Numerical integrators for learning dynamical systems from noisy data

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Decades of research have been spent on classifying the properties of numerical integrators when solving ordinary differential equations (ODEs). Here, a first step is taken to examine the properties of numerical integrators when used to learn dynamical systems from noisy data with neural networks. Mono-implicit Runge–Kutta (MIRK) methods are a class of integrators that can be considered explicit for inverse problems. The symplectic property is useful when learning the dynamics of Hamiltonian systems. Unfortunately, a proof shows that symplectic MIRK methods have a maximum order of $p = 2$. By taking advantage of the inverse explicit property, a novel integration method called the mean inverse integrator, tailored for solving inverse problems with noisy data, is introduced. As verified in numerical experiments on different dynamical systems, this method is less sensitive to noise in the data.

1 Introduction

Dynamical systems describing and enhancing properties of neural networks was a topic of study [1, 2, 3] also prior to the seminal work on neural ODEs [4]. On the other hand, neural networks can be utilized to learn solutions of pre-specified ordinary or partial differential equations from data using physics-informed neural networks [5, 6]. Similarly, Hamiltonian neural networks [7] combine numerical integrators and neural networks to approximate the Hamiltonian function of energy preserving dynamical systems. ODEs on Hamiltonian form have been widely studied in the field of geometric numerical integration [8] where the symplectic property of the ODE flow is a key characteristic. This property could inform the neural network architecture [9] or guide the choice of numerical integrator, yielding a theoretical guarantee that the learning target is actually a Hamiltonian function [10, 11].

Given an ODE

$$\dot{y} = f(y), \quad y(t) : [0, T] \to \mathbb{R}^n,$$

the initial value problem aims at computing solutions $y(t_i)$ when the vector field $f(y)$ and an initial value $y(t_0) = y_0$ is known. The focus of this study is the inverse problem, which assumes knowledge of multiple samples of the solution $S_N = \{y_i\}_{i=1}^N$ and aims instead at approximating the vector field with a neural network such that $f_\theta \approx f$. Famously, the Runge–Kutta (RK) integrators have been studied for decades for solving initial value problems. This begs the question of how such methods are best leveraged in the inverse case.

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

2 Numerical integration in inverse ODE problems

2.1 Inverse ODE problems on Hamiltonian form

Assuming that the ODE samples S_N are known. The inverse problem is the following optimization problem:

$$\text{find parameters } \theta \text{ satisfying} \qquad \min_{\theta} \sum_{n=0}^{N-1} \Big\| y_{n+1} - \Phi_{h,f_\theta}(y_n) \Big\|, \tag{1}$$

where f_θ is a neural network with parameters θ and Φ_{h,f_θ} is a one-step integration method with step length $h > 0$.

In particular, for Hamiltonian systems we follow the idea behind Hamiltonian neural networks [7] and learn the Hamiltonian $H : \mathbb{R}^n \to \mathbb{R}$, with $n = 2d,\ d \in \mathbb{Z}_+$. In this case, the neural network is of the form

$$f_\theta(y) := J\nabla H_\theta(y), \quad \text{where} \quad J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}, \tag{2}$$

such that the learned vector field always forms a Hamiltonian system. In the numerical experiments, this form will be used when learning the dynamics of the double pendulum problem. For the Lotka–Volterra problem, the vector field will be learned directly, obtaining an approximation $f_\theta : \mathbb{R}^n \to \mathbb{R}^n$.

2.2 Inverse explicit integrators

To guarantee properties such as symmetry and stability, numerical integrators often need to be implicit, requiring the solution of non-linear equations at every step. One such example is the implicit midpoint method

$$\hat{y}_{n+1} = \Phi_{h,f}(y_n, \hat{y}_{n+1}) = y_n + hf\big(\frac{y_n + \hat{y}_{n+1}}{2}\big). \tag{3}$$

However, in the setting of inverse ODE problems, trajectories $S_N = \{y_i\}_{i=0}^N$ are known while the vector field f is what we want to approximate. The value of the solution at time t_n and t_{n+1}, y_n and y_{n+1} are both known and can be inserted in Equation (3), yielding an explicit procedure to approximate f. The procedure, which we here denote as the *inverse injection*, is utilized successfully by multiple authors [12, 13, 14, 15] when learning dynamical systems from data. The midpoint method is an implicit scheme when utilized on an initial value problem, but is explicit for the inverse problem under the inverse injection. It would be highly valuable to identify other implicit Runge–Kutta schemes that are inverse explicit. Let us denote such methods as being *inverse explicit*. E.g. the Gauss-Legendre collocation methods of order $p > 3$ are not inverse explicit as their stages k_i are defined implicitly. However, it could be shown that the RK sub-class called mono-implicit Runge–Kutta (MIRK) methods [16, 17] are all inverse explicit. When used to solve initial value problems, these methods require only solving a system for the next step $\hat{y}_{n+1}$ and not for the stages k_i. They are thus explicit under the inverse injection.

MIRK methods are defined by coefficients $b, v \in \mathbb{R}^s$ and a lower triangular matrix $D \in \mathbb{R}^{s\times s}$ such that

$$\begin{aligned} k_i &= f\big(y_n + v_i(y_{n+1} - y_n) + h\sum_{j=1}^{s} d_{ij}k_j\big), \qquad i = 1, \dots, s, \\ y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i k_i, \end{aligned} \tag{4}$$

where $d_{ij} := [D]_{ij}$. Knowing a class of integration methods that are computationally efficient for inverse problems allows for the construction of numerical integrators tailored to specific problems, where high order, symmetry or symplecticity might be of importance. However, the following Theorem bounds the maximum order of symplectic MIRK methods and is proved in Appendix B.

Theorem 1. *The maximum order of a symplectic MIRK method is* $p = 2$.

Examples of two MIRK methods with order $p = 4$ and $p = 6$ can be found in Appendix A. Aside from Runge–Kutta methods, discrete gradient integration methods [18, 19] are inverse explicit and well suited to train Hamiltonian neural networks using a modified backpropagation algorithm [20].

2.3 Inverse ODE problems with noise

It is often the case that the samples S_N are not exact, but perturbed by noise. A noisy ODE sample is here defined by an independent, normally distributed perturbation

$$\tilde{y}_i = y_i + \delta_i, \quad \delta_i \sim \mathcal{N}(0, \sigma^2 I), \tag{5}$$

where $\mathcal{N}(0,\sigma^2 I)$ represents the multivariate normal distribution and we assume that $\sigma > 0$ is sufficiently small compared to the step size h. The flow of an ODE is the map $\varphi_{h,f}$, such that given an initial value $y(t_0)$, it yields the solution at time $t_0 + h$ of the ODE, $\varphi_{h,f}(y(t_0)) := y(t_0 + h)$. The flow map satisfies the following fundamental group property

$$\varphi_{h_1,f} \circ \varphi_{h_2,f}\big(y(t_0)\big) = \varphi_{h_1+h_2,f}(y(t_0)), \qquad h_1, h_2 > 0.$$

Replacing exact flows by numerical flows, the mean inverse integrator (MII) removes noise leveraging this group property. In fact, compositions of a one-step method $\Phi_{h,f}$ can be utilized to generate multiple approximations to the same point in the flow. Assuming we know the points $\{\tilde{y}_0, \tilde{y}_1, \tilde{y}_2, \tilde{y}_3\}$, then $\hat{y}_2$ can be approximated by computing the mean of the numerical flows Φ starting from different initial values:

$$\overline{y}_2 = \frac{1}{3}\big(\Phi_{h,f}(\tilde{y}_1) + \Phi_{h,f} \circ \Phi_{h,f}(\tilde{y}_0) + \Phi_{-h,f}(\tilde{y}_3)\big) = \frac{1}{3}\big(\tilde{y}_0 + \tilde{y}_1 + \tilde{y}_3 + h(\hat{f}_{0,1} + 2\hat{f}_{1,2} - \hat{f}_{3,2})\big),$$

where $\hat{f}_{n,n+1}$ is the vector field evaluation of an inverse explicit numerical integrator such that $\Phi_{h,f}(\tilde{y}_n, \tilde{y}_{n+1}) = \tilde{y}_n + h\hat{f}_{n,n+1}$. For the midpoint method we have $\hat{f}_{n,n+1} = f(\frac{\tilde{y}_n + \tilde{y}_{n+1}}{2})$. The mean approximation over the whole trajectory $\overline{y}_i$, for $i = 0, \ldots, N$, could be computed simultaneously, reusing multiple vector field evaluations in an efficient manner. E.g., when $N = 3$ we get

$$\begin{bmatrix} \overline{y}_0 \\ \overline{y}_1 \\ \overline{y}_2 \\ \overline{y}_3 \end{bmatrix} = \frac{1}{3}\left(\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{y}_0 \\ \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \end{bmatrix} + h \begin{bmatrix} -3 & -2 & -1 \\ 1 & -2 & -1 \\ 1 & 2 & -1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \hat{f}_{0,1} \\ \hat{f}_{1,2} \\ \hat{f}_{2,3} \end{bmatrix} \right). \tag{6}$$

The same structure is illustrated in Appendix C. In general, for a sample S_N and an inverse explicit integrator $f_{n,n+1}$ the mean inverse integrator is given by

$$\overline{Y} = \frac{1}{N}\left(UY + hV\hat{F}\right), \tag{7}$$

where

$$Y := [y_0, \ldots, y_N]^T \in \mathbb{R}^{(N+1)\times m} \quad \text{and} \quad \hat{F} := [\hat{f}_{0,1}, \ldots, \hat{f}_{N-1,N}]^T \in \mathbb{R}^{N \times m}.$$

Finally, $U \in \mathbb{R}^{(N+1)\times(N+1)}$ and $V \in \mathbb{R}^{(N+1)\times N}$ are given by

$$[U]_{ij} := \begin{cases} 0 & \text{if} \quad i = j \\ 1 & \text{else} \end{cases} \qquad \text{and} \qquad [V]_{ij} := \begin{cases} j - 1 - N & \text{if} \quad j \geq i \\ j & \text{else} \end{cases}.$$

By substituting the known vector field f, with a neural network f_θ and denoting the matrix with vector field evaluations by $\hat{F}_\theta$ such that $\overline{Y}_\theta := \frac{1}{N}(UY + hV\hat{F}_\theta)$, we can formulate the inverse problem in (1) as

$$\text{find parameters } \theta \text{ satisfying} \qquad \min_\theta \big\|Y - \overline{Y}_\theta\big\|. \tag{8}$$

Note that in the implementation of the algorithm for training f_θ, higher accuracy was achieved if the neural network was trained for some initial epochs using a one-step scheme as in Equation (1), before proceeding to use the mean inverse integrator. This could be understood as a pre-conditioning or pre-training of f_θ.

3 Experiments

The numerical integrators in Table 1 are utilized to learn vector fields from data obtained from the double pendulum and the Lotka–Volterra system. Both problems are defined in Appendix D. The inverse explicit methods are tested both as one-step methods and when used as temporal

Integration method	Name in plots	Order	Stages	Symm.	Sympl.
Implicit midpoint	Midpoint	2	1	yes	yes
MIRK4 from midpoint	MIRK4 mid	4	4	yes	no
MIRK6	MIRK6	6	5	no	no

Table 1: Methods used in experiments. *Symm.* is short for symmetric and *sympl.* for symplectic.

discretization in the mean inverse integrator. After using the integrators in training, approximated solutions $\tilde{y}_{n+1} = \Phi_{h,f_\theta}(y_n)$ are computed and the error is found over M different points by

$$e(f_\theta) = \frac{1}{M}\sum_{i=1}^{M} \|\tilde{y}_i - y_i\|_2 \quad \text{where} \quad y_i \in S_N^{\text{test}}.$$

For all test problems, the neural networks have 3 layers with a width of 100 neurons and $\tanh(\cdot)$ as the activation function and are trained with the *L-BFGS* algorithm for 40 epochs. The training data is generated by integrating $N = 500$ initial values with step sizes and number of steps given by $\{h = 0.4, n = 3\}$ and $\{h = 0.1, n = 12\}$. The points in the flow are perturbed by noise where $\sigma \in \{0, 0.05\}$. Error is measured in $M = 10$ points in the flow and the standard deviation is estimated by re-running 10 experiments with random initializations for both parameters θ and samples S_N.

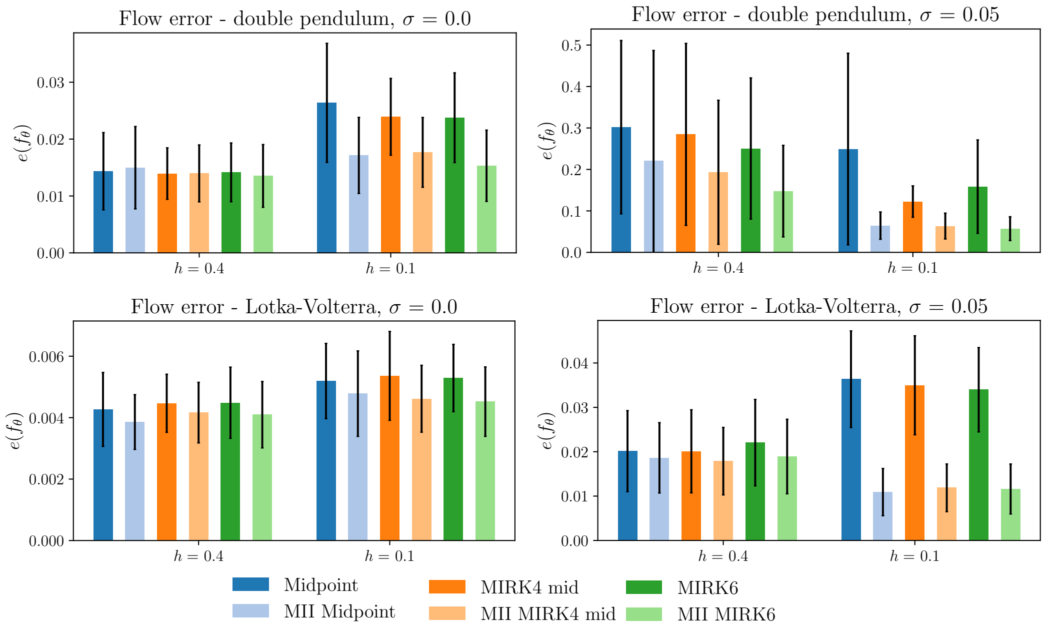

Figure 1: Error in the flow for the double pendulum and Lotka–Volterra problem. The height of bars represents the mean error over 10 experiments and the length of the black line represents the standard deviation.

4 Conclusion

The main contribution of this work is the characterization of the inverse explicit property of the MIRK methods and the novel mean inverse integrator. As seen in Figure 1, the mean inverse integrator yields lower error in the numerical flow, as well as lower standard deviation in the error estimate. The MII method has relatively lower error when the step size is $h = 0.1$, which might be due to increased discretization error for the MII method at larger step sizes h. The same phenomenon is observed when studying the roll-out in time in Figure 2. There is however a need to do a theoretical analysis of how the MII method balances smoothing of noise against increased discretization error. MIRK methods opens up for using a range of different numerical integrators in training. The results in Figure 1, particularly for the chaotic double pendulum problem, might indicate that both order and symmetry of integrators matter for accuracy when training on noisy data. There is an extensive literature on Runge–Kutta methods and MIRK methods in particular and there might be other concepts and methods which could extend the current toolbox for learning dynamical systems from data.

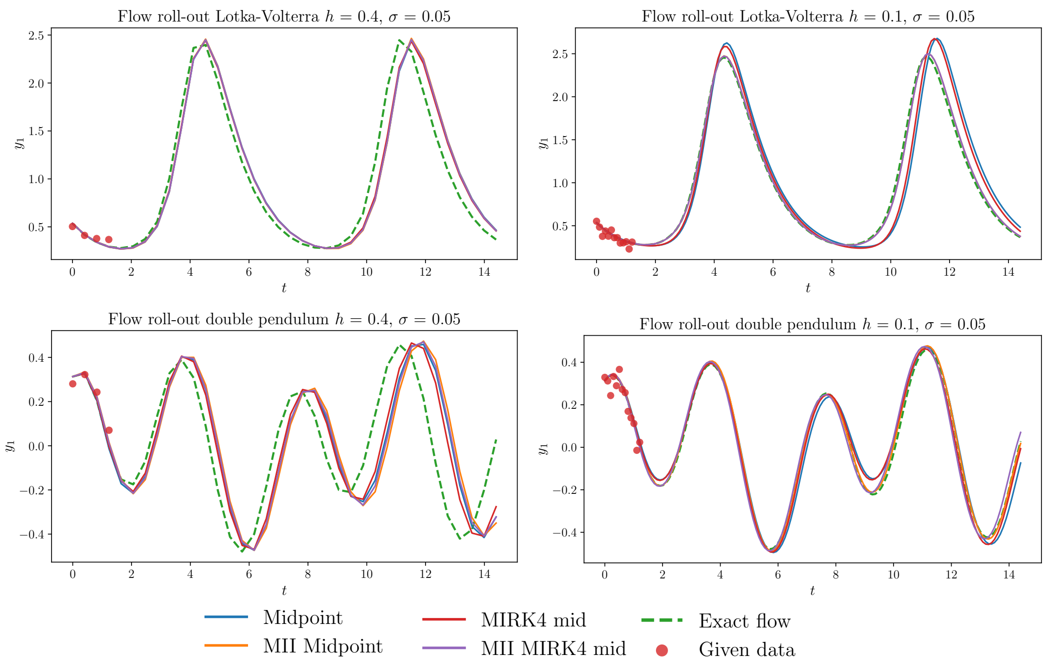

Figure 2: Roll-out in time of the y_1 variable including approximations trained by different integration methods on noisy data, $\sigma = 0.05$. The red dots illustrate the number of points and amount of noise in one of the $M = 500$ training trajectories.

References

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

[2] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020.

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

[4] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[5] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[6] Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, et al. NeuralPDE: Automating physics-informed neural networks (PINNs) with error approximations. *arXiv preprint arXiv:2107.09443*, 2021.

[7] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. *CoRR*, abs/1906.01563, 2019.

[8] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations; 2nd ed.* Springer, Dordrecht, 2006.

[9] Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

[10] Aiqing Zhu, Pengzhan Jin, Beibei Zhu, and Yifa Tang. Inverse modified differential equations for discovery of dynamics. *arXiv preprint arXiv:2009.01058*, 2020.

[11] Christian Offen and Sina Ober-Blöbaum. Symplectic integration of learned Hamiltonian systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(1):013122, 2022.

[12] Marco David and Florian Méhats. Symplectic Learning for Hamiltonian Neural Networks. *arXiv preprint arXiv:2106.11753*, 2021.

[13] Elena Celledoni, Andrea Leone, Davide Murari, and Brynjulf Owren. Learning Hamiltonians of constrained mechanical systems. *Journal of Computational and Applied Mathematics*, 417:114608, 2023.

[14] Sølve Eidnes, Alexander J Stasik, Camilla Sterud, Eivind Bøhn, and Signe Riemer-Sørensen. Port-Hamiltonian neural networks with state-dependent ports. *arXiv preprint arXiv:2206.02660*, 2022.

[15] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *arXiv preprint arXiv:1909.13334*, 2019.

[16] Jeff R Cash. A class of implicit Runge–Kutta methods for the numerical integration of stiff ordinary differential equations. *Journal of the ACM (JACM)*, 22(4):504–511, 1975.

[17] K Burrage, FH Chipman, and Paul H Muir. Order results for mono-implicit Runge–Kutta methods. *SIAM journal on numerical analysis*, 31(3):876–891, 1994.

[18] GRW Quispel and Grant S Turner. Discrete gradient methods for solving ODEs numerically while preserving a first integral. *Journal of Physics A: Mathematical and General*, 29(13):L341, 1996.

[19] Robert I McLachlan, G Reinout W Quispel, and Nicolas Robidoux. Geometric integration using discrete gradients. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 357(1754):1021–1045, 1999.

[20] Takashi Matsubara, Ai Ishikawa, and Takaharu Yaguchi. Deep energy-based modeling of discrete-time physics. *Advances in Neural Information Processing Systems*, 33:13100–13111, 2020.

A Mono-implicit Runge–Kutta methods

A general Runge–Kutta method with s stages is a one-step numerical integrator given by

$$\begin{aligned} k_i &= f\big(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\big), \qquad i = 1,\dots,s, \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_i k_i, \end{aligned} \tag{9}$$

and the method is specified by the coefficient matrix $A \in \mathbb{R}^{s\times s}$ and the vector $b \in \mathbb{R}^s$, where $a_{ij} = [A]_{ij}$, $b_i = [b]_i$, requiring that $c_i = \sum_{j=1}^{s} a_{ij}$ for $i = 1,\dots,s$. A method could be compactly represented by a *Butcher tableau* which structures the coefficients the following way:

$$\begin{array}{c|c} \text{c} & \text{A} \\ \hline & b^T \end{array}$$

A MIRK method defined in Equation (4) is specified by a coefficient vector $b \in \mathbb{R}^s$, $v \in \mathbb{R}^s$ in addition to the strictly lower triangular matrix $D \in \mathbb{R}^{s\times s}$. The MIRK methods are usually represented by an extended Butcher tableau with an extra column for v and the matrix D replaces the A matrix, yielding

$$\begin{array}{c|c|c} \text{c} & \text{v} & \text{D} \\ \hline & & b^T \end{array}$$

In [17] it is proved that the maximum order of an s-stage MIRK method is $p = s + 1$ and several methods with stages $s \leq 5$ are presented. The method called *MIRK4 mid* (left tableau below) is a symmetric MIRK method that could be understood as introducing two new stages to the 2−stage Gauss-Legendre collocation method of order $p = 4$, and is first presented in [14]. *MIRK6* (right tableau below) is found in [17] and is a MIRK method with $s = 5$ stages and order $p = 6$.

$$
\begin{array}{c|c|cccc}
\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{2}-\frac{\sqrt{3}}{6} & 0 & 0 & 0 & 0 \\
\frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{2}+\frac{\sqrt{3}}{6} & 0 & 0 & 0 & 0 \\
\frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{2} & 0 & -\frac{\sqrt{3}}{6} & 0 & 0 \\
\frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{2} & \frac{\sqrt{3}}{6} & 0 & 0 & 0 \\
\hline
 & & 0 & 0 & \frac{1}{2} & \frac{1}{2}
\end{array}
\qquad
\begin{array}{c|c|ccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{4} & \frac{5}{32} & \frac{9}{64} & -\frac{3}{64} & 0 & 0 & 0 \\
\frac{3}{4} & \frac{27}{32} & \frac{3}{64} & -\frac{9}{64} & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & -\frac{5}{24} & \frac{5}{24} & \frac{2}{3} & -\frac{2}{3} & 0 \\
\hline
 & & \frac{7}{90} & \frac{7}{90} & \frac{16}{45} & \frac{16}{45} & \frac{2}{15}
\end{array}
$$

B Proof of Theorem 1

Proof. A Runge–Kutta method as given by Equation (9) is symplectic if and only if

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0.$$

Inserting the MIRK coefficients a_{ij} as given by Equation (4), we get

$$\begin{aligned} b_i(v_i b_j + d_{ij}) + b_j(v_j b_i + d_{ji}) - b_i b_j &= 0 \\ b_i d_{ij} + b_j d_{ji} + b_i b_j (v_j + v_i - 1) &= 0. \end{aligned}$$

As D is strictly lower triangular, we get that

$$\begin{aligned} \text{either} \quad d_{ji} = 0 \quad \text{or} \quad d_{ij} = 0 \implies \quad & b_i d_{ij} + b_i b_j (v_j + v_i - 1) = 0, \\ \text{if } i = j \implies \quad & b_i^2 (2v_i - 1) = 0. \end{aligned}$$

Requiring d_{ij}, b_i and v_i to satisfy the symplecticity condition yields the following restriction

$$\begin{aligned} b_i d_{ij} + b_i b_j (v_j + v_i - 1) = 0, &\quad \text{for } i \neq j, \\ \text{and} \quad b_i = 0 \text{ or } v_i = \frac{1}{2}, &\quad \text{for } i = j. \end{aligned} \tag{10}$$

Without loss of generality, we assume that the m first entries of $b \in \mathbb{R}^s$ are zero. Enforcing the conditions of Equation (10) on $v \in \mathbb{R}^s$ we get for $1 \leq m \leq s$

$$\begin{aligned} b &= [0, \dots, 0, b_{m+1}, \dots, b_s]^T, \\ v &= [v_1, \dots, v_m, \frac{1}{2}, \dots, \frac{1}{2}]^T. \end{aligned}$$

In total, the MIRK coefficient matrix $A = D + vb^T$ gives a Butcher tableau of the form

$$
\begin{array}{c|cccc|ccc}
 & 0 & 0 & \dots & 0 & v_1 b_{m+1} & \dots & v_1 b_s \\
 & d_{21} & 0 & \dots & 0 & & & \\
 & d_{31} & d_{32} & & 0 & \vdots & & \vdots \\
 & \vdots & & \ddots & & & & \\
 & d_{m,1} & \dots & d_{m,m-1} & 0 & v_m b_{m+1} & \dots & v_m b_s \\
\hline
 & 0 & \dots & \dots & 0 & \frac{1}{2} b_{m+1} & \dots & \frac{1}{2} b_s \\
 & \vdots & & & \vdots & \vdots & & \vdots \\
 & 0 & \dots & \dots & 0 & \frac{1}{2} b_{m+1} & \dots & \frac{1}{2} b_s \\
\hline
 & 0 & \dots & \dots & 0 & b_{m+1} & \dots & b_s
\end{array}
$$

Since the lower left submatrix is the zero matrix, this leaves the stages $k_{m+1}, \dots, k_s$ unconnected to the first m stages. Furthermore as $b_i = 0$ for $i = 1, \dots, m$, these stages are not included in the computation of the final integration step. The method is thus reducible to the lower right submatrix of A and $b_{m+1}, \dots, b_s$. The reduced method is in general given by

$$\begin{array}{c|ccc} & \frac{1}{2}b_1 & \dots & \frac{1}{2}b_s \\ & \vdots & & \vdots \\ & \frac{1}{2}b_1 & \dots & \frac{1}{2}b_s \\ \hline & b_1 & \dots & b_s \end{array}$$

It is trivial to check that if $\sum_i^s b_i = 1$ the method satisfies order conditions up to order $p = 2$, which could be found in [8, Ch. III.1.1] to be

$$\sum_i b_i = 1, \qquad \text{and} \qquad \sum_{i,j} b_i a_{ij} = \frac{1}{2},$$

but fails to satisfy the first of the two conditions required for order $p = 3$, since

$$\sum_{i,j,k} b_i a_{ij} a_{ik} = \frac{1}{4} \sum_{i,j,k} b_i b_j b_k = \frac{1}{4} \neq \frac{1}{3}.$$

Hence, the maximum order of a symplectic MIRK method is $p = 2$. □

C Structure of the mean inverse integrator

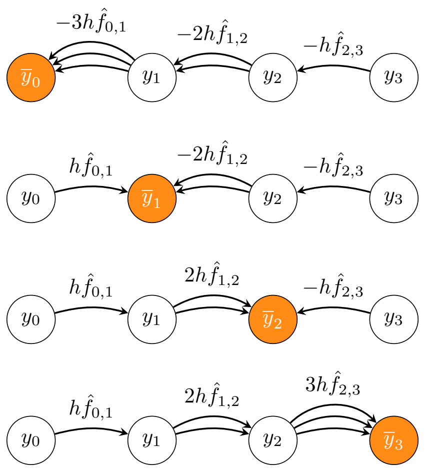

Figure 3: Illustration of the structure of the mean inverse integrator for $N = 3$.

D Test problems

Let q_i and p_i denote the angle and angular momentum of pendulum $i = 1, 2$. The double pendulum system has a Hamiltonian that is not separable, where $y = [q_1, q_2, p_1, p_2]^T \in \mathbb{R}^4$ and the Hamiltonian is given by

$$H(q_1, q_2, p_1, p_2) = \frac{\frac{1}{2}p_1^2 + p_2^2 - p_1 p_2 \cos(q_1 - q_2)}{1 + \sin^2(q_1 - q_2)} - 2\cos(q_1) - \cos(q_2).$$

The ODE is thus defined by the vector field $f(y) := J\nabla H(y)$ where the matrix J is the same as in Equation (2). The Lotka–Volterra problem is defined by the interaction of two species, of which population number is represented by $y_1 > 0$ and $y_2 > 0$ assuming that y_1 is the prey of a predator y_2. Assuming that all interaction parameters are given by $\alpha = \beta = \gamma = \delta = 1$ the vector field is given by

$$f(y) = \begin{bmatrix} y_1 - y_1 y_2 \\ y_1 y_2 - y_2 \end{bmatrix}.$$