An optimal control framework for adaptive neural ODEs

Joubine Aghili, Olga Mula

An optimal control framework for adaptive neural ODEs

Joubine Aghili[1] and Olga Mula[2]

[1]IRMA UMR 7501, Université de Strasbourg, CNRS, F-67000 Strasbourg, France.

[2]Technical University of Eindhoven, Den Dolech 2, P.O. Box 513, 5600 Eindhoven, Netherlands.

Abstract

In recent years, the notion of neural ODEs has connected deep learning with the field of ODEs and optimal control. In this setting, neural networks are defined as solutions of a given ODE which is solved with a certain time discretization. The learning task consists in finding the ODE parameters as the optimal values of a sampled loss minimization problem. In the limit of infinite time steps, and data samples, we obtain a notion of continuous formulation of the problem. The practical implementation involves two discretization errors: a sampling error, and a time-discretization error. In this work, we develop a general optimal control framework to analyse the interplay between the above two errors. We prove that to approximate the solution of the fully continuous problem at a certain accuracy, we not only need a minimal number of training samples, but we also need to solve the control problem on the sampled loss function with some minimal accuracy. The theoretical analysis allows us to develop rigorous adaptive schemes in time and sampling, and give rise to a notion of adaptive neural ODEs. The performance of the approach is illustrated in several numerical examples.

1 Introduction

1.1 Context

Neural networks produce structured parametric families of functions that have been studied for at least 70 years (see [1, 2]). It is however only in the last

decade that their popularity has surged. Thanks to the increase of computing power, and the development of easy-to-use computing tools for optimization and automatic differentiation, statistical learning of neural networks has produced state-of-the-art performance in a large variety of machine learning problems, from computer vision [3] (e.g. self-driving cars, X-rays diagnosis,...) to natural language processing [4] (e.g. Google translate, DeepL Translator,...) and reinforcement learning (e.g. superhuman performance at Go [5]). Despite this great empirical success, neural networks are not entirely well-understood and there is a pressing need to provide:

- solid *mathematical foundations* to understand their approximation power and why they may outperform other classes of functions,
- *algorithms* to find systematically an appropriate architecture for each problem and to train the network in order to deliver in practice the approximation capabilities predicted by the theory, and to guarantee robustness over generalization errors.

On the first point, significant advances have been made recently on the approximation power of neural networks (see, e.g., [6–9] for a selection of rigorous results). In particular, a few recent works have given theoretical evidence on the advantages of using deep versus shallow architectures for the approximation of certain relevant families of functions (see, e.g., [7, 10]). However, so far the obtained results do not seem to be informative on how to address the second point above, that is, how to build algorithms that discover automatically the right architecture for each problem, and that may allow to benefit from the theoretically high approximation properties of (deep) neural networks. Even though there are many contributions using evolutionary algorithms, e.g. [11, 12], the theoretical foundations of adaptive neural networks remains unclear. As a result, their training remains a key open issue subject to very active research. The present work is a contribution in this direction from the perspective of neural ODEs [13].

The most commonly applied training methods are based on the stochastic gradient descent ([14, 15]) and its variants (see, e.g., [16, 17]). It has the advantage of being easily implementable but it is difficult to tune properly in practice, especially in problems involving large data sets and using deep neural networks with many layers and coefficients which are very prone to over-parametrization issues. The difficulty in training deep networks raises the question regarding the benefits of applying a *shallow-to-deep adaptive strategy* in the network architecture in order perform the training in a more robust and rapid manner. The underlying intuition is that shallow networks give poor approximation properties but converge faster than deeper networks which, on the contrary, have higher approximation power.

There is a large body of emerging works which explore numerically the potential of training deep neural networks with shallow-to-deep strategies (see, e.g., [18–23]). In this work, we present an abstract setting and convergence results which should be understood as one possible theoretical justification for this type of adaptive approach. In order to understand the potential gain, it

is necessary to define a notion of the continuous underlying objects that are approximated. This is the reason why we have chosen to work from a perspective which connects the task of learning with neural ODEs with optimal control and dynamical ODE systems. In this view, there is a notion of a continuous neural network which is described by a continuous time-dependent ODE system. Its discretization fixes its architecture. For instance, ResNet (see [24]) can be regarded as an Explicit Euler scheme for the solution of certain dynamics. The weights involved in the ODE are seen as controls and correspond to the hyperparameters of the network which have to be optimized. The process of learning these parameters can then be recast as an optimal control problem over the admissible controls where the cost function is the empirical risk (sometimes with an extra regularization term). Necessary optimality conditions are then easily formulated through the Pontryagin Maximum Principle (PMP, [25, 26]).

1.2 Contribution of the present work

The optimal control point of view is appealing since it gives access to a fully continuous description of neural networks and of the underlying statistical learning process. The practical implementation involves two discretization errors:

1. **Sampling error:** The cost function of the fully continuous optimal control problem is the average of a loss function, sometimes also called risk. However, in practice, the average loss is replaced by an empirical mean which is built from N samples. So it is necessary to assess by how much the minimum of the sampled mean deviates from the continuous underlying problem. We analyze this point in Theorem 7.2.
2. **Time-integration error:** To solve the optimal control problem on the empirical mean of the loss, one solves the resulting PMP with Newton-type iterations involving the solution of forward and backward dynamical systems with certain time integration schemes (which fix the neural network architecture). The usual approach is to first fix a discretization and then solve the discrete version of the PMP. However, the discretization errors accumulate at each iteration and make us deviate from the time-continuous version. It is thus necessary to examine how much the final output deviates from the time-continuous one.

In this context, the main contributions of the present work are:

1. **Derivation of a full error analysis:**
   - We make a theoretical analysis on the interplay between the above two errors. The impact of the sampling error is analyzed in Theorem 7.2 and the discretization error in Theorem 6.2. From these results, it follows that to approximate the minimum of the underlying fully continuous problem at a certain accuracy, we not only need a minimal number of training samples, but we also need to solve the control problem on the empirical risk with some minimal accuracy. This requires, in turn, to use time-adaptive techniques. In our analysis,

both errors are additive and can actually be optimally balanced. The final result is given in Corolary 7.3 and it is formulated in probability.
- To solve the control problem on the empirical risk at a given target accuracy, we show that we have to compute the forward and backward propagations at each step of the Newton-type algorithms at increasing accuracy. As our theoretical proof of Theorem 6.2 will illustrate, tightening the accuracy of the time integration is necessary to guarantee convergence to the exact continuous problem and justifies the coarse-to-fine, resp. shallow-to-deep, strategy. This yields neural networks with depths that are automatically adapted.

2. **Practical coarse-to-fine adaptive algorithm:**
   - We next use the theory to derive actionable criteria to build time-integration schemes that are adaptively refined across the iterations. This has the advantage of reducing computational cost since early iterations use looser tolerances, thus avoiding unnecessary work due to oversolving, while later iterations use tighter tolerances to deliver accuracy. The coarse-to-fine strategy could also be understood as a dynamic regularization which helps to prevent over-parametrization.
   - We illustrate the behavior of the algorithm in several benchmark examples.

1.3 Potential impact, limitations and extensions

We believe the results of this article can contribute to the following key topics in deep learning:
- Automate the selection of neural network architecture during the training,
- Enhance the interpretability of the generalization errors by connecting them with sampling and discretization errors,
- Provide a first theoretical justification of the works adopting the principle of shallow-to-deep training of deep neural networks.
- Provide a theoretical framework for neural ODE strategies optimizing time-steps with a fixed number of layers.

1.4 Related works

To the best of our knowledge, the connection between deep learning, dynamical systems and optimal control is not new. It can be traced back at least to the works of LeCun and Pineda in the 1980's, in which the idea of back-propagation is connected to the adjoint variable arising in optimal control (see [27, 28]). The approach has gained a lot of attention in recent years following works on neural ODEs such as [13, 29, 30]. We refer to these works, and also to [31–34] for some selected references.

In the field of neural ODEs, there are by now many experimental studies exploring the effects of different time-stepping techniques (see e.g. [35, 36]), and their effects on stability and robustness (see, e.g., [30, 37]). The topic of adaptivity has also been explored. In works such as [13, 38, 39], high order

adaptive methods which automatically change the time step by estimating local truncation error have been developed . We also refer to [32] where the time step sizes are included as an additional parameter to be optimized. In all these works the number of layers is fixed during the training iterations. This idea is part of our work but we also understand adaptivity as the idea of increasing progressively the number of steps. In fact, given the present state of the art, our contribution should be undertsood as an abstract, theoretical framework that rigorously justifies convergence to the continuous problem, rather than a very concrete numerical algorithm. Our numerical implementation does not provide state of the art results but it is intended to reproduce as faithfully as possible the theoretical setting.

1.5 Outline of the paper

In section 2, we define precisely what we understand by deep learning and we recall its connection with optimal control. Section 3 gives the optimal control setting which we use in our subsequent developments. In particular, we formulate the fully continuous optimal control problem over the expectation of the loss and its sampled version involving an empirical mean of the loss. Section 4 recalls the Pontryagin's Maximum Principle for the sampled problem. We also recall in this section the Extended Method of Successive Approximations (E-MSA), originally introduced in [31], to solve the sampled problem. The algorithm is iterative, and it requires to compute at each iteration a forward and a backward time propagation followed by a certain maximization over the control variables. In its original formulation, it is assumed that these steps are performed exactly. In Section 5, we formulate our main algorithm, which is an inexact version of the E-MSA that we call Adaptive MSA (A-MSA). It is based on approximately realizing each propagation and maximization within a certain accuracy. To ensure convergence to the exact solution, the accuracy needs to be tightened across the iterations. In section 6, we prove that A-MSA converges to a time-continuous solution of the sampled PMP. In section 7, we connect the solution of the sampled problem with the continuous one involving the exact average of the loss function. For this, we use results obtained in [40]. This last step allows to connect how much the solution of A-MSA deviates from the fully continuous one in terms of the number of samples and the accuracy of the discretization. In section 8, we give guidelines to implement A-MSA in practice, and section 9 illustrates the performance of the algorithm in numerical examples. We conclude the paper in Section 10.

2 From deep learning to optimal control

2.1 Statistical Learning Problems

We consider the following *statistical learning problem*: Assume that we are given a *domain set* $X \subseteq \mathbb{R}^n$ and a *label set* $Y \subseteq \mathbb{R}^k$, with $n, k \in \mathbb{N}$. Further assume that there exists an unknown probability distribution μ on $X \times Y$.

Given a loss function $\mathcal{L} : Y \times Y \to \mathbb{R}^+$, the goal of the statistical learning problem is to find a function $v : X \to Y$, which we will call *prediction rule*, from a hypothesis class $\mathcal{V} \subset \{v : X \to Y\}$ such that the expected loss

$$\mathcal{J}(v) \coloneqq \mathbb{E}_{(x,y)\sim\mu}\mathcal{L}(v(x), y)$$

is minimized over $\mathcal{V}$. In other words, the task is to find

$$v^* \in \arg\min_{v\in\mathcal{V}} \mathcal{J}(v).$$

In general, the probability distribution μ is unknown and we are only given a set $\mathcal{S}_N$ of $N \in \mathbb{N}$ training samples

$$\mathcal{S}_N \coloneqq \{(x_i, y_i)\}_{i=1}^N.$$

The most common choice is then to consider the uniform distribution $\mu_N : X \times Y \to \mathbb{R}^+$,

$$\mu_N(x, y) = \frac{1}{N}\sum_{i=1}^N \delta_{(x_i,y_i)}(x, y)$$

which yields the so-called *empirical loss*

$$\mathcal{J}_N(v) \coloneqq \mathbb{E}_{(x,y)\sim\mu_N}\mathcal{L}(v(x), y) = \frac{1}{N}\sum_{i=1}^N \mathcal{L}(v(x_i), y_i)$$

and one finds v_N, an approximation of v, by minimizing over it,

$$v_N \in \arg\min_{v\in\mathcal{V}} \mathcal{J}_N(v). \tag{2.1}$$

In the following, the above optimization step will be called the *learning procedure*.

2.2 Neural Network Architectures and Deep Learning

While there exists numerous different architectures, we could simplify the discussion and say that neural networks are a class of functions with the basic general form

$$v : \mathbb{R}^n \to \mathbb{R}^k, \quad x \mapsto W_L\sigma(W_{L-1}\sigma(\dots(\sigma(W_1(x))))), \tag{2.2}$$

where:

- $L \in \mathbb{N}$ is the *depth*, i.e., the number of layers of the neural network,
- $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$ are affine maps for $\ell = 1, \dots, L$. For consistency, we must set $N_0 = n$ and $N_L = k$ but the rest of the dimensions $N_\ell \in \mathbb{N}$ can be freely chosen. We have for all $x \in \mathbb{R}^{N_{\ell-1}}$, $W_\ell(x) = A_\ell(x) + b_\ell$ for a matrix $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and a vector $b_\ell \in \mathbb{R}^{N_\ell}$.

- $\sigma : \mathbb{R} \to \mathbb{R}$ is a *nonlinear activation function* which is applied coordinate-wise in (2.2). Some popular choices are the ReLU function, $\sigma(x) = \max(0, x)$, or the hyperbolic tangent, $\sigma(x) = \tanh(x)$.

Deep learning describes the range of learning procedures to solve statistical learning problems where the hypothesis class $\mathcal{V}$ is taken to be a set of neural networks. One usually works with the class of neural networks with given depth L, activation function σ and fixed dimensions $N_1, \dots, N_{L-1}$ for the affine mappings,

$$\mathcal{NN}(L, \sigma, N_1, \dots, N_{L-1}) := \{v : X \to Y \ : \ v(x) = W_L \sigma(W_{L-1} \sigma(\dots(\sigma(W_1(x))))\}.$$

For this class, the task is to solve the empirical risk problem (2.1) associated to it.

2.3 Continuous Formulation as an Optimal Control Problem

Functions of the type (2.2) can be built by repeated composition of parametrizable functions

$$\phi_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, \quad x \mapsto \phi_\ell(x) = \sigma(W_\ell(x)), \quad i = 1, \dots, L-1,$$

followed by an affine step, that is,

$$v = W_L \circ \phi_{L-1} \circ \cdots \circ \phi_1, \quad \forall v \in \mathcal{NN}(L, \sigma, N_1, \dots, N_{L-1}).$$

This simple observation yields to numerous other possible architectures. The most relevant instance triggering the connection with optimal control are the so-called Residual Neural Networks (ResNet, [24]). They correspond to the choice

$$\phi_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, \quad x \mapsto \phi_\ell(x) = x + h\sigma(W_\ell(x)), \quad i = 1, \dots, L-1,$$

for a certain parameter[1] $h > 0$.

If we now set $N_{L-1} = \cdots = N_1 = N_0 = n$, the application of ϕ_ℓ can be interpreted as an Explicit Euler step from time $t_{\ell-1} = (\ell - 1)h$ to time $t_\ell = t_{\ell-1} + h$ of the dynamics

$$\dot{x}_t = f(x_t, \theta_t), \quad \text{with } x_0 = x,$$

where

$$f(x_t, \theta_t) = \sigma(W_t(x_t)), \quad \text{with } W_t(x) = A_t x + b_t,$$

[1] In fact, ResNets were originally defined for $h = 1$. The "augmented" definition with the paramater h was introduced in [30].

Network Architecture	Associated ODE discretization
ResNet [24], RevNet [42]	Forward Euler
Polynet [43]	Approximation of Backward Euler
FractalNet [44]	Runge-Kutta order 2

Table 1: Some neural network classes and their associated ODE discretization.

and θ_t gathers the parameters upon which the dynamics depend. In our case,

$$\theta_t = (A_t, b_t).$$

Note that the matrices A and b are now time-dependent, hence the t-subscript.

In this view, ResNet functions can be interpreted as the output of performing $L-1$ time steps of size h of the above dynamics, followed by an affine operation $W_L : \mathbb{R}^n \to \mathbb{R}^k$. This last operation is important since it maps the final output from the domain set in $\mathbb{R}^n$ to the label set in $\mathbb{R}^k$. In fact, it can be replaced by any linear or nonlinear mapping $g : \mathbb{R}^n \to \mathbb{R}^k$. Note also that h can be included as a parameter of the model class and that one can optimize over it as in [32].

Following similar lines, one can view certain classes of neural networks as discretizations of an underlying continuous dynamical system. Table 1 lists some popular classes and the associated numerical scheme (see, e.g., [41] for more details on these connections). We emphasize however that the connection with optimal control comes at the price of imposing that the input and intra-layer dimensions are equal ($N_{L-1} = \cdots = N_1 = N_0$), thus it cannot be made for all classes of neural networks.

3 Optimal control setting

We next define the notation for the optimal control setting which we will use in the rest of the paper. The notation is kept as consistent as possible with section 2 in order to further enhance the similarity between the usual deep learning approach and the present one.

Like in section 2, we assume that we are given a *domain set* $X \subset \mathbb{R}^n$ and a *label set* $Y \subset \mathbb{R}^k$, with $n, k \in \mathbb{N}$, and that there exists an unknown probability distribution μ on $X \times Y$ representing the distribution of the input-target pairs (x, y). Consider now a set of admissible controls or training weights $\Theta \subseteq \mathbb{R}^m$. Usually, we set $\Theta = \mathbb{R}^m$ in deep learning but the present methodology allows to consider constraints we will denote $\Theta \subseteq [U_{\min}, U_{\max}]^m$. Fix $T > 0$ and let f (feed-forward dynamics), Φ (terminal loss function) and R (regularizer) be functions

$$f : \mathbb{R}^n \times \Theta \to \mathbb{R}^n, \quad \Phi : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}, \quad R : \Theta \to \mathbb{R}.$$

Let $L^\infty([0,T], \Theta)$ be the set of essentially bounded measurable controls. In the following, we use bold-faced letters for path-time quantities. For example,

$\boldsymbol{\theta} \coloneqq \{\theta_t : 0 \leq t \leq T\}$ for any $\boldsymbol{\theta} \in L^\infty([0,T],\Theta)$. For every control $\boldsymbol{\theta} \in L^\infty([0,T],\Theta)$ and every value of the random variable $x \in X$, we define the state dynamics $\boldsymbol{u}^{\boldsymbol{\theta},x} \coloneqq \{u_t^{\boldsymbol{\theta},x} : 0 \leq t \leq T\}$ as the solution to the ordinary differential equation (ODE),

$$\begin{cases} \dot{u}_t^{\boldsymbol{\theta},x} &= f(u_t^{\boldsymbol{\theta},x}, \theta_t), \quad \forall t \in (0,T) \\ u_0^{\boldsymbol{\theta},x} &= x \end{cases} \tag{3.1}$$

The ODE is stochastic and its only source of randomness is the initial condition x.

With this notation, the deep learning optimization problem can be posed as the optimal control problem of finding

$$J^* = \inf_{\boldsymbol{\theta} \in L^\infty([0,T],\Theta)} \mathcal{J}(\boldsymbol{\theta}), \quad \text{subject to (3.1)}, \tag{3.2}$$

where

$$\mathcal{J}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{(x,y)\sim\mu}\left[\text{Loss}(x,y,\boldsymbol{\theta})\right],$$

and for any input-target pair $(x,y) \in \mathbb{R}^n \times \mathbb{R}^k$, the loss function is defined as

$$\text{Loss}(x,y,\boldsymbol{\theta}) \coloneqq \Phi(u_T^{\boldsymbol{\theta},x}, y) + \int_0^T R(\theta_t)\mathrm{d}t$$

Note that the regularizer R could in general also depend on the state u and Φ is the actual loss function upon which we want to act. It plays the same role as the loss function $\mathcal{L}$ of section 2.1. We can relate them by introducing the mapping $g : \mathbb{R}^n \to \mathbb{R}^k$ from section 2.3, and setting

$$\Phi(u_T^{\boldsymbol{\theta},x}, y) = \mathcal{L}(g(u_T^{\boldsymbol{\theta},x}), y).$$

Like in the setting of section 2.1, we are only given a set of N samples $\mathcal{S}_N = \{(x_i,y_i)\}_{i=1}^N$ and the probability distribution μ is not exactly known. For a given control $\boldsymbol{\theta} \in L^\infty([0,T],\Theta)$, each sample follows the dynamics

$$\begin{cases} \dot{u}_t^{\boldsymbol{\theta},i} &= f(u_t^{\boldsymbol{\theta},i}, \theta_t), \quad \forall t \in (0,T) \\ u_0^{\boldsymbol{\theta},i} &= x_i \end{cases} \tag{3.3}$$

for $i = 1,\ldots,N$. We then perform empirical risk minimization taking, e.g., the uniform distribution $\mu_N = \frac{1}{N}\sum_{i=1}^N \delta_{(x_i,y_i)}$. The optimal control problem becomes

$$J^*_{\mathcal{S}_N} = \inf_{\boldsymbol{\theta} \in L^\infty([0,T],\Theta)} \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}), \quad \text{subject to (3.3)}, \tag{3.4}$$

where

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{(x,y)\sim\mu_N}\mathrm{Loss}(x,y,\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Phi(u_T^{\boldsymbol{\theta},i},y_i) + \int_0^T R(\theta_t)\mathrm{d}t$$

Note that the solutions of the sampled optimal control problem (3.4) depend on the sample set $\mathcal{S}_N$ and are therefore random variables. However, for $\mathcal{S}_N$ fixed, problem (3.4) is deterministic and can thus be solved with deterministic optimal control techniques. In what follows, we adopt this viewpoint first to solve problem (3.4) at any target accuracy. By this we mean the following: for any given target accuracy $\varepsilon > 0$, we prove in Theorem 6.2 that it is possible to numerically compute a control $\boldsymbol{\theta}_\varepsilon$ such that

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}_\varepsilon) - J^*_{\mathcal{S}_N} \leq \varepsilon.$$

We then come back to the probabilistic point of view and prove in Corollary 7.3 that, provided that the number N of samples is sufficiently large, there exists a constant $C > 0$ such that

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}_\varepsilon) - J^* \leq C\varepsilon.$$

with high probabillity. The derivation of the latter bound relies on the results of [40].

4 Pontryagin's Maximum Principle and Method of Successive Approximations

In this section, we focus on the *sampled* optimal control problem (3.4). We first recall the necessary optimality conditions, usually known as the Pontryangin's Maximum Principle. We next introduce the main algorithm to solve the PMP which will be the starting point for our subsequent developments. In the following, the Euclidean norm of any vector $a \in \mathbb{R}^n$ is denoted by $\|a\|$ and the scalar product with any other vector $b \in \mathbb{R}^n$ is $a \cdot b$.

4.1 Pontryangin's Maximum Principle

We define the Hamiltoninan $H : [0,T] \times \mathbb{R}^n \times \mathbb{R}^n \times \Theta \to \mathbb{R}$ as

$$H(t,u,p,\theta) \coloneqq p \cdot f(u,\theta) - R(\theta). \tag{4.1}$$

The following classical Pontryagin's Maximum Principle (PMP) gives necessary optimality conditions to problem (3.4).

Theorem 4.1. *Let $\boldsymbol{\theta}^* \in L^\infty([0,T],\Theta)$ be an optimal control to (3.4). For $i = 1,\dots,N$, let $\boldsymbol{u}^{\boldsymbol{\theta}^*,i}$ be the associated state process with initial condition x_i.*

There exists a co-state process $\boldsymbol{p}^{\boldsymbol{\theta}^*,i} \in L^\infty([0,T],\mathbb{R}^n)$ *such that*

$$\begin{aligned} \dot{u}_t^{\boldsymbol{\theta}^*,i} &= f(u_t^{\boldsymbol{\theta}^*,i},\theta_t^*), \quad u_0^{\boldsymbol{\theta}^*,i} = x_i, \\ \dot{p}_t^{\boldsymbol{\theta}^*,i} &= -\nabla_u H(t,u_t^{\boldsymbol{\theta}^*,i},p_t^{\boldsymbol{\theta}^*,i},\theta_t^*), \quad p_T^{\boldsymbol{\theta}^*,i} = -\nabla_u \Phi(u_T^{\boldsymbol{\theta}^*,i},y_i), \end{aligned}$$

and, for each $t \in [0,T]$,

$$\frac{1}{N}\sum_{i=1}^N H(t,u_t^{\boldsymbol{\theta}^*,i},p_t^{\boldsymbol{\theta}^*,i},\theta_t^*) \geq \frac{1}{N}\sum_{i=1}^N H(t,u_t^{\boldsymbol{\theta}^*,i},p_t^{\boldsymbol{\theta}^*,i},\theta), \quad \forall \theta \in \Theta. \quad (4.2)$$

The proof of this theorem and its variants can be found in any optimal control theory reference (see, e.g., [45–47]). We omitted the case of an abnormal multiplier, which will not be considered in the following.

We emphasize that the PMP is only a necessary condition, so there can be cases where the solutions to the PMP are not global optima for (3.4). Nevertheless, in practice the PMP often gives good solution candidates, and when certain convexity assumptions are satisfied the PMP becomes sufficient (see [48]). In the next section, we discuss the numerical methods that we take as a starting point to solve the PMP.

4.2 MSA and Extended-MSA

A classical algorithm to find $\boldsymbol{\theta}^*$ and the corresponding forward and co-state dynamics $\boldsymbol{u}^{\boldsymbol{\theta}^*,i}$ and $\boldsymbol{p}^{\boldsymbol{\theta}^*,i}$ is the Method of Successive Approximations (see [49]). It is a fixed-point method based on the following steps. Starting from an initial guess of the optimal control $\boldsymbol{\theta}^0$, for each $k \geq 0$ we first solve for $i = 1,\dots,N$ the forward dynamics

$$\dot{u}_t^{\boldsymbol{\theta}^k,i} = \nabla_p H(t,u_t^{\boldsymbol{\theta}^k,i},p_t^{\boldsymbol{\theta}^k,i},\theta_t^k) = f(u_t^{\boldsymbol{\theta}^k,i},\theta_t^k), \quad u_0^{\boldsymbol{\theta}^k,i} = x_i. \quad (4.3)$$

The dynamics $\boldsymbol{u}^{\boldsymbol{\theta}^k,i}$ allows us to compute the backward dynamics

$$\dot{p}_t^{\boldsymbol{\theta}^k,i} = -\nabla_u H(t,u_t^{\boldsymbol{\theta}^k,i},p_t^{\boldsymbol{\theta}^k,i},\theta_t^k), \quad p_T^{\boldsymbol{\theta}^k,i} = -\nabla_u \Phi(u_T^{\boldsymbol{\theta}^k,i},y_i). \quad (4.4)$$

Finally, we update the control by using the maximization condition (4.2),

$$\theta_t^{k+1} \in \arg\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^N H(t,u_t^{\boldsymbol{\theta}^k,i},p_t^{\boldsymbol{\theta}^k,i},\theta), \quad \forall t \in [0,T]. \quad (4.5)$$

In this form, MSA converges only locally when it is initialized with a starting control guess that is sufficiently close to an optimal control $\boldsymbol{\theta}^*$. To overcome this limitation, an Extended MSA algorithm (E-MSA) based on an augmented Lagrangian strategy has been introduced in [31]. This algorithm is the starting point for our subsequent developments. It works as follows: fix some $\rho > 0$

and define the augmented Halmitonian

$$\widetilde{H}(t,u,p,\theta,v,q) \coloneqq H(t,u,p,\theta) - \frac{\rho}{2}\|v - f(u,\theta)\|^2 - \frac{\rho}{2}\|q + \nabla_u H(t,u,p,\theta)\|^2. \tag{4.6}$$

We can now formulate an extended PMP based on $\widetilde{H}$.

Proposition 4.2 (Extended PMP, see [31]). *Let $\boldsymbol{\theta}^* \in L^\infty([0,T],\Theta)$ be an optimal control to (3.4). For $i = 1,\dots,N$, let $\boldsymbol{u}^{\boldsymbol{\theta}^*,i}$ be the associated state process with initial condition x_i. There exists a co-state process $\boldsymbol{p}^{\boldsymbol{\theta}^*,i} \in L^\infty([0,T],\mathbb{R}^n)$ such that*

$$\begin{aligned} \dot{u}_t^{\boldsymbol{\theta}^*,i} &= \nabla_p \widetilde{H}(t, u_t^{\boldsymbol{\theta}^*,i}, p_t^{\boldsymbol{\theta}^*,i}, \theta_t^*), \quad u_0^{\boldsymbol{\theta}^*,i} = x_i, \\ \dot{p}_t^{\boldsymbol{\theta}^*,i} &= -\nabla_u \widetilde{H}(t, u_t^{\boldsymbol{\theta}^*,i}, p_t^{\boldsymbol{\theta}^*,i}, \theta_t^*), \quad p_T^{\boldsymbol{\theta}^*,i} = -\nabla_u \Phi(u_T^{\boldsymbol{\theta}^*,i}, y_i), \end{aligned}$$

and, for each $t \in [0,T]$,

$$\frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^*,i}, p_t^{\boldsymbol{\theta}^*,i}, \theta_t^*) \geq \frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^*,i}, p_t^{\boldsymbol{\theta}^*,i}, \theta), \quad \forall \theta \in \Theta.$$

The E-MSA algorithm consists in applying the MSA algorithm with the augmented Hamiltonian $\widetilde{H}$ instead of with H. Since $\nabla_p \widetilde{H} = \nabla_p H = f$ and $\nabla_u \widetilde{H} = \nabla_u H$, steps (4.3) and (4.4) remain the same as in MSA and the maximation step (4.5) is replaced by

$$\theta_t^{k+1} \in \arg\max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i}, p_t^{\boldsymbol{\theta}^k,i}, \theta), \quad \forall t \in [0,T]. \tag{4.7}$$

It has been proven in [31] that if the parameter ρ is taken sufficiently large, the E-MSA algorithm converges to the set of solution of the extended PMP for any initial guess of the control $\boldsymbol{\theta}^0$. However, this scheme cannot be realized in practice without discretizing. This introduces a perturbation of the time-continuous formulation of the algorithm since the resulting trajectories will be approximations of the continuous one. To guarantee convergence to the exact continuous solution, it is necessary to identify suitable approximation error tolerances that still guarantee convergence to the exact solution. This motivates to introduce an Adaptive MSA scheme that we describe in the next section. We prove that if the forward and backward propagations are solved with increasing accuracy at each step, then the algorithm converges towards a *continuous* solution of the Extended PMP.

5 A-MSA: Adaptive Method of Successive Approximations

The algorithm requires defining forward and backward time-integration schemes, which we introduce in section 5.1. We next present the algorithm and how predictions are made in sections 5.2 and 5.3. We end up by proving convergence in section 6.

5.1 Routines `solve_fwd` and `solve_bwd`

In the following, we work with time-integration routines `solve_fwd` and `solve_bwd` which can be cast in the abstract framework that we next describe. Relevant particular instances of it will be continuous Galerkin (cG) or Runge-Kutta collocation (RK-C) methods.

Setting:

We describe the framework in the case of the forward dynamics

$$\begin{cases} \dot{u}_t &= f(u_t), \quad \forall t \in (0,T) \\ u_0 &= x \end{cases} \tag{5.1}$$

for a generic Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$. The backward dynamics can be deduced similarly.

Let $0 = t_0 < t_1 < \cdots < t_L = T$ be a partition of $[0,T]$ and let $\mathcal{T}_\ell := (t_{\ell-1}, t_\ell]$, and $h_\ell := t_\ell - t_{\ell-1}$. For $q \in \mathbb{N}$, let $\mathcal{V}_q^{\mathcal{T}}$ be the space of continuous functions that are piecewise polynomials of degree q in the time mesh $\mathcal{T} = \cup_{\ell=1}^{L} \mathcal{T}_\ell$, i.e.,

$$\mathcal{V}_q^{\mathcal{T}} := \{v \in \mathcal{C}^0([0,T];\, \mathbb{R}^n) \,:\, v|_{\mathcal{T}_\ell} \in \mathbb{P}^{(q)}(\mathcal{T}_\ell), \quad \ell = 1, \ldots, L\},$$

where

$$\mathbb{P}^{(q)}(\mathcal{T}_\ell) := \{v \in \mathcal{C}^0(\mathcal{T}_\ell;\, \mathbb{R}^n) \,:\, \forall\, t \in \mathcal{T}_\ell,\, v(t) = \sum_{j=0}^{q} t^j v_j,\, v_j \in \mathbb{R}^n\}$$

is the space of polynomials of degree q in the interval $\mathcal{T}_\ell$.

Introducing a *projection operator*

$$\Pi^{(q)} : \mathcal{C}^0([0,T];\, \mathbb{R}^n) \mapsto \mathcal{V}_q^{\mathcal{T}}$$

the time discrete approximation U to the solution u is defined as follows: we seek $U \in \mathcal{V}_q^{\mathcal{T}}$ satisfying the initial condition $U(0) = x$ as well as

$$U_t' = \Pi^{(q-1)} f(U_t), \quad \forall t \in \mathcal{T}_\ell, \tag{5.2}$$

for $\ell = 1, \dots, L$. Note that since both terms belong to $\mathbb{P}^{(q-1)}(\mathcal{T}_\ell)$, (5.2) admits a Galerkin formulation

$$\int_{\mathcal{T}_\ell} v \cdot U_t' \, \mathrm{d}t = \int_{\mathcal{T}_\ell} v \cdot \Pi^{(q-1)} f(U_t) \, \mathrm{d}t, \quad \forall v \in \mathbb{P}^{(q)}(\mathcal{T}_\ell). \tag{5.3}$$

In the following, we use mainly (5.2) but (5.3) is of interest since it connects (RK-C) methods with (cG) methods. It is proven in [50], that the continuous Galerkin method corresponds to the choice $\Pi^{(q-1)} := P^{(q-1)}$, with $P^{(q-1)}$ denoting the (local) L^2 orthogonal projection onto $\mathbb{P}^{(q-1)}(\mathcal{T}_\ell)$ for each ℓ. Runge-Kutta collocation methods with pairwise distinct nodes in $\mathcal{T}_\ell$ can be obtained by choosing $\Pi^{(q-1)} := I^{(q-1)}$ with $I^{(q-1)}$ denoting the interpolation operator by elements of $\mathbb{P}^{(q)}(\mathcal{T}_\ell)$ at the nodes $t_{\ell-1} + \alpha_i(t_\ell - t_{\ell-1})$, $i = 1, \dots, q$, $\ell = 1, \dots, L$, with appropriate weights $0 \leq \alpha_1 < \cdots < \alpha_q \leq 1$. All RK-C methods with pairwise distinct nodes in $[0, 1]$ can be obtained by applying appropriate numerical quadrature to continuous Galerkin methods.

L^2 error estimation:

One simple way of estimating the error between u and U is via the following Grönwall inequality. Denoting $e_t = u_t - U_t$, we have

$$\dot{e}_t = f(u_t) - \Pi^{(q-1)} f(U_t) = (f(u_t) - f(U_t)) + \left(f(U_t) - \Pi^{(q-1)} f(U_t) \right)$$

Multiplying by e_t, we get

$$\begin{aligned} \frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} e_t^2 &= e_t \left(f(u_t) - f(U_t) \right) + e_t \left(f(U_t) - \Pi^{(q-1)} f(U_t) \right) \\ &\leq L e_t^2 + |e_t| |f(U_t) - \Pi^{(q-1)} f(U_t)| \\ &\leq (L + 1/2) e_t^2 + \frac{1}{2} |f(U_t) - \Pi^{(q-1)} f(U_t)|^2, \end{aligned}$$

where L is the Lipschitz constant on the second variable of g. By the Grönwall inequality, since $e(0) = 0$,

$$e_t^2 \leq \int_0^t |f(U_s) - \Pi^{(q-1)} f(U_s)|^2 e^{(2L+1)(t-s)} \mathrm{d}s$$

which yields an L^2 estimate of the error

$$\begin{aligned} \|u - U\|^2_{L^2([0,T],\mathbb{R}^n)} &\leq \int_0^T \int_0^t |f(U_s) - \Pi^{(q)} f(U_s)|^2 e^{(2L+1)(t-s)} \mathrm{d}s \mathrm{d}t \\ &\leq \frac{e^{(2^L+1)T}}{2L+1} \|f - \Pi^{(q-1)} f\|^2_{L^2([0,T],\mathbb{R}^n)} \end{aligned} \tag{5.4}$$

In the following, we denote Π_ζ a projector delivering an accuracy $\eta \geq 0$ for the involved dynamics

$$\|f - \Pi^{(q-1)} f\|_{L^2([0,T],\mathbb{R}^n)} \leq \eta.$$

From inequality (5.4), such a projector gives an accuracy in the solution U which is bounded by

$$\|u - U\|_{L^2([0,T],\mathbb{R}^n)} \leq \frac{e^{(2^L+1)T/2}}{(2L+1)^{1/2}}\eta$$

The routines:

In the following, the routine

$$\boldsymbol{u}^\eta = \texttt{solve\_fwd}(\eta;\ f, x)$$

yields an evolution

$$\boldsymbol{u}^\eta = \{u_t^\eta\ :\ t \in [0,T]\}$$

that approximates the exact solution $\boldsymbol{u}$ of the *forward* dynamics (5.1) with a projector Π_η. The routine

$$\boldsymbol{p}^\eta = \texttt{solve\_bwd}(\eta;\ f, x)$$

works similarly for the *backward* dynamics $\dot{p}_t = f(p_t)$ with $p_T = x$.

5.2 The learning phase of the algorithm

The A-MSA algorithm consists in performing steps (4.3) and (4.4) of the original MSA but with a numerical time-integrators which give inexact trajectories. We carry the discussion assuming that we use the routines `solve_fwd` and `solve_bwd` introduced above. At every iteration k, both routines involves projectors Π_k that deliver an accuracy ε_k which is yet to be determined. The accuracy will be tightened as k increases in a way that still guarantees convergence to the exact, continuous solution (3.4) of the sampled PMP problem. We deduce its value later on from the convergence analysis of section 6.

Starting from an initial guess of the optimal control $\boldsymbol{\theta}^0$, for each $k \geq 0$ we first solve at accuracy ε_k the forward dynamics

$$\boldsymbol{u}^{\boldsymbol{\theta}^k, i, \varepsilon_k} = \texttt{solve\_fwd}\left(\varepsilon_k;\ (t,u) \mapsto f(u, \theta_t^k), x_i\right), \quad \forall i = 1, \ldots, N.$$

Satisfying this accuracy may require to adapt the time discretization mesh and/or the numerical scheme as discussed in the previous section.

We then use the final state $u_T^{\boldsymbol{\theta}^k,i,\varepsilon_k}$ to compute an approximation of the *backward* dynamics at the same accuracy ε_k for all $i \in \{1,\dots,N\}$

$$\boldsymbol{p}^{\boldsymbol{\theta}^k,i,\varepsilon_k} = \texttt{solve\_bwd}\left(\varepsilon_k;\ (t,p) \mapsto -\nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p, \theta_t^k), -\nabla_u \Phi(u_T^{\boldsymbol{\theta}^k,i,\varepsilon_k}, y_i)\right).$$

This step may, again, require adaptivity to satisfy the target tolerance. To update the control, instead of finding the exact maximum like in (4.7), it is in fact sufficient to find controls that are away by a factor $0 < \gamma_k \leq 1$ of the maximum and for which $\gamma_k \to 1$ as $k \to \infty$. In other words, it suffices to find for all $t \in [0,T]$, a control $\theta_t^{k+1} \in \Theta$ such that for all $t \in [0,T]$

$$\frac{1}{N}\sum_{i=1}^{N} \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1}) \geq \gamma_k \max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta). \tag{5.5}$$

Note that when $\gamma_k = 1$, we fall back to the case where we look for the exact maximum. Algorithm 1 summarizes the whole procedure by describing a routine A-MSA$[\mathcal{S},\tau]$ which computes a control dynamics $\hat{\boldsymbol{\theta}}$ that solves the extended PMP problem until the difference between two successive iterates is smaller than $\tau > 0$. The algorithm also yields a numerical scheme associated to the parameter $\hat{\varepsilon}$ which is the accuracy of the forward and backward solvers used in the last iteration. These elements are then used for prediction as explained in the next section.

The practical implementation of the algorithm requires to specify the tolerances ε_k and γ_k. They can be theoretically derived from the convergence analysis of section 6. However, since the bounds of the analysis may not be sharp and that it involves quantities which are difficult to estimate in practice, devote section 8 to give indications on how to implement A-MSA in practice.

5.3 Predictions

The routine A-MSA$[\mathcal{S},\tau]$ gives a discrete control $\hat{\boldsymbol{\theta}}$ and an accuracy $\hat{\varepsilon}$. We also have a certain numerical scheme for the forward propagator, which can be interpreted as the final network architecture. As a result, for a given input data x, we can predict the output by first computing

$$\hat{\boldsymbol{u}} = \texttt{solve\_fwd}\left(\hat{\varepsilon};\ (t,u) \mapsto f(u,\hat{\boldsymbol{\theta}}), x\right),$$

and then taking

$$\hat{y} \coloneqq g(\hat{u}_T)$$

as the approximation of the true y. In other words, our neural network is the mapping

$$\mathcal{NN} : \mathbb{R}^n \to \mathbb{R}^k$$

Algorithm 1 Learning Algorithm: A-MSA$[\mathcal{S}_N, \tau] \to [\hat{\boldsymbol{\theta}}, \hat{\varepsilon}]$

Require: $\mathcal{S}_N = \{x_i, y_i\}_{i=1}^N$, target tolerance τ
Set of internal parameters: $\boldsymbol{\theta}^0$, ρ, maximum number of iterations $k_{\max}$.
$k \leftarrow 0$
$J, J_{\text{old}} \leftarrow J(\boldsymbol{\theta}^0)$
$\Delta \leftarrow \tau + 1$

while $\Delta > \tau$ or $k \leq k_{\max}$ **do**
  Update ε_k and γ_k
  for i in $\{1, \dots, N\}$ **do** ▷ In parallel
    $\boldsymbol{u}^{\boldsymbol{\theta}^k, i, \varepsilon_k} = \texttt{solve\_fwd}\left(\varepsilon_k;\ (t,u) \mapsto f(u, \theta_t^k), x_i\right)$
    $\boldsymbol{p}^{\boldsymbol{\theta}^k, i, \varepsilon_k} = \texttt{solve\_bwd}\left(\varepsilon_k;\ (t,p) \mapsto -\nabla_x H(t, u_t^{\boldsymbol{\theta}^k, i, \varepsilon_k}, p, \theta_t^k), -\nabla_u \Phi(u_T^{\boldsymbol{\theta}^k, i, \varepsilon_k}, y_i)\right)$
  end for
  for $t \in [0, T]$ **do** ▷ In parallel
    Find θ_t^{k+1} s.t.

$$\frac{1}{N}\sum_{i=1}^{N} \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k, i, \varepsilon_k}, p_t^{\boldsymbol{\theta}^k, i, \varepsilon_k}, \theta_t^{k+1}) \geq \gamma_k \max_{\theta \in \Theta} \frac{1}{N}\sum_{i=1}^{N} \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k, i, \varepsilon_k}, p_t^{\boldsymbol{\theta}^k, i, \varepsilon_k}, \theta)$$

  end for
  $J_{\text{old}} \leftarrow J; \quad J \leftarrow J(\boldsymbol{\theta}^{k+1}); \quad \Delta \leftarrow J_{\text{old}} - J$
end while

$\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^{k+1}; \quad \hat{\varepsilon} \leftarrow \varepsilon_k$
Output: $[\hat{\boldsymbol{\theta}}, \hat{\varepsilon}]$

$$x \to \mathcal{NN}(x) \coloneqq g(\hat{u}_T) = g\left(\texttt{solve\_fwd}\left(\hat{\varepsilon};\ (t,u) \mapsto f(u, \hat{\boldsymbol{\theta}}), x\right)(t = T)\right)$$

We summarize the prediction pipeline in Algorithm 2.

Algorithm 2 Prediction algorithm: $\mathcal{NN}(x) \to \hat{y}$

Input:
- Observation $x \in \mathbb{R}^n$,
- Neural network architecture $[\hat{\boldsymbol{\theta}}, \hat{\varepsilon}]$ (from A-MSA$[\mathcal{S}_N, \tau]$).

$\hat{\boldsymbol{u}} \leftarrow \texttt{solve\_fwd}\left(\hat{\varepsilon};\ (t,u) \mapsto f(u, \hat{\boldsymbol{\theta}}), x\right)$
Output: $g(\hat{u}_T)$

6 A priori convergence of A-MSA

In this section, we prove that, for suitably chosen tolerances $(\varepsilon_k)_k$, the sequence of controls $(\boldsymbol{\theta}^k)_{k\geq 0}$ computed with A-MSA is a minimizing sequence for problem (3.4), that is,

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \xrightarrow{k\to\infty} J^*_{\mathcal{S}_N}.$$

Our analysis is built upon the one presented in [31] for the exact E-MSA to which it is necessary to add perturbative arguments due to the inexact propagations.

We work with the same continuity assumptions as in [31] for the ideal case of exact propagations:

(A1) Φ is twice continuously differentiable. Φ and $\nabla\Phi$ satisfy the following Lipschitz condition: there exists $K > 0$ such that

$$|\Phi(u) - \Phi(u')| + \|\nabla\Phi(u) - \nabla\Phi(u')\| \leq K\|u - u'\|, \quad \forall (u, u') \in \mathbb{R}^n.$$

(A2) For all $t \in [0, T]$ and $\theta \in \Theta$, $u \to f(u, \theta)$ is twice continuously differentiable and satisfying the following Lipschitz condition: there exists $K' > 0$ such that

$$\|f(u,\theta)-f(u',\theta)\|+\|\nabla_u f(u,\theta)-\nabla_u f(u',\theta)\|_2 \leq K\|u-u'\|, \quad \forall (u, u') \in \mathbb{R}^n,$$

where $\|\cdot\|_2$ denotes the induced 2-norm.

Intermediate problem:

We first study the following intermediate problem. Consider a fixed input-output pair (x, y). For a given control $\boldsymbol{\theta} \in L^\infty([0, T], \Theta)$, let $\boldsymbol{u}^{\boldsymbol{\theta},x}$ and $\boldsymbol{p}^{\boldsymbol{\theta},x}$ be the exact solutions of the forward and backward propagations using x and y in the initial and final conditions,

$$\begin{cases} \dot{u}_t^{\boldsymbol{\theta},x} &= f(u_t^{\boldsymbol{\theta},x}, \theta_t), \quad u_0^{\boldsymbol{\theta},x} = x, \\ \dot{p}_t^{\boldsymbol{\theta},x} &= -\nabla_u H(t, u_t^{\boldsymbol{\theta},x}, p_t^{\boldsymbol{\theta},x}, \theta_t), \quad p_T^{\boldsymbol{\theta},x} = -\nabla_u \Phi(u_T^{\boldsymbol{\theta},x}, y), \end{cases}$$

Let $\boldsymbol{u}^{\boldsymbol{\theta},x,\zeta}$ and $\boldsymbol{p}^{\boldsymbol{\theta},x,\zeta}$ be the outputs of the same propagation but with accuracy $\zeta > 0$,

$$\begin{cases} \boldsymbol{u}^{\boldsymbol{\theta},x,\zeta} &= \texttt{solve\_fwd}\,(\zeta;\ (t,u) \mapsto f(u,\boldsymbol{\theta}), x) \\ \boldsymbol{p}^{\boldsymbol{\theta},x,\zeta} &= \texttt{solve\_bwd}\left(\zeta;\ (t,p) \mapsto -\nabla_u H(t, u_t^{\boldsymbol{\theta},x,\zeta}, p, \theta_t), -\nabla_u \Phi(u_T^{\boldsymbol{\theta},x,\zeta}, y)\right). \end{cases} \tag{6.1}$$

Similarly, $\boldsymbol{u}^{\boldsymbol{\varphi},x,\eta}$ and $\boldsymbol{p}^{\boldsymbol{\varphi},x,\eta}$ denote the η-accurate solutions but for another control $\boldsymbol{\varphi} \in L^\infty([0, T], \Theta)$.

The loss function associated to $\boldsymbol{u}^{\boldsymbol{\theta},x,\zeta}$ is denoted

$$\text{Loss}(x, y, \boldsymbol{\theta}, \zeta) \coloneqq \Phi(u_T^{\boldsymbol{\theta},x,\zeta}, y) + \int_0^T R(\theta_t)\mathrm{d}t,$$

and similarly for $\boldsymbol{u}^{\boldsymbol{\varphi},x,\eta}$. In addition, we define the quantity

$$\Delta H^{x,\zeta}_{\boldsymbol{\varphi},\boldsymbol{\theta}}(t) \coloneqq H(t, u_t^{\boldsymbol{\theta},\zeta}, p_t^{\boldsymbol{\theta},\zeta}, \varphi_t) - H(t, u_t^{\boldsymbol{\theta},\zeta}, p_t^{\boldsymbol{\theta},\zeta}, \theta_t), \tag{6.2}$$

which will play an essential role in what follows.

Lemma 6.1 is an important building-block in the proof of convergence of A-MSA. To not interrupt the flow of reading, we have deferred its proof to Appendix A.

Lemma 6.1. *Let $\boldsymbol{\theta}$, $\boldsymbol{\varphi} \in L^\infty([0,T],\Theta)$ be two controls and let $(x,y) \in \mathbb{R}^n \times \mathbb{R}^k$ be an input-output pair. Let $(\boldsymbol{u}^{\boldsymbol{\theta},x,\zeta}, \boldsymbol{p}^{\boldsymbol{\theta},x,\zeta})$ and $(\boldsymbol{u}^{\boldsymbol{\varphi},x,\eta}, \boldsymbol{p}^{\boldsymbol{\varphi},x,\eta})$ be solutions of the forward and backward associated problems with accuracy ζ and η like in (6.1). Then, there exist a constant $C > 0$ independent of $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$ but dependent on T, $\|\Pi_\zeta\|$ and $\|\Pi_\eta\|$ such that, if $\eta \leq 1$,*

$$\begin{aligned} &Loss(x, y, \boldsymbol{\varphi}, \eta) - Loss(x, y, \boldsymbol{\theta}, \zeta) \\ &\quad \leq -\int_0^T \Delta H^{x,\zeta}_{\boldsymbol{\varphi},\boldsymbol{\theta}}(t)\mathrm{d}t + C\left((\eta+\zeta)^2 + \|\nabla_w(\Delta H^{x,\zeta}_{\boldsymbol{\varphi},\boldsymbol{\theta}})\|^2_{L^2([0,T])}\right) \end{aligned} \tag{6.3}$$

Convergence of A-MSA:

We can use Lemma 6.1 to prove our main convergence result which we give in the following theorem. It involves the quantity

$$\begin{aligned} \lambda_k^2 &\coloneqq \frac{1}{N}\sum_{i=1}^N \|\nabla_w(\Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k})\|^2_{L^2([0,T])} \\ &= \frac{1}{N}\sum_{i=1}^N \left(\int_0^T \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1}) - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2\mathrm{d}t \right. \\ &\left. + \int_0^T \|\nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1}) - \nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2\mathrm{d}t\right) \end{aligned}$$

Theorem 6.2. *Let $\delta > 0$ be a fixed constant and suppose that we run the A-MSA algorithm with the following accuracies:*

- *(ε_k) is a positive and decreasing sequence such that*

$$0 \leq \varepsilon_k \leq \min\left(\varepsilon_{k-1}, \frac{\delta\lambda_k^2}{4C + \rho N^{-1/2}\lambda_k}\right), \quad \forall k \geq 1.$$

- *The maximization step* (5.5) *is performed with the parameter*

$$\gamma_k \geq \frac{\left|\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})/N\right|}{\delta\lambda_k^2 + \left|\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})/N\right|}, \quad \forall k \in \mathbb{N}.$$

Then, if $\rho > 2(C+2\delta+\varepsilon_0)$*, where* C *is the constant of Lemma 6.1, the sequence of controls* $(\boldsymbol{\theta}^k)$ *of A-MSA is a minimizing sequence for problem* (3.4)*, that is,*

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \to_{k\to\infty} J^*_{\mathcal{S}_N}.$$

Furthermore, $\lambda_k^2 \to 0$ *as* $k \to \infty$ *and we have*

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \leq -\kappa\lambda_k^2, \quad \kappa := |C + 2\delta - \frac{\rho}{2}| > 0, \quad \forall k \in \mathbb{N}.$$

Proof We start by applying inequality (6.3) from Lemma 6.1 to the controls $\boldsymbol{\varphi} = \boldsymbol{\theta}^{k+1}$ and $\boldsymbol{\theta} = \boldsymbol{\theta}^k$ of the A-MSA algorithm, with accuracies $\eta = \varepsilon_{k+1}$ and $\zeta = \varepsilon_k$ and for the samples $(x, y) = (x_i, y_i)$. We then take the average over i and obtain

$$\begin{aligned}\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) &= \frac{1}{N}\sum_{i=1}^N \mathrm{Loss}(x_i, y_i, \boldsymbol{\theta}^{k+1}, \varepsilon_{k+1}) - \mathrm{Loss}(x_i, y_i, \boldsymbol{\theta}^k, \varepsilon_k) \\ &\leq -\frac{1}{N}\sum_{i=1}^N \int_0^T \Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k}(t)\mathrm{d}t + C\left((\varepsilon_k + \varepsilon_{k+1})^2 + \lambda_k^2\right) \quad (6.4)\end{aligned}$$

We first do the proof when $\gamma_k = 1, \forall k \geq 1$. In this case, from the maximization step (5.5) in A-MSA,

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1}, \dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \dot{p}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}) \\ &\qquad \geq \frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k, \dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \dot{p}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}).\end{aligned}$$

Recalling definition (4.6) for $\widetilde{H}$, we reassemble the terms in the above inequality to derive

$$\begin{aligned}&-\frac{1}{N}\sum_{i=1}^N \int_0^T \Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k}(t)\mathrm{d}t \\ &\leq -\frac{\rho}{2N}\sum_{i=1}^N \int_0^T \left(\|\dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 + \|\dot{p}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - \nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})|^2\right) \\ &+ \frac{\rho}{2N}\sum_{i=1}^N \int_0^T \left(\|\dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2 + \|p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - \nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2\right)\mathrm{d}t \\ &\qquad (6.5)\end{aligned}$$

Since by construction

$$\dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} = \Pi_{\varepsilon_k} f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)$$

and

$$\|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - \Pi_{\varepsilon_k} f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|_{L^2([0,T],\mathbb{R}^n)} \leq \varepsilon_k,$$

we have by Cauchy-Schwartz inequality

$$\begin{aligned}
&\int_0^T \|u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 \mathrm{d}t \\
&\quad = \int_0^T \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - \Pi f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2 + \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 \\
&\quad - 2\left\langle f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - \Pi f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k), f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\right\rangle \mathrm{d}t \\
&\quad \geq \int_0^T \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - \Pi f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k)\|^2 + \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 \mathrm{d}t \\
&\quad - 2\varepsilon_k \left(\int_0^T \|f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^k) - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 \mathrm{d}t\right)^{1/2}
\end{aligned}$$

Since a similar bound holds for $\int_0^T \|p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - \nabla_u H(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1})\|^2 \mathrm{d}t$, we derive that

$$-\frac{1}{N}\sum_{i=1}^N \int_0^T \Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k}(t)\mathrm{d}t \leq -\frac{\rho}{2}\lambda_k^2 + \rho N^{-1/2}\varepsilon_k\lambda_k \tag{6.6}$$

Using (6.6), we can further bound (6.4) to obtain

$$\begin{aligned}
\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) &\leq -\frac{\rho}{2}\lambda_k^2 + \rho N^{-1/2}\varepsilon_k\lambda_k + C\left((\varepsilon_k + \varepsilon_{k+1})^2 + \lambda_k^2\right) \\
&\leq \left(C - \frac{\rho}{2}\right)\lambda_k^2 + \varepsilon_k(4C + \rho N^{-1/2}\lambda_k)
\end{aligned} \tag{6.7}$$

where we have used that (ε_k) is a decreasing sequence.

Since, for a fixed $\delta > 0$, we have chosen ε_k such that $\varepsilon_k \leq \delta\lambda_k^2/(4C+\rho N^{-1/2}\lambda_k)$, we infer that

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \leq \left(C + \delta - \frac{\rho}{2}\right)\lambda_k^2.$$

As a result, fixing $\rho > 2(C + \delta)$,

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \leq -\tilde{\kappa}\lambda_k^2, \quad \tilde{\kappa} := \left|C + \delta - \frac{\rho}{2}\right| > 0.$$

Moreover, we can rearrange and sum over $k = 0$ to K the above expression to get

$$\sum_{k=0}^K \lambda_k^2 \leq \tilde{\kappa}^{-1}\left(\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^0) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{K+1})\right) \leq \tilde{\kappa}^{-1}\left(\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^0) - \inf_{\boldsymbol{\theta}\in L^\infty([0,T],\Theta)} \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta})\right).$$

Therefore $\sum_{k=0}^K \lambda_k^2 < +\infty$ which implies that $\lambda_k^2 \to_{k\to\infty} 0$. Therefore, the A-MSA converges to a solution of the extended PMP.

Let us consider now the general case $0 < \gamma_k \leq 1$. From the maximization step (5.5),

$$\frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta_t^{k+1}) \geq \gamma_k \max_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^N \widetilde{H}(t, u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k}, \theta), \quad \forall t \in [0,T].$$

As a result, bound (6.5) receives an additional term and becomes,

$$-\frac{1}{N}\sum_{i=1}^{N}\int_0^T \Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k}(t)\mathrm{d}t \le -\frac{\rho}{2N}\sum_{i=1}^{N}\int_0^T \Big(\|\dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})\|^2 + \|p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - \nabla_u H(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})\|^2\Big)\mathrm{d}t$$

$$+\frac{\rho}{2N}\sum_{i=1}^{N}\int_0^T \Big(\|\dot{u}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - f(u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k})\|^2 + \|\dot{p}_t^{\boldsymbol{\theta}^k,i,\varepsilon_k} - \nabla_u H(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k})\|^2\Big)\mathrm{d}t + \frac{\gamma_k^{-1}-1}{N}\Big|\sum_{i=1}^{N}\widetilde{H}(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})\Big|.$$

It follows that bound (6.6) receives the same additional term

$$-\frac{1}{N}\sum_{i=1}^{N}\int_0^T \Delta H^{i,\varepsilon_k}_{\boldsymbol{\theta}^{k+1},\boldsymbol{\theta}^k}(t)\mathrm{d}t \le -\frac{\rho}{2}\lambda_k^2 + \rho N^{-1/2}\varepsilon_k\lambda_k + \frac{\gamma_k^{-1}-1}{N}\Big|\sum_{i=1}^{N}\widetilde{H}(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})\Big|.$$

Therefore (6.7) becomes

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \le \Big(C-\frac{\rho}{2}\Big)\lambda_k^2 + \varepsilon_k(4C+\rho N^{-1/2}\lambda_k) + \frac{\gamma_k^{-1}-1}{N}\Big|\sum_{i=1}^{N}\widetilde{H}(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})\Big|.$$

Choosing ε_k like before and setting

$$\gamma_k \ge \frac{\big|\sum_{i=1}^{N}\widetilde{H}(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})/N\big|}{\delta\lambda_k^2 + \big|\sum_{i=1}^{N}\widetilde{H}(t,u_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},p_t^{\boldsymbol{\theta}^k,i,\varepsilon_k},\theta_t^{k+1})/N\big|}$$

yields

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \le \Big(C+2\delta-\frac{\rho}{2}\Big)\lambda_k^2.$$

We conclude along the same lines as before by fixing now $\rho > 2(C+2\delta)$, and inferring that

$$\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^{k+1}) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) \le -\kappa\lambda_k^2, \quad \kappa := \big|C+2\delta-\frac{\rho}{2}\big| > 0.$$

□

Before going to the next section, a few remarks on Theorem 6.2 are in order:

- Note that since the sequence of controls $(\boldsymbol{\theta}^k)$ built by A-MSA is a minimizing sequence, it need not convergence in $L^\infty([0,T],\Theta)$ norm, nor does it necessarily contain a convergent subsequence. To guarantee that the minimizing sequence converges in norm, it is necessary to additional convexity and/or compactness properties.
- The convergence result is *a priori* in the sense that it cannot be directly implemented for the following reasons:
  - The bounds for (ε_k) and (γ_k) involve a constant C which is difficult to estimate in practice.

  - The bounds may be suboptimal due to the construction of the proof.
  - In practice, it is difficult to guarantee that the maximization step is performed within a certain fraction γ_k of the actual maximum.
- The main interest of the result lies in the fact that it reveals the necessity to tighten the accuracy in the time integration schemes and the maximization algorithms across the iterations in order to approach the continuous sampled optimal control problem (3.4).

7 Generalization error and convergence towards the fully continuous problem

The goal of this section is to connect the controls $(\boldsymbol{\theta}^k)$ of the A-MSA Algorithm 1 with the PMP solutions of the fully continuous problem (3.2). For this, we rely on recent results from [40] connecting the sampled PMP with the fully continuous PMP which we next briefly recall.

We assume in the following that $\boldsymbol{\theta}^*$ and $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ are solutions of (3.2) and (3.4) such that the Hamiltonian step attains a maximum in the *interior* of Θ. Consequently, the continuous PMP solution $\boldsymbol{\theta}^*$ satisfies

$$\boldsymbol{F}(\boldsymbol{\theta}^*)_t \coloneqq \mathbb{E}_\mu \nabla_\theta H(t, u_t^{\boldsymbol{\theta}^*}, p_t^{\boldsymbol{\theta}^*}, \theta_t^*) = 0$$

for a.e. $t \in [0,T]$ and where $\boldsymbol{F} : L^\infty([0,T],\Theta) \to L^\infty([0,T],\mathbb{R}^m)$. Similarly, an interior solution $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ of the sampled PMP is a random variable which satisfies

$$\boldsymbol{F}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})_t \coloneqq \frac{1}{N}\sum_{i=1}^N \nabla_\theta H(t, u_t^{\boldsymbol{\vartheta}^*_{\mathcal{S}_N},i}, p_t^{\boldsymbol{\vartheta}^*_{\mathcal{S}_N},i}, \vartheta_{N,t}) = 0.$$

Under these assumptions, the theorem below, proven in [40], describes the convergence of an interior solution $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ of the first order condition of the sampled PMP to an interior solution $\boldsymbol{\theta}^*$ of the continuous PMP. The additional local strong concavity assumption on the Hessian allows to guarantee that $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ is a global/local maximum of the sampled PMP. The result also guarantees convergence of loss function values.

Definition 7.1. For $\rho > 0$ and $x \in L^\infty([0,T],\Theta)$, define $S_\rho(x) \coloneqq \{y \in L^\infty([0,T],\Theta) : \|x-y\| \le \rho\}$. The mapping $\boldsymbol{F}$ is said to be stable on $S_\rho(x)$ if there exists a constant $K_\rho > 0$ such that for all $y, z \in S_\rho(x)$,

$$\|y - z\|_{L^\infty([0,T],\Theta)} \le K_\rho \|\boldsymbol{F}(x) - \boldsymbol{F}(y)\|_{L^\infty([0,T],\mathbb{R}^m)}.$$

Theorem 7.2 (Theorem 6, Corollary 1 and 2 from [40])**.** *Let $\boldsymbol{F}$ be a mapping which is stable on $S_\rho(\boldsymbol{\theta}^*)$ for some $\rho > 0$ and $\boldsymbol{\theta}^*$ a solution of $\boldsymbol{F} = 0$. Then there exists positive constants s_0, C, K_1, K_2 and $\rho_1 < \rho$ and a random variable*

$\boldsymbol{\vartheta}^*_{\mathcal{S}_N} \in S_{\rho_1}(\boldsymbol{\theta}^*)$ *such that*

$$\mathbb{P}[\|\boldsymbol{\theta}^* - \boldsymbol{\vartheta}^*_{\mathcal{S}_N}\| \geq Cs] \leq 4\exp\left(-\frac{Ns^2}{K_1 + K_2 s}\right), \quad s \in (0, s_0]$$

and $\boldsymbol{\vartheta}^*_{\mathcal{S}_N} \to \boldsymbol{\theta}^*$ *as* $N \to \infty$ *in probability. Moreover, there exists constants* K_1', K_2' *such that*

$$\mathbb{P}[\left|\mathcal{J}(\boldsymbol{\theta}^*) - \mathcal{J}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})\right| \geq s] \leq 4\exp\left(-\frac{Ns^2}{K_1' + K_2' s}\right), \quad s \in (0, s_0] \tag{7.1}$$

In addition, if for all $t \in [0,T]$*, the Hessian* $\mathbb{E}_\mu \nabla^2_{\theta,\theta} H(u_t^{\boldsymbol{\theta}^*}, p_t^{\boldsymbol{\theta}^*}, \theta_t^*) + \lambda_0 I$ *is negative definite, then* $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ *is also a strict local maximum of the sampled Hamiltonian* $\vartheta \mapsto \frac{1}{N}\sum_{i=1}^N H(u_t^{\boldsymbol{\vartheta}^*_{\mathcal{S}_N},i}, p_t^{\boldsymbol{\vartheta}^*_{\mathcal{S}_N},i}, \vartheta)$*. In particular, if the sampled Hamiltonian has a unique maximizer, then* $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ *is a solution of the sampled PMP (4.2) with the same high probability.*

Corollary 7.3. *If the sampled Hamiltonian has a unique maximizer* $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$*, then there exists constants* $\tilde{K}_1$ *and* $\tilde{K}_2$ *such that for any* $\varepsilon > 0$*, there exists* $k(\varepsilon) \in \mathbb{N}$ *such that for* $k \geq k(\varepsilon)$*, the control* $\boldsymbol{\theta}^k$ *at iteration* k *of A-MSA satisfies*

$$\mathbb{P}\left(|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) - \mathcal{J}(\boldsymbol{\theta}^*)| \geq 3\varepsilon\right) \leq 4\exp\left(-\frac{N\varepsilon^2}{\tilde{K}_1 + \tilde{K}_2\varepsilon}\right)$$

Proof We use that

$$\begin{aligned}\mathbb{P}\left(|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) - \mathcal{J}(\boldsymbol{\theta}^*)| \geq 3\varepsilon\right) \\ \leq \mathbb{P}\left(|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})| \geq \varepsilon\right) + \mathbb{P}\left(|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N}) - \mathcal{J}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})| \geq \varepsilon\right) \\ + \mathbb{P}\left(|\mathcal{J}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N}) - \mathcal{J}(\boldsymbol{\theta}^*)| \geq \varepsilon\right),\end{aligned}$$

and then bound each term as explained next. Let $\varepsilon > 0$ be fixed. With high probability, $\boldsymbol{\vartheta}^*_{\mathcal{S}_N}$ is a solution to the sampled PMP (3.4), that is $\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N}) = J^*_{\mathcal{S}_N}$. By Theorem 6.2, there exists $k(\varepsilon) \in \mathbb{N}$ such that for $k \geq k(\varepsilon)$, the control $\boldsymbol{\theta}^k$ at iteration k of A-MSA satisfies $|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^k) - \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})| \leq \varepsilon$, which bounds the first term. For the second term, by the infinite dimensional Hoeffding's inequality (see [51, Corollary 2]), $|\mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N}) - \mathcal{J}(\boldsymbol{\vartheta}^*_{\mathcal{S}_N})| \leq \varepsilon$ with probability $1 - \exp(-N\varepsilon^2/(\tilde{K}_1' + \tilde{K}_2'\varepsilon))$. The third term is also bounded by ε with probability $4\exp\left(-\frac{N\varepsilon^2}{K_1' + K_2'\varepsilon}\right)$ thanks to (7.1). □

8 Guidelines for numerical implementation

Algorithm 1 should in principle be implemented with tolerances ε_k and γ_k given in Theorem 6.2 to ensure convergence. These tolerances are in practice hard to guarantee, and the convergence analysis that leads to these quantities

may be suboptimal. A trade-off between the theory and numerical implementation consists simply in replacing the forward and backward propagations at prescribed accuracies by propagations where the time steps are progressively refined across the iterations. It is therefore necessary to prescribe a refinement strategy. For the Hamiltonian maximization step (line 13 of Algorithm 1), the best that one can do is to use an efficient optimizer of nonconvex problems.

In addition to these considerations, note that Algorithm 1 can be parallelized at several parts: for each sample i, the forward and backward propagations can be performed in parallel (see the for loop of line 8). Each propagation can additionally be parallelized by means of parallel in time algorithms as in [52] (see [53, 54] for some selected references on the topic). In addition, the Hamiltonian maximization step can also be parallelized (see `for` loop of line 12 of Algorithm 1). For our development, we have chosen to parallelize the Maximization step, since it is way more expensive in terms of computational time than the propagation step.

9 Numerical experiments

In this section, we present some numerical experiments, aiming primarily at illustrating the behavior of the A-MSA algorithm. We give special focus on studying the performance in the learning phase in terms of the value of the loss function. We also discuss computing times and examine expressivity in terms of generalization errors. Note that the final quality of approximation depends on the number of samples, on the network depth, and on the optimizers and their starting guesses. We proceed by increasing levels of difficulty in the numerical examples in order to disentangle the effect of each of the different aspects (note however that their effects are usually mixed in non-trivial applications). The implementation has been done with Python 3 and NumPy. Particular attention has been paid to vectorize most of the operations in order to delegate as much as possible the looping to internal, highly optimized C and Fortran functions. We have observed that the time to perform the ODE propagations is negligible with respect to the Hamiltonian maximization (line 12 of Algorithm 1). We have thus parallelized this step with MPI.

9.1 Approximation of the sine function

In this example, inspired from [31, Section 6], the task is to approximate the graph of the sine function,

$$y : X = [-\pi, \pi] \to Y = [-1, 1], \qquad x \mapsto y(x) = \sin(x).$$

Note that here the dimensions n and k of the domain and label sets are $n = k = 1$. The continuous learning problem (see (3.2)) is to minimize the $L^2(X)$

approximation error over the controls $\boldsymbol{\theta} \in L^\infty([0,T],\Theta)$, namely

$$\inf_{\boldsymbol{\theta}\in L^\infty([0,T],\Theta)} \frac{1}{2}\int_X |y(x) - g(u_T^{\boldsymbol{\theta},x})|^2 \,\mathrm{d}x.$$

For every point $x \in X$, $u_T^{\boldsymbol{\theta},x}$ is the final time state of the ODE

$$\begin{cases} \dot{u}_t^{\boldsymbol{\theta},x} &= \tanh(A_t \cdot u_t^{\boldsymbol{\theta},x} + b_t), \quad \forall t \in (0,T] \\ u_0^{\boldsymbol{\theta},x} &= x(1,\dots,1)^T \in \mathbb{R}^d. \end{cases}$$

Here, $\theta_t = (A_t, b_t) \in \mathbb{R}^{d\times d} \times \mathbb{R}^d \sim \mathbb{R}^{d^2+d}$ are the parameters to optimize. We constraint them to lie in $[\alpha_{\min}, \alpha_{\max}] = [-1,1]$ so $\Theta = [-1,1]^{d^2+d}$. The function g is defined as

$$g : \mathbb{R}^d \to \mathbb{R}, \quad z = (z_1,\dots,z_d) \mapsto g(z) \coloneqq \frac{1}{d}\sum_{i=1}^d z_i,$$

and note that we have chosen $\mu = \mathrm{d}x$ as the Lebesgue measure, $f = \tanh$ as the activation function, $\Phi(u_T^{\boldsymbol{\theta},x}, y) = \frac{1}{2}|y(x) - g(u_T^{\boldsymbol{\theta},x})|^2$ and we do not have any regularisation ($R = 0$).

In the sampled version which we consider in our experiments, we work with a set $\mathcal{S}_N = \{(x_i, y_i)\}_{i=1}^N$ of pairs generated with equidistant data points

$$x_i = -\pi + 2\pi\frac{i-1}{N-1}, \quad y_i = \sin(x_i), \quad i \in \{1,\dots,N\},$$

and perform empirical risk minimization taking the uniform distribution $\mu_N = \frac{1}{N}\sum_{i=1}^N \delta_{(x_i,y_i)}$ (see (3.4)). We solve this problem with the above described A-MSA algorithm where the final time is set to $T = 5$, the penalization parameter is set to $\rho = 5$, and the number of neurons per layer to $d = 3$. We use a basic explicit Euler scheme as a time integrator, which induces a ResNet architecture. We progressively refine the time discretization from $L = 3$ to $L = 32$ time steps (in other words, we increase the network depth from 3 to 32 layers) using three different strategies:

(A1) *Abrupt refinement*: We refine from shallow ($L = 3$) to deep ($L = 32$) at iteration $k = 250$.
(A2) *Fast refinement*: We add 10 layers every 50 iterations.
(A3) *Slow refinement*: We add 10 layers every 100 iterations.

Our goal is to compare the behavior of these refinement strategies with the non-adaptive training of a shallow and a deep network having $L = 3$ and $L = 32$ layers respectively.

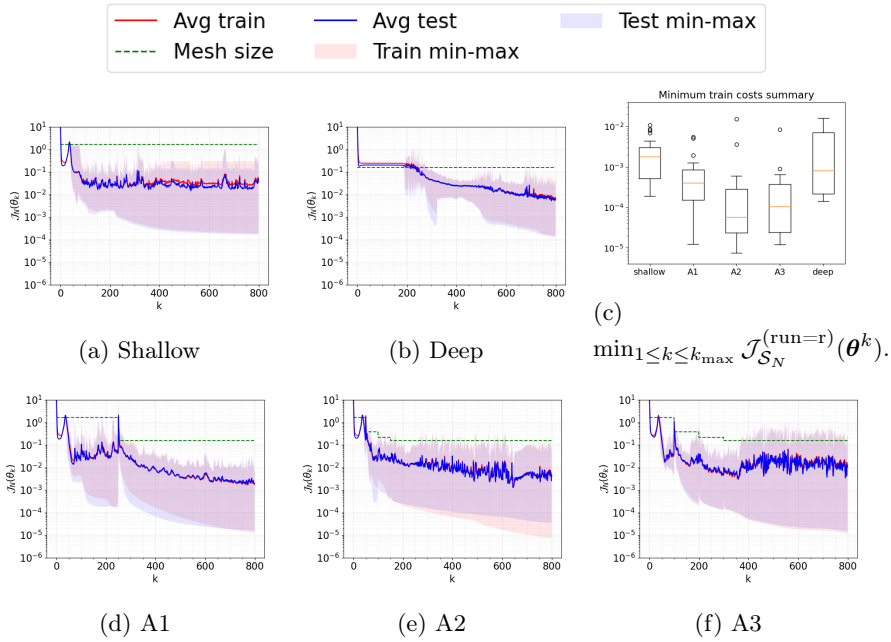
At each iteration $k \geq 0$ of A-MSA, the maximization step is performed with NumPy's L-BFGS-B optimizer, which is an iterative algorithm suited for nonconvex optimization problems. The optimizer is initialized by picking

the best candidate among a list Θ^0 of vectors, including the last best control $\boldsymbol{\theta}^k_{\text{best}} = \arg\min_{0\le i<k} \mathcal{J}_{\mathcal{S}_N}(\boldsymbol{\theta}^i)$ among all previous iterations before k and random perturbations of it, precisely

$$\Theta^0 := \bigcup_{q=0}^{5} \bigcup_{i=1}^{25} \{\boldsymbol{\theta}^k_{\text{best}} + 10^{-2q}\texttt{rand}(\alpha_{\min}, \alpha_{\max}), 10^{-2q}\texttt{rand}(\alpha_{\min}, \alpha_{\max})\}$$

where $\texttt{rand}(\alpha_{\min}, \alpha_{\max})$ is a uniform real distribution function in the interval $[\alpha_{\min}, \alpha_{\max}]$. As a result, the final output of the algorithm is random due to the choice of the initial guess. For this reason, we make $\bar{r} = 20$ runs of A-MSA, each one consisting in $k_{\max} = 800$ iterations.

In Figure 1, we first fix $N = 20$ samples and we plot the convergence history of the train loss $\mathcal{J}^{(\text{run=r})}_{\mathcal{S}_N}(\boldsymbol{\theta}^k)$ (see red colors). We also plot the generalization error via the *test loss* across the iterations k (see blue colors). The test loss is the quantity $\mathcal{J}_{\widetilde{\mathcal{S}}_N}(\boldsymbol{\theta}^k)$, defined in (3.4), where we use a test set $\widetilde{\mathcal{S}}_N$ different from the training set $\mathcal{S}_N$. $\widetilde{\mathcal{S}}_N$ is taken uniformly random at each iteration $1 \le k \le k_{\max}$. Since the history depends on each run r, the figure shows some statistics related to the repetitions. The continuous red curve shows the average value $\mathcal{J}^{(\text{av})}_{\mathcal{S}_N,k} := (1/\bar{r})\sum_{r=1}^{\bar{r}} \mathcal{J}^{(\text{run=r})}_{\mathcal{S}_N}(\boldsymbol{\theta}^k)$ over the runs and the red diffuse color shows its distribution around the average. The same presentation is given in blue for the generalization errors. We can first observe that the convergence history is, in average, relatively similar for all architectures. The average value of the loss function does not go below 10^{-2} for the shallow network (figure 1a), and the value reached by the deep network is only slightly better (figure 1b): at the end of the iterations, the cost function is in average below 7.10^{-2}. Similar values are reached by any of our three adaptive strategies (see figures 1d, 1e and 1f). As a consequence, if we take this average value as the performance indicator, we may conclude that the approximation power of the deep network is only marginally better than the shallow one. We may also conclude that the adaptive training strategy reduces to some extend the computational time for training (as we illustrate further on in section 9.4) but it does not bring extra approximation power. However, note that this indicator is not entirely appropriate because it does not correspond to any realized convergence history and, more importantly, because it does not inform about the best performance that we have at hand. It is crucial to remark that the variance around the average convergence value is significantly larger in the adaptive strategies compared to the non adaptive ones. This is particularly true for (A2) and (A3). Figure 1c shows that the minimal value $\min_{1\le k\le k_{\max}} \mathcal{J}^{(\text{run=r})}_{\mathcal{S}_N}(\boldsymbol{\theta}^k)$ reached during the $k_{\max} = 800$ iterations of each run is, in average, one order of magnitude lower than the one of the non-adaptive coarse and the non-adaptive deep network. This result illustrates that the adaptive strategy significantly contributes to reach better minima, thus allowing to benefit better from the higher approximation power of the deeper neural network.



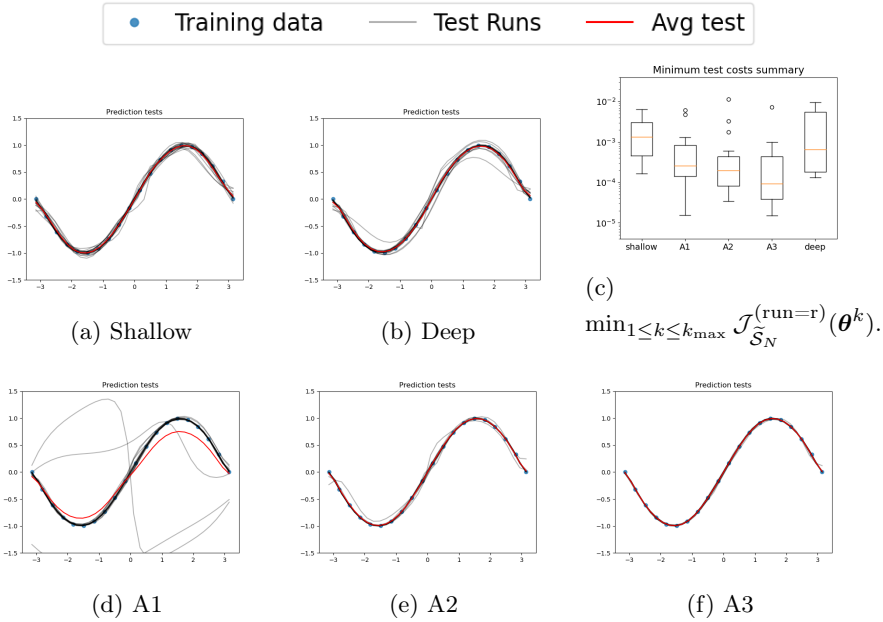
(a) Shallow (b) Deep (c) $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}^k)$.

(d) A1 (e) A2 (f) A3

Fig. 1: Sine function. Training loss $\mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}_k)$ (red) and test loss $\mathcal{J}_{\widetilde{\mathcal{S}}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}_k)$ (blue). Here $N = 20$ and $\bar{r} = 20$ runs.

Figure 2 shows the reconstruction of the sine function with the controls performing best in each training run, namely, realizing $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}^k)$ for every r. We see that the best over all realizations yields a very satisfactory approximation. Statistics on the generalization errors of these best controls are given in Figure 2c

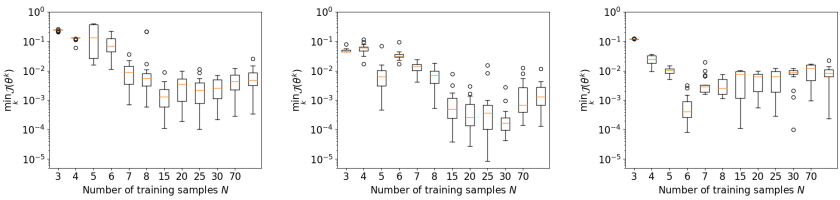
We can next examine in Figure 3 the impact of the number N of data samples in the generalization errors of the best controls. As one can expect, the generalization errors decreases when the number N of samples increases. For $N \geq 20$, the errors with the shallow network stagnate before reaching 10^{-3} in average regardless of the number of samples. This illustrates the limitations in the approximation power of the shallow network in the current example. The non-adaptive deep network ($L = 32$) presents a behavior which is only better for a reduced number of samples $N \leq 8$ but overall its performance is relatively similar to the coarse neural network. This comes from the difficulty in finding good quality optimizers in networks involving many coefficients. If we now adaptively train the deep network with strategy (A2), the generalization errors are in general lower than the ones reached by the non-adaptive strategies, especially when $N \geq 15$. The result illustrates that the adaptive strategy gives in average a better performance in terms of approximation. This is obtained at a reduced computing time as we show further on.



(a) Shallow (b) Deep (c) $\min_{1\le k\le k_{\max}} \mathcal{J}_{\widetilde{\mathcal{S}}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}^k)$.

(d) A1 (e) A2 (f) A3

Fig. 2: Sine function: Predictions of $\bar{r} = 20$ runs and average prediction.



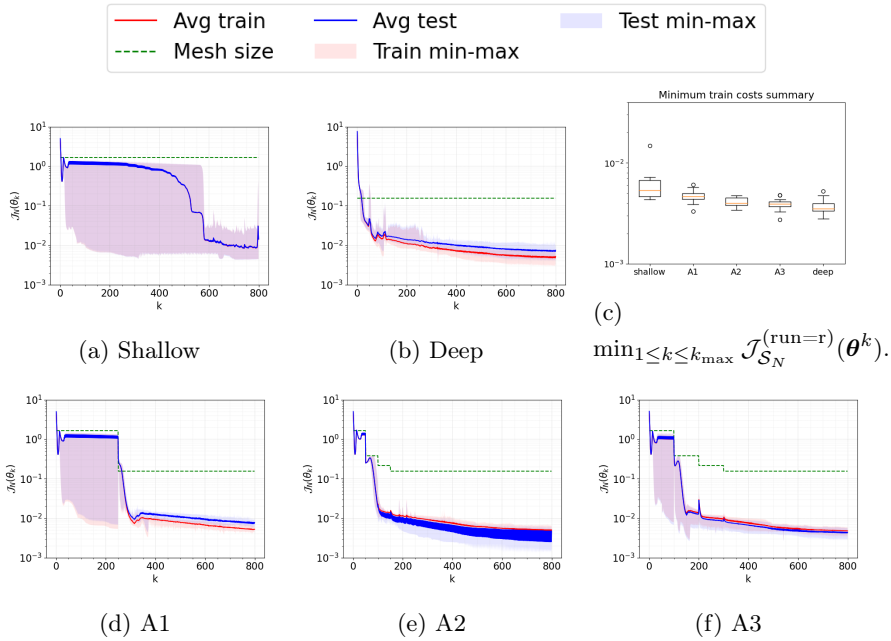
(a) Shallow: 3 layers (b) A2: $3 \to 32$ layers. (c) Deep: 32 layers

Fig. 3: Sine case – Generalization errors vs N for coarse, adaptive (A2) and thin networks

9.2 Noisy step function

We next examine the robustness of the method against noise.We consider the graph of the step function,

$$y : X = [-1, 1] \to Y = [-0.5, 0.5], \qquad x \mapsto y(x) = \begin{cases} 0.5 & \text{if } x \le 0 \\ -0.5 & \text{if } x > 0 \end{cases}$$



(a) Shallow (b) Deep (c) $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}^k)$.

(d) A1 (e) A2 (f) A3

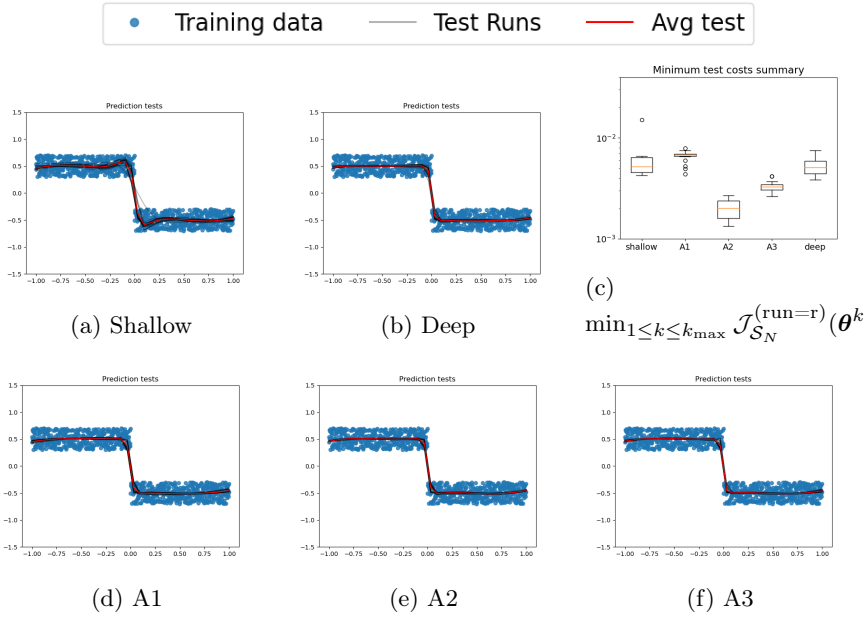
Fig. 4: Noisy step: Loss $\mathcal{J}_N^{(\mathrm{run=r})}(\boldsymbol{\theta}_k)$ for $N = 800$ and $\bar{r} = 20$ runs.

and we seek to approximate it when the given data are noisy. For each sample i, we get the pair (x_i, y_i) with

$$y_i = y(x_i) + \varepsilon_i, \quad i = 1, \ldots, N,$$

and the ε_i follow a uniform distribution in $[-0.2, 0.2]$.

We consider the same setting as before ($T = 5$, $\rho = 5$, $d = 3$) and fix $N = 800$. Figure 4 shows the convergence history of the training with the shallow, deep and adaptively deep neural networks. We consider the same adaptive strategies as before. All approaches reach more or less the same minimal value in the training (see Figure 4c. However, we observe that (A2) and (A3) give better generalization errors than the non-adaptive deep architecture (see Figure 5c). Note however that this superiority is not very large and all methods produce excellent reconstruction results as Figure 5 illustrates. We think that this is due to the fact that in this case all methods find configurations that are very close the global optimum, which is here given by the approximation error of the mapping $x \mapsto y(x)$.



(a) Shallow (b) Deep (c) $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run}=\mathrm{r})}(\boldsymbol{\theta}^k)$.

(d) A1 (e) A2 (f) A3

Fig. 5: Noisy step: Predictions of some runs.

9.3 Classification

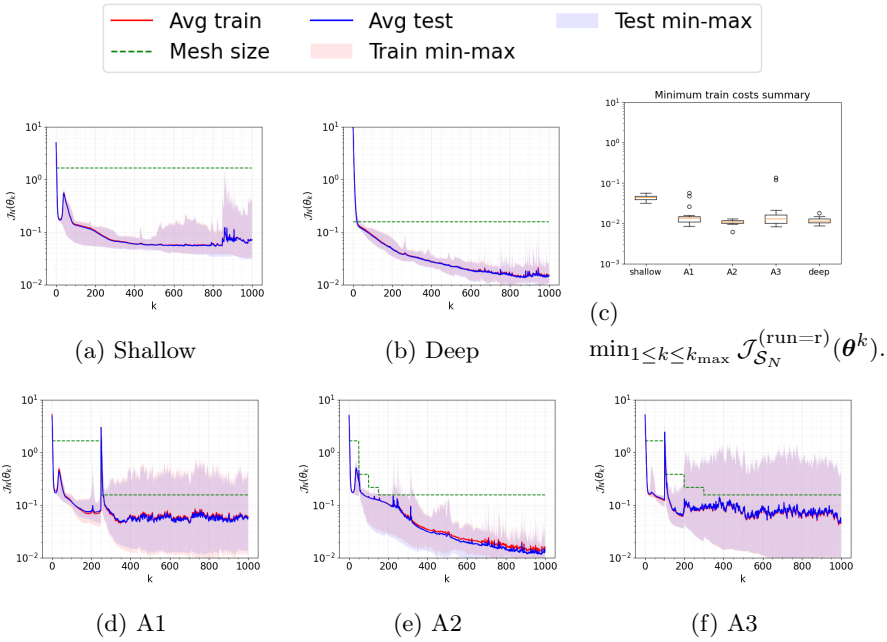
As a last example, we consider a simple 2D classification problem where the function to approximate is $y : X = [-1, 1]^2 \to Y = \{0, 1\}$ s.t.

$$y(x_1, x_2) = \begin{cases} 1 & \text{if } x_1^2 + x_2^2 \leq (0.5)^2, \\ 0 & \text{otherwise.} \end{cases}$$

The setting is the same as in the above examples but with slight changes. Each layer has now $d = 2 \times 3 = 6$ neurons per layers, the constraint on the controls is set to $[\alpha_{\min}, \alpha_{\max}] = [-2, 2]$ and the output function g is defined as

$$g : \mathbb{R}^d \to \mathbb{R}, \quad z = (z_1, \ldots, z_d) \mapsto g(z) \coloneqq \mathcal{H}\left(\frac{1}{d}\sum_{i=1}^{d} z_i\right)$$

where $\mathcal{H}$ is the usual Heaviside step function. The figure layout of the results is the same as in the other examples. Figures 6 and 7 show statistics on the loss function and examples of reconstruction. The blue-to-red scalar field in Fig 5 is the output of the trained network composed with a treshold filter $\mathcal{F} : x \mapsto \mathcal{H}(x - 0.5)$ from 1024 inputs points. The black crosses ($\times$) and white dots ($\circ$) are the $N = 800$ training points used for each case, the first refers to points with $z = 0$, the second to $z = 1$. Essentially, we observe that



(a) Shallow (b) Deep (c) $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run}=\mathrm{r})}(\boldsymbol{\theta}^k)$.

(d) A1 (e) A2 (f) A3

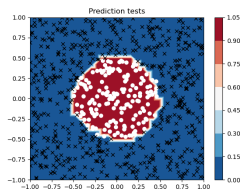
Fig. 6: Classification: Loss $\mathcal{J}_N^{(\mathrm{run}=\mathrm{r})}(\boldsymbol{\theta}_k)$ for $N = 800$ and 20 runs.

A-MSA yields the same quality of approximation as the non-adaptive deep neural network.

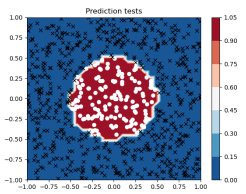
9.4 Computational times

We finish the section on numerical tests by examining the computational time in the learning approach. We present runtimes only for the noisy step example for the sake of brevity (similar results are obtained for the other tests). We consider three different criteria:

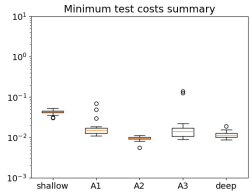
- **Complexity index (see Figure 8a):** we estimate the number of operations by measuring the runtime of a sequential run.
- **Parallel runtime index (see Figure 8b):** we measure the runtime of running in parallel the Hamiltonian maximization step of the learning algorithm. We place ourselves in a scenario with no constraints in the computing ressources and allocate one processor per layer. Note that the number of layers is increased in the adaptive approach so we use more and more ressources as the algorithm makes refinements.
- **Energy consumption index (see Figure 8a):** we estimate the total energy consumption of a parallel run as follows. For each iteration of the algorithm, we add the corresponding computing times of all the processors involved in that iteration. This gives an estimate in `cpu.seconds` of the energy consumption for each iteration. By summing over all iterations, we



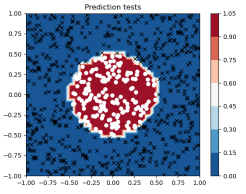
(a) Shallow



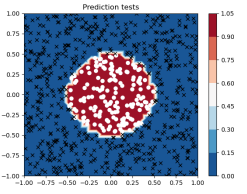
(b) Deep



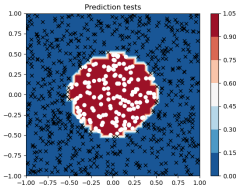
(c) $\min_{1\leq k\leq k_{\max}} \mathcal{J}_{\mathcal{S}_N}^{(\mathrm{run=r})}(\boldsymbol{\theta}^k)$.



(d) A1



(e) A2



(f) A3

Fig. 7: Classification: Predictions of $\bar{r} = 20$ runs and average prediction. Data labels: $\boxed{\times : 0}$, $\boxed{\circ : 1}$.

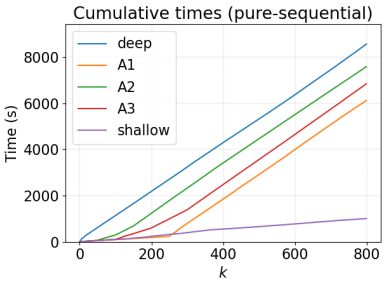
obtain the final estimate. This quantity is in fact equal to the cumulative sequential runtime which is given on Figure 8a.

Our runtimes have been measured on a cluster and we have observed that its occupation greatly impacts on the runtime results. To illustrate this issue, we show runtimes for the training of the shallow neural network in an empty node and a busy node, i.e. a node where others jobs are running in parallel. We have run the rest of the examples on empty nodes.

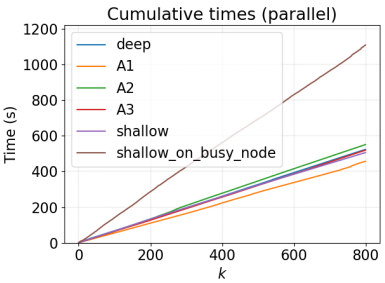
If we first consider Figure 8a, we see that the adaptive training strategy performs better than the non-adaptive one regarding the sequential runtimes and the energy consumption. This is comes as no surprise since the adaptive strategy performs less operations at the beginning of the iterations since they involve less layers. When computations are run in parallel, Figure 8b) shows that the run times are all very similar for all strategies. This is due to the fact that the tasks are well-balanced between processors since each of them does computations for one layer. We also observe that runtimes are greatly affected by the occupation of the cluster.

10 Conclusion

We have proven that the A-MSA algorithm converges to an underlying continuous learning problem in the limit of the time discretization and of the the number of samples. The convergence analysis requires that the time propagations and Hamiltonian maximizations are performed at each step with



(a) Complexity and energy consumption.



(b) Parallel runtime.

Fig. 8: Computational times for the noisy step example.

increasingly tight accuracy tolerances. Since a sharp estimation of these tolerances is difficult to obtain, we have implemented A-MSA with certain refinement strategies in order to examine its potential. The numerical experiments reveal that the adaptive strategy helps to benefit in practice from the higher approximation properties of deep networks by mitigating overparametrization issues: our results show that adaptivity increases the chances to find better quality minimizers compared to the non-adaptive training of deep neural networks. In addition, it appears that the adaptive strategy is clearly more performant in terms of complexity and energy consumption compared to the non-adaptive training of a deep neural network.

Some limitations and natural questions that arise for future works are the following:

- Our theoretical results involve numerous bounds and some of them might be suboptimal. Certain constants involved are also difficult to estimate in practice. As a result, the theoretical convergence result cannot be used to derive fully explicit adaptive strategies for implementation. However it gives hints on how to proceed in practice, and we propose practical guidelines to perform adaptivity. In future works, it would be desirable to find a sharper analysis with constants that are easier to compute in order to obtain a fully closed pipeline. This may however require very different arguments as our current ones.
- As we will explain in the next section, the optimal control point of view only describes certain classes of neural networks since it introduces certain restrictions in the architecture. As a result, our adaptive setting cannot automate the training of all existing classes of neural networks. So a natural question is how to extend the current ideas to other training techniques that are not based on optimal control.
- One important question is whether the flow map of a given dynamical system can represent the data at hand. The same issue also arises in the classical approach of deep learning in terms of the optimality of the choice

of the network. From the optimal control perspective, this representability problem can be viewed as a controllability problem. It would be interesting to transfer controllability results to the current deep learning framework to address this question.

Acknowledgments and Disclosure of Funding

This research has been funded by the Emergences Project of the Paris city council called "Models and Measures".

Declarations

Conflict of interest The authors declare no competing interests

References

[1] Hebb, D. O. The Organization of Behavior: A Neuropsychological Theory. New York: John Wiley and Sons, Inc., 1949. vol. 34, pp. 336–337. Wiley (1950). https://doi.org/10.1002/sce.37303405110. https://doi.org/10.1002/sce.37303405110

[2] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65–386 (1958)

[3] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

[4] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)

[5] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.*: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484 (2016)

[6] Yarotsky, D.: Error bounds for approximations with deep ReLU networks. Neural Networks **94**, 103–114 (2017)

[7] Daubechies, I., DeVore, R., Foucart, S., Hanin, B., Petrova, G.: Nonlinear approximation and (deep) relu networks. arXiv preprint arXiv:1905.02199 (2019)

[8] Grohs, P., Perekrestenko, D., Elbrächter, D., Bölcskei, H.: Deep neural network approximation theory. arXiv preprint arXiv:1901.02220 (2019)

[9] Gühring, I., Kutyniok, G., Petersen, P.: Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. Analysis and Applications **18**(05), 803–859 (2019). https://doi.org/10.1142/s0219530519410021

[10] Telgarsky, M.: Representation benefits of deep feedforward networks. arXiv preprint arXiv:1509.08101 (2015)

[11] Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks **5**(1), 54–65 (1994). https://doi.org/10.1109/72.265960

[12] Sun, Y., Xue, B., Zhang, M., Yen, G.G., Lv, J.: Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Transactions on Cybernetics **50**, 3840–3854 (2018)

[13] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in neural information processing systems **31** (2018)

[14] Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics, 400–407 (1951)

[15] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Physica-Verlag HD, Paris, France (2010). https://doi.org/10.1007/978-3-7908-2604-3_16

[16] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(Jul), 2121–2159 (2011)

[17] Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)

[18] Chen, T., Goodfellow, I., Shlens, J.: Net2net: Accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641 (2015)

[19] Wei, T., Wang, C., Rui, Y., Chen, C.W.: Network morphism. In: International Conference on Machine Learning, pp. 564–572 (2016)

[20] Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)

[21] Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., Yang, S.: Adanet: Adaptive structural learning of artificial neural networks. In: Proceedings

of the 34th International Conference on Machine Learning-Volume 70, pp. 874–883 (2017). JMLR. org

[22] Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. arXiv preprint arXiv:1808.05377 (2018)

[23] Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H., Jiang, Y.: Adaptive deep models for incremental learning: Considering capacity scalability and sustainability. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 74–82 (2019)

[24] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[25] Boltyanskii, V.G., Gamkrelidze, R.V., Pontryagin, L.S.: The theory of optimal processes. i. the maximum principle. Technical report, TRW SPACE TECHNOLOGY LABS LOS ANGELES CALIF (1960)

[26] Pontryagin, L.S.: Mathematical Theory of Optimal Processes. Routledge, U.K. (2018). https://doi.org/10.1201/9780203749319

[27] LeCun, Y., Touresky, D., Hinton, G., Sejnowski, T.: A theoretical framework for back-propagation. In: Proceedings of the 1988 Connectionist Models Summer School, vol. 1, pp. 21–28 (1988). CMU, Pittsburgh, Pa: Morgan Kaufmann

[28] Pineda, F.J.: Generalization of back propagation to recurrent and higher order neural networks. In: Neural Information Processing Systems, pp. 602–611 (1988)

[29] Weinan, E.: A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics **5**(1), 1–11 (2017). https://doi.org/10.1007/s40304-017-0103-z

[30] Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Problems **34**(1), 014004 (2018)

[31] Li, Q., Chen, L., Tai, C., E, W.: Maximum principle based algorithms for deep learning. The Journal of Machine Learning Research **18**(1), 5998–6026 (2018)

[32] Benning, M., Celledoni, E., Ehrhardt, M.J., Owren, B., Schönlieb, C..: Deep learning as optimal control problems: Models and numerical methods. Journal of Computational Dynamics **6**(2), 171–198 (2019). https://doi.org/10.3934/jcd.2019009

[33] Vialard, F.-X., Kwitt, R., Wei, S., Niethammer, M.: A Shooting Formulation of Deep Learning (2020)

[34] Ayyubi, H.A., Yao, Y., Divakaran, A.: Progressive growing of neural odes. arXiv preprint arXiv:2003.03695 (2020)

[35] Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: International Conference on Machine Learning, pp. 3276–3285 (2018). PMLR

[36] He, X., Mo, Z., Wang, P., Liu, Y., Yang, M., Cheng, J.: Ode-inspired network design for single image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1732–1741 (2019)

[37] Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., Holtham, E.: Reversible architectures for arbitrarily deep residual neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

[38] Zhang, T., Yao, Z., Gholami, A., Gonzalez, J.E., Keutzer, K., Mahoney, M.W., Biros, G.: Anodev2: A coupled neural ode framework. Advances in Neural Information Processing Systems **32** (2019)

[39] Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., Duncan, J.: Adaptive checkpoint adjoint method for gradient estimation in neural ode. In: International Conference on Machine Learning, pp. 11639–11649 (2020). PMLR

[40] E, W., Han, J., Li, Q.: A mean-field optimal control formulation of deep learning. Research in the Mathematical Sciences **6**(1), 10 (2019)

[41] Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 3282–3291. PMLR, Stockholmsmässan, Stockholm Sweden (2018). http://proceedings.mlr.press/v80/lu18d.html

[42] Gomez, A.N., Ren, M., Urtasun, R., Grosse, R.B.: The reversible residual network: Backpropagation without storing activations. In: Advances in Neural Information Processing Systems, pp. 2214–2224 (2017)

[43] Zhang, X., Li, Z., Change Loy, C., Lin, D.: Polynet: A pursuit of structural diversity in very deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 718–726 (2017)

[44] Larsson, G., Maire, M., Shakhnarovich, G.: Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648 (2016)

[45] Bertsekas, D.P.: Dynamic Programming and Optimal Control vol. 1. Athena scientific Belmont, MA, U.S. (1995)

[46] Clarke, F.: The maximum principle in optimal control, then and now. Control and Cybernetics **34**(3), 709 (2005)

[47] Athans, M., Falb, P.L.: Optimal Control: an Introduction to the Theory and Its Applications. Dover Publications Inc., U.S. (2013)

[48] Bressan, A., Piccoli, B.: Introduction to the Mathematical Theory of Control vol. 1. American institute of mathematical sciences Springfield, U.S. (2007)

[49] Chernousko, F.L., Lyubushin, A.A.: Method of successive approximations for solution of optimal control problems. Optimal Control Applications and Methods **3**(2), 101–114 (1982)

[50] Akrivis, G., Makridakis, C., , Nochetto, R.H.: Optimal order a posteriori error estimates for a class of runge–kutta and galerkin methods. Numerische Mathematik **114**(1), 133 (2009). https://doi.org/10.1007/s00211-009-0254-2

[51] Pinelis, I.F., Sakhanenko, A.I.: Remarks on inequalities for large deviation probabilities. Theory of Probability & Its Applications **30**(1), 143–148 (1986)

[52] Günther, S., Ruthotto, L., Schroder, J.B., Cyr, E.C., Gauger, N.R.: Layer-parallel training of deep residual neural networks. SIAM Journal on Mathematics of Data Science **2**(1), 1–23 (2020)

[53] Lions, J.L., Maday, Y., Turinici, G.: Résolution d'EDP par un schéma en temps pararéel. C. R. Acad. Sci. Paris (2001). t. 332, Série I, p. 661-668

[54] Maday, Y., Mula, O.: An adaptive parareal algorithm. Journal of Computational and Applied Mathematics **377**, 112915 (2020). https://doi.org/10.1016/j.cam.2020.112915

A Proof of Lemma 6.1

The proof of Lemma 6.1 makes use of certain bounds which we gather in the following Proposition A.1. They are derived by an immediate extension of the proofs given in [31] so we just sketch the proof for self-completness of the current work.

Proposition A.1. *We have the following bounds for all $t \in [0,T]$,*

$$\|p_t^{\boldsymbol{\theta},\zeta}\| \leq K' \coloneqq Ke^{\|\Pi_\zeta\|KT} \tag{A.1}$$

$$\|\delta u_t\| \leq K_{\delta u} \coloneqq \left(\sqrt{T}(\eta+\zeta) + \int_0^T \|f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\right) e^{KT} \tag{A.2}$$

$$\begin{aligned}\|\delta p_t\| \leq K_{\delta p} \coloneqq{}& \sqrt{T}e^{2KT}(1+K(1+K'T))(\eta+\zeta)\\ &+ e^{2KT}K(1+K'T)\int_0^T \|f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\\ &+ e^{KT}\int_0^T \|\nabla_u H(s,u_s^{\boldsymbol{\theta},\zeta},p_s^{\boldsymbol{\theta},\zeta},\varphi_s) - \nabla_u H(s,u_s^{\boldsymbol{\theta},\eta},p_s^{\boldsymbol{\theta},\eta},\theta_s)\|\mathrm{d}s\end{aligned} \tag{A.3}$$

Proof of Proposition A.1 Inequality (A.1) follows from the fact that

$$\dot{p}_t^{\boldsymbol{\theta},\zeta} = \Pi_\zeta\left(-p_t^{\boldsymbol{\theta},\zeta}\cdot\nabla_u f(u_t^{\boldsymbol{\theta},\zeta},\theta_t)\right)$$

thus taking the scalar product with $p_t^{\boldsymbol{\theta},\zeta}$ and using the Cauchy-Schwarz inequality, we derive the evolution

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|p_t^{\boldsymbol{\theta},\zeta}\|^2 \leq \|\Pi_\zeta\|\|\nabla_u f(u_t^{\boldsymbol{\theta},\zeta},\theta_t)\|_2\|p_t^{\boldsymbol{\theta},\zeta}\|^2, \quad \|p_T^{\boldsymbol{\theta},\zeta}\|^2 = \|\nabla_u\Phi(u_T^{\boldsymbol{\theta},\zeta},y)\|^2$$

and the result follows by applying the Grönwall's inequality and the fact that $\|\nabla_u\Phi(u_T^{\boldsymbol{\theta},\zeta},y)\| \leq K$ and $\|\nabla_u f(u_t^{\boldsymbol{\theta},\zeta},\theta_t)\|_2 \leq K$ by hypothesis (A1) and (A2).

To derive inequality (A.2), we start from

$$\delta\dot{u}_t = \dot{u}_t^{\boldsymbol{\varphi},\eta} - \dot{u}_t^{\boldsymbol{\theta},\zeta} = \Pi_\eta f(u_t^{\boldsymbol{\varphi},\eta},\varphi_t) - \Pi_\zeta f(u_t^{\boldsymbol{\theta},\zeta},\theta_t).$$

Integrating between 0 and $t \leq T$,

$$\delta u_t = \int_0^t \Pi_\eta f(s,u_s^{\boldsymbol{\varphi},\eta},\varphi_s) - \Pi_\zeta f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\mathrm{d}s$$

Taking norms,

$$\|\delta u_t\| \leq \int_0^t \|f(s,u_s^{\boldsymbol{\varphi},\eta},\varphi_s) - \Pi_\eta f(s,u_s^{\boldsymbol{\varphi},\eta},\varphi_s)\|\mathrm{d}s + \int_0^t \|f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s) - \Pi_\zeta f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s$$

$$\begin{aligned}&+ \int_0^t \|f(s,u_s^{\boldsymbol{\varphi},\eta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\\ &\leq \sqrt{T}(\eta+\zeta) + \int_0^T \|f(s,u_s^{\boldsymbol{\varphi},\eta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s)\|\mathrm{d}s\\ &\quad + \int_0^T \|f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\\ &\leq \sqrt{T}(\eta+\zeta) + K\int_0^T \|\delta u_s\|\mathrm{d}s + \int_0^T \|f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\end{aligned}$$

Thus by Grönwall's inequality,

$$\|\delta u_t\| \leq K_{\delta u} \coloneqq \left(\sqrt{T}(\eta+\zeta) + \int_0^T \|f(s,u_s^{\boldsymbol{\theta},\zeta},\varphi_s) - f(s,u_s^{\boldsymbol{\theta},\zeta},\theta_s)\|\mathrm{d}s\right) e^{KT}$$

Proceeding in a similar manner with $\delta\dot{p}_t$, by integrating between T and t, we derive

$$\int_T^t \delta\dot{p}_s \mathrm{d}s = \delta p_t - \delta p_T = \int_T^t \Pi_\eta \nabla_u H(s, u_s^{\boldsymbol{\varphi},\eta}, p_s^{\boldsymbol{\varphi},\eta}, \varphi_s) - \Pi_\zeta \nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \theta_s)\mathrm{d}s$$

and similarly as before, we infer that

$$\begin{aligned}\|\delta p_t\| \leq \sqrt{T}(\eta+\zeta) + K\|\delta u_T\| + KK' \int_0^T \|\delta u_s\| + K\int_0^T \|\delta p_s\|\mathrm{d}s\mathrm{d}s \\ + \int_0^T \|\nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \varphi_s) - \nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \theta_s)\|\mathrm{d}s\end{aligned}$$

Using bounds (A.2) and (A.3), and applying the Gronwall inequality, we derive the announced estimate

$$\|\delta p_t\| \leq K_{\delta p}$$

with

$$\begin{aligned}K_{\delta p} &= e^{KT}\Big(\sqrt{T}(\eta+\zeta) + KK_{\delta u}(1+K'T) \\ &\quad + \int_0^T \|\nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \varphi_s) - \nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \theta_s)\|\mathrm{d}s\Big) \\ &= \sqrt{T}e^{2KT}(1+K(1+K'T))(\eta+\zeta) \\ &\quad + e^{2KT}K(1+K'T)\int_0^T \|f(s, u_s^{\boldsymbol{\theta},\zeta}, \varphi_s) - f(s, u_s^{\boldsymbol{\theta},\zeta}, \theta_s)\|\mathrm{d}s \\ &\quad + e^{KT}\int_0^T \|\nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \varphi_s) - \nabla_u H(s, u_s^{\boldsymbol{\theta},\zeta}, p_s^{\boldsymbol{\theta},\zeta}, \theta_s)\|\mathrm{d}s\end{aligned}$$

□

We can now prove Lemma 6.1.

Proof of Lemma 6.1 By definition (4.1) of the Hamiltonian, for any $\boldsymbol{\theta} \in L^\infty([0,T],\Theta)$,

$$I(\boldsymbol{u}^{\boldsymbol{\theta}}, \boldsymbol{p}^{\boldsymbol{\theta}}, \boldsymbol{\theta}) \coloneqq \int_0^T \Big(p_t^{\boldsymbol{\theta}} \cdot f(u_t^{\boldsymbol{\theta}}, \theta_t) - H(t, u_t^{\boldsymbol{\theta}}, p_t^{\boldsymbol{\theta}}, \theta_t) - L(\theta_t)\Big)\,\mathrm{d}t = 0.$$

Thus

$$\begin{aligned}0 &= I(\boldsymbol{u}^{\boldsymbol{\varphi},\eta}, \boldsymbol{p}^{\boldsymbol{\varphi},\eta}, \boldsymbol{\varphi}) - I(\boldsymbol{u}^{\boldsymbol{\theta},\zeta}, \boldsymbol{p}^{\boldsymbol{\theta},\zeta}, \boldsymbol{\theta}) \qquad \text{(A.4)} \\ &= \underbrace{\int_0^T p_t^{\boldsymbol{\varphi},\eta} \cdot f(u_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - p_t^{\boldsymbol{\theta},\zeta} \cdot f(u_t^{\boldsymbol{\theta},\zeta}, \theta_t)\mathrm{d}t}_{:=\mathcal{I}_1} \\ &\quad - \underbrace{\int_0^T H(t, u_t^{\boldsymbol{\varphi},\eta}, p_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - H(t, u_t^{\boldsymbol{\theta},\zeta}, p_t^{\boldsymbol{\theta},\zeta}, \theta_t)\mathrm{d}t}_{:=\mathcal{I}_2} \\ &\quad - \underbrace{\int_0^T R(\varphi_t) - R(\theta_t)\mathrm{d}t}_{:=\mathcal{I}_3}\end{aligned}$$

Denoting

$$e_t^{f,\boldsymbol{\varphi},\eta} \coloneqq f(u_t^{\boldsymbol{\varphi},\eta},\varphi_t) - \Pi_\eta f(u_t^{\boldsymbol{\varphi},\eta},\varphi_t)$$

and similarly for $e_t^{f,\boldsymbol{\theta},\zeta}$, we have

$$\mathcal{I}_1 = \underbrace{\int_0^T p_t^{\boldsymbol{\varphi},\eta}\cdot\Pi_\eta f(u_t^{\boldsymbol{\varphi},\eta},\varphi_t) - p_t^{\boldsymbol{\theta},\zeta}\cdot\Pi_\zeta f(u_t^{\boldsymbol{\theta},\zeta},\theta_t)\mathrm{d}t}_{\coloneqq\mathcal{I}_{1,1}} + \underbrace{\int_0^T p_t^{\boldsymbol{\varphi},\eta}\cdot e_t^{f,\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta}\cdot e_t^{f,\boldsymbol{\theta},\zeta}\mathrm{d}t}_{\coloneqq\mathcal{I}_{1,2}}$$

In addition, since $\dot{u}_t^{\boldsymbol{\varphi},\eta} = \Pi_\eta f(u_t^{\boldsymbol{\varphi},\eta},\varphi_t)$ and similarly for $\boldsymbol{\theta}$, it follows that

$$\mathcal{I}_{1,1} = \int_0^T p_t^{\boldsymbol{\varphi},\eta}\cdot\dot{u}_t^{\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta}\cdot\dot{u}_t^{\boldsymbol{\theta},\zeta}\mathrm{d}t = \underbrace{\int_0^T p_t^{\boldsymbol{\theta},\zeta}\cdot\delta\dot{u}_t + \delta p_t\cdot\dot{u}_t^{\boldsymbol{\theta},\zeta}\mathrm{d}t}_{\coloneqq\mathcal{I}_{1,1,1}} + \underbrace{\int_0^T \delta p_t\cdot\delta\dot{u}_t\mathrm{d}t}_{\coloneqq\mathcal{I}_{1,1,2}}$$

where

$$\delta u_t \coloneqq u_t^{\boldsymbol{\varphi},\eta} - u_t^{\boldsymbol{\theta},\zeta}, \quad \text{and} \quad \delta p_t \coloneqq p_t^{\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta}.$$

By integration by parts, and the fact that $\dot{p}_t^{\boldsymbol{\theta},\zeta} = -\Pi_\zeta\nabla_u H(t,u_t^{\boldsymbol{\theta},\zeta},p_t^{\boldsymbol{\theta},\zeta},\theta_t)$, we have

$$\begin{aligned}
\mathcal{I}_{1,1,1} &= p_t^{\boldsymbol{\theta},\zeta}\cdot\delta u_t\Big|_0^T + \int_0^T \delta p_t\cdot\dot{u}_t^{\boldsymbol{\theta},\zeta} - \dot{p}_t^{\boldsymbol{\theta},\zeta}\cdot\delta u_t\mathrm{d}t \\
&= p_t^{\boldsymbol{\theta},\zeta}\cdot\delta u_t\Big|_0^T + \int_0^T \delta u_t\cdot\Pi_\zeta\nabla_u H(t,u_t^{\boldsymbol{\theta},\zeta},p_t^{\boldsymbol{\theta},\zeta},\theta_t) + \delta p_t\cdot\Pi_\zeta\nabla_p H(t,u_t^{\boldsymbol{\theta},\zeta},p_t^{\boldsymbol{\theta},\zeta},\theta_t)\mathrm{d}t \\
&= p_t^{\boldsymbol{\theta},\zeta}\cdot\delta u_t\Big|_0^T + \int_0^T \Pi_\zeta\nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t)\cdot\delta w_t\mathrm{d}t,
\end{aligned}$$

where we have used the shorthand notation $w = (u,p)$ in the last line. This notation will also be used in what follows.

We next address the intergral $\mathcal{I}_{1,1,2}$. By integration by parts,

$$\begin{aligned}
\mathcal{I}_{1,1,2} &= \frac{1}{2}\int_0^T \delta p_t\cdot\delta\dot{u}_t\mathrm{d}t + \frac{1}{2}\int_0^T \delta p_t\cdot\delta\dot{u}_t\mathrm{d}t \\
&= \frac{1}{2}\delta p_t\cdot\delta u_t\Big|_0^T - \frac{1}{2}\int_0^T \delta\dot{p}_t\cdot\delta u_t\mathrm{d}t + \frac{1}{2}\int_0^T \delta p_t\cdot\delta\dot{u}_t\mathrm{d}t \\
&= \frac{1}{2}\delta p_t\cdot\delta u_t\Big|_0^T + \frac{1}{2}\int_0^T \Big(\Pi_\eta\nabla_w H(t,w_t^{\boldsymbol{\varphi},\eta},\varphi_t) - \Pi_\zeta\nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t)\Big)\cdot\delta w_t\mathrm{d}t
\end{aligned}$$

Adding and subtracting $\nabla_w H(t,w_t^{\boldsymbol{\varphi},\eta},\varphi_t)$ and $\nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t)$ to the last formula,

$$\begin{aligned}
\mathcal{I}_{1,1,2} = \frac{1}{2}\delta p_t\cdot\delta u_t\Big|_0^T &+ \frac{1}{2}\int_0^T \big(\Pi_\eta\nabla_w H(t,w_t^{\boldsymbol{\varphi},\eta},\varphi_t) - \nabla_w H(t,w_t^{\boldsymbol{\varphi},\eta},\varphi_t)\big)\cdot\delta w_t\mathrm{d}t \\
&+ \frac{1}{2}\int_0^T \Big(\nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t) - \Pi_\zeta\nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t)\Big)\cdot\delta w_t\mathrm{d}t \\
&+ \frac{1}{2}\int_0^T \Big(\nabla_w H(t,w_t^{\boldsymbol{\varphi},\eta},\varphi_t) - \nabla_w H(t,w_t^{\boldsymbol{\theta},\zeta},\theta_t)\Big)\cdot\delta w_t\mathrm{d}t
\end{aligned}$$

By applying Taylor's theorem around $w = w^{\boldsymbol{\theta},\zeta} = (u_t^{\boldsymbol{\theta},\zeta}, p_t^{\boldsymbol{\theta},\zeta})$ to the function $w \mapsto \nabla_w H(t, w, \varphi_t)$, there exists $r_1(t) \in [0,1]$ such that

$$\nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) = \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) + \delta w_t \cdot \nabla_w^2 H(t, w_t^{\boldsymbol{\theta},\zeta} + r_1(t)\delta w_t, \varphi_t)$$

Thus

$$\begin{aligned}
\mathcal{I}_{1,1,2} = &\frac{1}{2}\delta p_t \cdot \delta u_t\Big|_0^T + \frac{1}{2}\int_0^T \left(\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\right) \cdot \delta w_t \mathrm{d}t \\
&+ \frac{1}{2}\int_0^T \left(\nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t) - \Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)\right) \cdot \delta w_t \mathrm{d}t \\
&+ \frac{1}{2}\int_0^T \left(\nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)\right) \cdot \delta w_t \mathrm{d}t \\
&+ \frac{1}{2}\int_0^T \delta w_t \cdot \nabla_w^2 H(t, w_t^{\boldsymbol{\theta},\zeta} + r_1(t)\delta w_t, \varphi_t) \cdot \delta w_t \mathrm{d}t
\end{aligned}$$

Since $\delta u_0 = 0$, the boundary terms from $\mathcal{I}_{1,1,1}$ and $\mathcal{I}_{1,1,2}$ are

$$\begin{aligned}
\left(p_t^{\boldsymbol{\theta},\zeta} + \frac{1}{2}\delta p_t\right) \cdot \delta u_t\Big|_0^T &= \left(p_T^{\boldsymbol{\theta},\zeta} + \frac{1}{2}\delta p_T\right) \cdot \delta u_T \\
&= -\nabla\Phi(u_T^{\boldsymbol{\theta},\zeta}) \cdot \delta u_T - \frac{1}{2}\left(\nabla\Phi(u_T^{\boldsymbol{\varphi},\eta}) - \nabla\Phi(u_T^{\boldsymbol{\theta},\zeta})\right) \cdot \delta u_T
\end{aligned}$$

By applying Taylor's theorem around $z = u_t^{\boldsymbol{\theta},\zeta}$ to the function $z \mapsto \nabla\Phi(z)$, there exists $r_2 \in [0,1]$ such that

$$\nabla\Phi(u_T^{\boldsymbol{\varphi},\eta}) = \nabla\Phi(u_T^{\boldsymbol{\theta},\zeta}) + \delta u_t \cdot \nabla^2\Phi(u_T^{\boldsymbol{\theta},\zeta} + r_2\delta u_T).$$

Therefore

$$\left(p_t^{\boldsymbol{\theta},\zeta} + \frac{1}{2}\delta p_t\right) \cdot \delta u_t\Big|_0^T = -\nabla\Phi(u_T^{\boldsymbol{\theta},\zeta}) \cdot \delta u_T - \frac{1}{2}\delta u_T \cdot \nabla^2\Phi(u_T^{\boldsymbol{\theta},\zeta} + r_2\delta u_T) \cdot \delta u_T$$

Since, again by Taylor's theorem, there exists $r_3 \in [0,1]$ such that

$$\Phi(p_T^{\boldsymbol{\varphi},\eta}) = \Phi(p_T^{\boldsymbol{\theta},\zeta}) + \delta u_t \cdot \nabla\Phi(u_T^{\boldsymbol{\theta},\zeta}) + \frac{1}{2}\delta u_t \cdot \nabla^2\Phi(u_T^{\boldsymbol{\theta},\zeta} + r_3\delta u_t) \cdot \delta u_T,$$

we finally get the expression for the the boundary terms from $\mathcal{I}_{1,1,1}$ and $\mathcal{I}_{1,1,2}$

$$\begin{aligned}
\left(p_t^{\boldsymbol{\theta},\zeta} + \frac{1}{2}\delta p_t\right) \cdot \delta u_t\Big|_0^T = &\Phi(u_T^{\boldsymbol{\theta},\zeta}) - \Phi(u_T^{\boldsymbol{\varphi},\eta}) - \\
&\frac{1}{2}\delta u_T \cdot \left(\nabla^2\Phi(u_T^{\boldsymbol{\theta},\zeta} + r_2\delta u_T) - \nabla^2\Phi(u_T^{\boldsymbol{\theta},\zeta} + r_3\delta u_T^\eta)\right) \cdot \delta u_T
\end{aligned}$$

We now turn to $\mathcal{I}_2$. By Taylor's theorem, there exists $r_4(t) \in [0,1]$ such that

$$\begin{aligned}
\mathcal{I}_2 &= \int_0^T H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)\mathrm{d}t \\
&= \int_0^T \Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta}(t)\mathrm{d}t + \int_0^T \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) \cdot \delta w_t \mathrm{d}t \\
&\quad + \frac{1}{2}\int_0^T \delta w_t \cdot \nabla_w^2 H(t, w_t^{\boldsymbol{\theta},\zeta} + r_4(t)\delta w_t, \varphi_t) \cdot \delta w_t \mathrm{d}t,
\end{aligned}$$

where we remind that, as defined in equation (6.2), $\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta}(t) \coloneqq H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) - H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)$. Injecting the above expressions for the terms of $\mathcal{I}_1$ and $\mathcal{I}_2$ into equation

(A.4), passing the terms corresponding to the loss function to the left hand side, and adding and subtracting $\int_0^T \Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) \cdot \delta w_t \mathrm{d}t$,

$$
\begin{aligned}
&\mathrm{Loss}(x, y, \boldsymbol{\varphi}) - \mathrm{Loss}(x, y, \boldsymbol{\theta}) && \text{(Diff-Loss)}\\
&= \Phi(u_T^{\boldsymbol{\varphi},\eta}) - \Phi(u_T^{\boldsymbol{\theta},\zeta}) + \int_0^T \left(R(\varphi_t) - R(\theta_t)\right) \mathrm{d}t\\
&= \int_0^T \Big(\Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t) - \Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t)\Big) \cdot \delta w_t \mathrm{d}t && \text{(T1)}\\
&+ \int_0^T p_t^{\boldsymbol{\varphi},\eta} \cdot e_t^{f,\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta} \cdot e_t^{f,\boldsymbol{\theta},\zeta} \mathrm{d}t && \text{(T2)}\\
&+ \frac{1}{2}\int_0^T \left(\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\right) \cdot \delta w_t \mathrm{d}t && \text{(T3)}\\
&+ \frac{1}{2}\int_0^T \Big(\nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t) - \Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)\Big) \cdot \delta w_t \mathrm{d}t && \text{(T4)}\\
&+ \frac{1}{2}\int_0^T \Big(\nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t)\Big) \cdot \delta w_t \mathrm{d}t && \text{(T5)}\\
&+ \frac{1}{2}\int_0^T \delta w_t \cdot \nabla_w^2 H(t, w_t^{\boldsymbol{\theta},\zeta} + r_1(t)\delta w_t, \varphi_t) \cdot \delta w_t \mathrm{d}t && \text{(T6)}\\
&- \int_0^T \Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta}(t) \mathrm{d}t && \text{(T7)}\\
&+ \int_0^T \Big(\Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t)\Big) \cdot \delta w_t \mathrm{d}t && \text{(T8)}\\
&- \frac{1}{2}\int_0^T \delta w_t \cdot \nabla_w^2 H(t, w_t^{\boldsymbol{\theta},\zeta} + r_4(t)\delta w_t, \varphi_t) \cdot \delta w_t \mathrm{d}t && \text{(T9)}\\
&- \frac{1}{2}\delta u_T \cdot \Big(\nabla^2 \Phi(u_T^{\boldsymbol{\theta},\zeta} + r_2 \delta u_T) - \nabla^2 \Phi(u_T^{\boldsymbol{\theta},\zeta} + r_3 \delta u_T^\eta)\Big)
\end{aligned}
$$

We next derive a bound for the difference $\mathrm{Loss}(x, y, \boldsymbol{\varphi}) - \mathrm{Loss}(x, y, \boldsymbol{\theta})$. We proceed to bound all the terms by order of appearance in the above formula. We will sometimes use that by (A.2), (A.3) and Jensen's inequality,

$$
\|\delta w_t\|^2 = \|\delta u_t\|^2 + \|\delta p_t\|^2 \lesssim (\eta + \zeta)^2 + \int_0^T \|\nabla_w H(s, w_s^{\boldsymbol{\theta},\zeta}, \varphi_s) - \nabla_w H(s, w_s^{\boldsymbol{\theta},\zeta}, \theta_s)\|^2 \mathrm{d}s
$$

which yields

$$
\|\delta w_t\|_{L^2([0,T])} \lesssim (\eta + \zeta) + \|\nabla_w(\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T],\mathbb{R}^n \times \mathbb{R}^n)}. \tag{A.6}
$$

Term (T1)+(T5):

By the Cauchy-Schwartz inequality and (A.6), we can bound term (T1) by

$$
\begin{aligned}
&\int_0^T \Big(\Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t) - \Pi_\zeta \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t)\Big) \cdot \delta w_t \mathrm{d}t\\
&\le \|\Pi_\zeta\| \|\nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \theta_t) - \nabla_w H(t, w_t^{\boldsymbol{\theta},\zeta}, \varphi_t)\|_{L^2([0,T])} \|\delta w_t\|_{L^2([0,T])}\\
&\lesssim (\eta + \zeta) \|\nabla_w(\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T])} + \|\nabla_w(\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|^2_{L^2([0,T],\mathbb{R}^n \times \mathbb{R}^n)}
\end{aligned}
$$

Thus

$$
(T1) + (T5) \lesssim (\eta + \zeta) + (\eta + \zeta) \|\nabla_w(\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T])} + \|\nabla_w(\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|^2_{L^2([0,T])}
$$

Term (T2):

Using (A.1) and the fact that $\|e_t^{f,\boldsymbol{\theta},\zeta}\|_{L^2([0,T],\mathbb{R}^n)} \leq \zeta$

$$\int_0^T p_t^{\boldsymbol{\varphi},\eta} \cdot e_t^{f,\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta} \cdot e_t^{f,\boldsymbol{\theta},\zeta} \mathrm{d}t \lesssim \eta + \zeta$$

Terms (T3) and (T8):

By the Cauchy-Schwarz inequality, we can bound term (T3)

$$\begin{aligned}
&|\int_0^T \left(\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\right) \cdot \delta w_t \mathrm{d}t| \\
&\leq \|\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\|_{L^2([0,T],\mathbb{R}^n\times\mathbb{R}^n)} \|\delta w_t\|_{L^2([0,T],\mathbb{R}^n\times\mathbb{R}^n)} \\
&\lesssim \eta \|\nabla_w (\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T],\mathbb{R}^n\times\mathbb{R}^n)}
\end{aligned}$$

where we have used that the propagator Π_η guarantees

$$\|\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\|_{L^2([0,T],\mathbb{R}^n\times\mathbb{R}^n)}, \lesssim \eta$$

and finally

$$\begin{aligned}
&|\int_0^T \left(\Pi_\eta \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t) - \nabla_w H(t, w_t^{\boldsymbol{\varphi},\eta}, \varphi_t)\right) \cdot \delta w_t \mathrm{d}t| \\
&\qquad \lesssim \eta(\eta + \zeta) + \eta \|\nabla_w (\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T],\mathbb{R}^n\times\mathbb{R}^n)}
\end{aligned}$$

Term (T4):

Using (A.1) and the fact that $\|e_t^{f,\boldsymbol{\theta},\zeta}\|_{L^2([0,T],\mathbb{R}^n)} \leq \zeta$, we have $\int_0^T p_t^{\boldsymbol{\varphi},\eta} \cdot e_t^{f,\boldsymbol{\varphi},\eta} - p_t^{\boldsymbol{\theta},\zeta} \cdot e_t^{f,\boldsymbol{\theta},\zeta} \mathrm{d}t \lesssim \eta + \zeta$.

Term (T6), (T9) and (T10):

By assumptions (A1), (A2), all second derivative terms are bounded element-wise by some constant K. Hence, we have $\|\delta w_t \cdot A \cdot \delta w_t\| \lesssim \|\delta w_t\|^2$ for A being a second derivative matrix.

Summary:

By using the above inequalities, in formula (Diff-Loss), we deduce that there exists a constant $C > 0$ depending on T, $\|\Pi_\zeta\|$ and $\|\Pi_\eta\|$ such that, if $\eta \leq 1$,

$$\mathrm{Loss}(x, y, \boldsymbol{\varphi}) - \mathrm{Loss}(x, y, \boldsymbol{\theta}) \leq -\int_0^T \Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta}(t) \mathrm{d}t + C\left((\eta + \zeta)^2 + \|\nabla_w (\Delta H_{\boldsymbol{\varphi},\boldsymbol{\theta}}^{x,\zeta})\|_{L^2([0,T])}^2\right)$$

Note that the term $(\eta+\zeta)^2$ accounts for the fact that we are not solving exactly the dynamics. □