PORT–HAMILTONIAN GRADIENT FLOWS

Michael Poli[* 1,3], **Stefano Massaroli**[* 2,3]
Jinkyoo Park[1], **Atsushi Yamashita**[2], **Hajime Asama**[2],[1]
[1]Department of Industrial & Systems Engineering, KAIST, Daejeon, South Korea
[2]Department of Precision Engineering, The University of Tokyo, Tokyo, Japan
[3]DiffEqML
[*]Equal contribution authors
{massaroli, yamashita, asama}@robot.t.u-tokyo.ac.jp
{poli_m, jinkyoo.park}@kaist.ac.kr,

ABSTRACT

In this paper we present a general framework for continuous–time gradient descent, often referred to as *gradient flow*. We extend *Hamiltonian gradient flows*, which ascribe mechanical dynamics to neural network parameters and constitute a natural continuous–time alternative to discrete momentum–based gradient descent approaches. The proposed Port-Hamiltonian Gradient Flow (PHGF) casts neural network training into a system–theoretic framework: a fictitious physical system is coupled to the neural network by setting the loss function as an energy term of the system. As autonomous port–Hamiltonian systems naturally tend to dissipate energy towards one of its minima by construction, solving the system simultaneously trains the neural network. We show that general PHGFs are compatible with both continuous–time data–stream optimization, where the optimizer processes a continuous stream of data, as well as standard fixed–step optimization. In continuous–time, PHGFs allow for the embedding of black–box adaptive–step ODE solvers and are able to stick to the energy manifold, thus avoiding divergence due to large learning rates. In fixed–step optimization, on the other hand, PGHFs open the door to novel fixed–step approaches based on symplectic discretizations of the Port–Hamiltonian with similar memory footprint and computational complexity as momentum optimizers.

Notation $\mathbb{N}$ and $\mathbb{R}$ denote the sets of natural and real numbers. Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. $\nabla\mathcal{L}$ is its transposed gradient, i.e. $\nabla\mathcal{L} := (\frac{\partial}{\partial \mathbf{x}}\mathcal{L}(\mathbf{x}))^\top$.

1 INTRODUCTION

Similarly to how continuous neural architectures are built upon residual networks [1; 2], it is easy to observe that the gradient descent dynamics

$$\boldsymbol{\theta}_{s+1} = \boldsymbol{\theta}_s - \gamma\nabla\,\mathcal{L}_{\boldsymbol{\theta}_s} \tag{1}$$

resemble the Euler discretization of an ordinary differential equation

$$\dot{\boldsymbol{\theta}}(s) = -\nabla\,\mathcal{L}_{\boldsymbol{\theta}(s)} \tag{2}$$

commonly referred to as *gradient flow*. There exists a long line of work on gradient flow [3], both as a theoretical approach able to shine light on certain properties of neural networks [4] as well as a grounded approach to accelerate convergence [5]. This work is concerned with a dynamical system perspective of *gradient–flow*, in line with *Hamiltonian descent methods* [6]. The ideas proposed in this paper build upon [7]. We introduce *port–Hamiltonian gradient flows* (PHGF) and discuss leveraging the proposed method in a continuous time, data–stream optimization setting [8] as well as a regular discretized setting which allows for a symplectic discretization of the port–Hamiltonian.

2 PORT–HAMILTONIAN GRADIENT FLOW

2.1 PRELIMINARIES AND BASIC ASSUMPTIONS

Let $(\mathcal{T}, \geq)$ and $(\mathcal{S}, \geq)$ be linearly ordered sets, called respectively the data time and the parameter update time sets. Typically $\mathcal{T} = \mathcal{K} = \mathbb{N}$ or $\mathcal{T} = \mathcal{K} = \mathbb{R}$. Furthermore, let $(\mathcal{X}, \mathfrak{S})$ be a measure space. We consider the empirical risk minimization problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{x}\in\mathcal{X}}\left[\ell_{\boldsymbol{\theta}}(\mathbf{x})\right] \tag{3}$$

being $\ell : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$ a non–convex function and $\mathbf{x}$ some random variable whose distribution is known only through a stream of i.i.d. samples $\{\mathbf{x}_t\}_{t\in\mathcal{T}} \subset \mathcal{X}$. We assume $\ell_{(\cdot)}(\mathbf{x})$ and $\ell_{\boldsymbol{\theta}}(\cdot)$ to be respectively smooth enough and $\mathfrak{S}$–measurable such that $\frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{\mathbf{x}\in\mathcal{X}}\left[\ell_{\boldsymbol{\theta}}(\mathbf{x})\right] = \mathbb{E}_{\mathbf{x}\in\mathcal{X}}\left[\frac{\partial}{\partial\boldsymbol{\theta}}\ell_{\boldsymbol{\theta}}(\mathbf{x})\right]$. For the sake of compactness, from now on we write $\mathcal{L}_{\boldsymbol{\theta}} := \mathbb{E}_{\mathbf{x}\in\mathcal{X}}\left[\ell_{\boldsymbol{\theta}}(\mathbf{x})\right]$. We will first consider the non–stochastic case, i.e. $\mathcal{L}_\theta$ is knows.

2.2 Port–Hamiltonian Gradient Flow

Port–Hamiltonian systems represent a general modeling framework capturing dynamic phenomena as the explicit effect of some flow of energy within the system or with the external world [9; 10; 11]. In [7], port–Hamiltonian systems are exploited for neural network training. In a similar fashion, we can define a general–purpose optimizer, the *port–Hamiltonian gradient flow* (PHGF), in mechanical–like PH form, with realization

$$\frac{d}{ds}\begin{bmatrix}\boldsymbol{\theta}\\ \boldsymbol{\omega}\end{bmatrix} = \begin{bmatrix}0 & \mathbb{I}_d\\ -\mathbb{I}_d & -\mathbf{B}\end{bmatrix}\begin{bmatrix}\nabla_{\boldsymbol{\theta}}\mathcal{H}\\ \nabla_{\boldsymbol{\omega}}\mathcal{H}\end{bmatrix}, \quad \mathbf{B} = \mathbf{B}^\top > 0 \tag{4}$$

where $\boldsymbol{\omega}$ is introduced as generalized momentum, $\boldsymbol{\omega} := \mathbf{M}\dot{\boldsymbol{\theta}}$ ($\mathbf{M} = \mathbf{M}^\top \succ 0$), and $\mathcal{H}$ is obtained from $\mathcal{L}$ by adding a fictitious *kinetic energy* term, i.e.

$$\mathcal{H} := \mathcal{L}_{\boldsymbol{\theta}} + \frac{1}{2}\boldsymbol{\omega}^\top\mathbf{M}^{-1}\boldsymbol{\omega}$$

It holds,

$$\frac{d}{ds}\mathcal{H} = -\nabla_{\boldsymbol{\omega}}^\top\mathcal{H}\,\mathbf{B}\nabla_{\boldsymbol{\omega}}\mathcal{H} = -\boldsymbol{\omega}^\top\mathbf{M}^{-1}\mathbf{B}\mathbf{M}^{-1}\boldsymbol{\omega} \le 0 \tag{5}$$

and, indeed, $\frac{d}{ds}\mathcal{H} = 0 \Leftrightarrow \boldsymbol{\omega} = 0$. Therefore, if $\mathcal{L}$ is bounded from below, the system will dissipate all its *energy* and $\boldsymbol{\theta}$ will converge to a local minimizer $\boldsymbol{\theta}^*$ of $\mathcal{L}_{\boldsymbol{\theta}}$ (with null momentum), i.e.

$$\nabla\mathcal{L}_{\boldsymbol{\theta}^*} = 0,\ \nabla^2\mathcal{L}_{\boldsymbol{\theta}^*} > 0,\ \boldsymbol{\theta}^* = \lim_{s\to\infty}\boldsymbol{\theta}(s)$$

In practice, the optimal value is obtained by integrating the ODE "long enough".

Note that, when training neural networks on finite datasets (i.e. $\mathcal{T}$ is a finite set, $|\mathcal{T}| = m$), the above result corresponds to the full–batch case: $\nabla_{\boldsymbol{\theta}}\mathcal{H} = \frac{1}{m}\sum_{t=0}^{m}\nabla_{\boldsymbol{\theta}}\ell_{\boldsymbol{\theta}}(\mathbf{x}_t)$.

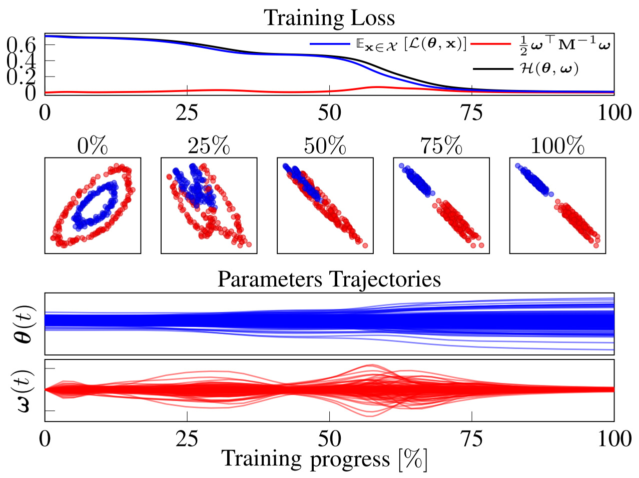

Figure 1: Nonlinear classification problem performed by a neural network trained with PHGF. As the energy is dissipated, the parameters converge with null momenta and the network is able to separate the data features.

On centennial adaptation rules: black–box ODE solvers Adaptive optimizers such as *RMSProp* [12], *Adam* [13] and *RAdam* [14] rely on handcrafted heuristics aimed at accelerating convergence by a rescaling of *per–parameter* learning rates. However, from a dynamical system perspective, iterating different adaptive methods can be seen as solving the ODE (2) with some *adaptive–step* numerical differential equation solver. So *why not exploit over one hundred years of results in the solution of differential equations to seek adaptation rules*?

Coming back to the port–Hamiltonian optimizer in differential form (4), it can be noticed how the *power balance equation* (5) provides the rate of decay (or, more technically, *dissipation*) of the Hamiltonian function $\mathcal{H}$. Thus, in a interval $[0, s]$, the net decrement of $\mathcal{H}$ is

$$\mathcal{H}(s) - \mathcal{H}(0) = -\int_0^s \boldsymbol{\omega}(\tau)^\top\mathbf{M}^{-1}\mathbf{B}\mathbf{M}^{-1}\boldsymbol{\omega}(\tau)d\tau \tag{6}$$

While we have no control on $\mathcal{H}(0)$, which solely depends on how $\boldsymbol{\theta}$ is initialized, by choosing $\mathbf{M}$ and $\mathbf{B}$, we can accurately control how the system dissipates energy, i.e. the rate of convergence towards a local minimizer of $\mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta},\mathbf{x})\right]$. In practice, when the ODE is solved with numerical schemes, the properties of convergence and control of the energy decay rate can be preserved out–of–the–box. To achieve this, fixed–step discretizations require a sufficiently small step size or *learning rate*, as is the case for SGD, whereas adaptive–step solvers can be used directly in conjunction with higher learning rates than possible in the fixed–step setting.

Going stochastic: gradient flow with port–Hamiltonian systems In a stochastic setting the data and parameter updates time coincide, i.e. $s \equiv t$. In this case, at each instant, only the gradient $\nabla \ell_{\boldsymbol{\theta}(t)}(\mathbf{x}_t)$ is available to the optimizer, i.e. an estimate of $\mathbb{E}_{\mathbf{x}\in\mathcal{X}}\left[\nabla_{\boldsymbol{\theta}} \ell_{\boldsymbol{\theta}(t)}(\mathbf{x})\right]$. Note that, in general, the above results on convergence translate (in expectation) to this stochastic case only if $\nabla_{\boldsymbol{\theta}} \ell_{\boldsymbol{\theta}(t)}(\mathbf{x}_t)$ is an unbiased estimate of the gradient at each instant t [15]. This can be achieved by uniform i.i.d data sampling as in discrete stochastic gradient methods. From a dynamical systems perspective, in fact, this is justified by following the results on continuous versions of gradient descent [3; 16]. Let $\boldsymbol{\chi} := (\boldsymbol{\theta}, \boldsymbol{\omega})$. In the full stochastic setting, symmetrically to [3], a port–Hamiltonian gradient flow can be represented symbolically by *stochastic differential equation* (SDE) in the form

$$d\boldsymbol{\chi} = \mathbf{F}\nabla_{\boldsymbol{\chi}} \mathcal{H}(\boldsymbol{\chi})dt + \mathbf{g}(\boldsymbol{\chi})dw$$

where $w(t)$ denotes a multivariate Wiener process, $\mathbf{F}$ is the system matrix in (4) and $\mathbf{g} : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ is appropriately chosen function. Then, under mild assumptions [17; 8], passivity and thus convergence in expectation of the SDE above is guaranteed once again by the port–Hamiltonian structure.

You don't always need to adapt: symplectic port–Hamiltonian discretizations In practice, while solving (4) in the stochastic setting, traditional black–box adaptive ODE solvers require either significant computation and low tolerances due to the stiffness caused by the data stream. Similarly, SDE solvers cannot be employed directly since $\nabla_{\boldsymbol{\chi}} \mathcal{H}(\boldsymbol{\chi})$ and the true form of $\mathbf{g}(\boldsymbol{\chi})$ are not available.

Can we reduce the computational burden without sacrificing convergence?

A simple possible solution is exploiting the PH structure of the optimizer and adopt *symplectic integrators*, which preserve energetic properties of the discretized model (e.g. energy balance) [18; 19].

2.3 SYMPLECTIC DISCRETIZED PHGF

The PHGF framework is compatible with standard fixed step discretizations of gradient flow methods. Here, the port–Hamiltonian formulation naturally leads to energy–preserving, symplectic discretizations. The full derivation of various symplectic PHGF schemes are reported in the appendix.

Definition 1 *Störmer–Verlet Port–Hamiltonian Gradient Flow (SV–PHGF)*

Assume to be given by an oracle, for all $t \in \mathcal{T}$, the gradient $\nabla_{\boldsymbol{\theta}} \ell_{\boldsymbol{\theta}_t}(\mathbf{x}_t)$. The SV–PHGF iterative update rule is then given by

$$\begin{aligned} \boldsymbol{\omega}_{t+1} &= \frac{2-\gamma}{2+\gamma\beta}\boldsymbol{\omega}_t - \frac{2\gamma}{2+\gamma\beta}\nabla_{\boldsymbol{\theta}} \ell_{\boldsymbol{\theta}_t}(\mathbf{x}_t) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \gamma \mathbf{M}^{-1}\boldsymbol{\omega}_{t+1} \end{aligned} \tag{7}$$

SV–PHGF leverages separability by construction of the generalized energy $\mathcal{H}_t := \ell_{\boldsymbol{\theta}_t}(\mathbf{x}_t) + \frac{1}{2}\boldsymbol{\omega}_t^\top \mathbf{M}^{-1}\boldsymbol{\omega}_t$ to adapt the Störmer–Verlet [20] implicit second order method. More specifically, separability allows for an explicit implementation of the Störmer–Verlet method that eliminates *leapfrogging* and requires a single backward call per update step. We consider a simple choice of the dissipation matrix $\mathbf{B} := \beta \mathbb{I}_d$ and identity inertia matrix $\mathbf{M}$, though the framework allows for different designs to speed convergence or introduce coupling terms between parameters.

3 EXPERIMENTS

We evaluate PHGF in a continuous data–stream setting as well as a regular discretized setting. We include more details on the experimental setup as supplementary material.

Symplectic PHGF We follow the setup of [14] and perform a comparison of Adam [13], RAdam [14], and SV–PHGF on MNIST. More specifically, we include two versions for both Adam and RAdam: the first using default hyperparameters, i.e. learning rate $lr = 10^{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, whereas the second with optimized variants obtained through a 500 run Optuna [21] randomized hyperparameter search for both learning rates and betas. For a clear comparison of the intrinsic properties of different approaches, no learning rate scheduling is introduced. Fig. 4 shows mean results for 10 runs. The symplectic nature of SV–PHGF allows it to accurately track the energy dissipation and leads to no oscillations in the loss at convergence.

Hybrid data–stream A different approach involves utilizing continuous PHGFs alongside batch training by integrating (4) for a finite time after each batch is sampled. In this setting, the *learning rate* assumes the role of *per–step integration time* i.e. integration interval for the ODE (4) defining the PHGF for a single sample or batch of the dataset. Fig. 2 shows the MNIST test accuracy of a 1 million parameter, 2 layer fully–connected neural network after 1 epoch of training with PHGFs coupled with the *Dormand–Prince* [22] solver. With sufficiently large β, the proposed method is generally robust to values of the learning rate that would result in instability and divergence for standard discrete gradient–based optimizers. This property allows for two distinct modes of operation for the PHGF: with long integration intervals the ODE always reaches a local minimizer for the energy calculated with a single batch or sample, which in turn promotes a parameter state space trajectory that moves from one energy local minima to another. On the other hand, short integration yields parameter dynamics more similar to those of momentum SGD.

4 Discussion

Deep learning optimizers as discretizations of PHGFs Continuous versions of commonly used deep learning optimizers such as Adam [23] can be adapted to the Port–Hamiltonian formulation and are therefore Port–Hamiltonian Gradient Flows. As a result, the proposed formalism can be leveraged to either inspire new optimization approaches equipped with implicit convergence guarantees, or to offer improvements to existing solutions.

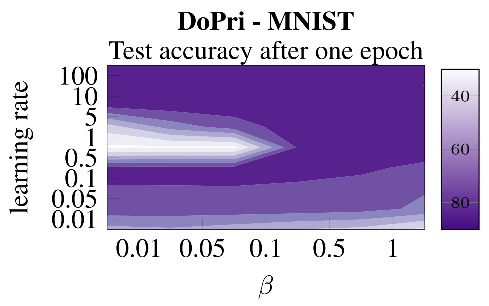

Figure 2: MNIST test accuracy after a single epoch of hybrid PHGF training. The method is robust to a wide range of learning rates.

On the design of inertia matrix and choice of ODE solvers Depending on the choice of ODE solver, PHGF yields different optimizers with no changes required to the formulation. Moreover, the design of ad–hoc inertia matrices $\mathbf{M}$ allow for the introduction of interconnections between parameters via coupling terms, which can affect convergence dynamics.

5 Conclusion

We introduce Port–Hamiltonian Gradient Flows (PHGF), a novel optimization framework inspired by the port–Hamiltonian framework equipped with implicit convergence guarantees. In continuous data–stream optimization settings, PHGFs can be coupled with black–box ODE solvers to yield different optimizers. Moreover, PHGFs are naturally compatible with computationally efficient fixed–step symplectic discretizations.

References

[1] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[2] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[3] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[4] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[5] Bin Shi, Simon S Du, Weijie J Su, and Michael I Jordan. Acceleration via symplectic discretization of high-resolution differential equations. *arXiv preprint arXiv:1902.03694*, 2019.

[6] Chris J Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.

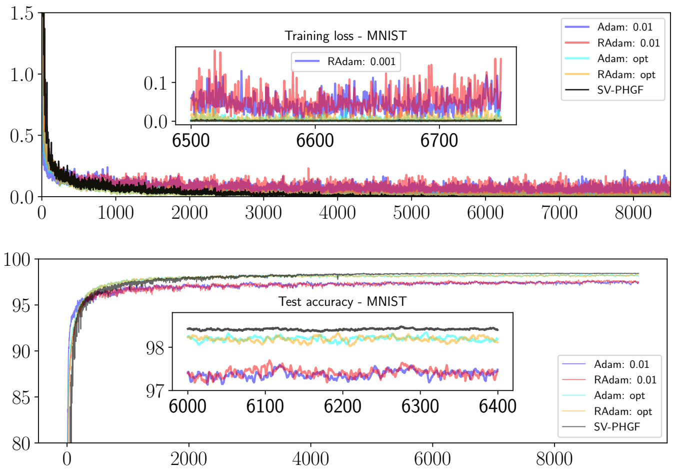

Figure 3: Training loss and test accuracy of different optimizers on MNIST. Mean results across 10 experiments. The SV–PHGF dissipates energy throughout training, reducing oscillations of the loss.

[7] Stefano Massaroli, Michael Poli, Federico Califano, Angela Faragasso, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Port-hamiltonian approach to neural network training. *arXiv preprint arXiv:1909.02702*, 2019.

[8] Francesco Cordoni, Luca Di Persio, and Riccardo Muradore. Stochastic port–hamiltonian systems. *arXiv preprint arXiv:1910.01901*, 2019.

[9] Romeo Ortega, Arjan J Van Der Schaft, Iven Mareels, and Bernhard Maschke. Putting energy back in control. *IEEE Control Systems Magazine*, 21(2):18–33, 2001.

[10] Romeo Ortega, Arjan Van Der Schaft, Fernando Castanos, and Alessandro Astolfi. Control by interconnection and standard passivity-based control of port-hamiltonian systems. *IEEE Transactions on Automatic control*, 53(11):2527–2542, 2008.

[11] Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx. *Modeling and control of complex physical systems: the port-Hamiltonian approach*. Springer Science & Business Media, 2009.

[12] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[15] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[16] Justin Sirignano and Konstantinos Spiliopoulos. Stochastic gradient descent in continuous time. *SIAM Journal on Financial Mathematics*, 8(1):933–961, 2017.

[17] Satoshi Satoh and Kenji Fujimoto. Passivity based control of stochastic port-hamiltonian systems. *IEEE Transactions on Automatic Control*, 58(5):1139–1153, 2012.

[18] Elena Celledoni and Eirik Hoel Høiseth. Energy-preserving and passivity-consistent numerical discretization of port-hamiltonian systems. *arXiv preprint arXiv:1706.08621*, 2017.

[19] Paul Kotyczka and Laurent Lefèvre. Discrete-time port-hamiltonian systems: A definition based on symplectic integration. *Systems & Control Letters*, 133:104530, 2019.

[20] Loup Verlet. Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967.

[21] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[22] Peter J Prince and John R Dormand. High order embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75, 1981.

[23] Anas Barakat and Pascal Bianchi. Convergence and dynamical behavior of the adam algorithm for non convex stochastic optimization. *arXiv preprint arXiv:1810.02263*, 2018.

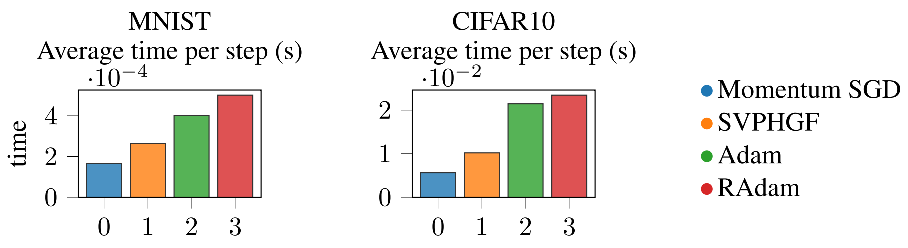

Figure 4: Time per optimization step on MNIST and CIFAR10.

A Appendix

A.1 Computational efficiency of symplectic PHGF

We perform a comparison of the *time per step* of different optimizers: SGD with momentum, SV–PHGF, Adam and RAdam on both MNIST and CIFAR10. All methods are tasked with optimizing the same architecture: a 2 layer fully–connected neural network with 1 million parameters for MNIST and a ResNet 56 with 1 million parameters for CIFAR10. As shown in Figure 4, SV–PHGF is computationally more efficient than both Adam and RAdam. As MNIST was trained with batch size 128 and CIFAR10 256, the result shows how the difference between optimizers remains significant for larger batches.