Neural non-canonical Hamiltonian dynamics based on discrete variational integrators for long-time simulations

C. Courtès[1,2], E. Franck[1,2], M. Kraus[3], L. Navoret[1,2], and L. Tremant[4]

[1]Institut de Recherche Mathématique Avancée, UMR 7501 Université de Strasbourg et CNRS,
7 rue René Descartes 67000 Strasbourg, France
[2]INRIA Nancy-Grand Est, MACARON Project, Strasbourg, France
[3]Max-Planck-Institut für Plasmaphysik, Garching bei München, Germany
[4]Univ. Artois, UR 2426, Laboratoire de Mathématiques de Lens (LML), F-62300 Lens, France

Contents

Abstract

This work focuses on learning non-canonical Hamiltonian dynamics from data, where long-term predictions require the preservation of structure both in the learned model and in numerical schemes. Previous research focused on either facet, respectively with a potential-based architecture and with degenerate variational integrators, but new issues arise when combining both. In experiments, the learnt model is sometimes numerically unstable due to the gauge dependency of the scheme, rendering long-time simulations impossible. In this paper, we identify this problem and propose two different training strategies to address it, either by directly learning the vector field or by learning a time-discrete dynamics through the scheme. Several numerical test cases assess the ability of the methods to learn complex physical dynamics, like the guiding center from gyrokinetic plasma physics.

1   Introduction

With the rise in machine learning research, the connection between neural networks and differential equations—particularly ordinary differential equations (ODEs)—has become an increasingly active area of study, explored through three main application domains. The first approach interprets certain neural networks, such as Residual Networks (ResNets), as discretizations of ODEs [CRBD18, DDT19, MPP$^+$20]. This perspective not only enables the design of novel architectures but also provides deeper insight into the internal dynamics of neural networks. The second line of research views gradient descent methods as discretizations of continuous-time gradient flows [JGH18], allowing a more refined analysis of learning dynamics by studying trajectories in parameter space. The third axis concerns time series prediction. Instead of using models such as Recurrent Neural Networks (RNNs) or Long short-term memory architectures (LSTM), when it is reasonable to assume that the data stems from an underlying continuous process, it becomes relevant to model these series with ODEs or stochastic differential equations (SDEs) [ZL24]. This work belongs to that third category.

Many contributions have been made to the learning of differential equations: the aim is then to determine the vector field, whose integral lines fit the observed data. Some, such as Sparse Identification of Non-linear Dynamics (SINDy) [BPK16, KKB18], aim to infer the analytic form of the governing equations from time transitions between pairs of states. Similar strategies have been developed using neural networks. Other works are based on unrolled training over multiple time steps [LTS22], either to learn discrete-time flows that are then composed, or to model continuous-time dynamics directly [BSR$^+$21]. In physical applications, learning ODEs is particularly relevant for model reduction [FFBM25, BHT$^+$24, RPS$^+$23, CFN$^+$24, CFN$^+$25] to capture the dynamics of latent variables or even to discover new physical laws [TID23]. From both the standpoint of physical modeling and long-term numerical stability, the preservation of conservative and dissipative structures is of key importance. This has naturally raised the question of whether such structures can be preserved by learning algorithms.

In the case of dissipative systems, some works have focused on the joint learning of a Lyapunov function and its associated ODE [KM19]. A large part of the literature, however, has focused on conservative ODEs, particularly those arising from Lagrangian

or Hamiltonian formalisms. For such systems, it is well known that preserving the underlying geometric structures, especially symplectic structures, is crucial to ensure long-term stability. In the context of machine learning, the seminal work on Hamiltonian Neural Networks (HNN) [GDY19] proposed to learn the vector field as the gradient of a Hamiltonian function, described with a neural network. It demonstrated that preserving the Hamiltonian structure—i.e., directly learning the Hamiltonian function—significantly improves the long-term stability of the learned dynamics. Since then, a range of approaches have been proposed to learn dynamics that respect Lagrangian or Hamiltonian structures [CGH+20, DM21, LSR+22], or to directly learn symplectic discrete-time flows [JZZ+20, EL24, LLCP22]. Most of these contributions focus on canonical Hamiltonian systems, where the dynamics are expressed in $\mathbb{R}^{2d}$ with the standard symplectic form. However, many physical systems (e.g., fluid mechanics, plasma physics, or constrained mechanical systems) do not follow this canonical form. This leads to non-canonical systems, where the underlying geometry is more complex and often described by generalized Poisson structures. Such dynamics are then described by both a Hamiltonian function and a skew-symmetric space-dependent matrix, defining the underlying Poisson structure: this matrix is invertible for non-canonical Hamiltonian system and non-invertible for general Poisson system. A few recent works have begun to extend previous methods to these non-canonical cases [CMY21] and to Poisson systems [JZKK20, Tap24, BTM21]. The present work falls within this direction. It focuses specifically on non-canonical Hamiltonian systems, building on the contributions of [CMY21]. We first introduce the additional challenges raised by the non-canonical setting—especially the fact that the geometric structure depends on the system state—before proposing two novel approaches to learn dynamics that preserve such structures.

1.1 Long-time numerical behaviour

In the seminal work [GDY19] on learning dynamics while preserving the Hamiltonian structure, the authors demonstrate the benefits of this approach. However, the respective roles of the two key ingredients—learning dynamics that preserve the Hamiltonian structure, and using a suitable numerical scheme to integrate the learned equation—remain to be fully clarified. In the first part of this work, we show that it is primarily the structure-preserving learning that improves medium-term prediction accuracy, while, over longer time horizons, the use of an appropriate numerical integrator remains essential.

In [CMY21], the authors propose an approach for the non-canonical case, based on enforcing the closedness of the symplectic form. This method shows promising long-term behavior, but it relies on a standard integration scheme (RK4), whose limitations become apparent when simulations are extended further in time. It is therefore natural to consider structure-preserving schemes, such as Discrete Variational Integrators (DVI) type [BFE22].

This example highlights a key point: learning an equation with the correct geometric structure is not enough. In complex cases such as non-canonical ODEs, even numerically adapted schemes like DVI may fail to accurately integrate the learned equation. We will illustrate this issue with a concrete example and analyze possible reasons behind such failure. Addressing this limitation is the main challenge this work aims to tackle.

1.2 Two training strategies depending on the dataset

Vector-field learning When using a scheme such as DVI to solve a non-canonical problem learned through the approach of [CMY21], it can happen that the scheme produces a

completely incorrect prediction. This stems from the fact that the scheme does not only simulate the evolution in the phase space, it also involves the evolution of the symplectic potential—it is not *gauge invariant*. This means that its error bound may increase if it is not penalised in the learning process, rendering it useless for simulating the learnt problem over either short or long times. We solve this problem by adding a penalisation term to the loss such which represents this error constant. To our knowledge, this error was not derived previously, and it provides insight into the gauge-dependency of the DVI.

Scheme learning   One might consider the situation where the vector field is not known at any collocation points but snapshots along trajectories are known. In that case, the dataset consists of pairs $(z_{|t=0}, z_{|t=h}) \in M^2$ with some time-step h. This is the original setting of Neural ODEs [CRBD18], where the authors fit a (learnt) vector field such that trajectories along it map the input to the output. For canonical Hamiltonian problems, specific architectures such as SympNets [JZZ+20] or HénonNets [BTM21, DBT23] explicitly produce a structure-preserving mapping from one snapshot to the next.

For non-canonical problems, however, no such architecture is yet known. Therefore, we choose a geometric numerical scheme and fit the learnt potentials such that using it for simulations will yield near-exact (discrete) trajectories. This method allows long-time simulations which can be more accurate than those on the reference model, though the time-step and numerical scheme must not change. This behaviour is similar to hypersolvers[1] [PMY+20, MYH22, BCLM23]. Because the scheme is implicit, a cheap initial guess is produced by a naive neural network with no specific architecture. This circumvents the stability issue of the DVI on the reference model and allows larger time-steps than usual.

1.3   Outline

The outline of this paper is as follows. Section 2 presents non-canonical Hamiltonian framework and the DVI numerical scheme, based on the Lagrangian formulation. Numerical tests also illustrate the advantages of the DVI for long-time simulations and issues related to gauge invariance when used in geometric model learning. In Section 3, we introduce the vector-field learning and the scheme learning strategies, adapted to or based on to DVI. Finally, a series of numerical experiments (Lotka-Volterra, massless charged particle and guiding center) is conducted in Section 5 to assess the ability of both methods to capture the dynamics.

2   Non-canonical Hamiltonian dynamics

Canonical Hamiltonian systems form a fundamental class of dynamical systems with a specific geometric structure. They are written as

$$\dot{z} = J^{-1}\nabla H(z),$$

where $z(t) \in \mathbb{R}^{2d}$ denotes the time-dependent coordinates, $H(z) \in \mathbb{R}$ denotes the Hamiltonian function representing the total energy of the system, and J is the canonical symplectic matrix given by

$$J = \begin{bmatrix} 0 & -\mathrm{Id} \\ \mathrm{Id} & 0 \end{bmatrix} \in M_{2d}(\mathbb{R}).$$

---

[1]For hypersolvers, the time-step may change, but the numerical scheme is tied to the network.

These systems have been extensively studied from theoretical and numerical perspectives, and more recently from a machine learning point of view. However, many physically relevant problems are not naturally expressed in canonical coordinates, even though they retain an underlying Hamiltonian structure. These systems, known as non-canonical Hamiltonian systems, take the more general form:

$$\dot{z} = W(z)^{-1}\nabla H(z), \tag{2.1}$$

where $W(z) \in M_{2d}(\mathbb{R})$, skew-symmetric, defines a *closed* and non-singular 2-form (i.e. its inverse defines a Poisson bracket).

2.1 A subclass of non-canonical problems

We will focus on a subclass of non-canonical problems, where the symplectic form W is defined as the *exterior derivative* of a one-form, which guarantees its closedness. It means that it is obtained from a mapping $A : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$, called the symplectic potential, through the relation

$$W = -\mathsf{d}A = (D_z A)^\mathsf{T} - D_z A, \tag{2.2}$$

where $D_z A$ denotes the Jacobian of A. In terms of coordinates, $(D_z A)_{ij} = \frac{\partial A_i}{\partial z^j}$.

The symplectic form must derive from a potential for two reasons. Firstly, in [CMY21], the authors observe that learning the symplectic form as a skew-symmetric matrix without additional structure generates neural models that are often unstable, with rapidly exploding solutions.[2] Secondly, we later use a numerical method (described next in §2.2) which requires that the symplectic potential takes the following simplified form:

$$A(z) = \begin{bmatrix} \vartheta(z) \\ 0 \end{bmatrix}. \tag{2.3}$$

With this choice, denoting $z = (x, y)^\mathsf{T}$ with $x, y \in \mathbb{R}^d$, the symplectic two-form is the block matrix

$$W = \begin{bmatrix} (D_x \vartheta)^\mathsf{T} - (D_x \vartheta) & -(D_y \vartheta) \\ (D_y \vartheta)^\mathsf{T} & 0 \end{bmatrix}, \tag{2.4}$$

with $D_x \vartheta$ and $D_y \vartheta$ denoting the Jacobian of ϑ w.r.t. x and y respectively as in (2.2). Then system (2.1) writes:

$$(D_y \vartheta)^\mathsf{T} \dot{x} = \nabla_y H,$$
$$\left((D_x \vartheta) - (D_x \vartheta)^\mathsf{T} \right) \dot{x} + (D_y \vartheta) \dot{y} = -\nabla_x H,$$

where the gradient is the transpose of the differential.

Remark 2.1. Canonical Hamiltonian systems belong to this category of problems, setting $\vartheta(x, y) = y$. Because every non-canonical system admits a local change of variable which transforms it into a canonical problem, this ansatz on the one-form may always be satisfied, at least locally, up to a change of variable. To our knowledge, the question of whether it can be satisfied globally for any (global) system of coordinates is still open.

---

[2]This might be due to the fact that the symplectic potential ensures the existence of Poincaré invariants (similarly to the Hamiltonian ensuring the existence of an energy invariant). Similarly, the behavior of geometric methods relies on generating functions [HLW06, VI.5], which use the symplectic potential.

This has the immediate advantage of characterising the invertibility of W by the local injectivity of $y \mapsto \vartheta(x, y)$ at fixed x, giving a direct criterion for well-posedness of the problem. The system may also readily be inverted, yielding the vector field

$$\begin{cases} \dot{x} = (D_y \vartheta)^{-\mathsf{T}} \nabla_y H, \\ \dot{y} = (D_y \vartheta)^{-\mathsf{T}} \left((\nabla_x \vartheta - \nabla_x \vartheta^{\mathsf{T}})(\nabla_y \vartheta)^{-\mathsf{T}} \nabla_y H - \nabla_x H \right). \end{cases} \tag{2.5}$$

From this identity, one may apply standard numerical methods, as in [CMY21] where high-order Runge-Kutta schemes with adaptive time-steps are used. Here, however, we are interested in schemes specifically designed for this type of problems, namely "degenerate variational integrators".

2.2 Degenerate variational integrators

Just as symplectic integrators are essential for the long-term numerical integration of canonical Hamiltonian systems [HLW06],preserving the geometric structure and qualitative behavior of the exact flow—it is equally important to design structure-preserving integrators for non-canonical Hamiltonian systems. These systems often possess a generalized symplectic or Poisson structure that must be respected to ensure physically meaningful simulations over long time intervals. One of the main approaches to achieving this goal is to recast the problem in a Lagrangian framework, and to construct discrete integrators that mimic the variational derivation of the equations of motion. This leads to the class of *degenerate variational integrators* (DVIs) [EFB$^+$18], which extend the ideas of variational integration to degenerate or non-canonical settings, allowing for the preservation of geometric invariants even in the absence of a constant symplectic form. We now aim to recall this Lagrangian formalism and the construction of the corresponding scheme.

Degenerate Lagrangian dynamics. System (2.1)-(2.2) can be obtained by first defining the Lagrangian

$$L(z, z_t) = A(z) \cdot z_t - H(z), \tag{2.6}$$

and the dynamics is then recovered by extremizing the action

$$\int_{t_0}^{t_1} L\left(z(t), \frac{d}{dt}z(t)\right) dt,$$

yielding the so-called Euler-Lagrange equations:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial z_t^i}\big(z(t), \dot{z}(t)\big)\right] = \frac{\partial L}{\partial z^i}\big(z(t), \dot{z}(t)\big), \quad \text{for } i \in \{1, ..., 2d\},$$

which, owing to the chain rule, can be written as

$$\sum_{j=1}^{2d}\left[\frac{\partial^2 L}{\partial z_t^i \partial z^j}\big(z(t), \dot{z}(t)\big)\dot{z}^j + \frac{\partial^2 L}{\partial z_t^i \partial z_t^j}\big(z(t), \dot{z}(t)\big)\ddot{z}^j\right] = \frac{\partial L}{\partial z^i}\big(z(t), \dot{z}(t)\big), \quad \text{for } i \in \{1, ..., 2d\}. \tag{2.7}$$

This particular choice of Lagrangian is called *degenerate*, as the vanishing of the Hessian w.r.t. z_t removes the second-order derivative $\ddot{z}$ from this last equation. Instead, it results in (2.1) with the two-form (2.2).

When considering a symplectic potential A as in Eq. (2.3), denoting $z = (x, y)^\mathsf{T}$ with $x, y \in \mathbb{R}^d$, the Lagrangian writes

$$L(x, y, x_t, y_t) = \vartheta(x, y) \cdot x_t - H(x, y). \tag{2.8}$$

To avoid any confusion with (2.6), in the sequel we call this a *properly degenerate* Lagrangian. We note that this choice arises naturally, using the so-called "tautological one-form" $\vartheta(x, y) = y$.

Degenerate variational integrators (DVI). Variational integrators are numerical schemes based on a discrete Lagrangian $(z_0, z_1) \mapsto L_h(z_0, z_1)$, for instance $L_h(z_0, z_1) = L(z_0, (z_1 - z_0)/h)$. The time-stepping scheme is then obtained from the *discrete* Euler-Lagrange equations, defined as

$$\nabla_{z_n}[L_h(z_{n-1}, z_n) + L_h(z_n, z_{n+1})] = 0.$$

Here we consider the *degenerate* variational integrator from [EFB$^+$18] which relies on the ansatz (2.8), and is derived from the discrete Lagrangian

$$L_h(x_n, x_{n+1}, y_{n+1}) = \vartheta(x_{n+1}, y_{n+1}) \cdot \frac{x_{n+1} - x_n}{h} - H(x_{n+1}, y_{n+1}). \tag{2.9}$$

The time-stepping scheme stems from the discrete Euler-Lagrange equations, which in this case are written

$$\begin{cases} \nabla_{x_n}\left[L_h(x_{n-1}, x_n, y_n) + L_h(x_n, x_{n+1}, y_{n+1})\right] = 0, \\ \nabla_{y_{n+1}}\left[L_h(x_n, x_{n+1}, y_{n+1}) + L_h(x_{n+1}, x_{n+2}, y_{n+2})\right] = 0, \end{cases}$$

Usually, this would be written differentiating w.r.t. y_n and not y_{n+1}, but since these equations have to be satisfied for every n, the identity above is equivalent to the usual one. This generates the time-stepping scheme

$$\begin{cases} \vartheta(x_{n+1}, y_{n+1}) = \vartheta(x_n, y_n) + (D_x\vartheta(x_n, y_n))^\mathsf{T}(x_n - x_{n-1}) - h\nabla_x H(x_n, y_n), \\ (D_y\vartheta(x_{n+1}, y_{n+1}))^\mathsf{T}x_{n+1} = (D_y\vartheta(x_{n+1}, y_{n+1}))^\mathsf{T}x_n + h\nabla_y H(x_{n+1}, y_{n+1}). \end{cases}$$
$$\tag{2.10}$$

While this may seem like a two-step method with the term x_{n-1} in the first equation, this term may be obtained explicitly using the second equation,

$$x_{n-1} = x_n - h\left(D_y\vartheta(x_n, y_n)\right)^{-\mathsf{T}}\nabla_y H(x_n, y_n).$$

As with the continuous problem, the scheme is well-defined if $D_y\vartheta$ is invertible (up to stability and regularity conditions). In practice, only x_{-1} is computed. For later time-steps, the term x_{n-1} can be used from previous iterations.

We implemented this scheme using the automatic differentiation (AD) routines of `PyTorch` to derive the Euler-Lagrange equations.[3] Because we only simulate problems in low dimensions, Equation (2.10) is solved using a Newton method—the Jacobian also being computed using AD. The initial guess of (x_{n+1}, y_{n+1}) in the Newton iterations is chosen using a single step of the explicit Euler method applied to (2.5).

---

[3]It should be noted that x_{n-1} and x_{n+1} are assumed *independent* of x_n when deriving the Euler-Lagrange equations. In `PyTorch` vernacular, they must be "detached" from x_n before differentiating.

Remark 2.2. The variational integrator is sensitive to gauge choice for the potential ϑ. Indeed, changing $\vartheta(x, y)$ into $\vartheta(x, y) + g(x)$, with a function $g(x)$ with symmetric Jacobian,

$$D_x g(x)^\mathsf{T} - D_x g(x) = 0,$$

does not modify the symplectic matrix W (see Eq. (2.4)) and thus System (2.5), but change the approximate solution computed with (2.10). Although the convergence holds whatever the gauge is, the error constant may become quite large. Numerical examples will be provided in Section 5.

2.3 Long time simulation

As discussed in this section, long time simulations using neural models requires both geometric numerical integration and structure-preserving learning strategies. We first compare the long-time behavior of the standard RK4 scheme with the DVI (§2.2), justifying the use of this specific scheme. Then, by partially reproducing the study of [CMY21], we illustrate the importance of preserving the underlying geometric structure during training, even without numerical considerations. Finally, we show that simply applying the DVI to a neural model may result in unstable solutions, prompting the development of specific training strategies in the next section.

Reference model. This study is conducted on a classical Lotka-Volterra problem which can be found in [HLW06],

$$\dot{x} = x(1 - y), \qquad \dot{y} = y(x - 2). \tag{2.11}$$

This corresponds to the structure (2.5) with maps

$$\vartheta(x, y) = -\ln(y)/x, \qquad H(x, y) = x + y - 2\ln(x) - \ln(y). \tag{2.12}$$

In this case, a change of variable $(q, p) = \big(\ln(x), \ln(y)\big)$ yields a canonical structure with Hamiltonian $(q, p) \mapsto e^q + e^p - 2q - p$. In general, however, such a change of variable is defined locally, not globally. We shall therefore treat this problem as purely non-canonical.

Using the DVI. Another important aspect of long-time simulations mentioned previously is the numerical scheme, which must be appropriate —we introduced the DVI to that effect. Figure 1 compares the results of long-time simulations on the reference model (2.11) using either the standard RK4 (explicit Runge-Kutta of order 4) or the DVI with a large time-step. In short time (clear yellow dots), the RK4 scheme is highly-accurate, as expected. In long time, however, the numerical solution does not stay on the closed orbit and becomes less and less accurate. Here the scheme dissipates energy as can be seen on the right-hand plot. In contrast, the DVI *does* generate a periodic solution, though its orbit is clearly inaccurate (as expected from a first-order scheme). This means that the error is fairly large in short time, but the preservation of qualitative properties ensures that the long-time behaviour of the numerical solution is closer to that of the exact solution.[4]

---

[4]By exact solution, we mean a highly-accurate numerical solution computed using the `solve_ivp` routine of SciPy with parameters `atol=1e-12` and `rtol=1e-10`.
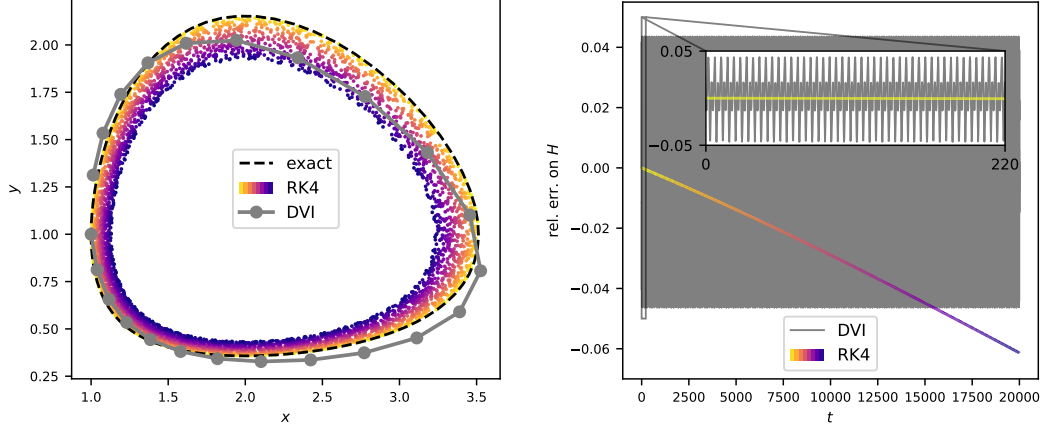
Figure 1: (Lotka-Volterra) Long-time behaviour of the RK4 and DVI numerical schemes, with initial condition $x_0 = 1, y_0 = 1$ and time-step $h = 0.2$. Left: solutions in phase space. Only one out of 16 time-steps is displayed for the RK4 solution (to reduce file size), and only the first time-steps of the DVI solution is displayed (the solution is periodic). Right: relative error on the Hamiltonian as a function of time.

Defining neural networks. In [CMY21], the authors enforce the symplectic structure by learning both the Hamiltonian $H_\Theta : \mathbb{R}^{2d} \to \mathbb{R}$ and the symplectic potential $A_\Theta : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$. Echoing the presentation of §2.1, this allows the construction of the symplectic form $z \mapsto W_\Theta(z) \in M_{2d}(\mathbb{R})$ with automatic differentiation (AD) using (2.2), to find the vector field

$$f_\Theta^{\text{non-can}}(z) := W_\Theta(z)^{-1} \nabla H_\Theta(z) \tag{2.13}$$

(where ∇H is also computed using AD). The networks A_Θ, H_Θ are fitted on collocation points $(z^{(i)}, \dot{z}^{(i)})_{i \in \mathcal{I}}$ where $\mathcal{I} \subset \mathbb{N}$ indexes the data and $\dot{z}^{(i)} = f(z^{(i)})$. This is done by minimizing the loss

$$\mathcal{L}_{\text{vf}}(\Theta) := \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \|\dot{z}^{(i)} - f_\Theta(z^{(i)})\|^2, \tag{2.14}$$

using an optimization method in batches.

Here we further impose the ansatz (2.3) nullifying the second half of the coordinates of A_Θ, and therefore learn only the first half of the symplectic potential $\vartheta_\Theta : \mathbb{R}^{2d} \to \mathbb{R}^d$. This allows the use of the Degenerate Variational Integrator [EFB+18] presented in Section 2.2. To recover the vector-field, the symplectic form is constructed with (2.4).

Hard-coding structure. To show that the structure of model (2.13) is important, let us compare the solutions of three different neural models with the reference (2.11). The baseline neural model is a single multi-layer perceptron (MLP)

$$z \mapsto f_\Theta^{\text{no-struct}}(z) \tag{2.15}$$

(with 2 hidden layers of dimension 40) which represents the vector-field. The "canonical" neural model, usually called HNN [GDY19], consists a single MLP H_Θ from which the vector-field is obtained by automatic differentiation (AD) as

$$z \mapsto f_\Theta^{\text{can}}(z) := J^{-1} \nabla H_\Theta(z). \tag{2.16}$$
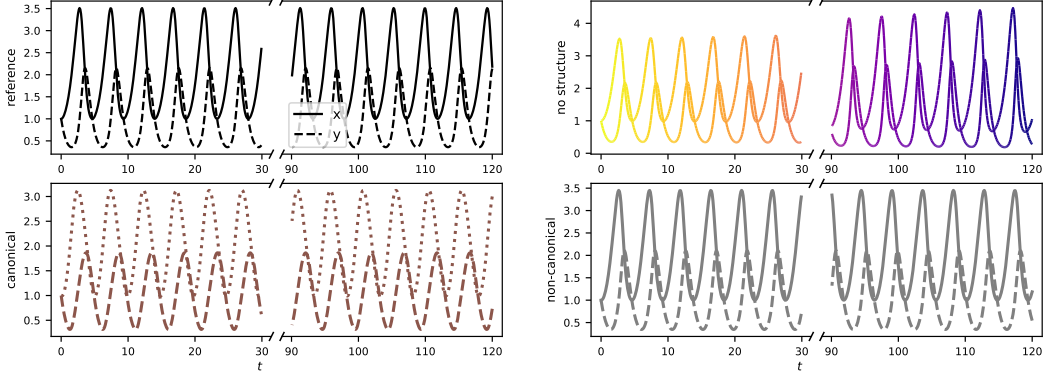
9

Figure 2: (Lotka-Volterra) Solutions with initial condition $x_0 = 1$, $y_0 = 1$ for the different models (reference model (2.11), no structure neural model (2.15), canonical neural model (2.16), non-canonical neural model (2.13)), obtained using `solve_ivp`.

The non-canonical model (2.13), consists of two separate MLPs, the symplectic potential ϑ_Θ and the Hamiltonian H_Θ, both of 2 hidden layers of size 30. For all networks, two trainings are performed using the Adam optimizer and batches of size 500. The first training uses a learning rate $\mathrm{lr} = 10^{-2}$ for 50 epochs, and the second $\mathrm{lr} = 10^{-3}$ for 150 epochs. The dataset and the MLP architecture will be detailed in the next sections.

Figure 2 displays a highly accurate numerical solution of each model, computed using the `solve_ivp` routine of SciPy with parameters `rtol=1e-10` and `atol=1e-12`. The same solutions are shown in phase-space in Figure 3 (left).Instead of yielding a closed (periodic) orbit, the baseline model diverges over time, making it unsuitable for long-time simulation. While the canonical model generates a closed orbit, it is also visible that it cannot accurately recreate the non-canonical vector-field. Despite the short training and the small size of the networks, the non-canonical model recovers accurate dynamics.

On the right-hand graphs, it appears that on the trajectory of this model, the (reference) Hamiltonian is preserved up to $\sim 2\%$, while the baseline without structure diverges. This is computed by evaluating the Hamiltonian function (2.12) at every time for each model's trajectory. We do not compare the learnt Hamiltonian functions with the reference one, though each geometric neural model preserves its own Hamiltonian.

Unexpected instability. We now show that the DVI is sensitive to the gauge of the model, which turn out to be important when the model is learnt from data. Consider the rightmost plot of Figure 4, which shows that the solution of the neural model with structure obtained as in [CMY21] (dotted black) is accurate compared to the reference model (gray). In purple is the numerical solution using the DVI for two different times steps. Comparing it to the reference model (left plot), the DVI is apparently less stable for the neural model. In fact, when plotting several trajectories, many diverge after a few iterations (this will be shown in §5.1).

Since $d = 1$, the equations of the continuous dynamics (2.5) are simplified, as the antisymmetric part of $D_x \vartheta = \partial_x \vartheta$ simply vanishes:

$$\begin{cases} \dot{x} = \partial_y H(x,y)/\partial_y \vartheta(x,y), \\ \dot{y} = -\partial_x H(x,y)/\partial_y \vartheta(x,y). \end{cases}$$
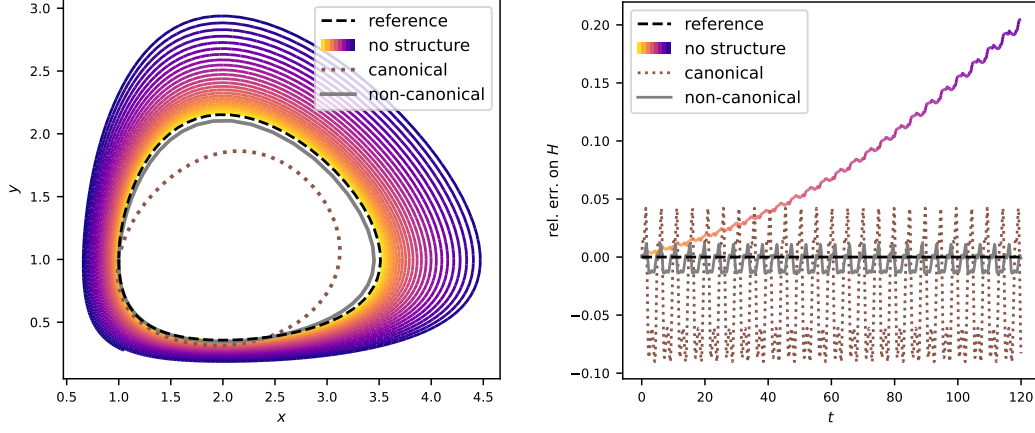
Figure 3: (Lotka-Volterra) Solutions with initial condition $x_0 = 1$, $y_0 = 1$ for the different models (reference model (2.11), no structure neural model (2.15), canonical neural model (2.16), non-canonical neural model (2.13)), obtained using `solve_ivp`. Left: phase portrait. Right: time evolution of the relative error of the Hamiltonian (2.12) for the different solutions.
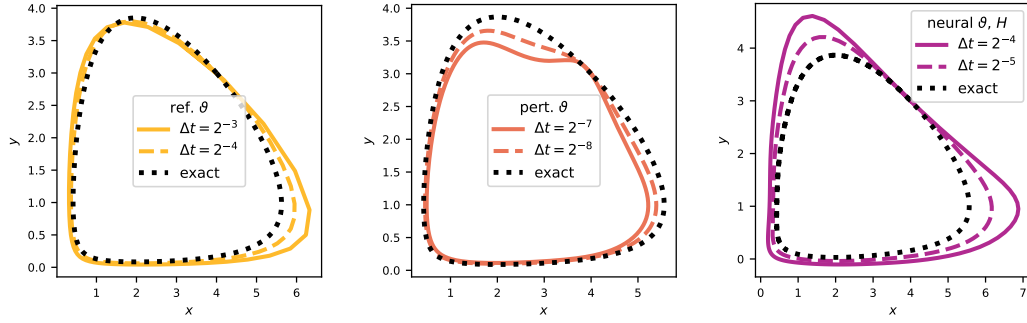

Figure 4: (Lotka-Volterra) Solutions with initial condition $x_0 = 4$, $y_0 = 3$ obtained using the first-order DVI scheme with different time steps for different models. Left: reference model (2.12). Middle: a perturbed model $\vartheta(x, y) \leftarrow \vartheta(x, y) + \frac{1}{2}\cos(2x)$. Right: a non-canonical neural model (2.13). Exact solutions refers to refined solutions obtained using `solve_ivp` on the same model.

However, this simplification does not occur for the discrete scheme, which writes

$$
\begin{cases}
\dfrac{\ln(y_{n+1})}{x_{n+1}} = \dfrac{\ln(y_n)}{x_n}\left(1 - h(1 - y_n)\right) + h\left(1 - \dfrac{2}{x_n}\right), \\
x_{n+1} = x_n + hx_{n+1}(1 - y_{n+1}),
\end{cases}
\tag{2.17}
$$

where we plugged in $x_n - x_{n-1} = hx_n(1 - y_n)$. While the second component is a standard implicit Euler method on $\dot{x}$, the first component is more exotic, involving a logarithm which would never appear when considering only the vector field.

The expression of the scheme shows that, while the continuous dynamics remain unchanged under any gauge perturbation of the potential (see Remark 2.2), this is not the case for the scheme. It thus becomes clear that if the learning process has captured a potential with a large perturbation depending only on x, then the scheme may produce completely different results. This is illustrated in the center plot of Figure 4, where we deliberately modify the potential to make the problem more stiff. We argue that such gauge perturbations may occur during training, and propose a way to avoid them in the next section.

3 Learning the vector field with regularization

As previously discussed, the degenerate Lagrangian systems considered here are invariant under certain potential perturbations. Consequently, a learning procedure that preserves this structure may generate a whole family of potentials leading to the same trajectory. However, this property does not generally hold when using a structure-preserving scheme such as the Discrete Variational Integrator (DVI), which is not gauge-invariant and thus leads to the numerical issues discussed earlier (see Fig. 4). Therefore, to apply a DVI to a learned non-canonical system, one must prevent the learning process from producing potentials that significantly deviate from the true one. A fairly natural approach is to add a penalization term that constrains the space of admissible potentials. This can also be seen as introducing a prior on the potential being learned.

As in (2.14), the networks $\vartheta_\Theta, H_\Theta$ are fitted on collocation points $(z^{(i)}, \dot{z}^{(i)})_{i \in \mathcal{I}}$ where $\mathcal{I} \subset \mathbb{N}$ indexes the data and $\dot{z}^{(i)} = f(z^{(i)})$ is given by the reference model. The loss now also includes a regularization term $(z, \dot{z}) \mapsto r_\Theta(z, \dot{z}) \in \mathbb{R}^{2d}$ with weight $\varepsilon > 0$, and is therefore written

$$
\mathcal{L}_{\mathrm{vf}}(\Theta) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left[\|\dot{z}^{(i)} - W_\Theta^{-1}(z^{(i)}) \nabla H_\Theta(z^{(i)})\|^2 + \varepsilon \, \|r_\Theta(z^{(i)}, \dot{z}^{(i)})\|^2 \right],
\tag{3.1}
$$

As always, the optimization is performed in batches.

The key challenge now is to propose a penalization term that effectively prevents undesirable symplectic potentials.

3.1 The regularization term

When performing a standard (local) error analysis on the DVI (see §A.2), denoting x_h the solution after one time-step, we find the dominant error term

$$
\begin{cases}
x(h) - x_h = -\dfrac{h^2}{2}\ddot{x} + \mathcal{O}(h^3), \\
y(h) - y_h = \dfrac{h^2}{2} D_y \vartheta^{-1}\left(\dddot{\vartheta} + D_x \vartheta \ddot{x}\right) + \mathcal{O}(h^3).
\end{cases}
\tag{3.2}
$$

On one hand, this is unsurprising: the error of a scheme of order p generally only involves time-derivatives of order $p+1$ and higher. On the other, these time-derivatives are usually in the coordinates (here x and y), whereas here the gauge dependency is clear: the solution is accurate only if ϑ is sufficiently smooth, the notion of sufficient depending on the order of the scheme used.

Since our final goal here is to apply the first-order DVI, the penalization term $(z, z_t) \mapsto r_\Theta(z, z_t)$ is chosen to be precisely this dominant error term (removing the value h which can be arbitrary). It is computed by inverting the linear system

$$\begin{bmatrix} D_x\vartheta_\Theta(z) + \left(D_x\vartheta_\Theta(z)\right)^\mathsf{T} & D_y\vartheta_\Theta(z) \\ \left(D_y\vartheta_\Theta(z)\right)^\mathsf{T} & 0 \end{bmatrix} r_\Theta(z, z_t) = D_z[\nabla_z L_\Theta(z, z_t)]z_t, \qquad (3.3)$$

where $L_\Theta(z, z_t) = \vartheta_\Theta(x, y)^\mathsf{T} x_t - H_\Theta(x, y)$, with the differentials only applied to the corresponding argument, not to z_t. The derivation of this expression is detailed in Appendix A.2.

As before, this cost function involves a number of derivatives and thus requires automatic differentiation to compute the derivatives of the Lagrangian and Hamiltonian, as well as to account for the derivative of the matrix inverse with respect to the potential parameters appearing in the matrix. Since the matrices are small, differentiating through the matrix inversion does not necessarily require additional treatment, such as applying the implicit function theorem [BR24]. Now that the modified learning problem has been introduced, we propose to provide the technical details of the model and those necessary for the learning process.

3.2 Hyperparameters of neural networks

Parameterization. The parameterized symplectic potenial ϑ_Θ and Hamiltonian H_Θ are both defined as standard multi-layer perceptron (MLP) neural networks, of the form

$$z \mapsto \mathcal{F}_{\Theta_L} \circ \rho_{\mu_{L-1}} \circ \mathcal{F}_{\Theta_{L-1}} \circ ... \circ \rho_{\mu_1} \circ \mathcal{F}_{\Theta_1}(z), \qquad (3.4)$$

with parametric linear layers $\mathcal{F}_{\Theta_\ell}$ and activation layers ρ_{μ_ℓ} and

$$\Theta = (\Theta_L, \mu_{L-1}, \Theta_{L-1}, ..., \mu_1, \Theta_1).$$

The linear layers correspond to an affine mapping $\mathcal{F}_{\Theta_\ell}(h) = M_\ell h + b_\ell$ with parameters $\Theta_\ell = (M_\ell, b_\ell)$ a weight matrix and vector bias respectively. For activation layers, we chose the so-called "self-scalable $\tanh$" [GSC$^+$23], $\rho_{\mu_\ell}(h) = \tanh_\odot(h) + \mu_\ell \odot h \odot \tanh_\odot(h)$ with a parameter vector μ_ℓ, where $\odot$ denotes the componentwise product (or Hadamard product) and $\tanh_\odot$ is the hyperbolic tangent applied componentwise. This activation function was originally developed for multiscale physics-informed neural networks where the network is fitted using its differentials, which we also do here.

Normalizing the inputs. The different coordinates might involve different scales, both for the input z and the (indirect) output $\dot{z}$. For the input, we simply add a coordinate-wise normalization layer before each neural network, such that each coordinate has values in $[0, 1]$, i.e. there is a pre-processing layer

$$\begin{bmatrix} z^1 \\ \vdots \\ z^{2d} \end{bmatrix} \mapsto \begin{bmatrix} \frac{z^1 - z^1_{\min}}{z^1_{\max} - z^1_{\min}} \\ \vdots \\ \frac{z^{2d} - z^{2d}_{\min}}{z^{2d}_{\max} - z^{2d}_{\min}} \end{bmatrix}, \qquad (3.5)$$

the minima and maxima being determined *a priori* from the training set.

Normalizing the outputs. It is standard for the outputs of neural networks to be normalized to facilitate learning. Here, this is crucial for the dynamics on all components to be accurate. In the forthcoming numerical examples, this will be especially true for the guiding center test case, in which some components differ by up to 4 orders of magnitude (see Figure 23).

Since the outputs of neural networks do not correspond to the vector field directly, they cannot be rescaled. The vector-field data also cannot be normalized, since it corresponds to a physical quantity and determines the dynamics. Therefore, the normalization is done in the training loss (3.1): the standard Euclidean norm is replaced with the data-informed norm:

$$\|u\|^2 = u^\mathsf{T} M^{-1} u, \tag{3.6}$$

where $M = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \dot{z}^{(i)}(\dot{z}^{(i)})^\mathsf{T}$ is a Gram matrix associated with the data.

Because the Gram matrix is symmetric positive definite,[5] we compute this norm using the inverse its symmetric (positive) square root $M = M^{1/2}M^{1/2}$, taking the Euclidean norm of $v = M^{-1/2}\delta\dot{z}$. This is obtained from a singular value decomposition $M = U\Sigma U^\mathsf{T}$ with $\Sigma = \mathrm{diag}(\sigma_1^2, ..., \sigma_{2d}^2)$ and $U^\mathsf{T}U = I_{2d}$ (owing to the symmetry of M), setting $M^{-1/2} = U\Sigma^{-1/2}U^\mathsf{T}$. Here this is computed "on the fly" for each batch, but it could be computed prior to the training. If this decomposition is too costly, one might prefer a componentwise scaling $M = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \mathrm{diag}\big(\dot{z}_1^{(i)}, ..., \dot{z}_{2d}^{(i)}\big)^2$, to which a low-rank correction might also be applied.

4   Learning the time-discrete dynamics through the scheme

The second learning strategy aims at fitting the symplectic potential and the Hamiltonian so as to minimize the one-step error of the numerical scheme on a trajectory. As such, the training data consists of collocation triples $(z_0^{(j)}, z_1^{(j)}, z_2^{(j)})_{j\in\mathcal{J}}$ indexed on $\mathcal{J} \subset \mathbb{N}$, which correspond to data at times $(t^{(j)}, t^{(j)} + h, t^{(j)} + 2h)$ for a given time step $h > 0$. We look for a symplectic potential $\tilde{\vartheta}_\Theta$ and a Hamiltonian $\tilde{H}_\Theta$ that nullify the mean one-step error of the scheme, denoted S_Θ:

$$S_\Theta(z_0, z_1, z_2; h) = \begin{bmatrix} \tilde{\vartheta}_\Theta(x_2, y_2) - \Big(\tilde{\vartheta}_\Theta(x_1, y_1) + \big(D_x\tilde{\vartheta}_\Theta(x_1, y_1)\big)^\mathsf{T}(x_1 - x_0) - h\nabla_x\tilde{H}_\Theta(x_1, y_1)\Big) \\ (D_y\tilde{\vartheta}_\Theta(x_2, y_2))^\mathsf{T} x_2 - \Big((D_y\tilde{\vartheta}_\Theta(x_2, y_2))^\mathsf{T} x_1 + h\nabla_y\tilde{H}_\Theta(x_2, y_2)\Big) \end{bmatrix}. \tag{4.1}$$

This means that the learnt quantities may not exactly match the physical ones but a "modified" symplectic potential and Hamiltonian. Consequently, simulations on the learnt dynamics will be more accurate, similarly to HyperSolvers [PMY+20].

Remark 4.1. While similar, this approach presents two key differences with those of Deep Solvers [SCL20, BCLM23] and HyperSolvers [PMY+20]. First, the network does not learn a correction term, it learns the entire Lagrangian. Second, the time-step is considered fixed, it is not an input of the neural networks. This first difference makes the learning more difficult, but the second makes it more simple—overall, it is just different. As pointed

---

[5]To avoid the case where M is only symmetric positive *semi*-definite, we actually add a small offset εI_d.

out in [DM21], using an scheme which preserves structure theoretically allows the network to reach a near-zero learning error.

4.1 The loss of scheme-learning

Naively minimizing the mean squared norm of this error would be unsuccessful, because the trivial symplectic potential and Hamiltonian, $\tilde{\vartheta}_\Theta = 0$ and $\tilde{H}_\Theta = 0$, are actually solutions. This was already taken into account in [OO23, OO24], where authors learn Lagrangians from other variational integrators by minimizing this error. To circumvent this, they add a penalization term on the Jacobian

$$J_\Theta(z_0, z_1, z_2; h) = D_{z_2} S_\Theta(z_0, z_1, z_2; h),$$

encouraging either its determinant or its smallest singular value to be non-zero. Our approach also combines an error term and a regularization term (weighed by some $\varepsilon > 0$), with the loss

$$\mathcal{L}_{\mathrm{sch}}(\Theta) = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \left[\left\| (J_\Theta^{(j)})^{-1} S_\Theta^{(j)} \right\|_{\mathrm{sch}}^2 + \varepsilon \, \log_{10}\big(\kappa(J_\Theta^{(j)})\big) \right], \qquad (4.2)$$

where $S_\Theta^{(j)}$ (resp. $J_\Theta^{(j)}$) is the one-step error (resp. its Jacobian) evaluated at the j-th data point $(z_0^{(j)}, z_1^{(j)}, z_2^{(j)})$. The optimization is performed in batches. Let us briefly explain this choice of loss function.

Error term. Due to the multiscale nature of the problems we consider, we found it necessary to employ a similar strategy as in the continuous case (3.1). By minimizing the one-step error directly, the multiscale components of the data are not captured, and no data-informed rescaling is possible. We therefore minimize the error of Newton-Raphson iterations near the solution, which are homogeneous to the variations in phase-space and enable the use of a data-informed norm

$$\|u\|_{\mathrm{sch}}^2 = u^\mathsf{T} M_{\mathrm{sch}}^{-1} u, \qquad (4.3)$$

where the Gram matrix $M_{\mathrm{sch}} := \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} (z_2^{(j)} - z_1^{(j)})(z_2^{(j)} - z_1^{(j)})^\mathsf{T}$ is the discrete equivalent of (3.6). If this is zero, then so is the error of the scheme (4.1), while avoiding the trivial quantities.

Regularization term. This error term ensures the accuracy of the scheme, but may fail if the Jacobian stops being invertible. To enforce this invertibility, we add a penalization term on its condition number, which is

$$\kappa(J) = \|J\| \, \|J^{-1}\| . \qquad (4.4)$$

This also ensures that the linear system of Newton iterations are well-conditioned. The logarithm ensures that this error term remains small even when the physical system presents large variations on the dataset.

This term is useful mostly at the start of the training, when the neural model varies a lot and the Jacobian may become degenerate. Unlike in VF learning where the regularization term needs to be minimized for the DVI to be well-behaved, here it seems to have no impact on the final learnt model and requires only a very small weight of $\varepsilon = 10^{-8}$.

4.2 Using very large time-steps

The scheme (2.10) being geometric means that formally, there exists a *modified* one-form $z \mapsto \vartheta_h(z)$ and a *modified* Hamiltonian $z \mapsto H_h(z)$. Both are close to the original quantity up to $\mathcal{O}(h^2)$.

Owing to backward error analysis, we argue that the learnt dynamics are inaccurate outside of discrete times $t \in \{0, h, 2h, ...\}$, or with another scheme. Therefore it seems fairly natural for long-time simulations to wish to choose h very large. What could limit the size of the time-step with our approach?

1. The modified quantities differ from the original ones up to an error of size $\mathcal{O}(h^2)$, and a large error may be compounded by iterations;

2. The modified quantities ϑ_h and H_h learnt by the network may only exist formally, meaning they can only be computed up to a flat function in $h \to 0$ (i.e. all the Taylor coefficients vanish at 0); the approximation is then valid only for small h.

3. During simulations, one must compute an initial guess of z_{n+1}. Standard approaches may yield guesses too inaccurate for optimisation techniques to solve the scheme, especially if it lands outside of the training set.

For the first remark, the main risk is to learn an inaccurate symplectic form, which would yield a non-symplectic scheme—this is different from SympNets, which work with a known symplectic form. Also, much like SympNets, there is indeed no guarantee that the Hamiltonian is exactly preserved along iterations: there might be a drift due to compounding errors. This drift exists for every method of learning, and should not invalidate our approach. For the second remark, it is important to known that these formal results are an "at worst" case, and it may still be beneficial to evaluate this approach for specific test cases. For example, let us ignore structure and consider the Euler scheme applied to $\dot{y} = f(y)$ of flow $(t, y_0) \mapsto \varphi^t(y_0)$. The modified vector field f_h verifies the identity $\varphi^h = \mathrm{id} + h\, f_h$, and therefore is well-defined for all timesteps. Finally, for the third remark, it may be useful to use a neural network for the initial guess. This baseline NN should not require a particular structure, but may be invertible [AKW$^+$19].

5 Numerical experiments

The different learning strategies are tested on the same Neural Degenerate Lagrangian architecture: the symplectic one-form and the Hamiltonian are different MLPs, each with their own parameters, and normalized inputs. The optimizer is always Adam.

5.1 Lotka-Volterra

We consider here the same Lotka-Volterra problem as the one introduced in §2.3, where the difficulties of the learning strategies have been exposed. We therefore choose not to recall them here, but this time we will follow the approaches described in Sections 3 and 4.

Hyperparameters. The dataset consists in 2000 trajectories, separated between a training, a test and a validation set, representing 80%, 15% and 5% of the data respectively. As in §2.3, their initial conditions are uniformly sampled in the subset of phase space such that $H(x, y) \leq 4.4$, and evolve over 5 time-steps with $h = 0.1$. This time-step and maximum
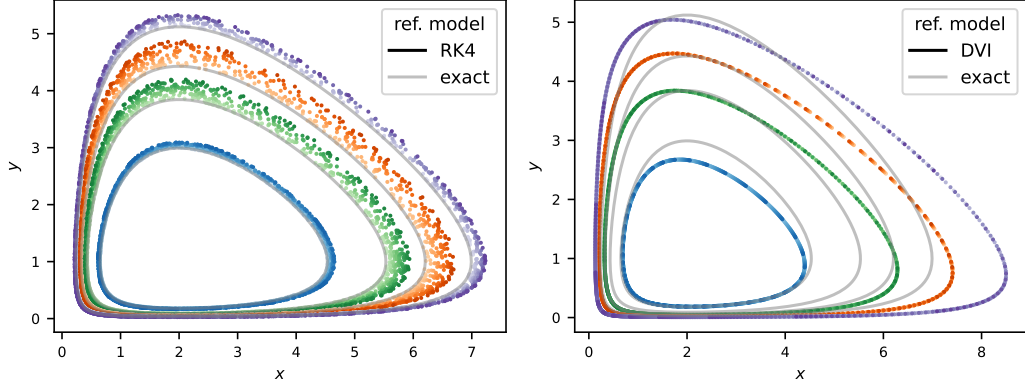
Figure 5: (Lotka-Volterra) Numerical solutions of the reference model (2.11) with different initial data. Left: RK4 solver with time-step $h = 0.1$ and 250k steps. Right: DVI solver with time-step $h = 0.1$ and 100k steps. Points are displayed every 51 steps. Exact solutions are refined numerical solutions of the reference model.
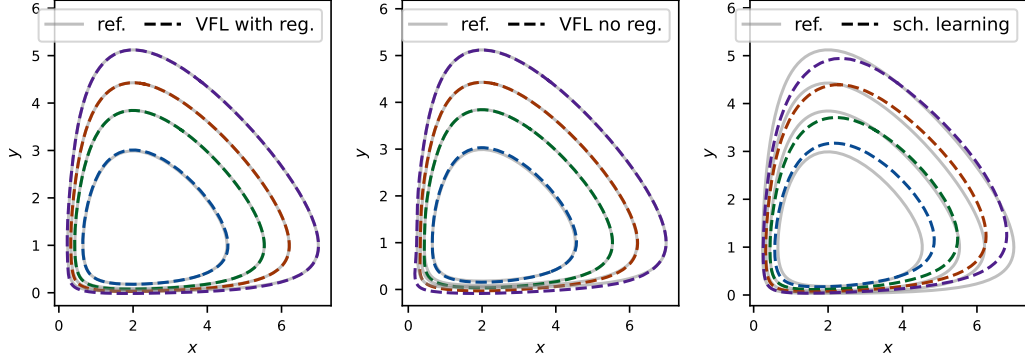

Figure 6: (Lotka-Volterra) Refined numerical solutions for each neural model (dashed lines), compared with the refined solutions of the reference model (solid lines), obtained by using `solve_ivp`. In reading order: VF learning (VFL) with and without regularization, and scheme learning.

energy are chosen such that trajectories at the edge of the domain are barely stable when using the DVI scheme.

Each network ϑ_Θ and H_Θ, independently of the learning strategy, consists of three layers of size 30, and the inputs are normalized as described in 3.2. The models undergo four trainings, first during 20 epochs with a learning rate of 10^{-2}, then 500 epochs with lr 10^{-3} and finally two trainings of 500 epochs with lr 10^{-4}. The regularization parameter is set to $\varepsilon = 10^{-6}$ for VF learning (§3) and $\varepsilon = 10^{-8}$ for scheme learning (§4).

Results of vector-field learning. In Figure 6, we compute highly-accurate solutions of the neural models using the `solve_ivp` routine of Scipy with parameters `rtol=1e-10` and `atol=1e-12` over a single period, and compare them to exact solutions of the reference model that are computed the same way. Over short times, both VF-learning neural networks closely match the reference model, though the regularized model is slightly more accurate near the origin. This matches the training metrics in Figure 7, where the error term in the loss is slightly larger for the non-regularized model, but both are small (of order 10^{-4}). It appears that our new regularization term has no negative impact on the
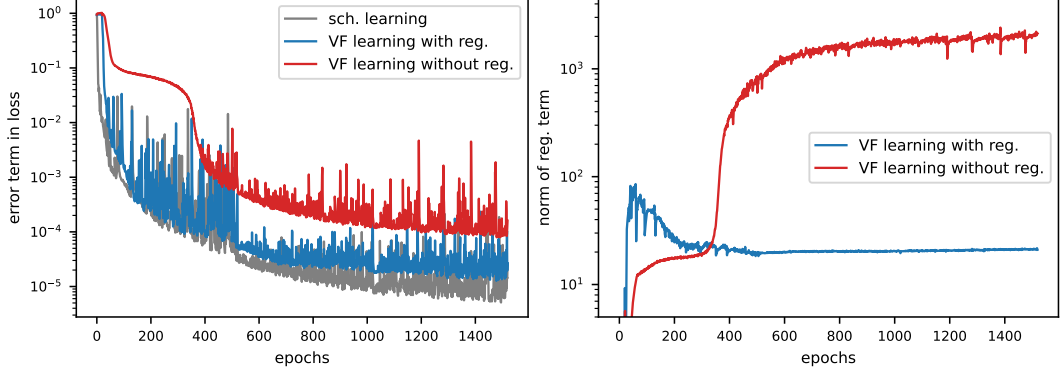
Figure 7: (Lotka-Volterra) Evolution of the loss and the regularization term on the test dataset throughout the training.
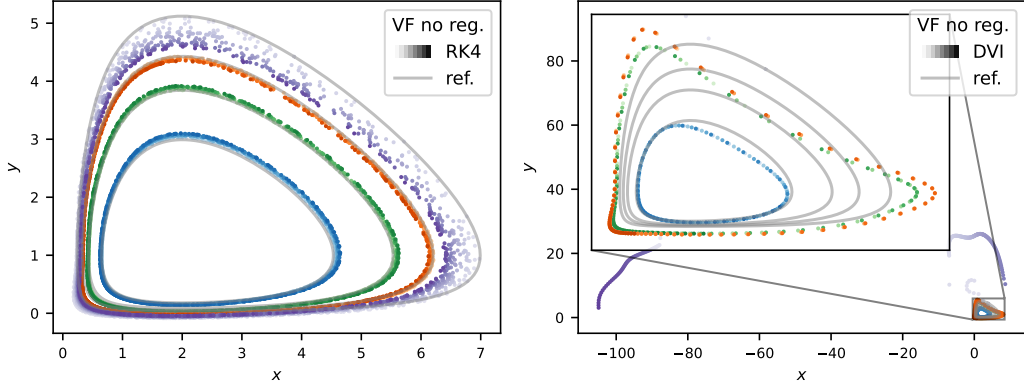

Figure 8: (Lotka-Volterra) Numerical solutions of the non-canonical neural model, obtained without regularization, for different initial conditions. Left: RK4 solver with time-step $h = 0.1$ for 250k steps. Right: DVI solver with time-step $h = 0.1$ for 200 steps. Points are displayed every 51 steps for the RK4 solution, every step is shown for the DVI solution. Reference solutions are refined numerical solutions of the reference model.

accuracy of the neural model.

When observing the long-time behavior of both models in Figures 8 and 9 (and the reference model in Figure 5), the qualitative properties of the numerical schemes still hold. Over short time intervals, the DVI is less accurate than RK4 due to its first-order nature and the large time step used. However, over longer time periods, the DVI maintains a periodic solution that does not degrade, unlike the RK4 scheme which produces a dissipative solution. Because of this periodicity, we stop after 100k time-steps for the (more costly) DVI, while we perform 250k time-steps for the RK4 scheme.

Comparing the behavior of the DVI on both VF-learning models, it is clear that the penalization method successfully addresses the gauge issue identified in Section 2 (Figure 4 on the right): without it, the DVI is unstable, and with it, the DVI is more accurate than on the reference model. This matches the trace of the regularization term r_Θ during training in Figure 7 (with weight $\varepsilon = 0$ for the model "without regularization"), which is much larger if it is not included in the loss. This further supports the idea that learning the structure alone is not sufficient. One must also use a time integration scheme that is adapted to this structure and, in this case, take this scheme into account during the training process.
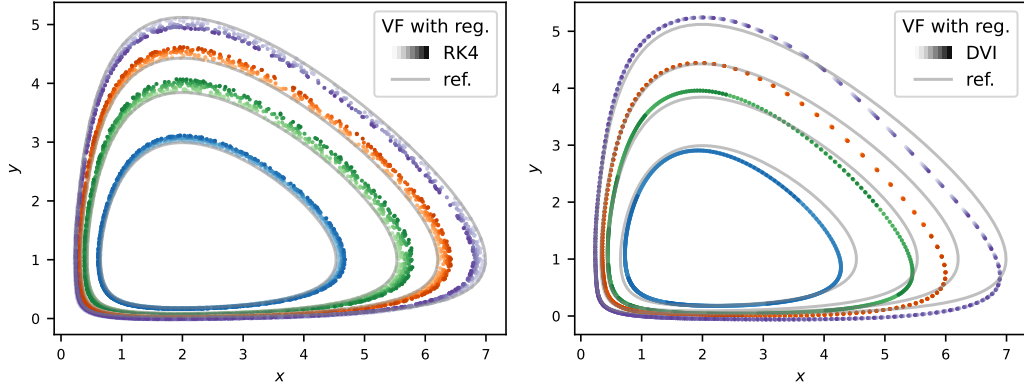
18

Figure 9: (Lotka-Volterra) Numerical solutions of the VF-learning neural model, obtained with regularization, for different initial conditions. Left: RK4 solver with time-step $h = 0.1$ for 250k steps. Right: DVI solver with time-step $h = 0.1$ for 100k steps. Points are displayed every 51 steps for the RK4 solution, every step is shown for the DVI solution. Reference solutions are refined numerical solutions of the reference model.
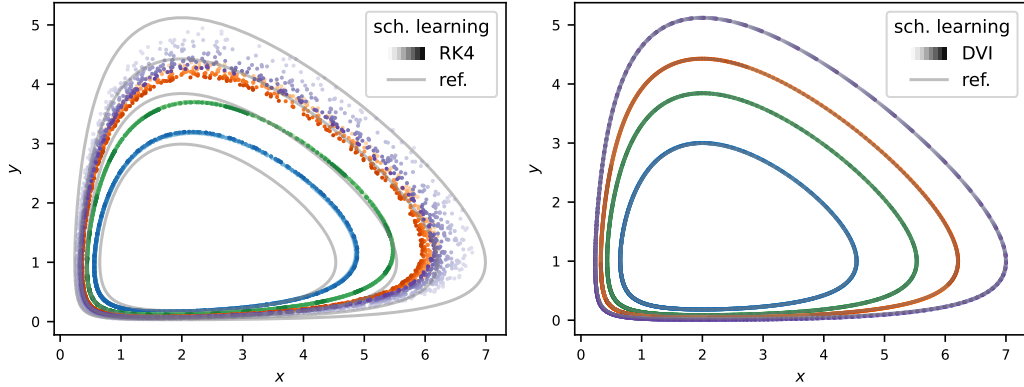

Figure 10: (Lotka-Volterra) Numerical solutions of the scheme-learning neural model for different initial conditions. Left: RK4 solver. Right: DVI solver. They are obtained using a time-step $h = 0.1$ (same as in the dataset) for 100k steps. Points are displayed every 31 steps. Reference solutions are refined numerical solutions of the reference model.
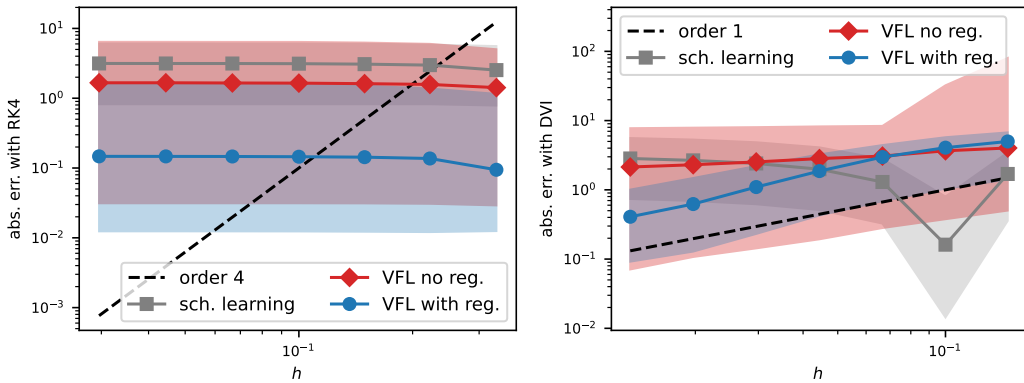

Figure 11: (Lotka-Volterra) Short-time numerical error for the different learning strategies with the RK4 scheme (left) and the first-order DVI (right), as measured on the validation set up to $t = 10.0$. The values drawn being the median (solid line), as well as the 5th and 95th percentiles (filled area).

Results of scheme learning. The simulations in Figure 10 using the DVI demonstrate that the scheme-learning corrects the error of the DVI, generating highly accurate closed solutions over long times. This suggests that the training process learns a modified potential and Hamiltonian that compensate for the scheme-induced error. Since the DVI scheme is used during training to perform time stepping, the results are consistent with those reported in [JZZ$^+$20], where learning a discrete flow was shown to yield higher accuracy than learning the underlying ODE.

However, it is important to note that the learned model is specific to the scheme and time step used during training, and its performance is likely to degrade when either is altered. The short-time error analysis in Figure 11 shows that the median numerical error is lowest when using the DVI with time-step $h = 0.1$, which matches the training data. When using smaller time-steps, the error degrades and plateaus while the VF-learning models converge with order 1 as expected (up to a learning error). The numerical solution using RK4 scheme is also much less accurate, even with a time-step $h = 0.1$. This can also be seen in Figure 6, where the exact solution of the scheme-learning model is much less accurate than the numerical solution yielded by the DVI in Figure 10.

5.2 Massless charged particle

Consider a standard Lorentz force in two dimensions (z the 2D space variable and v the 2D velocity) with magnetic potential $z \mapsto A(z)$ and electric potential $z \mapsto \varphi(z)$, applied to a particle of mass m and charge q. Classically, the Lagrangian is $(z, v) \mapsto \frac{1}{2}m|v|^2 + qA(z) \cdot v - q\varphi(z)$. Writing $z = (x, y)$, the magnetic potential can be chosen as $(\vartheta, 0)$ and still generate the same magnetic field $B = \partial_x A_y - \partial_y A_x = -\partial_y \vartheta$, setting $\vartheta(x, y) = A_x(x, y) - \int_0^y \partial_x A_y(x, y) \mathrm{d}y$. In the limit $m \to 0$, the kinetic energy disappears and the velocity can be replaced by z_t in the (rescaled) Lagrangian (with $q = 1$),

$$L(x, y, x_t) = \vartheta(x, y)x_t - \varphi(x, y).$$

For this experiment, the magnetic potential is chosen to increase from the origin as $A(x, y) = \frac{A_0}{2}(1 + x^2 + y^2)(-y, x)^\mathsf{T}$, i.e.

$$\vartheta(x, y) = -A_0\, y \left(1 + 2x^2 + \tfrac{2}{3}y^2\right), \tag{5.1}$$

which generates a magnetic field $B(x, y) = A_0(1 + 2x^2 + 2y^2)$. The electrostatic potential is

$$\varphi(x, y) = E_0\big(2 - \cos(x) - \sin(y)\big). \tag{5.2}$$

According to (2.5), this yields the dynamics $\dot{x} = -\partial_y \varphi/B$ and $\dot{y} = \partial_x \varphi/B$, i.e.

$$\dot{x} = \frac{E_0 \cos(y)}{A_0(1 + 2x^2 + 2y^2)}, \qquad \dot{y} = \frac{E_0 \sin(x)}{A_0(1 + 2x^2 + 2y^2)}. \tag{5.3}$$

This is coherent with the classical Lorentz dynamics $m\dot{v} = q\big(E(z) + \dot{z} \times B(z)\big)$ in the limit $m \to 0$, with $E = -\nabla \varphi$ and $B = \nabla \times A$. In the sequel, we set $A_0 = 1$ and $E_0 = 1$.

Hyperparameters We generate 15k initial conditions (x_0, y_0) in the ball of radius π centered at $(0, \pi/2)$ such that the electrical energy is not greater than 1.5, i.e. in the set $\{(x, y) \in B_\pi(0, \pi/2) \mid \varphi(x, y) \le 1.5\}$. This is done using a Latin hypercube method in polar coordinates around $x = 0, y = \pi/2$ and filtering the data based on the value of φ. For scheme-learning, the time-step is set to be $h = 0.5$.
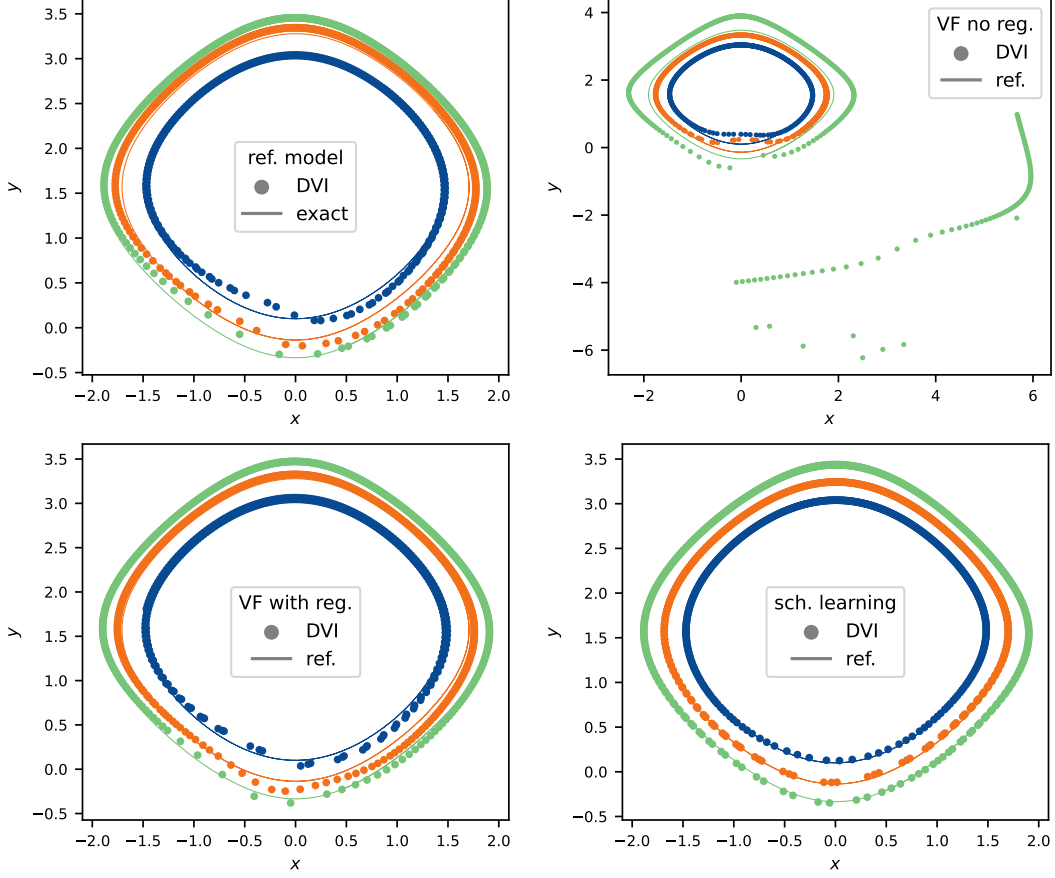
Figure 12: (Massless charged particle). Simulations of the reference model (5.3) (top left) VF-learning model trained without regularization (top right), VF-learning model with regularization (bottom left), scheme-learning model (bottom right), with the DVI scheme on three different trajectories with time-step $h = 0.5$ for 500 time-steps.

Each network ϑ_Θ and H_Θ, independently of the learning strategy, consists of two layers of size 50, and the inputs are normalized. The models undergo three trainings, first during 20 epochs with a learning rate of 10^{-2}, then 500 epochs with lr 10^{-3} and finally 500 epochs with lr 10^{-4}. The regularization parameter is set to $\varepsilon = 10^{-4}$ for VF learning and $\varepsilon = 10^{-8}$ for scheme learning.

Results   In the top left part of Figure 12, we can observe that the reference model (5.3), solved with the DVI scheme, exhibits a periodic behavior for the three trajectories. Since the time step is chosen very large, we observe some phase error. On the top right part, we see that the VF-learning model on one trajectory (the most external one) generates a completely wrong solution and the periodicity is lost. As with the Lotka-Volterra problem, it is very likely that a modified potential have been learned.

On Figure 12 (bottom left), we can observe that the regularization allows us, as before, to capture the correct trajectories, even though the learning error (in phase-space, Figure 13) is very similar to the non-regularized model. Although not identical to the reference solutions, the errors appear to be of the same order as when DVI is applied to the real system with a large time step. The dominant error therefore seems to be that of the
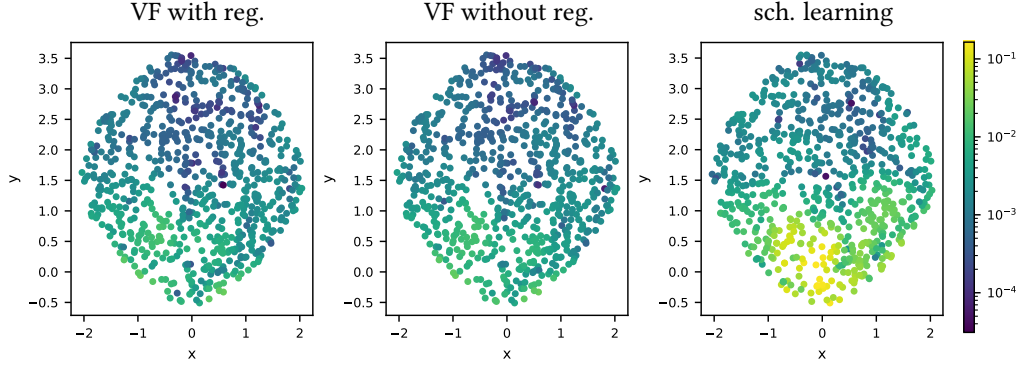
Figure 13: (Massless charged particle) Loss values on the validation set for the VF-learning models with (left) or without (center) regularization, and for the scheme-learning model (right).

schematic rather than that of learning.

On the bottom right of Figure 12, we can see that the approach that uses the scheme to learn, thus capturing an efficient discrete flow, partially corrects the DVI error. Even though the learning error (Figure 13) is highest where the DVI seems to be most inaccurate (near the origin), the numerical solutions are closer to the reference solutions when using this model. This result is comparable to those obtained for the Lotka-Volterra system, which validates the previous conclusions.

5.3 Guiding center

This test case is an asymptotic model of plasma physics in tokamaks with a strong magnetic field. The position is expressed in poloidal-toroidal coordinates $X = (r, \theta, \phi)$, where r is the minor-radial position, θ the geometric poloidal angle, and ϕ the geometric toroidal angle. The momentum is reduced to a single coordinate u in the toroidal direction, parallel to the magnetic field. Owing to the particular geometry and reduced dynamics, the problem is non-canonical and its Lagrangian is commonly written

$$L(X, u, X_t) = \big(A(X) + ub(X)\big) \cdot X_t - H(X, u),$$

where $A = (0, A_\theta, A_\phi)$ is a magnetic potential from which the vector field $\mathbf{B} = \nabla \times A$ is derived, and $b = \mathbf{B}/\|\mathbf{B}\|$ is the magnetic field unit vector.

We assume that the magnetic field is only in the toroidal direction, i.e. $ub(X) \cdot X_t = u\big(R_0 + r \cos(\theta)\big)\phi_t$. Reordering the coordinates into $z = (\theta, \phi, r, u)$, the Lagrangian is then properly degenerate, i.e. it admits the (x, y) decomposition.[6] Following [EFB+18], we write

$$L(\theta, \phi, r, u, \theta_t, \phi_t) = A_\theta(r, \theta)\theta_t + \big(A_\phi(r) + u(R_0 + r \cos \theta)\big)\phi_t - H(r, \theta, u).$$

with the symplectic (or magnetic) potential

$$A_\theta(r, \theta) = \frac{B_0 R_0^2}{\cos^2(\theta)} \left(\frac{r \cos(\theta)}{R_0} - \log \left(1 + \frac{r \cos(\theta)}{R_0} \right) \right), \qquad A_\phi(r) = -\frac{B_0 r^2}{2q_0}. \quad (5.4)$$

---

[6]If the problem is not properly degenerate, then it is possible to define an equivalent Lagrangian which is, as in [QGT09]. This is done by recognizing an identity $A_r(X)\dot{r} = \frac{\mathrm{d}}{\mathrm{d}t}f(X) - (\partial_\theta f)\dot{\theta} - (\partial_\phi f)\dot{\phi}$ for a well-chosen f, and removing $\frac{\mathrm{d}}{\mathrm{d}t}f(X)$ by invariance.
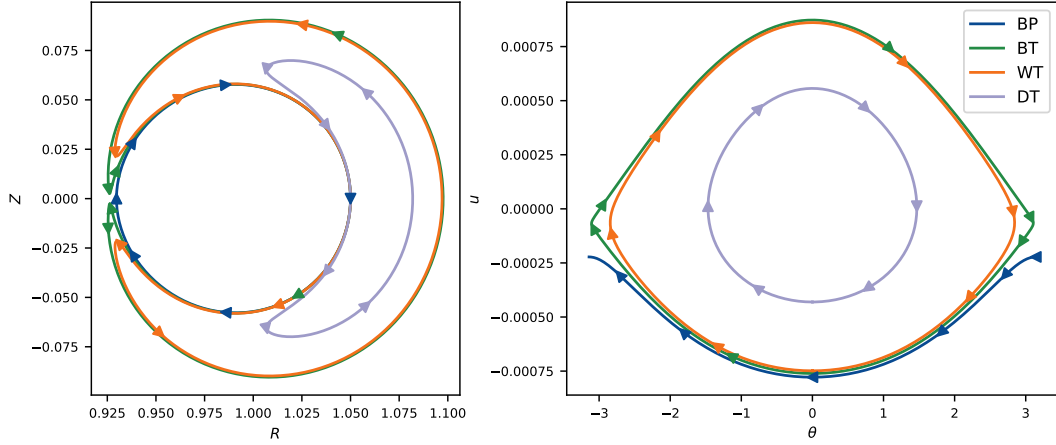
Figure 14: (Guiding center) The trajectories of interest in the poloidal (R, Z) plane (left) and in the (θ, u) plane (right). The initial conditions are $r_0 = 0.05$, $\theta_0 = 0$, $\phi_0 = 0$ and varying u_0, from smallest to largest in the legend (or, in absolute value, from largest to smallest).

The parameters are R_0 the radial position of the magnetic axis; B_0 the magnitude of the magnetic field at R_0; q_0 the (dimensionless) safety factor, regarded as constant. The Hamiltonian is a combination of kinetic and magnetic energies,

$$H(r, \theta, u) = \frac{1}{2}u^2 + \mu B(r, \theta), \qquad B(r, \theta) = \frac{B_0}{1 + \frac{r\cos(\theta)}{R_0}}\sqrt{1 + \left(\frac{r}{q_0 R_0}\right)^2}, \qquad (5.5)$$

where the additional parameter μ is the (constant) magnetic moment of the particle and $B = \|\mathbf{B}\|$ is the magnitude of the magnetic field. In the sequel we set $B_0 = 1$, $R_0 = 1$, $\mu = 2.25 \cdot 10^{-6}$ and $q_0 = 2$.

Reference behaviour We are interested in four trajectories, with the same initial position $r_0 = 0.05$, $\theta_0 = 0$ and $\phi_0 = 0$, and different initial velocities:

- **BP** – barely passing trajectory, $u_0 = -7.782 \cdot 10^{-4}$;

- **BT** – barely trapped trajectory, $u_0 = -7.610 \cdot 10^{-4}$;

- **WT** – well trapped trajectory, $u_0 = -7.487 \cdot 10^{-4}$;

- **DT** – deeply trapped trajectory, $u_0 = -4.306 \cdot 10^{-4}$ (also called "banana orbit").

As can be seen in Figure 14 with the BP trajectory, if the initial velocity is large enough in absolute value, then, similarly to a pendulum, the velocity does not change sign, i.e. the particle never "turns back" in the tokamak (hence the terminology of *passing* vs *trapped*). In the poloidal plane, also called the (R, Z) plane with $R = R_0 + r\cos(\theta)$ and $Z = r\sin(\theta)$, the trajectory changes from a crescent shape to a nearly circular orbit. Since the plots in (R, Z) and in (θ, u) are closely related, we will only plot the former when evaluating our neural networks.

To perform simulations, we choose 20 time-steps per period of the DT trajectory, i.e. $h = T_{\text{DT}}/20$ with $T_{\text{DT}} = 37974.6$. Over long-times, using the RK4 scheme fails to capture
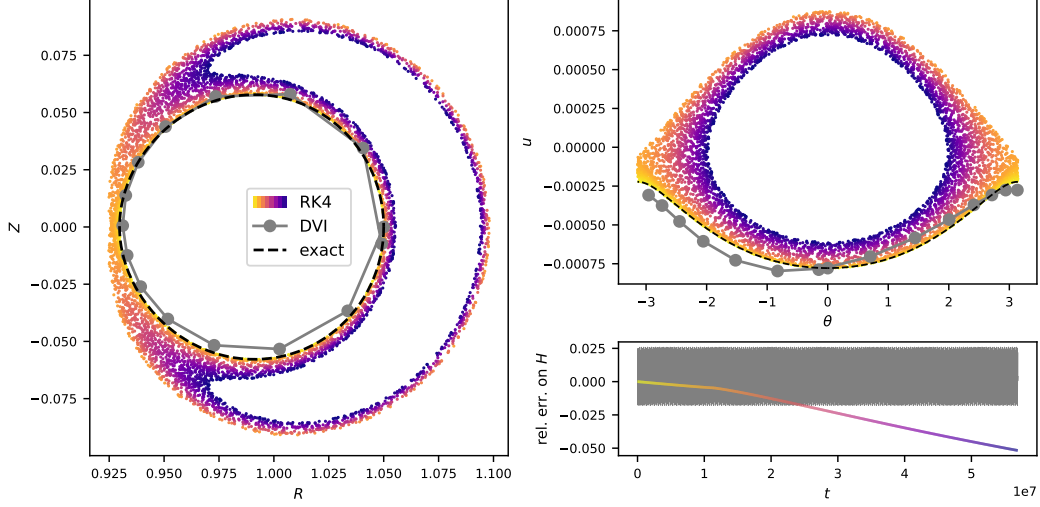
23

Figure 15: (Guiding center) Behaviour of the numerical solutions applying the RK4 or the DVI schemes to the reference model, with time-step $h = T_{\mathrm{DT}}/20$ up to time $t = 1500T_{\mathrm{DT}}$. In phase space, the RK4 solution is plotted every 5 time-step, and only the first period is plotted for the DVI scheme.

the passing behaviour, Figure 15. Using the DVI scheme (Figure 16), the BT trajectory is be wrongly categorized as "passing." This test case shows that even without considering very long time spans, it is crucial to use a structure-preserving scheme to capture the full set of physical solutions. Learning while preserving the structure may not be sufficient in practice.

Pre-processing of the input   We hard-code axisymmetry, the independence of the output w.r.t. ϕ, by discarding this coordinate from the inputs. Additionally, to take into account the angular nature of θ, we apply a transformation from [DN21]. Each coordinate is repeated $k = 6$ times, $v \in \mathbb{R} \mapsto v\mathbb{1} := (v, ..., v)^{\mathsf{T}} \in \mathbb{R}^k$, the angles are shifted and only their cosine is kept. This yields a pre-processing layer of the form

$$
\begin{pmatrix} \theta \\ \phi \\ r \\ u \end{pmatrix} \mapsto \begin{bmatrix} \cos_{\odot}(\theta\mathbb{1} + \varphi_{\Theta}) \\ r\mathbb{1} \\ u\mathbb{1} \end{bmatrix},
$$

where $\varphi_{\Theta} \in \mathbb{R}^k$ are parameters to be learnt (initialised randomly, uniformly in $[0, 2\pi]$) and $\cos_{\odot}$ is the cosine applied componentwise. The inputs are then normalized such that all coordinates are in $[0, 1]$, as for other problems.

The output of the Hamiltonian is also rescaled with an affine function which maps $[0, 1]$ to $[\min \|\dot{z}\|, \max \|\dot{z}\|]$, the extrema being determined on the training set. For scheme learning (§4), this is replaced by the finite-difference $\frac{1}{h}\|z_{n+1} - z_n\|$.

Training details   We uniformly generate 600 initial conditions in the tokamak with $r_0^2 \in [0.03^2, 0.055^2]$, $\theta_0 \in \left[-\frac{\pi}{10}, \frac{\pi}{10}\right]$, $\phi_0 \in [0, 2\pi]$ and $u_0 \in [-9 \cdot 10^{-4}, -3 \cdot 10^{-4}]$. Each trajectory then evolves for 60 time-steps with $h = T_{\mathrm{DT}}/20$, and is assigned to either the
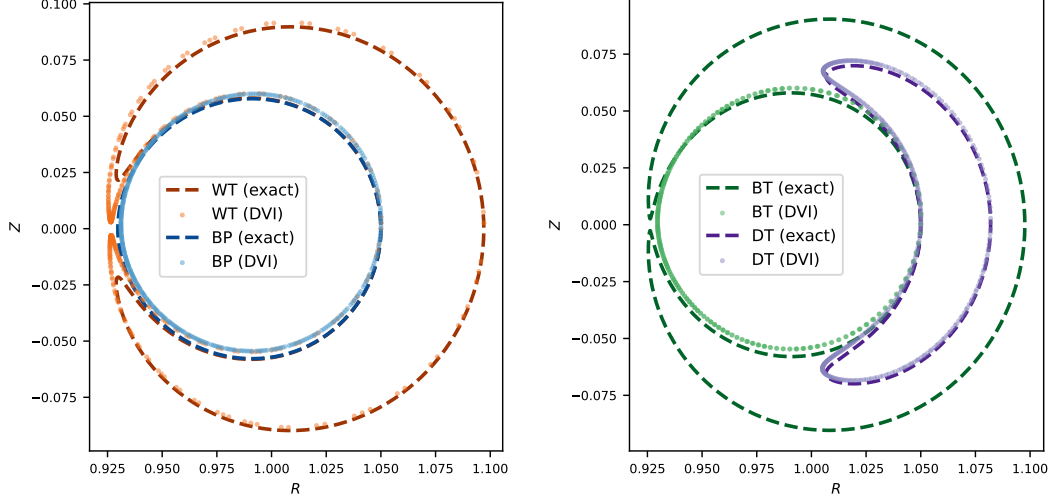
Figure 16: (Guiding center) Comparison on the reference model between the orbits from of the exact solution and of the numerical solution obtained with the DVI using a time-step $h = T_{\mathrm{DT}}/20$.

training, test or validation set with ratios of 80%, 15% and 5% respectively. This construction provides some "padding" around the specific trajectories we are interested in.

Each network consists of 3 layers of dimension 48, meaning there are 5.8k parameters in ϑ_Θ and 5.8k parameters in H_Θ, for a total of about 12k parameters per model.[7] The models are trained over 3 training: 20 epochs with learning rate 10^{-2}, 500 with lr 10^{-3} and 500 with lr 10^{-4}. The regularization weight is $\varepsilon = 1$ for VF learning (though we will observe no difference when setting $\varepsilon = 0$), and $\varepsilon = 10^{-8}$ for scheme learning.

Additionally, the first two trainings use an approximation of (4.1) in order to reduce the number of function evaluations. This uses a first-order asymptotic expansion of $\vartheta(x_2, y_2)$ around (x_1, y_1) and is written

$$
\begin{aligned}
\widetilde{J}_h^{-1} \widetilde{S}_h(z_0, z_1, z_2) = & \\
& \begin{bmatrix} y_2 - y_1 + (D_y \tilde{\vartheta}_\Theta)^{-1} \left((D_x \tilde{\vartheta}_\Theta)^{\mathsf{T}} (x_1 - x_0) - D_x \tilde{\vartheta}_\Theta (x_2 - x_1) - h \nabla_x \tilde{H}_\Theta \right) \\ x_1 - x_0 + h (D_y \tilde{\vartheta}_\Theta)^{-\mathsf{T}} \nabla_y \tilde{H}_\Theta \end{bmatrix},
\end{aligned} \quad (5.6)
$$

where every $\tilde{\vartheta}_\Theta$ and $\tilde{H}_\Theta$ are evaluated at (x_1, y_1). Because this requires only 1 function evaluation instead of 2, and uses a smaller matrix inversion (only on $D_y \vartheta_\Theta$), this greatly speeds up the otherwise slow training.

Behaviour of the models In Figure 17, it appears that the numerical solutions of the barely passing (BP) trajectory behave qualitatively the same as for the reference model. Using the RK4 scheme, the solution loses energy over time, ending up in a trapped trajectory. This is confirmed in Figure 18, where this change of regime is seen in the slope of the energy decrease. As for the other test cases, the scheme-learning model is less accurate than the VF-learning models with this numerical scheme: the energy oscillates more. However, it oscillates less when using the DVI, a difference of 10^2 in amplitude.

---

[7]Because there are 2 outputs for ϑ_Θ and only 1 for H_Θ, there are 48 more parameters in ϑ_Θ (there is no final bias for either network).
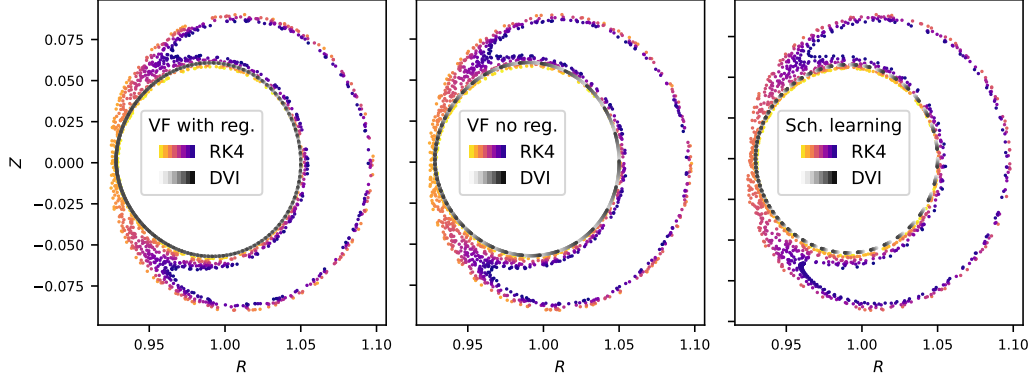
Figure 17: (Guiding center) Numerical solutions for the BP trajectory of the different neural models, obtained.
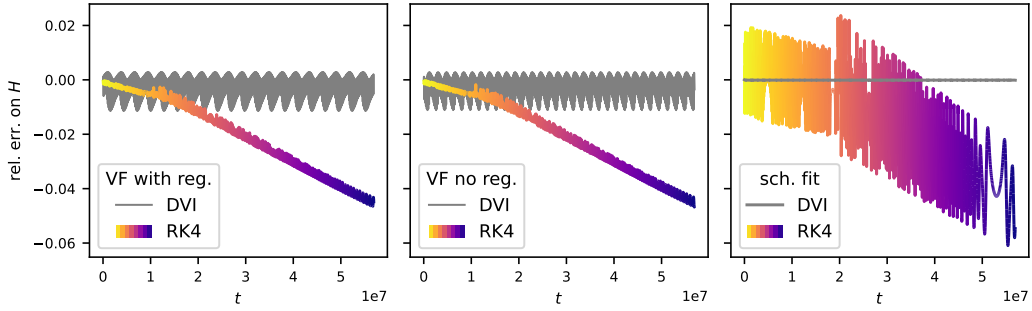

Figure 18: (Guiding center) Evolution of the relative error on H for the BP trajectory of the different neural models.

The center plots show that for this particular test case, the issue of gauge invariance does not appear. The solution of the DVI scheme behaves well even for the model trained without the regularization term. In fact, when monitoring the regularization term during training, both VF-learning models have similar values. This may be due to the hard-coding of the axisymmetry, to the relatively small size of the networks or to the fairly short training, which causes the maps to be smoother than with simpler problems. We now focus on the model trained *with* the regularization term to verify that even when including this term, the learnt dynamics are accurate.

The observations made on the BP trajectory are confirmed in Figures 19 and 20, where we compare the exact solutions (computed with solve_ivp as in the other experiments), the DVI-computed numerical solutions, and the reference trajectory. The VF-learning model produced very accurate exact solutions (in dashed lines), while the scheme-learning model is very accurate in its numerical solution using the DVI (scatter dots). Comparing the VF-learning model with the reference model (Figure 16), the former actually behaves better than the latter when using the DVI—especially with the BT trajectory, which remains trapped with the neural model (BP also remains passing). The impact of correcting terms in scheme learning is clear with the BT trajectory, which has the wrong behavior when considering the exact solution, but is more accurate than VF learning when applying the DVI. This is all confirmed in Figures 21 and 22. As in previous cases, this confirms that the network learns a very precise discrete flow, which effectively cancels or minimizes the
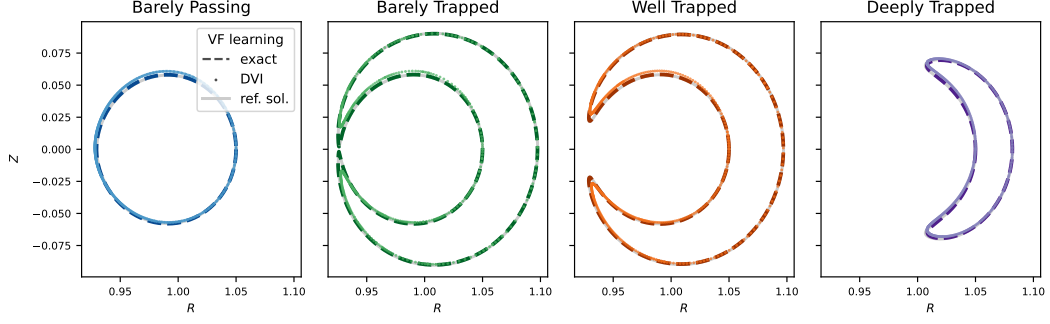
26

Figure 19: (Guiding center - VF-learning) Comparisons of the exact solutions of the VF-learning neural model (dashed line), the DVI-computed solutions of the VF-learning neural model with $h = T_{\mathrm{DT}}/20$ (scatter dots) and the reference trajectories (gray line).
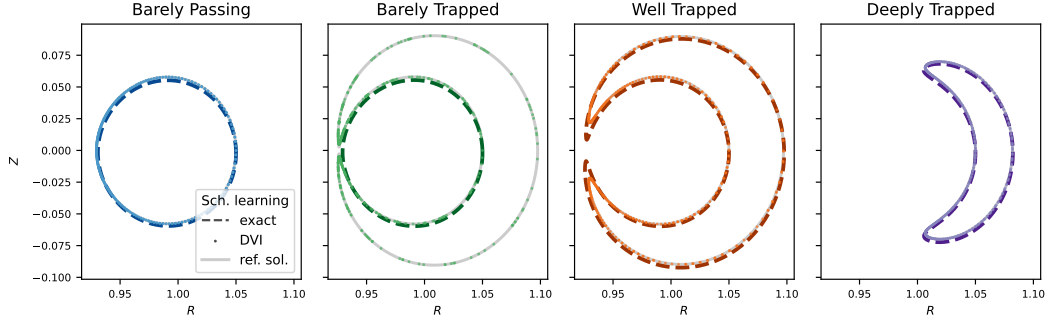

Figure 20: (Guiding center - scheme learning) Comparisons of the exact solutions of the scheme-learning neural model (dashed line), the DVI-computed solutions of the scheme-learning neural model with $h = T_{\mathrm{DT}}/20$ (scatter dots) and the reference trajectories (gray line).

scheme's error terms. The scheme-learning model is a better choice for long-time simulations, as long as the scheme and time-steps match the training.

In Figure 23, we illustrate the importance of using a multiscale norm (in our case, a Gram-informed norm). Without it, some components are completely ignored by the loss, since $|\dot{u}| \sim 10^{-4} \ll |\dot{r}| \sim 10^{-2} \ll |\dot{\theta}| \sim 1$. The trained model is then qualitatively inaccurate, and does not capture the diversity of trajectories displayed by the reference model.

6 Conclusion

In this paper, we build learning-based models to capture families of temporal trajectories, given by non-canonical Hamiltonian systems. We showed that, unlike the canonical case, one cannot directly apply a DVI scheme to a model trained with an imposed non-canonical structure. Indeed, non-canonical models are defined only up to a potential, whereas the DVI scheme is not invariant under such a potential shift. As a result, one may end up with a situation where the learning procedure captures the system correctly up to a potential, but injecting this model into the scheme leads to severe stability issues.

This highlights that, for more complex cases, one must not only impose the structure
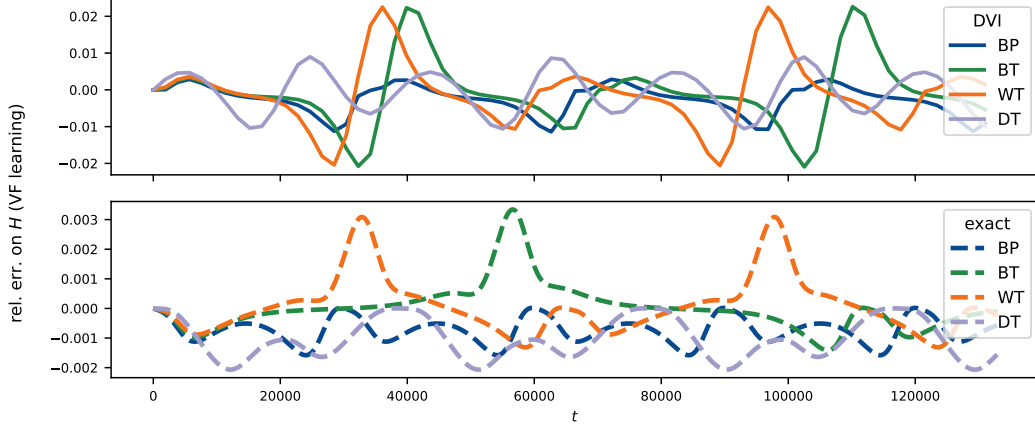
Figure 21: (Guiding center - VF learning) Relative variations of the Hamiltonian (5.5) for each trajectory of the VF-learning model using the DVI, with $h = T_{\mathrm{DT}}/20$, (top) and a refined simulation (bottom).

Figure 22: (Guiding center - scheme learning) Relative variations of the Hamiltonian (5.5) for each trajectory of the scheme-learning model using the DVI, with $h = T_{\mathrm{DT}}/20$, (top) and a refined simulation (bottom).
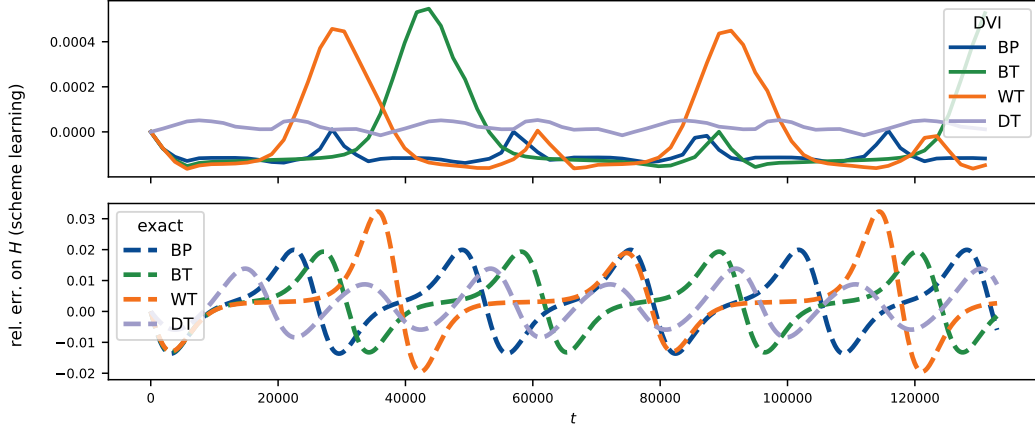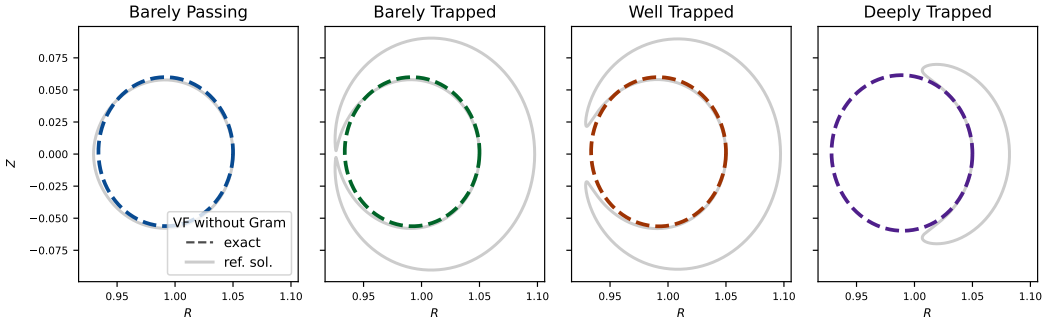

Figure 23: (Guiding center - VF-learning without Gram-informed norm) Exact solutions of the VF-learning neural model (dashed line) trained without the multiscale norm (3.6), compared to the reference trajectories (gray line).

during training but also ensure compatibility between the learned model, its invariance properties, and the numerical scheme.

To address this issue, we proposed two approaches.

1. learn the continuous-time model but add a penalty on the residual terms of the scheme, which mitigates the problem. However, the scheme remains of low order and becomes inaccurate for large time steps.

2. learn using the scheme itself, which amounts to learning a discrete flow. This resolves the issue and improves accuracy (since it partly corrects the scheme's consistency error), but the method is limited to the training time step.

Our next objective is to tackle even more complex problems, such as Poisson systems. One could also enrich the learning procedure with loss functions that account for the integration of the entire trajectory, either using discrete approaches (unrolling) or continuous ones (Neural ODEs).

References

[AKW+19] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E.W. Pellegrini, R.S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing Inverse Problems with Invertible Neural Networks, February 2019.

[BCLM23] M. Bouchereau, P. Chartier, M. Lemou, and F. Méhats. Machine Learning Methods for Autonomous Ordinary Differential Equations, April 2023.

[BFE22] J. W. Burby, J. M. Finn, and C. L. Ellison. Improved accuracy in degenerate variational integrators for guiding centre and magnetic field line flow. *J. Plasma Phys.*, 88(2):835880201, April 2022.

[BHT+24] C. Bonneville, X. He, A. Tran, J.S. Park, William Fries, Daniel A Messenger, Siu Wun Cheung, Yeonjong Shin, David M Bortz, D. Ghosh, J.-S. Chen, J. Belof, and Y. Choi. A comprehensive review of latent space dynamics identification algorithms for intrusive and non-intrusive reduced-order-modeling. *arXiv preprint arXiv:2403.10748*, 2024.

[BPK16] S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15):3932–3937, 2016.

[BR24] M. Blondel and V. Roulet. The elements of differentiable programming. *arXiv preprint arXiv:2403.14606*, 2024.

[BSR+21] M. Biloš, J. Sommer, S.S. Rangapuram, T. Januschowski, and S. Günnemann. Neural flows: Efficient alternative to neural odes. *Adv. Neural Inf. Process. Syst.*, 34:21325–21337, 2021.

[BTM21] J.W. Burby, Q. Tang, and R. Maulik. Fast neural Poincaré maps for toroidal magnetic fields. *Plasma Phys. Controlled Fusion*, 63(2):024001, February 2021.

[CFN+24] R. Côte, E. Franck, L. Navoret, G. Steimer, and V. Vigon. Hamiltonian reduction using a convolutional auto-encoder coupled to an hamiltonian neural network. *Commun. Comput. Phys.*, 37:315–352, 2024.

[CFN+25] R. Côte, E. Franck, L. Navoret, G. Steimer, and V. Vigon. Reduced particle in cell method for the vlasov-poisson system using auto-encoder and hamiltonian neural. *arXiv preprint arXiv:2506.15203*, 2025.

[CGH+20] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks, July 2020.

[CMY21]   Y. Chen, T. Matsubara, and T. Yaguchi. Neural Symplectic Form: Learning Hamiltonian Equations on General Coordinate Systems. In *Adv. Neural Inf. Process. Syst.*, volume 34, pages 16659–16670. Curran Associates, Inc., 2021.

[CRBD18]   R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D.K. Duvenaud. Neural Ordinary Differential Equations. In *Adv. Neural Inf. Process. Syst.*, volume 31. Curran Associates, Inc., 2018.

[DBT23]   V. Duruisseaux, J.W. Burby, and Q. Tang. Approximation of nearly-periodic symplectic maps via structure-preserving neural networks. *Sci. Rep.*, 13(1):8351, May 2023.

[DDT19]   E. Dupont, A. Doucet, and Y.W. Teh. Augmented neural odes. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[DM21]   M. David and F. Méhats. Symplectic learning for Hamiltonian Neural Networks, June 2021.

[DN21]   S. Dong and N. Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *J. Comput. Phys.*, 435:110242, June 2021.

[EFB⁺18]   C. L. Ellison, J. M. Finn, J. W. Burby, M. Kraus, H. Qin, and W. M. Tang. Degenerate variational integrators for magnetic field line flow and guiding center trajectories. *Phys. Plasma*, 25(5):052502, May 2018.

[EL24]   S. Eidnes and K.O. Lye. Pseudo-Hamiltonian neural networks for learning partial differential equations. *J. Comput. Phys.*, 500:112738, March 2024.

[FFBM25]   N. Farenga, S. Fresca, S. Brivio, and A. Manzoni. On latent dynamics learning in nonlinear reduced order modeling. *Neural Networks*, page 107146, 2025.

[GDY19]   S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In *Adv. Neural Inf. Process. Syst.*, volume 32. Curran Associates, Inc., 2019.

[GSC⁺23]   R. Gnanasambandam, B. Shen, J. Chung, X. Yue, and Z. Kong. Self-scalable tanh (stan): Multi-scale solutions for physics-informed neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[HLW06]   E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin Heidelberg, 2 edition, 2006.

[JGH18]   A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Adv. Neural Inf. Process. Syst.*, 31, 2018.

[JZKK20]   P. Jin, Z. Zhang, I.G. Kevrekidis, and G.E. Karniadakis. Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks, December 2020.

[JZZ⁺20]   P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G.E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, December 2020.

[KKB18]   E. Kaiser, J.N. Kutz, and S.L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proc. R. Soc. A*, 474(2219):20180335, 2018.

[KM19]   J.Z. Kolter and G. Manek. Learning stable deep dynamics models. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[LLCP22]   Y. Lu, S. Lin, G. Chen, and J. Pan. ModLaNets: Learning Generalisable Dynamics via Modularity and Physical Inductive Bias. In *Proceedings of the 39th International Conference on Machine Learning*, pages 14384–14397. PMLR, June 2022.

[LSR⁺22]   Y. Lishkova, P. Scherer, S. Ridderbusch, M. Jamnik, P. Liò, S. Ober-Blöbaum, and C. Offen. Discrete Lagrangian Neural Networks with Automatic Symmetry Discovery, November 2022.

[LTS22] K. Lee, N. Trask, and P. Stinis. Structure-preserving sparse identification of nonlinear dynamics for data-driven modeling. In *Mathematical and Scientific Machine Learning*, pages 65–80. PMLR, 2022.

[MPP$^+$20] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama. Dissecting neural odes. *Adv. Neural Inf. Process. Syst.*, 33:3952–3963, 2020.

[MYH22] F.B. Mathiesen, B. Yang, and J. Hu. Hyperverlet: A Symplectic Hypersolver for Hamiltonian Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4):4575–4582, June 2022.

[OO23] S. Ober-Blöbaum and C. Offen. Variational learning of Euler–Lagrange dynamics from data. *J. Comput. Appl. Math.*, 421:114780, March 2023.

[OO24] C. Offen and S. Ober-Blöbaum. Learning of discrete models of variational PDEs from data. *Chaos*, 34(1):013104, January 2024.

[PMY$^+$20] M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park. Hypersolvers: Toward Fast Continuous-Depth Models. In *Adv. Neural Inf. Process. Syst.*, volume 33, pages 21105–21117. Curran Associates, Inc., 2020.

[QGT09] H. Qin, X. Guan, and W.M. Tang. Variational symplectic algorithm for guiding center dynamics and its application in tokamak geometry. *Phys. Plasma*, 16(4):042510, April 2009.

[RPS$^+$23] F. Regazzoni, S. Pagani, M. Salvador, L. Dede, and A. Quarteroni. Latent dynamics networks (ld-nets): learning the intrinsic dynamics of spatio-temporal processes. *arXiv e-prints*, pages arXiv–2305, 2023.

[SCL20] X. Shen, X. Cheng, and K. Liang. Deep Euler method: Solving ODEs by approximating the local truncation error of the Euler method, March 2020.

[Tap24] B.K. Tapley. Symplectic neural networks based on dynamical systems. *arXiv preprint arXiv:2408.09821*, 2024.

[TID23] W. Tenachi, R. Ibata, and F.I. Diakogiannis. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 2023.

[ZL24] A. Zhu and Q. Li. DynGMA: A robust approach for learning stochastic differential equations from data. *J. Comput. Phys.*, 513:113200, 2024.

A Using degenerate variational integrators

A.1 The Jacobian of variational integrators

Usual variational integrators are based on discrete Lagrangians $(z_0, z_1) \mapsto L_h(z_0, z_1)$ and lead to the well-known discrete Euler-Lagrange equation

$$D_2 L_h(z_{n-1}, z_n) + D_1 L_h(z_n, z_{n+1}) = 0. \tag{A.1}$$

Many papers actually focus on learning a (continuous) Lagrangian from which they obtain discrete Euler-Lagrange equations which constitute the loss. To avoid the trivial solution $L \to 0$, a non-triviality condition is required. A natural way to do this is to ensure that the Jacobian w.r.t. z_{n+1} in (A.1)

$$J_{n+1}^{[\text{DEL}]} = D_2 D_1 L_h(z_n, z_{n+1}) \tag{A.2}$$

is nicely invertible, such that Newton-Raphson iterations are well-defined. Let us mention two approaches which try to minimize the average value on the training dataset of the following quantities,

$$[\text{LSR}^+22]: \quad \frac{1}{1 - e^{-0.01 \det(J)}}, \qquad\qquad [\text{OO24}]: \quad \text{ReLU}\left(1 - \frac{10}{\|J^{-1}\|^2}\right).$$

Both kill two birds with one stone by penalising trivial Lagrangians and ensuring the invertibility of the Jacobian. With the midpoint scheme $L_h(z_0, z_1) = L\left(\frac{z_0 + z_1}{2}, \frac{z_1 - z_0}{h}\right)$, this induces non-degenerate Lagrangian, since $D_2 D_1 L_h = \frac{1}{4}\partial_{zz}^2 L - \frac{1}{h^2}\partial_{\dot z \dot z}^2 L$.

In the case of degenerate Lagrangians, the principal term $\partial_{\dot z \dot z}^2 L$ vanishes, therefore this reasoning must be adapted. This is also apparent for the first-order DVI (2.10), as the DEL equation changes due to the difference between the x and y components. It becomes

$$\begin{cases} D_2 L_h(x_{n-1}, x_n, y_n) + D_1 L_h(x_n, x_{n+1}, y_{n+1}) = 0, \\ \qquad\qquad\qquad\qquad D_3 L_h(x_n, x_{n+1}, y_{n+1}) = 0. \end{cases}$$

The Jacobian of the system w.r.t. (x_{n+1}, y_{n+1}) is no longer (A.2) but a block matrix

$$J_{n+1}^{[\text{DEL}]} = \begin{bmatrix} D_2 D_1 & D_3 D_1 \\ D_2 D_3 & D_3^2 \end{bmatrix} L_h(x_n, x_{n+1}, y_{n+1}).$$

Explicitly, the blocks are

$$D_2 D_1 L_h = \left(-\frac{1}{h}\vartheta_{i,j}\right)_{i,j} \qquad D_2 D_3 L_h = \left(\vartheta_{k,\alpha,j} \cdot \frac{x_{n+1}^k - x_n^k}{h} + \frac{1}{h}\vartheta_{j,\alpha} - H_{,\alpha,j}\right)_{\alpha,j}$$

$$D_3 D_1 L_h = \left(-\frac{1}{h}\vartheta_{i,\beta}\right)_{i,\beta} \qquad D_3^2 L_h = \left(\vartheta_{k,\alpha,\beta} \cdot \frac{x_{n+1}^k - x_n^k}{h} - H_{,\alpha,\beta}\right)_{\alpha,\beta}$$

A.2 Rigorous error estimate of the scheme

Given initial conditions x_0, y_0 denote $x_h \approx x(h), y_h \approx y(h)$ the result of after one step of the scheme (2.10). We wish to find an expression of the leading term in the error $\begin{bmatrix} x(h) - x_h \\ y(h) - y_h \end{bmatrix}$. For this, we perform expansions, exploiting the assumed result $z_h = z_0 + h\dot z(0) + \mathcal{O}(h^2)$,

$$\vartheta_{j,\alpha}(z_h) = \vartheta_{j,\alpha} + h\vartheta_{j,\alpha,k}\dot z^k(0) + \mathcal{O}(h^2), \qquad H_{,\alpha}(z_h) = H_{,\alpha} + hH_{,\alpha,k}\dot z^k(0) + \mathcal{O}(h^2),$$

where the maps ϑ_j, H and their differentials are evaluated at z_0. Therefore, the part of the scheme updating x can be written

$$\left(\vartheta_{j,\alpha} + h\vartheta_{j,\alpha,k}\dot z^k + \mathcal{O}(h^2)\right)(x_h^j - x_0^j) = hH_{,\alpha} + h^2 H_{,\alpha,k}\dot z^k + \mathcal{O}(h^3),$$

where the maps and $\dot z$ are evaluated time $t = 0$. In the sequel, the argument of ϑ, H and their differentials is always x_0, and the time-differentials are always evaluated at time $t = 0$, so that we may lighten the presentation. Using $x_h^j - x_j^0 = h\dot x^j + \mathcal{O}(h^2) = \mathcal{O}(h)$, we find

$$\vartheta_{j,\alpha}x_h^j = \vartheta_{j,\alpha}x_0^j + hH_{,\alpha} + h^2\left(H_{,\alpha,k} - \vartheta_{j,\alpha,k}\dot x^j\right)\dot z^k + \mathcal{O}(h^3).$$

On the other hand, $x(h) = x_0 + h\dot{x} + \frac{1}{2}h^2\ddot{x} + \mathcal{O}(h^3)$, where the second-order derivative may be obtained from the differential equation (2.5), as

$$0 = \vartheta_{j,\alpha}\ddot{x}^j + \vartheta_{j,\alpha,k}\dot{x}^j\dot{z}^k - H_{,\alpha,k}\dot{z}^k.$$

Therefore, with $\vartheta_{j,\alpha}\dot{x}^j = H_{,\alpha}$, the dominant error term is

$$\vartheta_{j,\alpha}\big(x^j(h) - x_h^j\big) = \frac{h^2}{2}\left(\vartheta_{j,\alpha,k}\dot{x}^j - H_{,\alpha,k}\right)\dot{z}^k + \mathcal{O}(h^3),$$

i.e. in matrix-vector form, recognizing the Lagrangian,

$$x(h) - x_h = -\frac{h^2}{2}\ddot{x} + \mathcal{O}(h^3) = \frac{h^2}{2}(D_y\vartheta)^{-\mathsf{T}}D_z\big[\nabla_y L\big]\dot{z} + \mathcal{O}(h^3), \qquad \text{(A.3)}$$

where the differentials w.r.t. y and z are not applied to $\dot{x}$.

For the estimate in y, we recognize $x_{-h} = x_0 - h\dot{x}_{|t=0}$, therefore the scheme is

$$\vartheta_i(x_h, y_h) = \vartheta_i + h\vartheta_{j,i}\dot{x}^j - hH_{,i}.$$

A straightforward Taylor expansion also yields

$$\vartheta_i(x_h, y_h) = \vartheta_i + \vartheta_{i,j}(x_h^j - x_0^j) + \vartheta_{i,\nu}(y_h^\nu - y_0^\nu) + \frac{h^2}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'} + \mathcal{O}(h^3).$$

Reordering the terms, we find

$$\vartheta_{i,\nu}y_h^\nu = \vartheta_{i,\nu}y_0^\nu + h\vartheta_{j,i}\dot{x}^j - h\vartheta_{i,j}\frac{x_h^j - x_0^j}{h} - hH_{,i} - \frac{h^2}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'} + \mathcal{O}(h^3)$$

Since $x_h = x_0 + h\dot{x} + h^2\ddot{x} + \mathcal{O}(h^3)$, we find

$$\vartheta_{i,\nu}y_h^\nu = \vartheta_{i,\nu}y_0^\nu + h\underbrace{\left((\vartheta_{j,i} - \vartheta_{i,j})\dot{x}^j - H_{,i}\right)}_{=\vartheta_{i,\nu}\dot{y}^\nu} - h^2\left(\vartheta_{i,j}\ddot{x}^j + \frac{1}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'}\right) + \mathcal{O}(h^3).$$

The exact solution satisfies $y(h) = y_0 + h\dot{y} + \frac{1}{2}h^2\ddot{y} + \mathcal{O}(h^3)$, therefore the error term is

$$\vartheta_{i,\nu}\left(y^\nu(h) - y_h^\nu\right) = \frac{h^2}{2}\vartheta_{i,\nu}\ddot{y}^\nu + h^2\left(\vartheta_{i,j}\ddot{x}^j + \frac{1}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'}\right) + \mathcal{O}(h^3). \qquad \text{(A.4)}$$

The second-order derivative on y is obtained from

$$0 = \vartheta_{i,\nu}\ddot{y}^\nu + \vartheta_{i,\nu,k}\dot{y}^\nu\dot{z}^k + \left(\vartheta_{i,j} - \vartheta_{j,i}\right)\ddot{x}^j + \left(\vartheta_{i,j,k} - \vartheta_{j,i,k}\right)\dot{x}^j\dot{z}^k + H_{,i,k}\dot{z}^k,$$

which, by combining $\vartheta_{i,j,k'}\dot{x}^j + \vartheta_{i,\nu,k'}\dot{y}^\nu = \vartheta_{i,k,k'}\dot{z}^k$, is reordered

$$\vartheta_{i,\nu}\ddot{y}^\nu = (\vartheta_{j,i} - \vartheta_{i,j})\ddot{x}^j - \vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'} + (\vartheta_{j,i,k}\dot{x}^j - H_{,i,k})\dot{z}^k.$$

This may finally be injected in the error term to find

$$\vartheta_{i,\nu}\left(y^\nu(h) - y_h^\nu\right) = \frac{h^2}{2}\left((\vartheta_{j,i} - \vartheta_{i,j})\ddot{x}^j - \vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'} + (\vartheta_{j,i,k}\dot{x}^j - H_{,i,k})\dot{z}^k\right)$$
$$+ h^2\left(\vartheta_{i,j}\ddot{x}^j + \frac{1}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'}\right) + \mathcal{O}(h^3),$$

After simplification, this is written in the vector form

$$y(h) - y_h = \frac{h^2}{2}(D_y\vartheta)^{-1}\left((D_x\vartheta + D_x\vartheta^\mathsf{T})\ddot{x} + D_z[\nabla_x L]\dot{z}\right) + \mathcal{O}(h^3) \qquad \text{(A.5)}$$

where, again, the differentials w.r.t. x and z are only applied to the maps of the Lagrangian, not to $\dot{x}$. The second-order derivative $\ddot{x}$ may also be replaced using the error term $x(h) - x_h$, which yields

$$\begin{bmatrix} D_x\vartheta + D_x\vartheta^\mathsf{T} & D_y\vartheta \\ D_y\vartheta^\mathsf{T} & 0 \end{bmatrix}\begin{bmatrix} x(h) - x_h \\ y(h) - y_h \end{bmatrix} = \frac{h^2}{2}D_z[\nabla_z L]\dot{z} + \mathcal{O}(h^3). \qquad \text{(A.6)}$$

Remark A.1. Noticing the identity $\vartheta_{i,j}\ddot{x}^j + \frac{1}{2}\vartheta_{i,k,k'}\dot{z}^k\dot{z}^{k'} = \frac{1}{2}\left(\ddot{\vartheta} + D_x\vartheta\ddot{x} - D_y\vartheta\ddot{y}\right)$, the error term in (A.4) is simply

$$D_y\vartheta\left(y(h) - y_h\right) = \frac{h^2}{2}\left(\ddot{\vartheta} + D_x\vartheta\,\ddot{x}\right) + \mathcal{O}(h^3),$$

i.e., the regularity of ϑ impacts the accuracy of the scheme. Similarly, the symmetric part of $D_x\vartheta$ appears in this error term, even though has no influence on the original dynamics. All this explains why the gauge choice matters in this context.

B Brief study of the guiding center

B.1 Deriving the dynamics

It is standard to write these equations using the *effective* magnetic potential $A_\phi^\dagger := \vartheta_\phi = A_\phi + u$. Assuming only $A_{\theta,\phi} = A_{\theta,u} = 0$, the Euler-Lagrange equations are

$$A_{\theta,r}\dot{r} + A_{\theta,\theta}\dot{\theta} = A_{\theta,\theta}\dot{\theta} + A_{\phi,\theta}^\dagger\dot{\phi} - H_{,\theta}$$
$$A_{\phi,r}^\dagger\dot{r} + A_{\phi,\theta}^\dagger\dot{\theta} + A_{\phi,\phi}^\dagger\dot{\phi} + A_{\phi,u}^\dagger\dot{u} = A_{\phi,\phi}^\dagger\dot{\phi} - H_{,\phi}$$
$$0 = A_{\theta,r}\dot{\theta} + A_{\phi,r}^\dagger\dot{\phi} - H_{,r}$$
$$0 = A_{\phi,u}^\dagger\dot{\phi} - H_{,u}.$$

Considering the lines in order $(4, 3, 1, 2)$, this can be transformed into a triangular system,

$$\dot{\phi} = \frac{H_{,u}}{A_{\phi,u}^\dagger}, \quad \dot{\theta} = \frac{H_{,r} - A_{\phi,r}^\dagger\dot{\phi}}{A_{\theta,r}}, \quad \dot{r} = \frac{A_{\phi,\theta}^\dagger\dot{\phi} - H_{,\theta}}{A_{\theta,r}}, \quad \dot{u} = -\frac{A_{\phi,\theta}^\dagger\dot{\theta} + A_{\phi,r}^\dagger\dot{r} + H_{,\phi}}{A_{\phi,u}^\dagger}.$$

These dynamics are well defined only if $A_{\theta,r}$ and $A_{\phi,u}^\dagger$ are non-zero, which corresponds to the invertibility of $D_y\vartheta = \begin{pmatrix} A_{\theta,r} & 0 \\ A_{\phi,r} & A_{\phi,u}^\dagger \end{pmatrix}$. Reordering into (θ, ϕ, r, u), injecting the expression of the Hamiltonian and of $A_\phi^\dagger = A_\phi + u$, along with the assumption $A_{\phi,\theta} = A_{\phi,\phi} = 0$, finally yields

$$\dot{\theta} = \frac{\mu B_{,r} - A_{\phi,r}u}{A_{\theta,r}}, \qquad \dot{\phi} = u, \qquad \dot{r} = -\frac{\mu B_{,\theta}}{A_{\theta,r}}, \qquad \dot{u} = \frac{\mu B_{,\theta}A_{\phi,r}}{A_{\theta,r}},$$

Denoting $\rho = r\cos(\theta)/R_0$, $Z = r\sin(\theta)/R_0$, $\widetilde{r} = r/(q_0 R_0)$, these quantities are

$$A_{\theta,r} = \frac{B_0 r}{1+\rho}, \qquad A_{\phi,r} = -\frac{B_0 r}{q_0}, \qquad B_{,r} = \frac{B_0\sqrt{1+\widetilde{r}^2}}{r(1+\rho)}\left(\frac{\widetilde{r}^2}{1+\widetilde{r}^2} - \frac{\rho}{1+\rho}\right),$$

$$A_{\theta,\theta} = \frac{B_0 r^2 Z}{\rho^3}\left(2\log(1+\rho) - 2\rho + \frac{\rho^2}{1+\rho}\right), \qquad B_{,\theta} = \frac{B_0 Z\sqrt{1+\widetilde{r}^2}}{(1+\rho)^2}.$$

$$(\text{B.1})$$

B.2   Computing the magnetic potential and its differentials

The expressions of A_θ and $A_{\theta,\theta}$ are unstable for small values of $\cos(\theta)$, even though the functions are well-defined in this limit. Therefore, we compute them using an integral form,

$$A_\theta(\theta, r) = B_0 r^2 \int_0^1 \frac{t\,dt}{1+\rho t}, \qquad A_{\theta,\theta}(\theta, r) = B_0 r^2 Z \int_0^1 \frac{t^2\,dt}{(1+\rho t)^2}.$$

where, again, $\rho = r\cos(\theta)/R_0$ and $Z = r\sin(\theta)/R_0$. This is more costly to evaluate and less accurate for $\rho \approx -1$, but uniformly stable for all values of θ.

We compute A_θ and $A_{\theta,\theta}$ using this form and evaluating the integral with a Gauss-Legendre integration. A highly-accurate reference value was computed in `Julia` using `BigFloat` and 50 collocation points. Using double-precision with 20 collocation points, this method computes the integral in $A_{\theta,\theta}$ with an error of $1.7 \cdot 10^{-11}$ for $\rho = -0.9$, and of less than 10^{-15} for $-0.8 \le \rho \le 0.9$. The error is slightly smaller for the integral in A_θ. For comparison, using $\rho = 10^{-6}$ in the closed formula from (B.1) yields an error of $2.1 \cdot 10^{-4}$ with double-precision floats.

Some floating-point magic tricks might make the closed formula more accurate, using e.g. `log1p`,[8] but we did not find any, and they might be processor-dependent.

---

[8]For example, computing $\big(\texttt{log1p}(x)/x - 1\big)/\big((1+x) - 1\big)$ is more accurate than $\big(\texttt{log1p}(x)/x - 1\big)/x$ for small values of x.