Acta Polytechnica **00**(0):1–9, 0000

PREPRINT 2024-12-09

OPTIMAL CONTROL BASED ALGORITHMS FOR DEEP LEARNING : MODELS AND NUMERICAL METHODS

Zakaria Baba[*], Rida Assalouh

Ecole Polytechnique CMAP , Palaisau

[*] corresponding author: `zakaria.baba@polytechnique.edu,rida.assalouh@polytechnique.edu`

Abstract. In this work we propose a new framework for training deep learning algorithms by formulating the minimization problem as an optimal control problem. Specifically, we are investigating necessary conditions of optimality using Pontryagin's Minimum Principle (PMP).While earlier works have focused on solving weaker conditions than PMP, we are exploring a variant of the Method of Successive Approximations (MSA) based on perturbations of the control on small optimal intervals during each iteration of the algorithm. This approach provides theoretical guarantees for convergence and error estimation. However, the current method requires fine meshes, which can impact computational efficiency. To address this, we have opted for adaptive mesh techniques, which accelerate the algorithm while maintaining its robustness.Why this formalization? This method, apart from offering theoretical guarantees, is particularly interesting because it does not require calculating gradients with respect to model parameters, making it suitable for cases where the parameter space is discrete. Additionally, as MSA typically has a steeper descent than traditional gradient-based algorithms, our formalization ensures faster convergence. Similar to traditional backpropagation methods used to train neural networks, information is propagated forward through the network to compute predictions, and then backward to adjust weights based on errors. However, in the MSA+ framework, once the state and co-state values are determined, the parameter optimization step is decoupled across the different layers of the network.

Keywords: Res-Net, DeepL, Optimal control.

1. Introduction

This article summarizes basic instructions for authors how to format a document which will be sent to Acta Polytechnica or Acta Polytechnica CTU Proceedings.

The article should be structured into the following sections: Introduction, Materials and methods, Results, Discussion, Conclusions, List of symbols, Acknowledgments, References and Appendices. Materials and methods and Results sections can be renamed. Results and Discussion sections can be merged into one section. List of symbols, Acknowledgments and Appendices sections are optional. More details can be found in Section **??**.

2. ResNets: Structure and Motivations

The increasing need for deeper neural networks motivated Kaiming He, Xiangyu Zhang, and Shaoqing Ren to propose ResNets (*Residual Networks*) in 2015 ([1]). These networks introduce a novel structure aimed at addressing the degradation problem observed in traditional neural networks. This issue, where deeper networks experience stagnation or even a drop in accuracy, is illustrated in Figure 1.

The authors reformulate the network layers as residual functions with respect to their inputs, rather than learning unreferenced functions. Each ResNet block learns a residual function, defined by the equation:

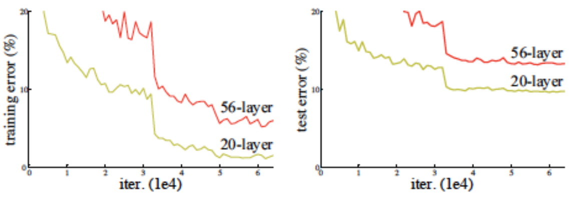

Figure 1. Impact of depth on traditional and residual networks (adapted from [1]).

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{W}) + \mathbf{x},$$

where $\mathbf{x}$ is the input to the block, $\mathbf{y}$ is the output, and $\mathcal{F}(\mathbf{x}, \mathbf{W})$ represents the transformation learned by the network, such as a combination of convolutions, nonlinearities, and normalization layers, with $\mathbf{W}$ being the weights.(Figure 1)

In their paper, the authors demonstrate that: extremely deep residual networks are easier to optimize, whereas "plain" networks (those that simply stack layers) show higher training errors as depth increases; ResNets can effectively leverage the accuracy improvements associated with greater depth, producing results significantly better than earlier networks.

The reader can clearly observe that this corresponds to an Euler approximation of an ODE, with a starting state $f(0)$ and an output $f(T)$.

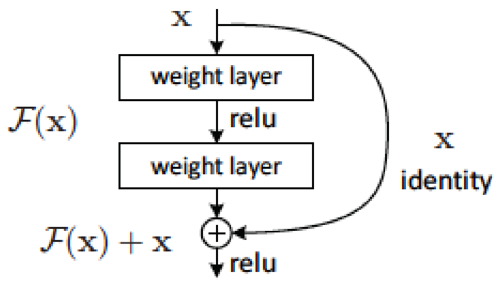
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Resnet structure

This interpretation has inspired significant research, framing the training of dense neural networks as an optimal control problem. Early pioneering works, such as those by LeCun and Pineda in the 1980s, established a connection between backpropagation and the adjoint method. More recent studies have expanded on this perspective within the framework of what we now call Neural ODEs. Research has focused on adaptive time-stepping methods, stability, and robustness, with high-order techniques that adjust time steps based on local truncation errors and approaches that optimize step sizes ([2], [3], [4], [5]).

In this report, we first provide an overview of the methods used in the continuous-time framework (Neural ODEs). In the first section, we rigorously formulate the training problem as an optimal control problem.

Viewing the problem from this perspective, two approaches are available: applying gradient descent on the control, which is now a function in L^2, or solving optimal control problems using methods based on the Pontryagin Maximum Principle (PMP), which are specifically designed to find optimal controls in a more drastic way .

Several efforts have been dedicated to the first approach (gradient descent [2], [6]). In this report, we first provide an overview of these methods, the different approaches taken, their limitations, and the results achieved within this framework.

The second approach is far less explored in the literature. Initially introduced by [7], the authors demonstrated the significance of a Hamiltonian perspective on gradient descent, where convergence towards a Pontryagin minimum is sought using the well-known Method of Successive Approximations (MSA). However, this method is heavily restricted to the linear case and is thus unsuitable for the nonlinear nature of neural networks. To address this, they introduced a weaker condition, the Extended PMP, which provides convergence guarantees but not necessarily to a Pontryagin minimum.

Building on the work of [7], [8] proposed a second variant called A-MSA, which aims to counter two intrinsic errors of the original model: sampling error and discretization error (Euler, for instance, being a particularly poor discretizer).

Both approaches come with convergence guarantees, but the question remains: convergence to what? The goal of our study is to present an adaptation of the MSA model that ensures convergence towards the Pontryagin minimum.

This approach remains very costly to implement, but heuristics can make it more feasible. In this report, we present the evolution of our research in this framework .

3. Optimal Control Training Formulation

Models such as ResNet are built recursively, following the general form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + hf(\mathbf{x}_k,$$

which can be interpreted as an Euler discretization of the following ODE:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), u(t)), \tag{2}$$

with a starting state $\mathbf{x}(0)$ and an output $\mathbf{x}(T)$.

The optimization of the parameter θ arises as a central question. - In the context of Equation (**??**), θ is a parameter vector in $\mathbb{R}^p$. - In the context of Equation (2), θ becomes a time-dependent function:

$$\theta : [0, T] \to \Theta \subset \mathbb{R}^p.$$

The function f is defined as the composition of a linear transformation and a nonlinear activation function, typically of the ReLU or sigmoid type. We assume that $\nabla_{\mathbf{x}} f$ is continuous with respect to θ, t, and $\mathbf{x}$.

We also define U, the set of admissible functions for θ, as:

$$U = \{\theta : [0, T] \to \Theta \mid \theta \text{ is Lebesgue-measurable}\}.$$

For simplicity, in this paper, we restrict U to the subspace L^2. The optimization problem is then stated as:

$$u^* = \arg\min_u \mathcal{L}(u) = \arg\min_u \mathbb{E}_{(x,y)\sim\mathbb{P}(x,y)} \left[l(x(T;u), y)\right], \tag{3}$$

where the loss function is defined as:

$$\mathcal{L}(u) = \mathbb{E}_{(x,y)\sim\mathbb{P}(x,y)} \left[l(x(T;u), y)\right]. \tag{4}$$

However, this loss is intractable because the true data distribution $\mathbb{P}(x, y)$ is unknown. Therefore, we approximate it using the empirical loss:

$$\hat{\mathcal{L}}(u) = \frac{1}{n}\sum_{i=1}^{n} l(x_i(T;u), y_i),$$

where $(x_i(T;u))_{1\le i\le n}$ are solutions to Equation (2) with initial conditions $x(0) = x_i$.

For simplicity, throughout this paper, we will use the notation $\mathcal{L}$ to refer to the empirical loss $\hat{\mathcal{L}}$.

Viewed from this perspective, it is an optimal control problem. The reader can observe that this formulation raises two main challenges: how to determine the optimal control and how to ensure that this control can be discretized effectively to provide an optimal solution for Res-Net.

To address the first question, we will present the theoretical framework of the two approaches used in this context:

(1.) Gradient descent method applied to control,

(2.) Pontryagin's approach.

The first approach might initially seem irrelevant for optimizing a neural network by formulating it as a continuous problem with gradient descent, particularly given the complexity of calculating gradients in an L^2 space. However, the breakthrough lies in how this approach aids in analyzing the stability and behavior of the network. To motivate this perspective, the reader can refer to [9], which explores methods for analyzing the behavior of different architectures, such as backward error analysis and the role of the adjoint in providing insights into the network's sensitivity to perturbations in initial data. Indeed, there is a vast amount of research and references on the problem of optimal control and the analysis of its behavior. Transitioning to a continuous formulation can therefore provide a deeper understanding of various aspects of our network.

4. Gradient based Approach

4.1. Primitive Gradient Approach

In this context, we will begin by presenting our first straightforward approach to using gradient descent to address the optimal control problem. To do so, we will first compute the gradient of our loss with respect to the control.

We will start by stating the conditions we impose on our dynamics to calculate the gradient.

Assumption 1. *The map $f : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}^d$ is continuous. Moreover, for every compact set $U \subset \mathbb{R}^q$, there exists $C_U > 0$ such that, for all $(x, u) \in \mathbb{R}^d \times U$,*

$$|f(x, u)| \le C_U(1 + |x|).$$

Assumption 2. *i)The function $\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ is continuous.*

ii)The probability f is compactly supported.

Assumption 3. *i)The map f is continuously differentiable.*

ii)For every $x \in \mathbb{R}^d$, $\ell(\cdot, x)$ is continuously differentiable. We denote its Jacobian by $\hat{y} \mapsto \partial_1\ell(\hat{y}, x)$. The map $\partial_1\ell(\hat{y}, x)$ is bounded with respect to $(\hat{y}, x)$ in bounded sets.

Assumption 4. *The Jacobian of $\nu(x, u)$ with respect to (x, u) is locally Lipschitz continuous.*

Proposition 1. *Let Assumptions 1, 2, 3, and 4 hold. Then, $f : L^\infty(I, \mathbb{R}^q) \to \mathbb{R}$ is Gâteaux-differentiable at any point $u \in L^\infty(I, \mathbb{R}^q)$. Its Gâteaux-derivative in the direction $v \in L^\infty(I, \mathbb{R}^q)$ is given by:*

$$df(u; v) = \langle \nabla f(u), v \rangle_{L^2},$$

where $\nabla f(u) \in L^\infty(I, \mathbb{R}^q)$ is defined as:

$$\nabla L(u)(t) \mapsto -\int p_t(x(u, x), u, x)^\top f_u(x(u, x), u_t) df(x).$$

with p the solution of

$$p_T^\top = -\partial_1\ell(x_1, x),$$
$$\dot{p}_t^\top = -p_t^\top f_x(x_t, u_t).$$

Proof. Refer to the appendix for an outline of the proof, primarily using Grönwall's inequalities and clever gradient calculations. ■

4.2. Gradient Descent Approach

Once we have established the gradient with respect to the control, we can implement a gradient descent approach using the following algorithm:

Algorithm 1 Gradient Descent (GD)

```
1: Initialization: Choose an initial control $u_0$ and a convergence tolerance.
2: while True do
3:     Solve the state equation to obtain $x_k(t)$ using the control $u_k(t)$.
4:     Solve the adjoint equation to obtain $p_{k-1}(t)$.
5:     Update the control $u_{k+1}(t)$ using:
           $u_{k+1}(t) = u_k(t) - \mu \nabla L(u_k)(t)$
6:     Check the stopping condition: Stop if $\|u_{k+1} - u_k\|$ is below the convergence tolerance.
7: end while
```

Experimental Results

In order to test the numerical results of this method, we studied how our ODE-Net can learn to model the sine function. We implemented manually the gradient according to formula (XX) over a sample of 30 points between 0 and 2π (so we considered the empirical risk (5) instead of the original one). We tested the model on new unseen points, we obtain this curve :

We observe that our ODE-net succeeds to learn the sine fonction and the gradient descent converges in very few steps.

When we tried with other tyes of functions namely in higher dimensions, we observed that in a lot of cases, the gradient descent algorithm requires very low learning rates to converge, otherwise we encounter the exploding gradients problem. However, when using very small learning rates, the algorithm becomes

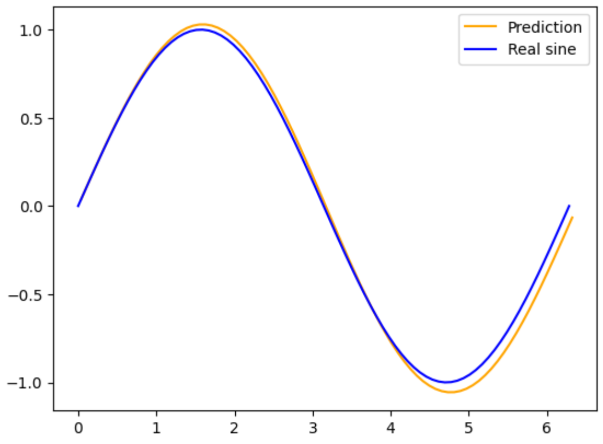

Figure 3. Learning the sine function

extremely slow. Consenquently, classical gradient descent can be time-consumming in some cases.

After realizing that ou ODE-net model succeeded to learn the sine function, we started asking questions about the universality of ODE-nets. We first found in the literature some papers saying that Resnets cannot learn non inversible functions, which is a priori a contradiction since we just learned the sine function. This is the subject of the following paragraph.

Universal Approximation of Resnets

An important aspect one can analyze in order to understand the representation power of a model is to determine the class of mappings that it can learn. For instance, the Universal Approximation Theorem states that a neural network with a single hidden layer can approximate any continuous function with compact support by any arbitrary accuracy, as the width goes to infinity. In contrast, Resnets can show some good representation properties when the depth goes to infinity. The main idea of this section is to show that ODE-Nets, eventhough they are continous limits of discrete ResNets, can show some weaknesses in representation that are solved by using discrete ResNets.

4.2.1. ODE-Nets representation power

Let us consider the example g(x) = -x There is no ODE-Net that can model this function.

Why this approach is not enough

We implemented gradient descent following Algorithm 1. **(Include and analyze the results here)**.

4.3. Discretized Gradient Approach [6]

This approach is inspired by the works of M. Benning, E. Celledoni, M. J. Ehrhardt, and al. in [6].

Instead of viewing the optimal control problem in the continuous setting, we reformulate it in the discrete framework as follows:

We consider $x^0, x^1, \dots, x^N$ and $u^0, u^1, \dots, u^N$, with a cost function $L(x^N)$. The optimal control problem becomes:

$$\min_{(x^{[j]},u^{[j]})} L(x^{[N]}),$$

subject to:

$$x^{[j]} = D_{\Delta t}(x^{[j-1]}, u^{[j-1]}), \quad x^{[0]} = x.$$

Here, $D_{\Delta t}$ represents a numerical integration of the ODE.

If we restrict ourselves to Euler integration, we may encounter instabilities, but for simplicity, we will adopt this method to explain the resolution process. However, a broader class of methods can be applied in this framework. For Euler discretization, we have:

$$x^{[j+1]} = x^{[j]} + \Delta t f(x^{[j]}, u^{[j]}).$$

Using this discretization approach, it can be shown (see [6] for details in the general case of Runge–Kutta methods) that, in the Eulerian case, we have:

$$x^{[j+1]} = x^{[j]} + \Delta t\, f(x^{[j]}, u^{[j]}), \tag{5}$$

$$p^{[j+1]} = p^{[j]} - \Delta t\,(\partial_x f(x^{[j]}, u^{[j]}))^\top p^{[j+1]}, \tag{6}$$

$$\partial_{u^{[j]}} L(x^{[N]}) = \Delta t\, \partial_u f(x^{[j]}, u^{[j]})^\top p^{[j+1]}. \tag{7}$$

Algorithm 2 Training ODE-inspired neural networks with gradient descent.

```
Require: Initial guess for the controls $u$, step size $\tau$
1: for $k = 1, \dots$ do
2:     Forward propagation: compute $x$ via (16)
3:     Backpropagation: compute $p$ via (17)
4:     Compute gradient $g$ via (24) and (41)
5:     Update controls: $u = u - \tau g$
6: end for
```

La difference entre les deux approche de decente de gradient et quue la premiere n'entraine pas un reseaux de neurone mais reformiule le problem de minimisation du risque comme un probleme de control optimal .

5. Pontryagin's Maximum Principle APPRACH

5.1. Pontryagin's Maximum Principle: Introduction

Pontryagin's Maximum Principle provides necessary conditions for optimality. Although it only offers necessary conditions, PMP generally yields strong candidates for optimality. We will recall the principle as presented in Pontryagin (1987) and then introduce a variant adapted to our specific case.

5.1.1. Pontryagin's Maximum Principle: Initial Formulation

Consider the minimization problem:

$$J = \phi(x(t_f)) + \int_{t_0}^{t_f} F(x(t), u(t))\, dt \tag{*}$$

subject to:

$$\dot{x} = f(x,u), \quad x(t_0) = x_0, \quad \text{and} \quad x(t_f) = x_f. \quad (**)$$

The Hamiltonian $H : [0,T] \times \mathbb{R}^n \times \mathbb{R}^n \times \Theta \to \mathbb{R}$ is defined as:

$$H(t,u,p,\theta) := p \cdot f - F.$$

Theorem 1 (PMP)**.** *Necessary conditions for $u \in U$ to minimize (*) subject to (**) are:*

$$\dot{p} = -\frac{\partial H}{\partial x}^{\top},$$

where $H = H(x,u,p) = F(x,u) + p^{\top} f(x,u)$, and

$$H(x^*,u^*,p^*) = \min_{u \in U} H(x,u,p).$$

If the final state $x(t_f)$ is free, then in addition to the above conditions, the following end-point condition must be satisfied:

$$p(t_f) = \nabla_x \phi \Big|_{t=t_f}.$$

Training a neural network in the continuous case can be interpreted as searching for a control that minimizes the error on the reachable state of our controlled ODE. However, the control problem typically considers only a single input, whereas in our case, we are specifically interested in minimizing the final state across multiple points in our dataset.

To address this, we rewrite the Pontryagin Maximum Principle (PMP) for the case of n starting points. This approach remains generalizable as long as the number of points is finite, because it can be interpreted as an increase in the dimension of our ODE. Let us properly state the PMP variant.

5.1.2. Pontryagin's Maximum Principle: Variant

Taking into account that F is identically null we have

Theorem 2 (PMP 2)**.** *Let $u^* \in L^1([0,T],U)$ be an optimal control for (3.4). For $i = 1,\dots,N$, let $x_{u^*,i}$ be the associated state process with initial condition x_i.*

There exists a co-state process $p_{u^*,i} \in L^1([0,T],\mathbb{R}^n)$ such that:

$$\dot{p}_{u^*,i}(t) = -\nabla_x H(t, x_{u^*,i}(t), p_{u^*,i}(t), u^*(t)),$$

$$p_{u^*,i}(T) = -\nabla_x L(x_{u^*,i}(T), y_i),$$

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

$$\frac{1}{N}\sum_{i=1}^{N} H(t, x_{u^*,i}(t), p_{u^*,i}(t), u^*(t)) \leq$$

$$\frac{1}{N}\sum_{i=1}^{N} H(t, x_{u^*,i}(t), p_{u^*,i}(t), u), \quad \forall u \in U. \quad (***)$$

The proof of the PMP and its variant can be found in [10].

By solving the minimality equation (***) we should converge to the Pontryagin minimum, which represents an excellent optimality point.

Since the pioneering work of [11], an algorithm known as the Method of Successive Approximations (MSA) has been theoretically proven to converge to the Pontryagin maximum in linear systems. This convergence method, which we will present in the next section, differs from gradient descent due to its ability to adapt to constraints that can be imposed on the control, as it does not require gradient calculations. However, in the general, nonlinear case, there is no guarantee of convergence.

We will first examine the naive approach to MSA, explain the impossibility of convergence in this context, and then propose solutions to address this problem.

5.1.3. MSA Algorithm

Algorithm 3 Simple MSA

```
1: Initialisation : Choisir un contrôle initial $u_0$ et une tolérance de convergence.
2: while convergence non atteinte do
3:     Résoudre l'équation d'état pour obtenir $x_{k-1}(t)$ avec le contrôle $u_{k-1}(t)$.
4:     Résoudre l'équation de l'adjoint pour obtenir $p_{k-1}(t)$.
5:     Mettre à jour le contrôle $u_k(t)$ en maximisant l'Hamiltonien :
           $u_k(t) = \arg\max_u H(x_{k-1}(t), p_{k-1}(t), u)$    (8)
6:     Vérifier la condition d'arrêt : arrêter si $\|u_k - u_{k-1}\|$ est inférieur à la tolérance de convergence.
7: end while
```

In the particular linear case where:

$$f(x,t,u) = A(t)x + b(t,u),$$

the convergence of the simplest MSA has been proven under this special case with a quadratic form of ϕ [12].

However, in the general nonlinear case, this classical MSA model can diverge in many situations. To better understand this, consider the inequality established by [7], with the proof reported in the Appendix.

Lemma 1. *Suppose $(A1)-(A2)$ hold. Then, there exists a constant $C > 0$ such that for any $u, \tilde{u} \in U$,*

$$L(u) - L(\tilde{u}) \geq$$

$$-\int_0^T \Delta_{u,\tilde{u}} H(t)\,dt + C\int_0^T \|f(t;x_t^u;u_t) - f(t;x_t^{\tilde{u}};\tilde{u}_t)\|^2\,dt$$

$$+C\int_0^T \|\nabla_x H(t;x_t^u;p_t^u;u_t) - \nabla_x H(t;x_t^{\tilde{u}};p_t^{\tilde{u}};\tilde{u}_t)\|^2\,dt,$$

where:

$$\Delta_{u,\tilde{u}}H(t) := H(t; x_t^u; p_t^u; u_t) - H(t; x_t^{\tilde{u}}; p_t^{\tilde{u}}; \tilde{u}_t).$$

What this lemma suggests is that the first term in the inequality shows that step 5 of the algorithm indeed results in a decrease in the risk L. However, the last two terms on the right-hand side demonstrate that this descent can be nullified if the replacement induces too many errors in the Hamiltonian dynamics. These two terms can be seen as feasibility errors of the new solution $\tilde{u}$.

One initial solution would be to enforce feasibility in the Hamiltonian by augmenting it. A second variant could involve exploring a control-changing method that optimizes over an interval to reduce risk.

Before exploring these two options, let us revisit the utility of this MSA approach. Step 5 ensures a drastic descent in control while requiring neither gradient computation nor error propagation. This means that, under convergence conditions, this approach should be faster and more adaptive (e.g., to constraints on control such as compactness, finiteness, convexity, etc.).

5.2. EMSA Algorithm

To include feasibility conditions in our augmented Hamiltonian, we define:

$$\tilde{H}(t, x, p, v, q, u) := H(t, x, p, u) - \frac{1}{2}\rho\|v - f(t, x, u)\|^2 - \frac{1}{2}\rho\|q + \nabla_x H(t, x, p, u)\|^2.$$

Proposition 2 (Extended PMP)**.** *Suppose that u^* is an essentially bounded solution to the optimal control problem . Then, there exists an absolutely continuous co-state process p^* such that the tuple (x_t^*, p_t^*, u_t^*) satisfies the necessary conditions:*

$$\begin{aligned} \dot{x}_t^* &= \nabla_p \tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*), \\ x_0^* &= x, \end{aligned} \tag{9}$$

$$\begin{aligned} \dot{p}_t^* &= -\nabla_x \tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*), \\ p_T^* &= -\nabla_x \phi(x_T^*), \end{aligned} \tag{10}$$

$$\tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*) \geq \tilde{H}(t, x_t^*, p_t^*, u, \dot{x}_t^*, \dot{p}_t^*), \tag{11}$$

Proof. If u^* is optimal, then by the Pontryagin Maximum Principle (PMP), there exists a co-state process p^* such that PMP conditions are verified

For all $t \in [0, T]$ and $u \in U$, we have:

$$\nabla_x \tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*) = \nabla_x H(t, x_t^*, p_t^*, u_t^*),$$

$$\nabla_p \tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*) = \nabla_p H(t, x_t^*, p_t^*, u_t^*),$$

which implies that (9) and (10) are satisfied.

Lastly, we can write:

$$\tilde{H}(t, x_t^*, p_t^*, u_t^*, \dot{x}_t^*, \dot{p}_t^*) = H(t, x_t^*, p_t^*, u_t^*) - \frac{1}{2}\rho\|\dot{x}_t^* - f(t, x_t^*, u_t^*)\|^2 - \frac{1}{2}\rho\|\dot{p}_t^* + \nabla_x H(t, x_t^*, p_t^*, u_t^*)\|^2,$$

which ensures condition (11). ■

We can easily show that when using an MSA applied to the augmented Hamiltonian, we obtain a descent direction for our algorithm. Let us examine this more closely:

Using Lemma 2 with $u = u^{k+1}$, $\phi = u^k$, we have:

$$\begin{aligned} L(u^{k+1}) - L(u^k) \leq -\mu_k + C \int_0^T &\|f(t, x_t^{u^k}, u_t^k) \\ &- f(t, x_t^{u^k}, u_t^{k+1})\|^2 \, dt \\ + C \int_0^T &\|\nabla_x H(t, x_t^{u^k}, p_t^{u^k}, u_t^k) \\ &- \nabla_x H(t, x_t^{u^k}, p_t^{u^k}, u_t^{k+1})\|^2 \, dt. \end{aligned} \tag{12}$$

With:

$$\mu_k = \int_0^T \Delta L_{u_k, u_{k+1}}(t) \, dt.$$

From the Hamiltonian maximization step in Algorithm 2, we know that:

$$\begin{aligned} H(t, x_t^{u^k}, p_t^{u^k}, u_t^k) \leq H(t, &x_t^{u^k}, p_t^{u^k}, u_t^{k+1}) \\ &- \frac{1}{2}\rho\|f(t, x_t^{u^k}, u_t^k) - f(t, x_t^{u^k}, u_t^{k+1})\|^2 \\ &- \frac{1}{2}\rho\|\nabla_x H(t, x_t^{u^k}, p_t^{u^k}, u_t^k) \\ &- \nabla_x H(t, x_t^{u^k}, p_t^{u^k}, u_t^{k+1})\|^2. \end{aligned} \tag{13}$$

Hence, we have:

$$L(u^{k+1}) - L(u^k) \leq -(1 - \frac{2C}{\rho})\mu_k. \tag{14}$$

Pick $\rho > 2C$, then we indeed have:

$$L(u^{k+1}) - L(u^k) \leq -D\mu_k, \quad D = \left(1 - \frac{2C}{\rho}\right) > 0. \tag{15}$$

We have just proven the convergence of the following algorithm, but towards a weaker condition than the Pontryagin minimum.

For a deeper analysis of this method, the reader is encouraged to explore ([7], [8]).

5.3. Optimal Interval MSA (OI-MASA)

As discussed in the previous section, the Extended MSA is theoretically not convergent to a Pontryagin Minimum Principle (PMP). Instead, Algorithm 4 solves a weaker PMP condition. In this section, we present a brand-new algorithm inspired by [13], which provides theoretical convergence guarantees. However, this method remains intractable in practice, so we also present heuristics for its implementation.

Algorithm 4 Extended MSA Algorithm

```
Require: Initialize: $u^0 \in U$, hyperparameter $\rho$.
1: for $k = 0$ to #Iterations do
2:     Solve $\dot{x}_t^{u^k} = f(t, x_t^{u^k}, u_t^k)$,   $x_0^{u^k} = x$.
3:     Solve $\dot{p}_t^{u^k} = -\nabla_x H(t, x_t^{u^k}, p_t^{u^k}, u_t^k)$,   $p_T^{u^k} = -\nabla_x \Phi(x_T^{u^k})$.
4:     Set $u_t^{k+1} = \arg\max_{u\in U} \tilde{H}(t, x_t^{u^k}, p_t^{u^k}, u, \dot{x}_t^{u^k}, \dot{p}_t^{u^k})$ for each $t \in [0,T]$.
5: end for
```

To motivate this approach, let us revisit the inequality from Lemma 1. The degree of infeasibility in the new triplet (x^n, p^n, u^{n+1}) prevents our updated control from achieving a descent in the loss. Modifying u more gradually could ensure a descent. This is exactly what [7] proposes by seeking a weaker E-MSA condition while incorporating feasibility constraints into the Hamiltonian to be maximized.

In our approach, we aim to adjust the control over small intervals that are optimal in the sense of minimizing the loss. Specifically, we strive to balance the minimization of the loss with the optimal direction of increase for the Hamiltonian.

Let us first define the elements and variables of this approach before designing our algorithm.

5.3.1. Formalisation:

For a control u_k and a hamiltonian H we define: $u_k(\tau,h)$ et $R_k(\tau,h)$ as :

$$L_k(\tau,h) = \mathbb{E}_{(x,y)\sim\mathbb{P}(x,y)}\left[l(x_{u_k(\tau,h)}(T), y)\right], \tag{16}$$

Where the control $u_k(\tau,h)$ refers to :

$$u_k(\tau,h)(t) = \begin{cases} u_k^*, & \text{if} t \in [\tau-h, \tau+h], \\ u_k(t), & \text{otherwise.} \end{cases} \tag{17}$$

$u_k(\tau,h)$ is the modified control towards the Hamiltonian-maximizing control, but only within an interval.

We will choose this interval in a way that optimizes our risk:

$$(\tau^*, h^*) \in \arg\min_{\tau,h} R_k(\tau,h).$$

Let O be the operator that associates:

$$O : u \mapsto u(\tau,h).$$

Thus, we will establish the following algorithm:

5.3.2. Convergence Result

Starting with this lemma:

Lemma 2. *Given any $h > 0$, a $\tau(h) = \tau(h,u)$ exists such that*

$$\int_{\tau(h,u)}^{\tau} \Delta_\tau H(t)\,dt \geq \frac{h\cdot\mu(u)}{T}. \tag{3.5}$$

Algorithm 5 Optimal Interval MSA Method

```
1: Initialization: Choose an initial control $u_0$ and a convergence tolerance.
2: while convergence not reached do
3:     Step 1: Solve the state forward equation to obtain $x_k(t)$ with the control $u_k(t)$.
4:     Step 2: Solve the backward adjoint equation to obtain $p_k(t)$.
5:     Step 3: Compute the control that maximizes the Hamiltonian:
           $$u_k^*(t) = \arg\max_u H(x_k(t), p_k(t), u).$$
6:     Step 4: Determine the parameters $\tau^*$ and $h^*$ that minimize the risk:
           $$(\tau^*, h^*) \in \arg\min_{\tau,h} R_k(\tau,h).$$
7:     Step 5: Update the control $u_k$ according to:
           $$u_{k+1} = u_k(\tau^*, h^*).$$
8:     Step 6: Check the stopping condition: stop if $\|u_{k+1} - u_k\|$ is less than the convergence tolerance.
9: end while
```

Proof. Assume the contrary, i.e., an $h^* > 0$ exists such that, for all $\tau \in [0,T]$,

$$\int_{\tau}^{\tau^*} \Delta_\tau H(t)\,dt < \frac{h^*\cdot\mu(u)}{T}.$$

Put $\tau_i = ih^*$, $i = 0,1,\ldots,N(h^*)$, where $N(h^*) = \lfloor T/h^* \rfloor$ is the integral part. We then have

$$\mu(u) \leq \sum_{i=0}^{N(h^*)} \int_{\tau_i}^{\tau_{i+1}} \Delta_\tau H(t)\,dt <$$

$$\frac{h^*\cdot N(h^*)}{T}\cdot\int_0^T \Delta_\tau H(t)\,dt \leq \mu(u).$$

Which is contradictory. ■

Using this lemma, we can prove the following theorem.

Theorem 3. *Let a constant $b > 0$ exist such that, for all admissible $u(t)$,*

$$|x_u(t)| \leq b, \quad t \in [0,T]. \tag{3.1}$$

Then,

$$L(Ou) \leq L(u) - \beta\mu^2(u), \quad \beta = const > 0; \tag{3.2}$$

$$\lim_{k\to\infty} \mu_k = 0, \quad \mu_k = \mu(u^{(k)}), \quad u^{(k)} = +O^{(k)}u^{(0)}. \tag{3.3}$$

Proof. See Appendix. ■

We have just proven the convergence of our algorithm to the Pontryagin maximum. However, this algorithm is extremely costly due to the double minimization over τ and h. Consequently, we use a heuristic that overcomes this problem. The heuristic consists of separating variables in step 4, by τ^*, the point that makes the most change in the difference of Hamiltonians of u_k and u_k^*:

$$\tau^* \in \arg\max_{\tau} \Delta_{u_k, u_k^*} H.$$

Algorithm 6 OI-MSA Heuristic 1

```
1: Initialization: Choose an initial control $u_0$ and a convergence tolerance.
2: while convergence not reached do
3:     Step 1: Solve the state forward equation to obtain $x_k(t)$ with the control $u_k(t)$.
4:     Step 2: Solve the backward adjoint equation to obtain $p_k(t)$.
5:     Step 3: Compute the next control $u_k^*$ by maximizing the Hamiltonian:
           $u_k^* = \arg\max_u H(x_k(t), p_k(t), u).$
6:     Step 4: Find the jump point $\tau$ in the Hamiltonian where the change is maximal between $u_k^*$ and $u_k$.
7:     Step 5: Determine the interval width $h$ around $\tau$ such that the control over the interval $[\tau - h, \tau + h]$ minimizes the risk $R_k(\tau, h)$.
8:     Step 6: Update the control $u_k$ as follows:
           $u_k(t) = \begin{cases} u_k^*(t), & \text{if } t \in [\tau - h, \tau + h], \\ u_k(t), & \text{otherwise.} \end{cases}$
9:     Step 7: Check the stopping condition: stop if $\|u_k - u_{k-1}\|$ is less than the convergence tolerance.
10: end while
```

6. Conclusions

Acknowledgements

List funding sources according to the funder's requirements.

References

[1] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition, 2015. 1512.03385

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Neural ordinary differential equations, 2019. 1806.07366

[3] W. E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics* **5**:1 – 11, 2017.

[4] T. Zhang, Z. Yao, A. Gholami, et al. Anodev2: A coupled neural ode evolution framework, 2019. 1906.04596

[5] J. Zhuang, N. Dvornek, X. Li, et al. Adaptive checkpoint adjoint method for gradient estimation in neural ode, 2020. 2006.02493

[6] M. Benning, E. Celledoni, M. J. Ehrhardt, et al. Deep learning as optimal control problems: models and numerical methods, 2019. 1904.05657

[7] Q. Li, L. Chen, C. Tai, W. E. Maximum principle based algorithms for deep learning, 2018. 1710.09513

[8] J. Aghili, O. Mula. An optimal control framework for adaptive neural ODEs, 2023. Working paper or preprint.

[9] E. Hairer, C. Lubich, G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, vol. 31 of *Springer Series in Computational Mathematics*. Springer, 2nd edn., 2006. https://doi.org/10.1007/3-540-30666-8

[10] F. L. Lewis, V. L. Syrmos. *Optimal Control: An Introduction to the Theory and Its Applications*. Wiley-Interscience, 2012.

[11] I. A. Krylov, F. L. Chernousko. On the method of successive approximations for solution of optimal control problems. *J Comp Mathem and Mathematical Physics* **2**(6), 1962.

[12] J. Bergmann, K. Nolte. Zur konvergenz des algorithmus von krylow und Černous'ko. i. *Math Operationsforsch Statist, Ser Optimization* **8**(3):11–77, 1977.

[13] A. A. Lyubushin. Modifications and convergence of successive approximations for optimal control problems. *USSR Computational Mathematics and Mathematical Physics* **19**:53–61, 1980. (Received 14 November 1978). https://doi.org/10.1016/0041-5553(79)1201-0053

A. Appendices

Insert supplementary large data sets, extensive tables or figures and more detailed experimental data, etc.

A.1. External data sets

Use this first Appendix for description and link to external datasets.

A.2. Large data sets

Any supplementary large sets of data (e.g. graphs, tables) that are not appropriate to insert into the text.

LaTeX class errors and warnings

- **Warning:** Text of ‘abstract‘ is 611 characters too long