

750 A Higher order approximations

751 In [16], the authors consider variational backward error analysis for learning Lagrangian systems
 752 in absence of external forces ($F = 0$). To each fixed variational discretization of a Lagrangian
 753 systems with Lagrangian L and step-size h (e.g. [7]), there exist an inverse modified Lagrangian
 754 L_{invmod} (expressed as a formal power series in powers of h) such that applying the chosen variational
 755 discretization to L_{invmod} gives a symplectic integrator whose modified Lagrangian is L . When
 756 applying variational discretization to learn Lagrangians it is L_{invmod} which is learned and accurate
 757 approximations of L can be deduced from L_{invmod} . This technique requires the computation of
 758 higher order derivatives of the learned Lagrangian (with respect to q and \dot{q}).

759 Our approach to obtain higher order approximations of the Lagrangian and Forces, is simply to
 760 include more data points for each approximation and to follow a strategy similar to symmetric
 761 multistep variational integrators, which are known to have good geometric properties, [60, 61].

762 For a method of order 4, we proceed as follows and consider

$$\begin{aligned} L_{\Delta}(q_{n-2}, q_{n-1}, q_{n+1}, q_{n+2}) &= hL_{\theta}(\bar{q}_n, \bar{v}_n), \\ F_{\Delta}^{+}(q_{n-2}, q_{n-1}, q_{n+1}, q_{n+2}) &= \frac{h}{2}F_{\theta}(\bar{q}_n, \bar{v}_n), \\ F_{\Delta}^{-}(q_{n-2}, q_{n-1}, q_{n+1}, q_{n+2}) &= \frac{h}{2}F_{\theta}(\bar{q}_n, \bar{v}_n), \end{aligned}$$

763 where

$$\begin{aligned} \bar{q}_n &= \frac{1}{12}(-q_{n-2} + 8q_{n-1} + 4q_{n+1} + q_{n+2}), \\ \bar{v}_n &= \frac{1}{12h}(q_{n-2} - 8q_{n-1} + 8q_{n+1} - q_{n+2}). \end{aligned}$$

764 For general higher order methods, we use optimal order finite differences for the approximation
 765 of the derivative with coefficients δ_j ($j = -k, \dots, k$), $\delta_j = -\delta_{-j}$, so that we have a derivative
 766 approximation of order $2k$,

$$\begin{aligned} \bar{v}_n &= \frac{1}{h} \sum_{j=-k}^k \delta_j q_{n-j}, \\ \bar{q}_n &= (1 - \delta_1)q_{n+1} - \sum_{j=-k, j \neq 0, 1}^k \delta_j q_{n-j}. \end{aligned}$$

and where $\delta_0 = 0$ and

$$\delta_j = \frac{(-1)^{j-1}}{j} \frac{k!^2}{(k-j)!(k+j)!}, \quad j = 1, \dots, k.$$

767 B Generalised Lagrangian Neural Networks

We briefly describe here the approach of Xiao et al. [3], see also [2]. This methodology requires both
 position and velocity data. Starting from [2], by expanding the term $\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}(q, \dot{q}))$ differentiating with
 respect to t , and assuming $\frac{\partial^2 L}{\partial \dot{q}^2}$ to be invertible, the authors derive the following explicit expression
 of the forced Euler-Lagrange equations

$$\ddot{q} = \left(\frac{\partial^2 L}{\partial \dot{q}^2} \right)^{-1} \left(\frac{\partial L}{\partial q} - \dot{q} \cdot \frac{\partial^2 L}{\partial q \partial \dot{q}} + F \right),$$

768 then use a numerical integration to approximate $(q, \dot{q}) \approx (\hat{q}, \hat{\dot{q}})$ at the times where the data is observed,
 769 and obtain a loss function as the mean of terms of the form $\|q - \hat{q}\|^2 + \|\dot{q} - \hat{\dot{q}}\|^2$. This approach
 770 relies on $\frac{\partial^2 L}{\partial \dot{q}^2}$ being at least locally invertible resulting in a so-called regular Lagrangian L , [11, 379].
 771 However, no specific technique for regularizing the Lagrangian is described in the paper. A notable
 772 drawback of this method is its reliance on instantaneous position and velocity data.

Without access to observed velocities, we use their midpoint approximation. Denoting with $x(t) := [q(t), \dot{q}(t)]$, we approximate $x(t_{i+\frac{1}{2}})$ using (q_i, q_{i+1}) :

$$x(t_{i+\frac{1}{2}}) \approx \bar{x}(q_i, q_{i+1}), \quad \bar{x}(q_i, q_{i+1}) := [\bar{q}_{i+\frac{1}{2}}, \bar{\dot{q}}_{i+\frac{1}{2}}],$$

here we use the definitions of Section 3.1 for $\bar{q}_{i+\frac{1}{2}}, \bar{\dot{q}}_{i+\frac{1}{2}}$.

The loss function we use is

$$\mathcal{L} = \frac{1}{N_{\mathcal{T}}(N-1)} \sum_{\mathcal{T}} \sum_{n=1}^{N-1} \|\hat{x}(q_{n-1}, q_n) - \bar{x}(q_n, q_{n+1})\|_2^2, \quad (16)$$

where $\hat{x}(q_{n-1}, q_n)$ is the prediction to the model.

One significant limitation of this approach lies in the requirement to compute the inverse of the Hessian $\left(\frac{\partial^2 L}{\partial \dot{q}^2}\right)^{-1}$ for each data point, as noted by [3]. This process is significantly resource-intensive. In the proposed method, the Hessian is only necessary to evaluate when regularizing the model, thereby avoiding the need to determine the Hessian at every data point. This offers a notable computational advantage. Furthermore, the GLNN requires that the Hessian matrix is invertible at all points to avoid invalid computations, which makes the training process challenging.

In the implementation of Neural ODEs we use the same approximation of the velocities and the same loss function.

C Hyperparameters

To prevent overfitting and ensure sufficient training, we save the model parameters at each epoch and select the version corresponding to the lowest validation loss as the final model.

Table 3: Network parameters

Task		DFLNN		GLNN		Neural ODE
		L_{θ}	F_{θ}	L_{θ}	F_{θ}	
Task 1: Damped Double Pendulum	Hidden dim	30	30	30	30	30
	Hidden Layers	3	3	3	3	3
	Activation	GELU	GELU	GELU	GELU	GELU
Task 2: Dissipative Charged Particle	Hidden dim	30	30	30	30	30
	Hidden Layers	3	3	3	3	3
	Activation	GELU	GELU	GELU	GELU	GELU
Task 3: Pixel Damped Pendulum	Hidden dim	30	30	30	30	30
	Hidden Layers	3	3	3	3	3
	Activation	GELU	GELU	GELU	GELU	GELU
Task 4: Human Motion Capture	Hidden dim	30	30	30	30	30
	Hidden Layers	3	3	3	3	3
	Activation	GELU	GELU	GELU	GELU	GELU

Table 3 summarize the dimensions of each feed-forward neural network element. The hyperparameters related to the loss function are detailed in Table 4, while Table 5 outlines the hyperparameters employed during training using the Adam optimizer.

D Supporting Experiments

Section 4 provided an overview of our results, including an analysis of the model’s performance relative to the time resolution of the dataset. Figure 6 presents the results for Task 1, whereas Figure 7 shows the findings for Task 2. These figures demonstrate both the performance of the proposed model with inductive biases of the linear external force structure and that of a model without any

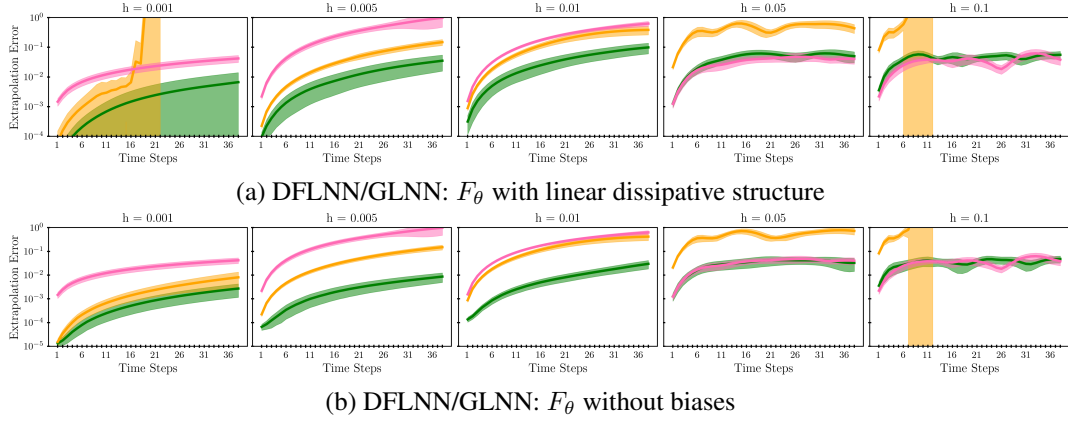


Figure 6: Results for Task 1 with different time resolution over the dataset. Solid Green lines are DFLNN (proposed), yellow lines are the GLNN model, and pink lines are a Neural ODE. (left to right) The extrapolation error of the trained models as the step size (h) increases, while maintaining constant linear damping. Findings revealed that baseline models could only match the proposed model's performance at very large step sizes. In this scenario, a larger step size leads to a dataset where each trajectory quickly converges towards uniform motion due to damping.

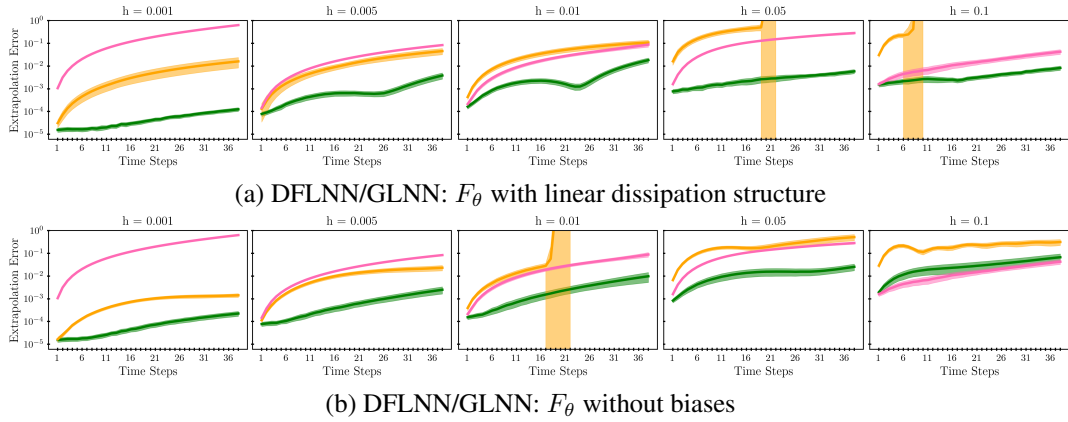


Figure 7: Results for Task 2 with different time resolution over the dataset. Solid Green lines are DFLNN (proposed), yellow lines are the GLNN model, and pink lines are a Neural ODE. (left to right) The extrapolation error of the trained models as the step size (h) increases, while maintaining constant linear damping. Findings revealed that baseline models could only match the proposed model's performance at very large step sizes. In this scenario, a larger step size leads to a dataset where each trajectory quickly converges towards uniform motion due to damping.

Table 4: Loss parameters. '-' indicates that the hyperparameter is not defined for the specific model.

Task		DFLNN	GLNN	Neural ODE
Task 1: Damped Double Pendulum	ω_{physics}	0.5	-	-
	ω_{reg}	0.5	-	-
	R	100	-	-
	dropout rate (for F^{Free})	0.5	0.5	-
Task 2: Dissipative Charged Particle	ω_{physics}	0.5	-	-
	ω_{reg}	0.5	-	-
	R	100	-	-
	dropout rate (for F^{Free})	0.5	0.5	-
Task 3: Pixel Damped Pendulum	ω_{physics}	0.9	-	-
	ω_{reg}	0.1	-	-
	ω_{AE}	1.0	1.0	1.0
	R	20	-	-
	dropout rate (for F^{Free})	0.5	0.5	-
Task 4: Human Motion Capture	ω_{physics}	0.5	-	-
	ω_{reg}	0.5	-	-
	ω_{AE}	1.0	1.0	1.0
	R	100	100	-
	dropout rate (for F^{Free})	0.5	0.5	-

Table 5: Training parameters. '-' indicates that the entire dataset is provided at each iteration.

Task	Model	Learning rate	Batch size	Epochs
Task 1: Damped Double Pendulum	DFLNN	0.001	-	20.000
	GLNN	0.001	-	20.000
	NeuralODE	0.001	-	20.000
Task 2: Dissipative Charged Particle	DFLNN	0.001	-	20.000
	GLNN	0.001	-	20.000
	NeuralODE	0.001	-	20.000
Task 3: Pixel Damped Pendulum	DFLNN	0.001	1000	5.000
	GLNN	0.001	1000	5.000
	NeuralODE	0.001	1000	5.000
Task 4: Human Motion Capture	DFLNN	0.001	-	20.000
	GLNN	0.001	-	20.000
	NeuralODE	0.001	-	20.000

assumptions about the external force structure. The data indicate that, although both models perform comparably, the version incorporating inductive bias achieves better accuracy. Furthermore, the proposed model exhibited more consistent performance with regard to the dataset's time resolution in comparison to the baseline models.

Computational resources For all experiments presented in this paper, it was sufficient to train the proposed model on a CPU with a maximal training time of ~ 30 minutes. One exception is Task 3, where we introduced an autoencoder with convolutional layers to treat image data. For this task, we utilized a GPU, and the training time increased by a factor of ~ 6 .

E Limitations

The computational cost of training the proposed model is affected by the number of regularization points, as we introduce a log term of the Hessian. In our experiment, we have only been considering a

806 limited number of regularization points ($R \leq 100$) as this was sufficient for the presented experiments.
 807 Hence, very large numbers of regularization points have not been in the scope of this study.

808 Moreover, the objective for this study was to explore the impact of the discrete Euler-Lagrange
 809 equations instead of focusing on hyperparameter tuning. Consequently, we did not conduct an
 810 extensive hyperparameter search, and limited the experiments to small neural networks with 3 layers,
 811 each containing 30 neurons, for both the proposed and baseline models as described in section C
 812 Deep networks and other network architectures have not been considered.

813 F Datasets

814 The data used in Tasks 1 and 2 was integrated using `scipy.integrate.odeint`. We will now
 815 comment on the equations being used when integrating the systems considered in Task 1 and 2.

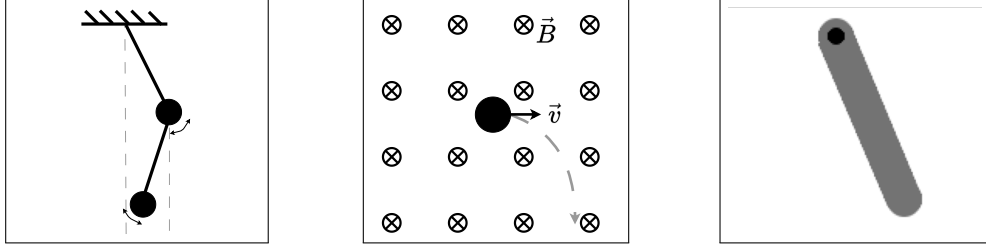


Figure 8: Synthetic datasets. From the left-panel: a damped double pendulum, a charged particle moving through a magnetic field, and a damped single pendulum represented with pixel frames.

816 F.1 Damped Double pendulum

817 Consider two masses, m_1 and m_2 , each connected in sequence to a stationary pivot via rigid rods
 818 of lengths l_1 and l_2 , respectively. The displacement angles θ_1 and θ_2 define the system's position
 819 relative to equilibrium. For a damped double pendulum, where each pendulum encounters a damping
 820 force proportional to its velocity, the motion is governed by the following equations:

$$\ddot{\theta}_1 = \frac{-g(2m_1 + m_2) \sin(\theta_1) - m_2 g \sin(\theta_1 - \theta_2) - 2 \sin(\theta_1 - \theta_2) m_2 \left(\dot{\theta}_2^2 l_2 + \dot{\theta}_1^2 l_1 \cos(\theta_1 - \theta_2) \right)}{l_1(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2))} - b_1 \dot{\theta}_1 \quad (17)$$

$$\ddot{\theta}_2 = \frac{2 \sin(\theta_1 - \theta_2) \left(\dot{\theta}_1^2 l_1 (m_1 + m_2) + g(m_1 + m_2) \cos(\theta_1) + \dot{\theta}_2^2 l_2 m_2 \cos(\theta_1 - \theta_2) \right)}{l_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2))} - b_2 \dot{\theta}_2 \quad (18)$$

821 where g represents the gravitational acceleration, and b_1 and b_2 denote the damping coefficients for
 822 the first and second pendulums, respectively.

823 F.1.1 Dissipative Charged particle traveling in a Magnetic Field

824 The Lorentz force characterizes the motion of a charged particle moving within a magnetic field.
 825 Introducing dissipation results in energy loss within the system over time. This concept is applicable
 826 to various physical systems, including plasmas or conductors, where particles experience resistive or
 827 damping forces as they navigate through magnetic fields. We utilize Newton's second law to model
 828 the particle's dynamics. The total force acting on the particle combines the Lorentz force with a
 829 damping force representing dissipation.

830 The expression for the Lorentz force is given by:

$$\mathbf{F}_{\text{Lorentz}} = q(\mathbf{v} \times \mathbf{B}), \quad (19)$$

831 where q represents the particle's charge, \mathbf{v} denotes the velocity vector, and \mathbf{B} stands for the magnetic
 832 field vector. A dissipative force that is linear in velocity reads:

$$\mathbf{F}_{\text{damping}} = -b\mathbf{v}, \quad (20)$$

833 with b being the damping coefficient. From Newtons second law, the acceleration \mathbf{a} of the particle is
 834 then:

$$\mathbf{a} = \frac{q}{m}(\mathbf{v} \times \mathbf{B}) - \frac{b}{m}\mathbf{v} \quad (21)$$

835 where m denotes the mass of the particle. Specifically described for a three-dimensional Cartesian
 836 coordinate system:

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dz}{dt} = v_z \quad (22)$$

$$\frac{dv_x}{dt} = \frac{q}{m}(v_y B_z - v_z B_y) - \frac{b}{m}v_x \quad (23)$$

$$\frac{dv_y}{dt} = \frac{q}{m}(v_z B_x - v_x B_z) - \frac{b}{m}v_y \quad (24)$$

$$\frac{dv_z}{dt} = \frac{q}{m}(v_x B_y - v_y B_x) - \frac{b}{m}v_z \quad (25)$$

837 In the experiments considered in this paper, the magnetic field remains constant over time.