1} = A\xi_k + \sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k$$
$$y_k = C\xi_k$$

where noise signals w_k and $p_k^{(j)}$ satisfy:

$$\mathbb{E}[w_k] = 0, \qquad \mathbb{E}[p_k^{(j)}] = 0,$$
$$\mathbb{E}[w_k w_\tau^\top] = 0 \ \forall k \neq \tau, \qquad p_k^{(j)} \text{ are i.i.d.},$$
$$\mathbb{E}[w_k w_k^\top] = \Sigma \succ 0, \qquad \mathbb{E}[p_k^{(j)} p_k^{(j)}] = \sigma_j^2.$$

If there exists $P \succ 0$ such that $Tr(PB_w \Sigma B_w^\top) \leq \gamma^2$, and

$$A^\top PA - P + \sum_{j=1}^{N} \sigma_j^2 A_j^\top PA_j + C^\top C \preceq 0 \tag{15}$$

Then we have:

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[y_k^\top y_k] \leq \gamma^2$$

The proof can be found in Appendix A.1.

3.3 Modeling network nonlinearity

We have so far dealt with linear models. However, the gradient g (Equation 1) not a linear function of θ. In order to model the nonlinear training dynamics, we need to model the nonlinearity in g_k.

In particular, if we use squared-error loss, $\nabla_{\boldsymbol{y}} \mathcal{L}$ is simply the identity matrix, and the nonlinearity in g_k is due to the (time-varying) Jacobian J_k.

Given the initial Jacobian J_0, we express J_k as J_0 with multiplicative perturbation:

$$J_k = (I + M_k) J_0 (I + N_k)^\top,$$

where M_k, N_k are "small". We express the perturbation in additive terms and drop the second-order term:

$$J_k = (I + M_k) J_0 (I + N_k)^\top \approx J_0 + M_k J_0 + J_0 N_k^\top$$

We certify convergence by express the optimizer and the uncertainty (perturbation on Jacobian) as linear matrix inequalities (LMI), then construct and solve an SDP problem. The interleaved system is shown in Figure 2.

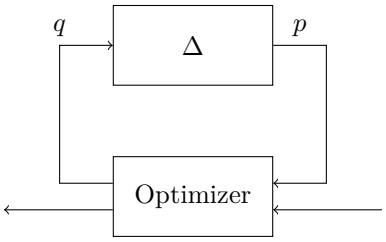


Figure 2: Diagram for the interleaved system with the optimizer and the Jacobian uncertainty.

Adding the components related to Jacobian uncertainty to the state equation, we have:

Combined State Equation

The combinedstate equation for the system with Jacobian uncertainty and minibatch noise is:

$$\xi_{k+1} = A\xi_k + \sum_{j=1}^{N} A_j \xi_k \cdot p_k^{(j)} + BJ_0^\top \underbrace{(M_k + M_k^\top) J_0 C \xi_k}_{p_1} + B \underbrace{M_k J^\top J \bar{C} \xi_k}_{p_2} + BJ^\top J \underbrace{M_k^\top \bar{C} \xi_k}_{p_3} + B_w w_k \tag{16}$$

where A, A_j, B_w are defined in Equation (13). The input and output quantities are:

$$\begin{aligned} q_1 &= JC\xi_k & p_1 &= \Delta_1 q_1 \quad (\Delta_1 = M_k + M_k^\top) \\ q_2 &= J^\top JC\xi_k & p_2 &= \Delta_2 q_2 \quad (\Delta_2 = M_k) \\ q_3 &= C\xi_k & p_3 &= \Delta_3 q_3 \quad (\Delta_3 = M_k^\top) \end{aligned}$$

We can rewrite the state equation in matrix form:

$$\begin{bmatrix} \xi_{k+1} \\ \hline q_{1,k} \\ q_{2,k} \\ q_{3,k} \end{bmatrix} = \left[\begin{array}{c|ccc|ccc|c} A & A_1 & \cdots & A_N & BJ^\top & B & BJ^\top J & B_w \\ \hline JC & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ J^\top JC & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ C & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \xi_k \\ \hline p_k^{(1)} \\ \vdots \\ p_k^{(N)} \\ \hline p_{1,k} \\ p_{2,k} \\ p_{3,k} \\ \hline w_k \end{bmatrix}. \tag{17}$$

And we define:

$$\begin{aligned} &\hat{A} = A, \quad \hat{B} = \begin{bmatrix} BJ^\top & B & BJ^\top J \end{bmatrix}, \\ &\hat{C}_1 = JC, \quad \hat{C}_2 = J^\top JC, \quad \hat{C}_3 = C, \\ &\hat{D}_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \hat{D}_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \hat{D}_3 = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

We certify the stability of the system under uncertainty in $\Delta_1, \Delta_2, \Delta_3$ by applying ℓ_2 constraints for (p_1, q_1), (p_2, q_2) and (p_3, q_3) pairs.

Theorem 5 (Certifying convergence rate for nonlinear model)**.** *Consider the system in Equation* (17) *where the noise signals w_k and $p_k^{(j)}$ satisfy:*

$$\begin{aligned} &\mathbb{E}[w_k] = 0, && \mathbb{E}[p_k^{(j)}] = 0, \\ &\mathbb{E}[w_k w_\tau^\top] = 0 \ \forall k \neq \tau, && p_k^{(j)} \text{ are i.i.d.}, \\ &\mathbb{E}[w_k w_k^\top] = \Sigma \succ 0, && \mathbb{E}[p_k^{(j)} p_k^{(j)}] = \sigma_j^2. \end{aligned}$$

Given $\delta_1, \delta_2, \delta_3 \geq 0$, if there exists a positive-definite matrix P, $0 < \rho < 1$, and $\lambda_1, \lambda_2, \lambda_3 \geq 0$ such that:

$$\begin{bmatrix} I \\ \hat{A} & \hat{B} \end{bmatrix}^\top \begin{bmatrix} -\rho^2 P & \\ & P \end{bmatrix} \begin{bmatrix} I \\ \hat{A} & \hat{B} \end{bmatrix} + \sum_{j=1}^{N} \sigma_j^2 \begin{bmatrix} A_j & 0 \end{bmatrix}^\top P \begin{bmatrix} A_j & 0 \end{bmatrix} + \sum_{l=1}^{3} \lambda_l \begin{bmatrix} \hat{C}_l & \hat{D}_l \end{bmatrix}^\top \begin{bmatrix} \delta_l^2 I & \\ & -I \end{bmatrix} \begin{bmatrix} \hat{C}_l & \hat{D}_l \end{bmatrix} \succeq 0,$$

then the expected loss $\frac{1}{2}\mathbb{E}[y_k^\top y_k]$ asymptotically converges with rate ρ.

Theorem 6 (Steady-state risk for nonlinear model)**.** *Consider the system in Equation* (17) *where the noise signals* w_k *and* $p_k^{(j)}$ *satisfy:*

$$\begin{aligned} &\mathbb{E}[w_k] = 0, && \mathbb{E}[p_k^{(j)}] = 0, \\ &\mathbb{E}[w_k w_\tau^\top] = 0 \ \forall k \neq \tau, && p_k^{(j)} \text{ are i.i.d.}, \\ &\mathbb{E}[w_k w_k^\top] = \Sigma \succ 0, && \mathbb{E}[p_k^{(j)} p_k^{(j)}] = \sigma_j^2. \end{aligned}$$

Given $\delta_1, \delta_2, \delta_3 \geq 0$*, if there exists a positive-definite matrix* P*,* $0 < \rho < 1$*, and* $\lambda_1, \lambda_2, \lambda_3 \geq 0$ *such that:*

$$\begin{bmatrix} I \\ \hat{A} & \hat{B} \end{bmatrix}^\top \begin{bmatrix} -\rho^2 P & \\ & P \end{bmatrix} \begin{bmatrix} I \\ \hat{A} & \hat{B} \end{bmatrix} + \sum_{j=1}^{N} \sigma_j^2 \begin{bmatrix} A_j & 0 \end{bmatrix}^\top P \begin{bmatrix} A_j & 0 \end{bmatrix} + \sum_{l=1}^{3} \lambda_l \begin{bmatrix} \hat{C}_l & \hat{D}_l \end{bmatrix}^\top \begin{bmatrix} \delta_l^2 I & \\ & -I \end{bmatrix} \begin{bmatrix} \hat{C}_l & \hat{D}_l \end{bmatrix} + \begin{bmatrix} C & 0 \end{bmatrix}^\top \begin{bmatrix} C & 0 \end{bmatrix} \preceq 0,$$

and that $Tr(P B_w \Sigma B_w^\top) \leq \gamma^2$*, then we have:*

$$\lim_{K \to \infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[y_k^\top y_k] \leq \gamma^2.$$

4 Simulations

4.1 Quadratic model with minibatch noise

First, we consider the quadratic model with minibatch noise. In this case, there is no uncertainty in the model Jacobian ($\delta_1 = \delta_2 = 0$). This scenario is similar to the noisy quadratic model in Zhang et al. (2019), but with important differences: 1) we model minibatch training as multiplicative noise (Section 3.2), as opposed to an additive gradient noise in Zhang et al. (2019); 2) in addition to the classic (heavy-ball) momentum, we also analyze the Nesterov accelerated gradient (NAG), which is not considered in Zhang et al. (2019) due to its lack of closed-form solution. With only minibatch noise and our assumption of overparameterized model, the steady-state risk is zero. Therefore, our analysis focuses on the convergence rate.

> Insight 1: in minibatch training, heavy-ball is fragile to learning rate increases past the critical damping threshold. Large learning rates leads to worse convergence rate and sometimes instability.

The NQM (Zhang et al., 2019) predicts that, as batch size increases, the optimal learning rate first linearly increases, resulting in perfect scaling of convergence rate and batch size. However, once going beyond a certain threshold, increasing the learning rate no longer improves the convergence rate, resulting in diminishing returns of scaling up of the batch size. This is due to the fact that the optimization enters the curvature-dominated regime, where large learning rates leads to oscillations, characteristic of an overdamped system. According to the NQM, when increasing the learning rate in the curvature-dominated regime, the convergence rate holds constant. In full-batch training ($\mathcal{B} = N$), this matches the prediction of our framework (the "$bs = 2000$" curve in Figure 3a). However, we find that in minibatch training, increasing the batch size past the critical damping threshold actually hurts the convergence rate, and may lead to instability. The performance degradation at large learning rates worsens with smaller batch sizes. This effect is more pronounced at large momentum values (Figures 3a, 10a and 11a show $\beta = 0.999, 0.9, 0.9$ respectively).

We note that our insight on the fragility of heavy-ball momentum in minibatch training is consistent with empirical observations when using heavy-ball momentum in deep learning. Unlike the NQM, our LMI framework is able to predict this effect due to proper modeling of the minibatch noise.

> Insight 2: in minibatch training, NAG is more robust than heavy-ball momentum at large learning rates.

In contrast to heavy-ball momentum, our LMI framework predits that NAG is much more robust at large learning rates.Figure 3b shows that, with a reasonably large batch size (such as 100) and a very large momentum value ($\beta = 0.999$), increasing the minibatch size past the critical damping threshold does not hurt the convergence rate. In Figure 4, we see that compared to HB, NAG can benefit from a larger learning rate with large momentum values. This prediction agrees with observations and anaysis in other works. Sutskever et al. (2013) noted that the difference of HB and NAG is only distinct when the learning rate is reasonably large, and that NAG behaves more stably than HB in many situations. Sutskever et al. (2013) argues that the NAG's comparative stability is due to the fact that it changes the momentum in a quicker and more responsive way. Rather than relying on heuristics, our framework provides a principled and quantitative explanation.

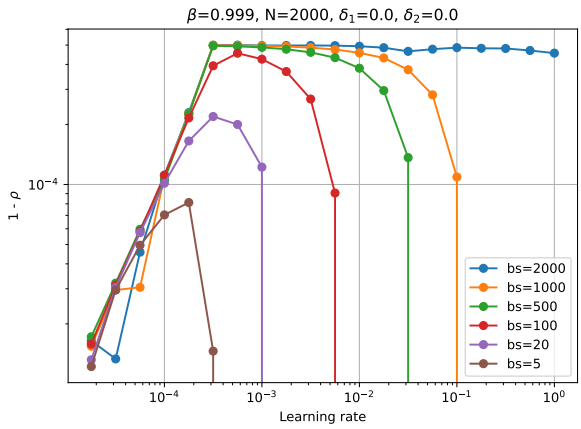
> Insight 3: NAG extends scaling slightly further than heavy-ball momentum, especially at large momentum values.

An interesting observation for NAG is that although increasing the learning rate past the critical damping threshold initially yields no benefit in convergence rate, if the learning rate is further increased, we start observing an "up-tick" again in Figure 3b. When plotted against the batch size, we see that NAG extends the benefit of parallelization slightly further than heavy-ball momentum (Figures 3c and 3d). However, we note that this observation is only prominent for very large very large momentum values (e.g. $\beta = 0.999$). For $\beta = 0.99$ (Figures 10b and 10d), we see a much lesser extension effect for batch size scaling. For $\beta = 0.9$ (Figures 11b and 11d), there is little difference from heavy-ball momentum. This limits the practical value of this insight, as using a very large momentum value slows down convergence (Figure 5) and makes training unstable.

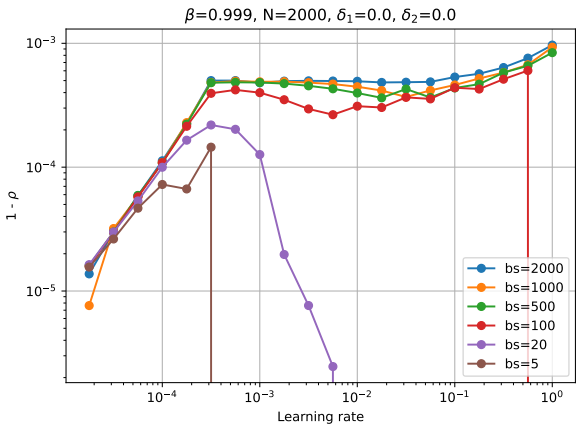
4.2 Nonlinear training dynamics

> Insight 4: NAG is more robust to training nonlinearities than heavy-ball momentum.

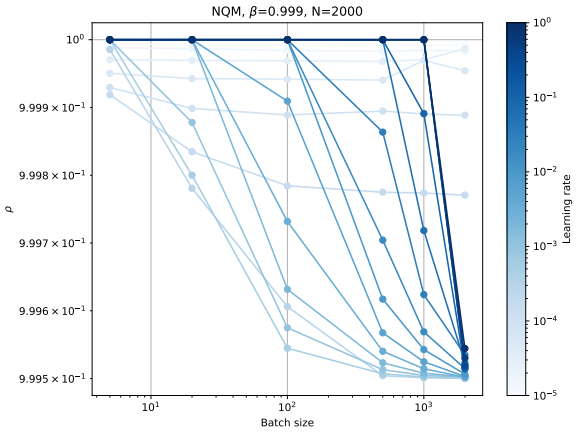
We quantify the different optimizers' robustness to nonlinear training dynamics using our Jacobian uncertainty modeling in Section 3.3. In Figure 6, we show the convergence rate as a function of Jacobian uncertainty. Compared to heavy-ball momentum, we can certify a better convergence rate for NAG under model uncertainty, both for the full-batch (Figure 6) and minibatch (Figure 12) cases. This insights adds to the previous arguments for robustness of NAG.



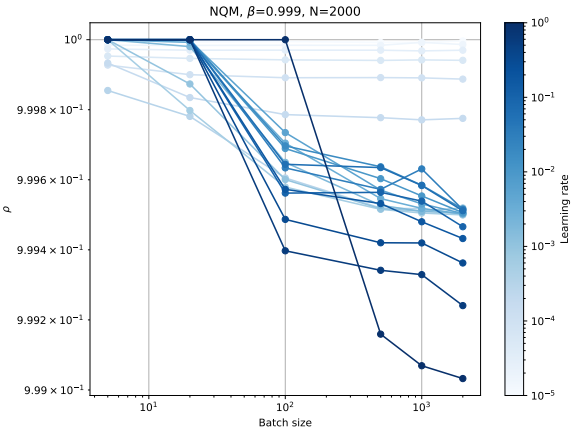
(a) Convergence rate vs. learning rate (HB, $\beta = 0.999$)



(b) Convergence rate vs. learning rate (NAG, $\beta = 0.999$)



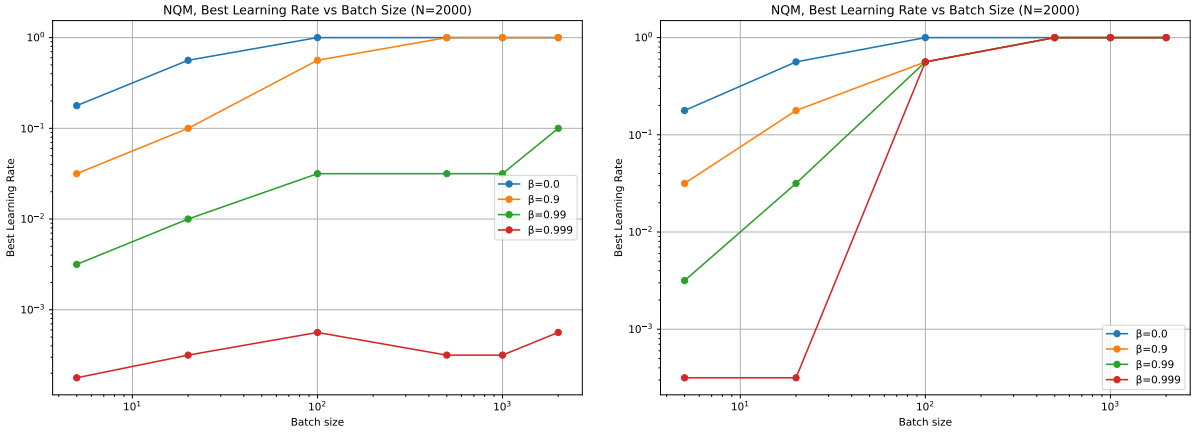
(c) Convergence rate vs. batch size (HB, $\beta = 0.999$)



(d) Convergence rate vs. batch size (NAG, $\beta = 0.999$)

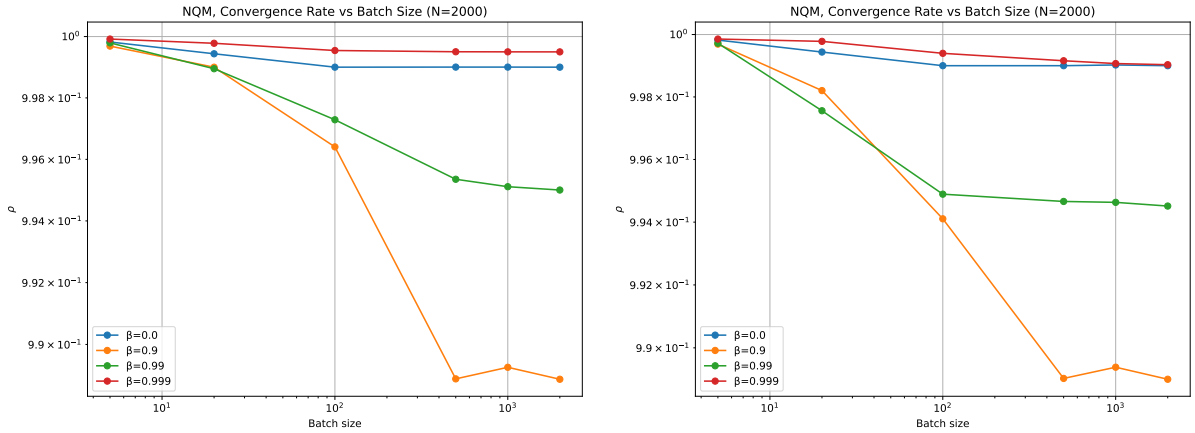
Figure 3: **With large momentum, NAG is more robust than HB at larger learning rates at small batch sizes.** With large momentum and learning rates, the system becomes overdamped.

e



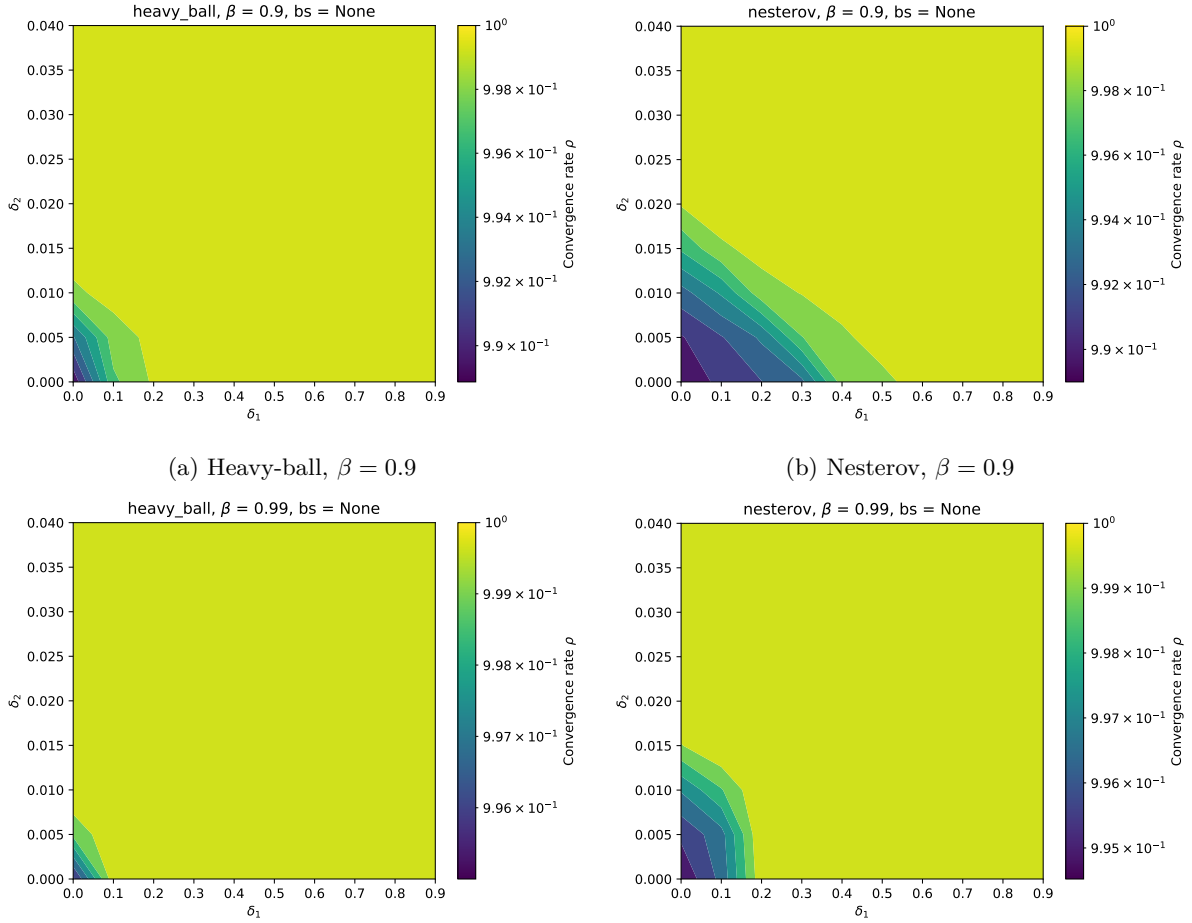
(a) Heavy Ball (b) NAG

Figure 4: **NAG allows larger learning rates than heavy-ball momentum.** Optimal learning rate vs. batch size in NQM with minibatch noise, for heavy-ball momentum and NAG, for different momentum values. With large momentum values, the optimal learning rate for NAG still scales well with batch size, whereas heavy-ball momentum suffers from instability at large learning rates.



(a) Heavy Ball (b) NAG

Figure 5: Convergence rate of heavy-ball and NAG with different momentum values.



(a) Heavy-ball, $\beta = 0.9$

(b) Nesterov, $\beta = 0.9$

(c) Heavy-ball, $\beta = 0.99$

(d) Nesterov, $\beta = 0.99$

Figure 6: **NAG is more robust than HB when subjected to nonlinearity.** Convergence rate as a function of the amount of Jacobian uncertainty (full batch). For each point on the heatmap, we plot the best convergence rate searched over learning rate.

5 Background: heavy-ball momentum and Nesterov accelerated gradient

Theoretical & empirical results It is often observed in practice that NAG enjoys better performance and robustness than heavy-ball momentum (Sutskever et al., 2013; Choi et al., 2019). However, theoretical evidences are nuanced and often lacking. In the class of smooth, convex functions, NAG (with proper learning rate schedule) has the optimal convergence rate in first-order optimizers (Nesterov, 1983), whereas heavy-ball momentum has no convergence guarantee (Lessard et al., 2016). In more realistic scenarios, however, there is little theoretical evidence on the advantages of NAG. For certain types of overparameterized neural networks, NAG converges to the global minimum with a similar or worse rate than HB (Bu et al., 2021; Liu et al., 2022a;b). For stochastic setting with small batch sizes, neither momentum methods can outperform SGD (Kidambi et al., 2018).

"Lookahead" gradient in NAG There have been many works that attempt to explain NAG's superior performance and robustness. Intuitively, NAG evaluates the gradient at a partial update location, instead of at the current location as heavy-ball momentum (Figure 7). This "lookahead" step in gradient evaluation allows NAG to adapt the momentum in a quicker and more responsive way, and to reduce oscillation along high-curvature directions, which leads to NAG being more tolerent to larger momentum values (Sutskever et al., 2013). The analysis in Sutskever et al. (2013) gives helpful intuition, but lacks rigor. As we will show in the following paragraphs, the "lookahead" step has a more principled explanation using the control interpretations.

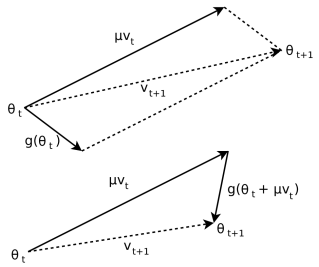


Figure 7: Taken from Figure 1 in Sutskever et al. (2013). (Top) Heavy-ball momentum. (Bottom) Nesterov accelerated gradient. Heavy-ball momentum evaluates the gradient $g(\theta_t)$ at the current location, while the NAG gradient evaluation $g(\theta_{t+1})$ has a "lookahead" step.

Continuous-time limits and ODEs Another line of work focuses on the continuous-time limit of the optimizer dynamics. Polyak (1964) motivated the proposed momentum method as "the method of a small heavy sphere" that moves in a potential field (hence the name "heavy-ball momentum"). A more accurate physical analogy should be "a point mass moving in a potential field through a viscous medium" (Qian, 1999). The viscous medium causes friction that is proportional to the velocity. The dynamics can be described by the following second-order ODE:

$$m\ddot{x} + b\dot{x} + \nabla V(x) = 0, \tag{18}$$

where x is the position of the point mass (analogous to θ in optimizers), m is the mass, b is the viscous damping coefficient, and $\nabla V(x)$ is the potential field. We obtain the heavy-ball momentum method by discretizing Equation (18) using the forward Euler method (Qian, 1999). Let Δt be the time step, the discretized dynamics is:

$$m\frac{x_{t+\Delta t} - 2x_t + x_{t-\Delta t}}{\Delta t^2} + b\frac{x_{t+\Delta t} - x_t}{\Delta t} + \nabla V(x_t) = 0.$$

Rearranging, we recover the heavy-ball momentum method with learning rate $\eta = \frac{\Delta t^2}{m+b\Delta t}$ and momentum $\beta = \frac{m}{m+b\Delta t}$:

$$x_{t+\Delta t} = x_t + \frac{m}{m+b\Delta t}(x_t - x_{t-\Delta t}) - \frac{\Delta t^2}{m+b\Delta t}\nabla V(x_t).$$

However, the naive approach for taking continuous-time limit is not useful for distinguishing NAG from HB. If we take $\Delta t \to 0$, HB and NAG have the same limiting ODE (Equation (18)) (Shi et al., 2022). To address this, Shi et al. (2022) proposed an alternative limiting process that yields high-resolution ODEs, which differentiates NAG and HB. The high-resolution ODE for HB and NAG are given by:

$$\textbf{HB:} \quad m\ddot{x} + b\dot{x} + (1 + \frac{b\Delta t}{2m})\nabla V(x) = 0, \tag{19a}$$

$$\textbf{NAG:} \quad m\ddot{x} + \big(b + \frac{\Delta t}{m}\nabla^2 V(x)\big)\dot{x} + (1 + \frac{b\Delta t}{2m})\nabla V(x) = 0. \tag{19b}$$

The high-resolution ODEs have kept the Δt terms, and in the infinitesimal limit recover Equation (18). Notably, NAG has an additional damping term, which is proportional to the Hessian of the loss function. According to the ODE, NAG takes more cautious steps along high-curvature directions, which generally helps reduce oscillations. The high-resolution ODEs are shown to better match simulations of HB and NAG than the naive version, and empirical results support that NAG signficantly reduces oscillation (Shi et al., 2022).

Control interpretations Rather than treating the optimizer as an inherent dynamical system that responds to an external force $\nabla V(x)$ (as in Equation (18)), we can adopt the opposite view and treat the optimizer as a controller (Hu & Lessard, 2017a). In the control view, $\nabla V(x)$ (analogous to gradient $\nabla \mathcal{L}$ in optimization) is the plant that has its own dynamics, and we would like to design a controller (optimizer) that stabilizes the plant (make the gradient $\nabla \mathcal{L}$ go to zero). This is visualized in the block diagram in Figure 8.

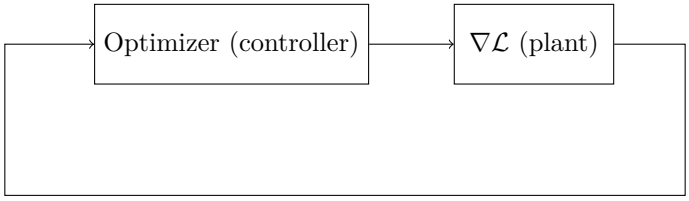


Figure 8: Block diagram showing the control view of the optimization process.

Since first-order optimizers are linear time-invariant (LTI) systems, we can analyze their properties using transfer functions. Applying the z-transform to the optimizer update equations, we get the following transfer functions for gradient descent, HB and NAG:

$$G_{\text{GD}}(z) = -\frac{\eta}{z-1}, \quad G_{\text{HB}}(z) = -\frac{\eta}{z-1}\cdot\frac{z}{z-\beta}, \quad G_{\text{NAG}}(z) = -\frac{\eta}{z-1}\cdot\frac{(1+\beta)z-\beta}{z-\beta}$$

From the transfer function, we can see that gradient descent corresponds to an integral controller. An integral controller (transfer function $\frac{K_I}{z-1}$) produces control signal that is proportional to the sum of past errors, which is necessary to eventually drive the error to zero. Unsurprisingly, the transfer functions of HB and NAG both contain an integral control component.

The acceleration of the momentum methods is reflected by the additional components in their transfer functions. Heavy-ball has an additional $\frac{z}{z-\beta}$ term, which is a lag compensator. A lag compensator boosts low-frequency gain and introduces a phase lag (Figure 9). In optimization, our objective is to asymptotically drive the gradient $\nabla \mathcal{L}$ to zero, which is a low frequency signal to track (in fact, frequency $\omega = 0$). A boost in low-frequency gain results in faster asymptotic convergence rate. This is apparent when we set the frequency $\omega = 0$, we have $z = e^{j\omega} = 1$, and $\frac{z}{z-\beta} = \frac{1}{1-\beta}$, recovering heavy-ball momentum's $\frac{1}{1-\beta}$ improvement in convergence rate for quadratic loss functions.

NAG is also the combination of an integrator and a lag compensator. With the same learning rate and momentum parameter, it has the same low-frequency gain as HB ($G_{NAG}(z) = G_{HB}(z)$ when $z = 1$), hence the same asymptotic convergence rate. However, it has a zero at $\frac{\beta}{1+\beta}$ instead of 0, which leads to a smaller phase lag. On the Bode plot (Figure 9), this leads to a flatter slope at the crossover frequency and a larger

phase margin compared to HB, which contribute to NAG's improved robustness. The comparative fragility of HB may explain its lack of convergence guarantee for general strongly convex functions (Hu & Lessard, 2017a).

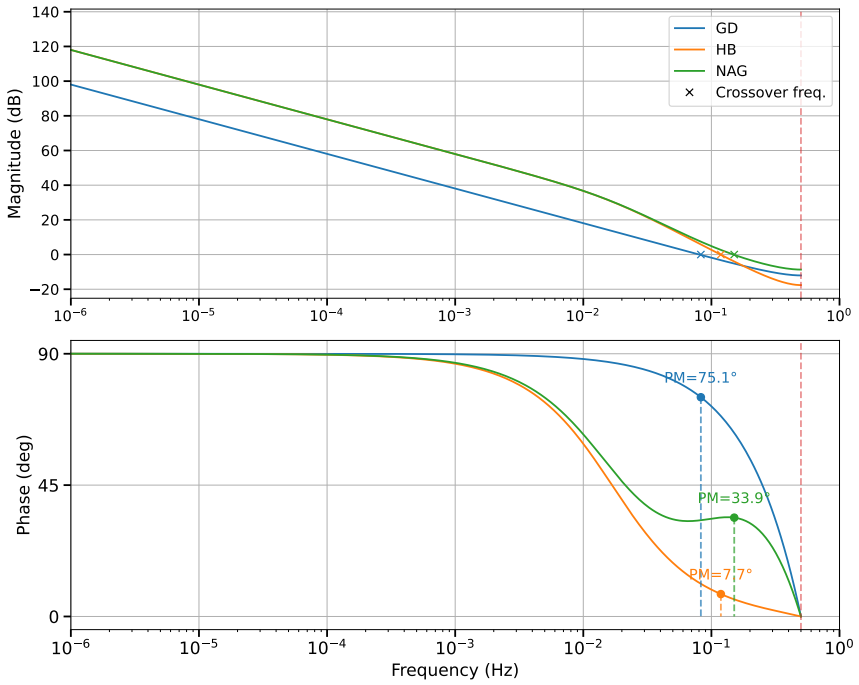


Figure 9: Bode plot showing gradient descent, HB and NAG (learning rate $\eta = 0.5$, momentum $\beta = 0.9$). The Bode plot shows the magnitude and phase of the optimizer responses, plotted against gradient signal frequency from 0 to the Nyquist frequency (with sampling rate 1, Nyquist freq. $= 1/2$, shown as the red dashed line). At low frequency, both HB and NAG have the same magnitude ($20 \log_{10} \frac{1}{1-\beta} = 20$dB higher than GD), meaning that they both have $\frac{1}{1-\beta}$ improvement in asymptotic convergence rate for quadratic loss functions. However, the high frequency responses are different: NAG has a flatter slope at crossover frequency, and the smaller phase lag of NAG leads to a larger phase margin (the phase at crossover frequency, taken difference with $0°$ due to positive feedback) than HB. These indicate that NAG is more robust to high-frequency gradient noise.

Additionally, NAG can be viewed as incorporating derivative control in its dynamics (Hu & Lessard, 2017a). We can rewrite the NAG update as:

$$u_{k+1} = u_k + \beta(u_k - u_{k-1}) - \eta \nabla \mathcal{L}(u_k) - \eta\beta(\nabla \mathcal{L}(u_k) - \nabla \mathcal{L}(u_{k-1})) \tag{20}$$

The last term in Equation (20) is a form of derivative control. Derivative control predicts future behavior based on local trends. This matches NAG's "lookahead" interpretation in Sutskever et al. (2013). Alltogether, NAG can be viewed as a form of the celebrated Proportional–integral–derivative (PID) controller (Astrom, 1995; Hu & Lessard, 2017a).

6 Related work

The noisy quadratic model (NQM) (Zhang et al., 2019) gives important insights on how the optimization performance scales with batch sizes, and how such scaling interacts with factors such as momentum, moving average and preconditioning. This work extends the insights of NQM by adopting proper minibatch noise models, and by incorporating nonlinear training dynamics. However, a limitation of our approach is that we focus on asymptotic performance, whereas NQM predicts finite-step loss values.

Lessard et al. (2016) first proposed to use the robust control framework for the analysis and design of optimizers. In this framework, the optimization is viewed as a dynamical system, and the convergence is expressed as the feasibility problem of a semi-definite program (SDP). Our analysis is based on the same framework. In Lessard et al. (2016), much of the effort is focused on designing integral-quadratic constraints (IQCs) for classical problem types in optimization (such as quadratic, smooth convex). In contrast, we focus on the case of neural network training, and propose a model for nonlinear dynamics and minibatch noise. Several follow-up works take dissipativity theory approach (Hu & Lessard, 2017b; Lessard, 2022), which are mathematically equivalent to the original IQC framework (Lessard et al., 2016), but may result in smaller LMIs. Hu et al. (2021) adapts the framework to stochastic gradient descent. However, they seek the worst-case performance with respect to the gradient noise, which is too conservative for minibatch training. Instead, we take the average-case performance with respect to our minibatch noise model.

A competing line of work is on the Performance Estimation Problem (PEP) (Drori & Teboulle, 2014; Taylor et al., 2017; 2018; Taylor & Bach, 2019). PEP also studies the worst-case performance of optimization problems by numerically solving SDPs. In contrast to the robust control based methods, PEP treats the optimization steps explicitly. This gives PEP the advantange of allowing finite-step loss bounds, whereas the robust control framework only certifies asymptotic performance. However, PEP scales poorly with the number of optimization steps, especially with complex problem classes and stochastic settings. We defer it to future work to obtain finite-step, NQM-like performance bounds using PEP.

References

Karl J Astrom. Pid controllers: theory, design, and tuning. *The international society of measurement and control*, 1995.

Zhiqi Bu, Shiyun Xu, and Kan Chen. A dynamical view on optimization algorithms of overparameterized neural networks. In *International conference on artificial intelligence and statistics*, pp. 3187–3195. PMLR, 2021.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1):451–482, 2014.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *2017 American Control Conference (ACC)*, pp. 3114–3119. IEEE, 2017a.

Bin Hu and Laurent Lessard. Dissipativity theory for nesterov's accelerated method. In *International Conference on Machine Learning*, pp. 1549–1557. PMLR, 2017b.

Bin Hu, Peter Seiler, and Laurent Lessard. Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Mathematical programming*, 187:383–408, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Laurent Lessard. The analysis of optimization algorithms: A dissipativity approach. *IEEE Control Systems Magazine*, 42(3):58–72, 2022.

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.

Xin Liu, Zhisong Pan, and Wei Tao. Provable convergence of nesterov's accelerated gradient method for over-parameterized neural networks. *Knowledge-Based Systems*, 251:109277, 2022a.

Xin Liu, Wei Tao, and Zhisong Pan. A convergence analysis of nesterov's accelerated gradient method in training deep linear neural networks. *Information Sciences*, 612:898–925, 2022b.

Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165): 1–73, 2021.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, pp. 543, 1983.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Bin Shi, Simon S Du, Michael I Jordan, and Weijie J Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, pp. 1–70, 2022.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Adrien Taylor and Francis Bach. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In *Conference on Learning Theory*, pp. 2934–2992. PMLR, 2019.

Adrien Taylor, Bryan Van Scoy, and Laurent Lessard. Lyapunov functions for first-order methods: Tight automated convergence guarantees. In *International Conference on Machine Learning*, pp. 4897–4906. PMLR, 2018.

Adrien B Taylor, Julien M Hendrickx, and François Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161:307–345, 2017.

Nikhil Vyas, Yamini Bansal, and Preetum Nakkiran. Limitations of the ntk for understanding generalization in deep learning. *arXiv preprint arXiv:2206.10012*, 2022.

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pp. 11727–11737. PMLR, 2021.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

A Appendix

A.1 Proofs

Proof of Theorem 1. Multiply both sides of equation 8 by $\xi_k - \xi^*$, we get:

$$(\xi_k - \xi^*)^\top A^\top P A(\xi_k - \xi^*) - \rho^2(\xi_k - \xi^*)^\top P(\xi_k - \xi^*) \leq 0 \tag{21}$$

Applying the dynamical equation equation 6 to $(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)$ and taking expectation, we get:

$$\begin{aligned}
\mathbb{E}[(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)] &= \mathbb{E}_{\epsilon_{k-1}}[\big(A(\xi_{k-1} - \xi^*) + B_w\epsilon_{k-1}\big)^\top P\big(A(\xi_{k-1} - \xi^*) + B_w\epsilon_{k-1}\big)] \\
&= \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top P A(\xi_{k-1} - \xi^*)] + 2\mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top P B_w\epsilon_{k-1}] + \mathbb{E}[\epsilon_{k-1}^\top B_w^\top P B_w\epsilon_{k-1}] \\
&= \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top P A(\xi_{k-1} - \xi^*)] + \mathrm{Tr}(B_w^\top P B_w)
\end{aligned}$$

Substituting into equation 21, taking the expectation and multiplying by ρ^{-2k}, we get a telescoping series:

$$\begin{aligned}
0 &\geq \sum_{l=0}^{k-1} \rho^{-2l}\mathbb{E}[(\xi_l - \xi^*)^\top A^\top P A(\xi_l - \xi^*) - \rho^2(\xi_l - \xi^*)^\top P(\xi_l - \xi^*)] \\
&= \rho^{-2(k-1)}\mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top P A(\xi_{k-1} - \xi^*)] \\
&\quad - \rho^{-2(k-2)}\mathbb{E}[(\xi_{k-1} - \xi^*)^\top P(\xi_{k-1} - \xi^*)] + \rho^{-2(k-2)}\mathbb{E}[(\xi_{k-2} - \xi^*)^\top A^\top P A(\xi_{k-2} - \xi^*)] \\
&\quad - \cdots \\
&\quad - \rho^0\mathbb{E}[(\xi_1 - \xi^*)^\top P(\xi_1 - \xi^*)] + \rho^0\mathbb{E}[(\xi_0 - \xi^*)^\top A^\top P A(\xi_0 - \xi^*)] \\
&\quad - \rho^2\mathbb{E}[(\xi_0 - \xi^*)^\top P(\xi_0 - \xi^*)] \\
&= \rho^{-2(k-1)}\Big(\mathbb{E}[(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)] - \mathrm{Tr}(B_w^\top P B_w)\Big) \\
&\quad + \rho^{-2(k-2)}\Big(- \mathbb{E}[(\xi_{k-1} - \xi^*)^\top P(\xi_{k-1} - \xi^*)] + \mathbb{E}[(\xi_{k-1} - \xi^*)^\top P(\xi_{k-1} - \xi^*)] - \mathrm{Tr}(B_w^\top P B_w)\Big) \\
&\quad + \cdots \\
&\quad + \rho^0\Big(- \mathbb{E}[(\xi_1 - \xi^*)^\top P(\xi_1 - \xi^*)] + \mathbb{E}[(\xi_1 - \xi^*)^\top P(\xi_1 - \xi^*)] - \mathrm{Tr}(B_w^\top P B_w)\Big) \\
&\quad - \rho^2\mathbb{E}[(\xi_0 - \xi^*)^\top P(\xi_0 - \xi^*)] \\
&= \rho^{-2(k-1)}\mathbb{E}[(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)] - \rho^2\mathbb{E}[(\xi_0 - \xi^*)^\top P(\xi_0 - \xi^*)] - \Big(1 + \rho^{-2} + \cdots + \rho^{-2(k-1)}\Big)\mathrm{Tr}(B_w^\top P B_w)
\end{aligned}$$

Rearranging, we have:

$$\begin{aligned}
\mathbb{E}[(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)] &\leq \rho^{2k}\mathbb{E}[(\xi_0 - \xi^*)^\top P(\xi_0 - \xi^*)] + \rho^{2(k-1)}\Big(1 + \rho^{-2} + \cdots + \rho^{-2(k-1)}\Big)\mathrm{Tr}(B_w^\top P B_w) \\
&= \rho^{2k}\mathbb{E}[(\xi_0 - \xi^*)^\top P(\xi_0 - \xi^*)] + \frac{1 - \rho^{2k}}{1 - \rho^2}\mathrm{Tr}(B_w^\top P B_w)
\end{aligned}$$

The quadratic loss function is:

$$\mathcal{L}_k = \frac{1}{2}\theta_k^\top H\theta_k = \frac{1}{2}\xi_k^\top \bar{C}^\top H\bar{C}\xi_k = \xi_k^\top M\xi_k, \quad M := \frac{1}{2}\bar{C}^\top H\bar{C}$$

Since M is positive-semidefinite, we replace P with $M^{1/2}(M^{1/2})^\dagger P(M^{1/2})^\dagger M^{1/2}$ and get:

$$\begin{aligned}
\frac{1}{2}\mathbb{E}[(\xi_k - \xi^*)^\top M(\xi_k - \xi^*)] &\leq \mathrm{cond}((M^{1/2})^\dagger P(M^{1/2})^\dagger) \cdot \rho^{2k}\frac{1}{2}\mathbb{E}[(\xi_0 - \xi^*)^\top M(\xi_0 - \xi^*)] \\
&\quad + \sigma_{\min}^{-1}((M^{1/2})^\dagger P(M^{1/2})^\dagger) \cdot \frac{1 - \rho^{2k}}{1 - \rho^2}\mathrm{Tr}(B_w^\top P B_w)
\end{aligned}$$

We restrict our attention to the dimensions in M with non-zero eigenvalues, because the singular dimensions do not contribute to the loss. This way, $\sigma_{\min}^{-1}((M^{1/2})^\dagger P(M^{1/2})^\dagger)$ and $\mathrm{cond}((M^{1/2})^\dagger P(M^{1/2})^\dagger)$ are well-defined. The loss $\mathcal{L}_k$ has exponential convergence with rate ρ^2. □

Proof of Theorem 2. Let P_k be the variance of the state ξ at time k,

$$P_k = \mathbb{E}[(\xi_k - \bar{\xi}_k)(\xi_k - \bar{\xi}_k)^\top], \quad \bar{\xi}_k := \mathbb{E}[\xi_k]$$

Notice that since ϵ_k is zero-centered i.i.d. noise, $\bar{\xi}$ evolves autonomously.

$$\bar{\xi}_{k+1} = A\bar{\xi}_k$$

Without loss of generality, we assume $\xi_0 = 0$. Then, we have $\bar{\xi}_k = 0$ for all k, and $P_k = \mathbb{E}[\xi_k \xi_k^\top]$. Substituting in the solution of equation 6 $\xi_k = A^k \xi_0 + \sum_{l=0}^{k-1} A^{k-1-l} B_w \epsilon_l$, we get:

$$P_{k+1} = AP_kA^\top + B_w B_w^\top$$

Since A is stable ($\mathrm{spec}(A)$ has radius less than 1), $\lim_{k\to\infty} P_k = P$ where,

$$APA^\top - P + B_w B_w^\top = 0$$

The steady-state risk is the variance of the output y_k ($\bar{y}_k := \mathbb{E}[y_k] = C\bar{\xi}_k = 0$):

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathrm{Tr}(\mathbb{E}[y_k y_k^\top]) = \mathrm{Tr}(CPC^\top).$$

□

Proof of Theorem 3. Multiply Equation (14) by $\xi_k - \xi^*$ on both sides, we have:

$$(\xi_k - \xi^*)^\top A^\top PA(\xi_k - \xi^*) - \rho^2(\xi_k - \xi^*)^\top P(\xi_k - \xi^*) + \sum_{j=1}^{N} \sigma_j^2 (\xi_k - \xi^*)^\top A_j^\top PA_j(\xi_k - \xi^*) \preceq 0 \tag{22}$$

Applying the dynamical equation Equation (12) to $(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)$ and take expectation, we have:

$$\begin{aligned}
\mathbb{E}[(\xi_k - \xi^*)^\top P(\xi_k - \xi^*)] &= \mathbb{E}\Bigg[\Bigg(A(\xi_{k-1} - \xi^*) + \sum_{j=1}^{N} A_j(\xi_{k-1} - \xi^*) \cdot p_{k-1}^{(j)} + B_w\epsilon_{k-1}\Bigg)^\top \\
&\qquad P\Bigg(A(\xi_{k-1} - \xi^*) + \sum_{j=1}^{N} A_j(\xi_{k-1} - \xi^*) \cdot p_{k-1}^{(j)} + B_w\epsilon_{k-1}\Bigg)\Bigg] \\
&= \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top PA(\xi_{k-1} - \xi^*)] + \sum_{j=1}^{N} \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A_j^\top PA_j(\xi_{k-1} - \xi^*)] \cdot \mathbb{E}[(p^{(j)})^2] \\
&\qquad + \mathbb{E}[\epsilon_{k-1}^\top B_w^\top PB_w\epsilon_{k-1}] \\
&= \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A^\top PA(\xi_{k-1} - \xi^*)] + \sum_{j=1}^{N} \sigma_j^2 \mathbb{E}[(\xi_{k-1} - \xi^*)^\top A_j^\top PA_j(\xi_{k-1} - \xi^*)] + \mathrm{Tr}(B_w^\top PB_w) \\
&= \mathbb{E}\Bigg[(\xi_{k-1} - \xi^*)^\top \Bigg(A^\top PA + \sum_{j=1}^{N} \sigma_j^2 A_j^\top PA_j\Bigg)(\xi_{k-1} - \xi^*)\Bigg] + \mathrm{Tr}(B_w^\top PB_w)
\end{aligned}$$

Substituting into Equation (22) and multiplying by ρ^{-2k}, we get a telescoping series:

$$
\begin{aligned}
0 \geq & \sum_{l=0}^{k-1} \rho^{-2l} \mathbb{E}[(\xi_l - \xi^*)^\top A^\top P A (\xi_l - \xi^*) - \rho^2 (\xi_l - \xi^*)^\top P (\xi_l - \xi^*) + \sum_{j=1}^{N} \sigma_j^2 (\xi_l - \xi^*)^\top A_j^\top P A_j (\xi_l - \xi^*)] \\
&= \rho^{-2(k-1)} \mathbb{E}[(\xi_{k-1} - \xi^*)^\top \left(A^\top P A + \sum_{j=1}^{N} \sigma_j^2 A_j^\top P A_j \right) (\xi_{k-1} - \xi^*)] \\
&\quad - \rho^{-2(k-2)} \mathbb{E}[(\xi_{k-1} - \xi^*)^\top P (\xi_{k-1} - \xi^*)] + \rho^{-2(k-2)} \mathbb{E}[(\xi_{k-2} - \xi^*)^\top \left(A^\top P A + \sum_{j=1}^{N} \sigma_j^2 A_j^\top P A_j \right) (\xi_{k-2} - \xi^*)] \\
&\quad - \cdots \\
&\quad - \rho^0 \mathbb{E}[(\xi_1 - \xi^*)^\top P (\xi_1 - \xi^*)] + \rho^0 \mathbb{E}[(\xi_0 - \xi^*)^\top \left(A^\top P A + \sum_{j=1}^{N} \sigma_j^2 A_j^\top P A_j \right) (\xi_0 - \xi^*)] \\
&\quad - \rho^2 \mathbb{E}[(\xi_0 - \xi^*)^\top P (\xi_0 - \xi^*)] \\
&= -\rho^{-2(k-1)} \left(\mathbb{E}[(\xi_k - \xi^*)^\top P (\xi_k - \xi^*)] - \mathrm{Tr}(B_w^\top P B_w) \right) \\
&\quad - \left(\rho^{-2(k-2)} + \cdots + \rho^0 \right) \mathrm{Tr}(B_w^\top P B_w) \\
&\quad - \rho^2 \mathbb{E}[(\xi_0 - \xi^*)^\top P (\xi_0 - \xi^*)] \\
&= -\rho^{-2(k-1)} \mathbb{E}[(\xi_k - \xi^*)^\top P (\xi_k - \xi^*)] - \rho^2 \mathbb{E}[(\xi_0 - \xi^*)^\top P (\xi_0 - \xi^*)] - \left(1 + \rho^{-2} + \cdots + \rho^{-2(k-1)} \right) \mathrm{Tr}(B_w^\top P B_w)
\end{aligned}
$$

The rest of the proof follows the same steps as in the proof of Theorem 1. □

Proof. Let $V(\xi_k) = \xi_k^\top P \xi_k$. Then,

$$
\begin{aligned}
V(\xi_{k+1}) - V(\xi_k) = & \xi_k^\top A^\top P A \xi_k + 2 \xi_k^\top A^\top P \Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big) \\
& + \Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big)^\top P \Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big) - \xi_k^\top P \xi_k
\end{aligned}
$$

Taking expectation, many terms (marked in red) drop out:

$$
\begin{aligned}
\mathbb{E}[V(\xi_{k+1})] - \mathbb{E}[V(\xi_k)] = & \mathbb{E}[\xi_k^\top A^\top P A \xi_k] - \mathbb{E}[\xi_k^\top P \xi_k] + 2\mathbb{E}[\xi_k^\top A^\top P \Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big)] \\
& + \mathbb{E}[\Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big)^\top P \Big(\sum_{j=1}^{N} A_j \xi_k \cdot p^{(j)} + B_w w_k \Big)] \\
= & \mathbb{E}[\xi_k^\top A^\top P A \xi_k] - \mathbb{E}[\xi_k^\top P \xi_k] + \sum_{j=1}^{N} \mathbb{E}[\xi_k^\top A_j^\top P A_j \xi_k \cdot (p^{(j)})^2] + \mathbb{E}[w_k^\top B_w^\top P B_w w_k] \\
& + 2 \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} \mathbb{E}[\xi_k^\top A_i^\top P A_j \xi_k \cdot p^{(i)} p^{(j)}] + 2 \sum_{j=1}^{N} \mathbb{E}[\xi_k^\top A_j^\top P B_w \cdot p^{(j)} w_k] \\
= & \mathbb{E}\left[\xi_k^\top \left(A^\top P A - P + \sum_{j=1}^{N} \sigma_j^2 A_j^\top P A_j \right) \xi_k \right] + \mathbb{E}[w_k^\top B_w^\top P B_w w_k]
\end{aligned}
$$

Multiply Equation (15) by ξ_k on both sides and substitute in the above, we have:

$$\mathbb{E}[V(\xi_{k+1})] - \mathbb{E}[V(\xi_k)] \leq -\mathbb{E}[\xi_k^\top C^\top C \xi_k] + \mathbb{E}[w_k^\top B_w^\top P B_w w_k] = -\mathbb{E}[y_k^\top y_k] + \mathrm{Tr}(P B_w \Sigma B_w^\top)$$

Taking average from $k = 0$ to $K - 1$, we have:

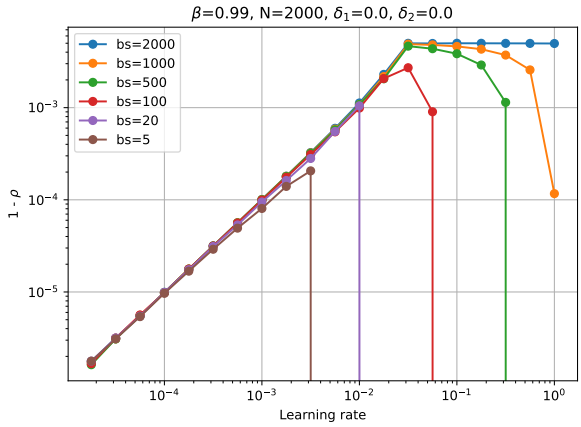
$$\frac{1}{K}\Big(\mathbb{E}[V(\xi_K)] - \mathbb{E}[V(\xi_0)]\Big) \leq \mathrm{Tr}(P B_w \Sigma B_w^\top) - \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[y_k^\top y_k]$$

Since the system is well-posed, $\mathbb{E}[V(\xi_K)] - \mathbb{E}[V(\xi_0)]$ is bounded. Therefore, the left-hand-side reduces to zero as $K \to \infty$, and we have:

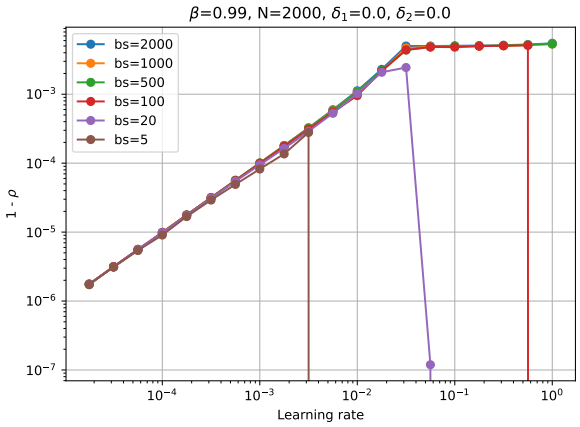
$$\lim_{K\to\infty} \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[y_k^\top y_k] \leq \mathrm{Tr}(P B_w \Sigma B_w^\top) \leq \gamma^2.$$

□

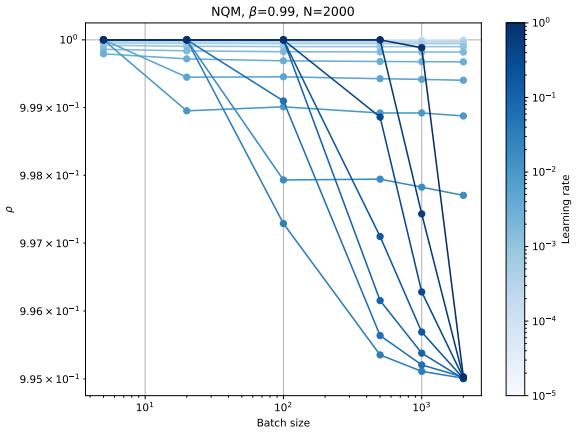
A.2 Additional simulation results



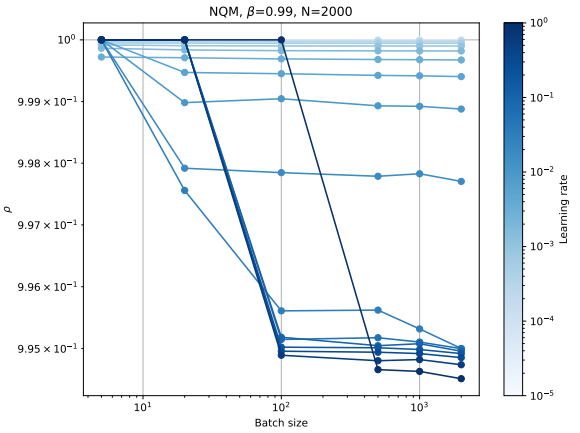
(a) Convergence rate vs. learning rate (HB, $\beta = 0.99$)



(b) Convergence rate vs. learning rate (NAG, $\beta = 0.99$)

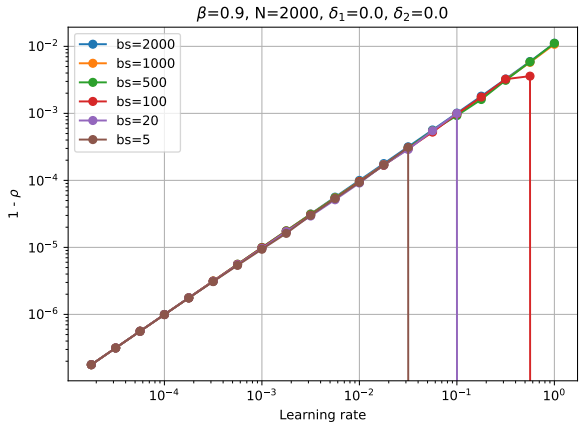


(c) Convergence rate vs. batch size (HB, $\beta = 0.99$)

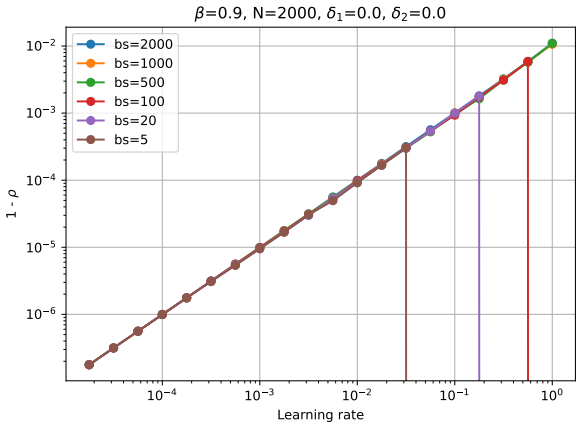


(d) Convergence rate vs. batch size (NAG, $\beta = 0.99$)

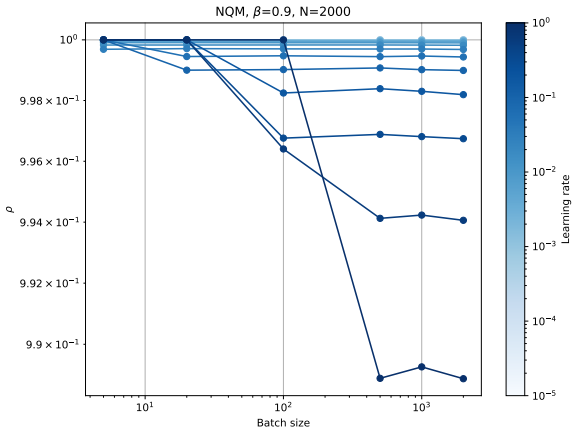
Figure 10: **With large momentum, NAG is more robust than HB at larger learning rates at small batch sizes.** With large momentum and learning rates, the system becomes overdamped.



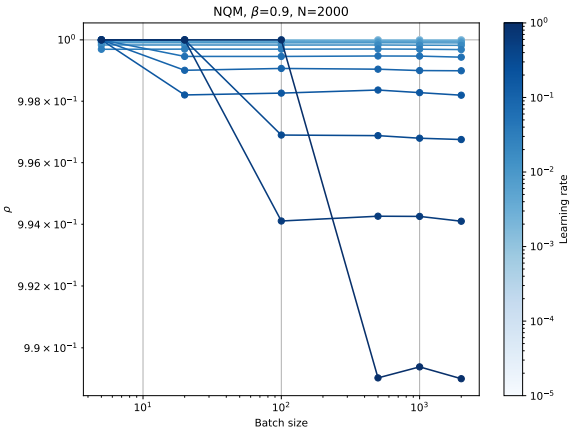
(a) Convergence rate vs. learning rate (HB, $\beta = 0.9$)



(b) Convergence rate vs. learning rate (NAG, $\beta = 0.9$)

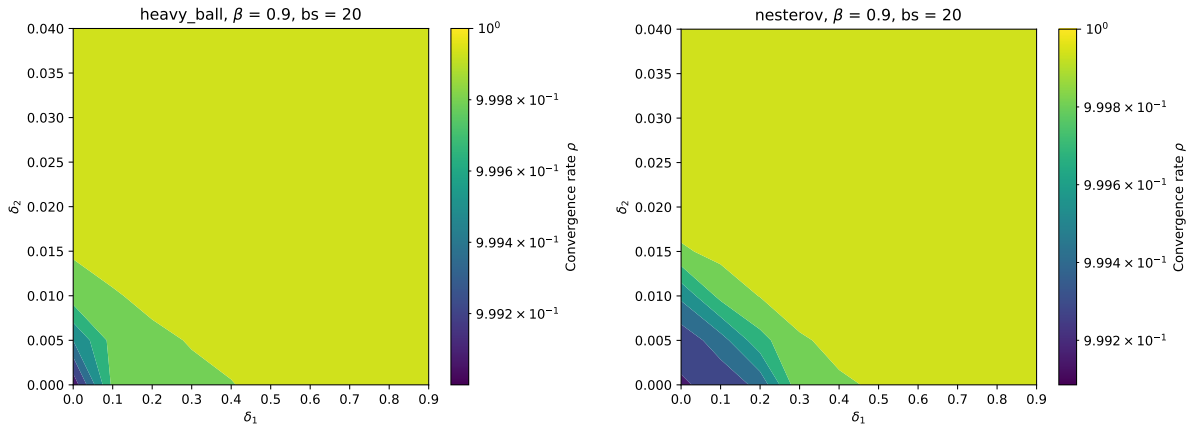


(c) Convergence rate vs. batch size (HB, $\beta = 0.9$)



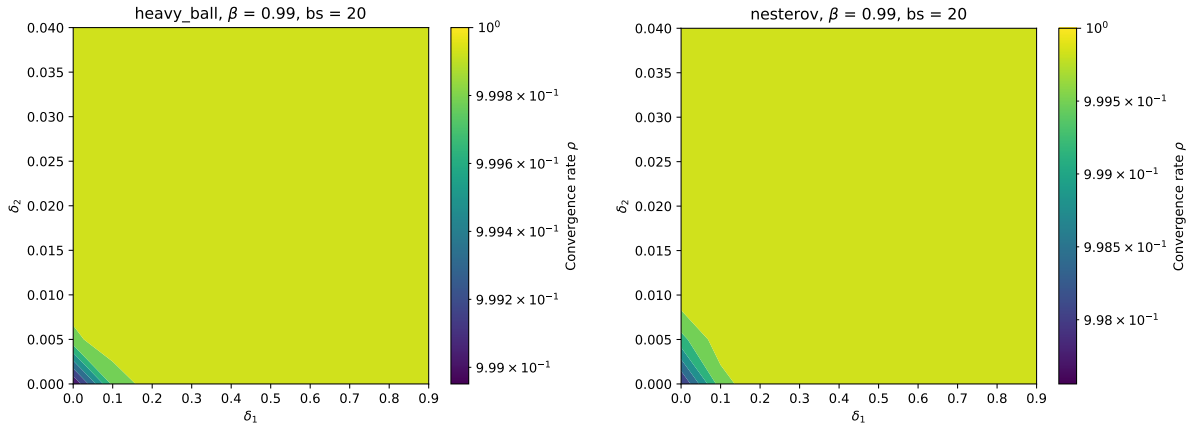
(d) Convergence rate vs. batch size (NAG, $\beta = 0.9$)

Figure 11: **With large momentum, NAG is more robust than HB at larger learning rates at small batch sizes.** With large momentum and learning rates, the system becomes overdamped.



(a) Heavy-ball, $\beta = 0.9$

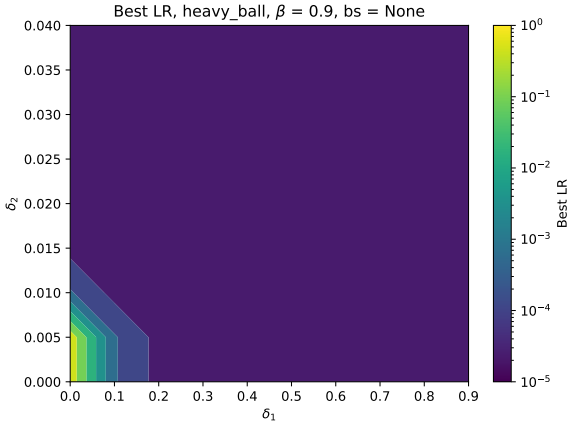
(b) Nesterov, $\beta = 0.9$



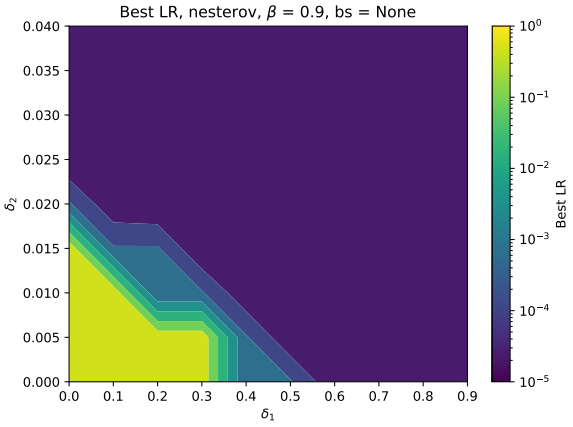
(c) Heavy-ball, $\beta = 0.99$

(d) Nesterov, $\beta = 0.99$

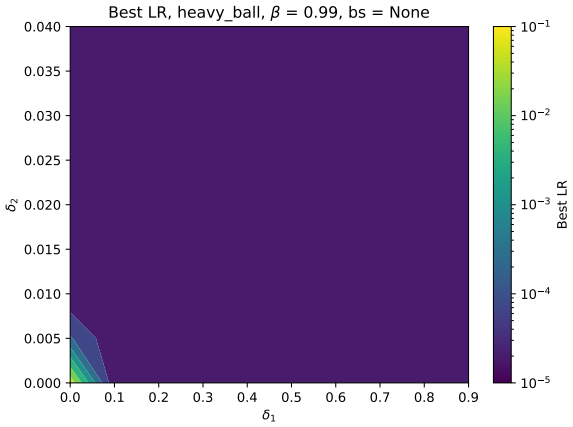
Figure 12: Convergence rate as a function of the amount of Jacobian uncertainty (batch size 20). For each point on the heatmap, we plot the best convergence rate searched over learning rate.



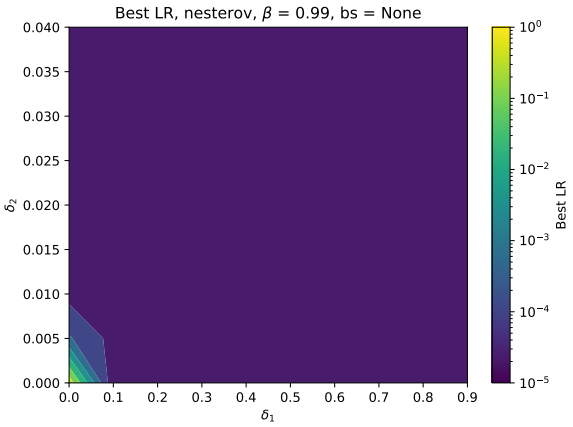
(a) Heavy-ball, $\beta = 0.9$



(b) Nesterov, $\beta = 0.9$

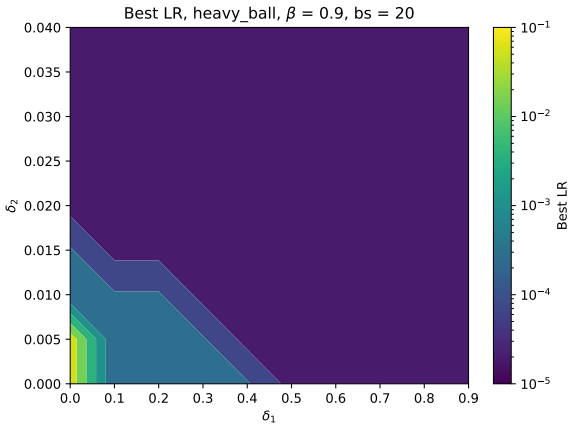


(c) Heavy-ball, $\beta = 0.99$

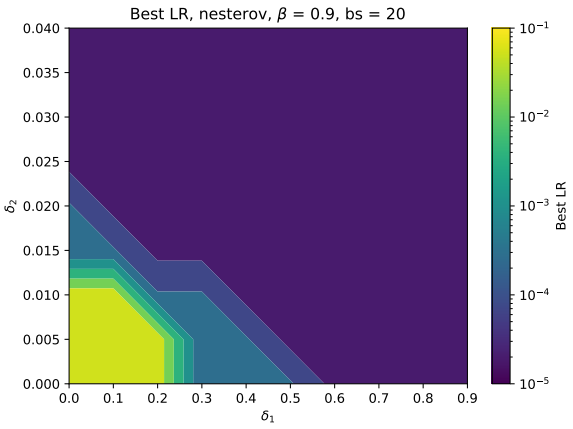


(d) Nesterov, $\beta = 0.99$

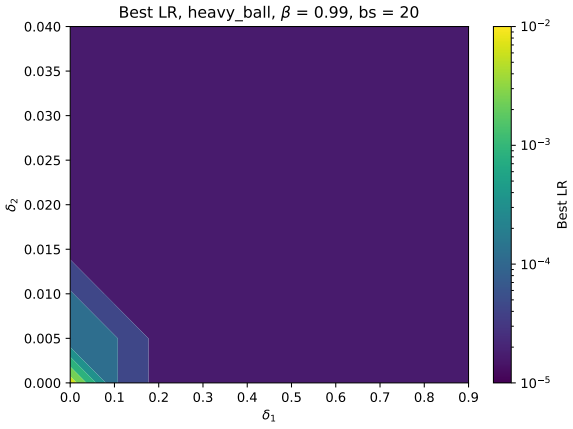
Figure 13: Optimal learning rate used to certify the convergence rates in Figure 6.



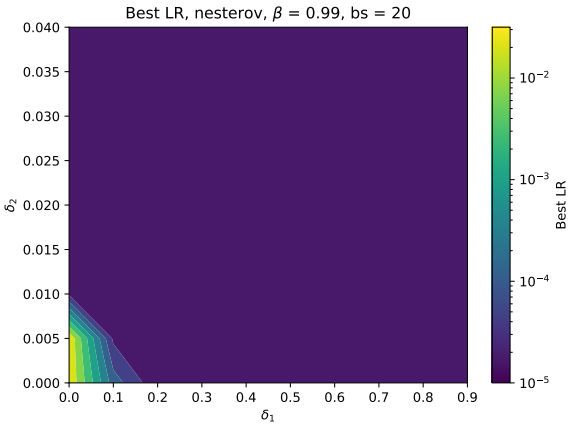
(a) Heavy-ball, $\beta = 0.9$



(b) Nesterov, $\beta = 0.9$



(c) Heavy-ball, $\beta = 0.99$



(d) Nesterov, $\beta = 0.99$

Figure 14: Optimal learning rate used to certify the convergence rates in Figure 12.