Geometric Numerical Integration and Scientific Machine Learning

Benedikt Brantner (as a stand-in for Michael Kraus)
(benedikt.brantner@ipp.mpg.de, michael.kraus@ipp.mpg.de)

Max–Planck–Institut für Plasmaphysik
Technische Universität München, Zentrum Mathematik

Outline

Ordinary Differential Equations and Runge-Kutta Methods

Numerical Solution of Ordinary Differential Equations

- consider the *initial value problems* with vector field f

$$\dot{x}(t) = f(t, x(t)), \qquad x(t_0) = x_0, \qquad f : \mathbb{R}^d \to \mathbb{R}^d, \qquad t, t_0 \in \mathcal{I} \subset \mathbb{R}$$

- the solution $x(t)$ is a curve $x : \mathcal{I} \to \mathbb{R}^d$, which satisfies the initial value problem
- a numerical one-step method for the solution of an initial value problem is an update rule of the form

$$\frac{x_{n+1} - x_n}{h} = \bar{f}(t_n, t_{n+1}; x_n, x_{n+1}) \qquad \text{with} \qquad x_n \approx x(t_n), \quad t_n = t_0 + nh,$$

such that

$$\bar{f}(t_n, t_n; x_n, x_n) = f(t_n, x_n) + \mathcal{O}(h)$$

The Explicit Euler Method

- idea: Taylor expansion of the solution $x(t)$ at t_0:

$$x(t_0 + h) = x(t_0) + h\dot{x}(t_0) + \mathcal{O}(h^2) = x(t_0) + h\, f(t_0, x_0) + \mathcal{O}(h^2).$$

- Neglect higher-order terms:

$$x_1 = x_0 + h\, f(t_0, x_0)$$

- Proceed iteratively:

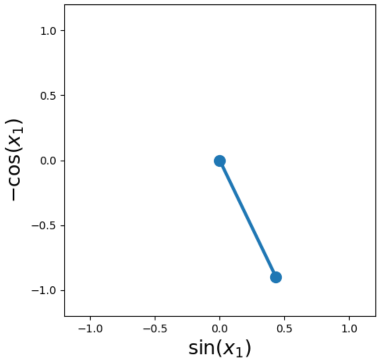
$$x_{n+1} = x_n + h\, f(t_n, x_n)$$

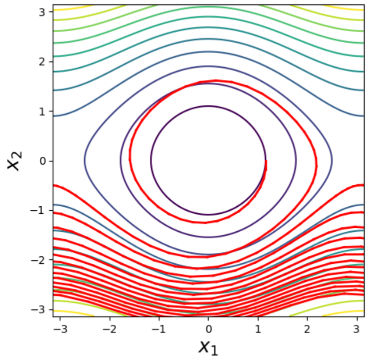
- the computational cost of the method lies in the evaluation of the function f!

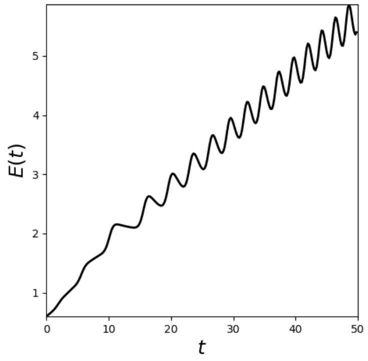
The Mathematical Pendulum with the Explicit Euler Method

- mathematical pendulum: two degrees of freedom $\{x_1, x_2\}$
  (mass $m = 1$, massless rod of length $l = 1$, gravitational acceleration $g = 1$)
- equations of motion

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\sin x_1, \qquad\qquad x_1(0) = \arccos(0.4), \qquad x_2(0) = 0$$







The Implicit Euler Method

- $x_{n+1} = x_n + hf(t_{n+1}, x_{n+1})$, and x_{n+1} is now determined implicitly.
- Newton method:

$$x_{n+1}^{(k+1)} = x_{n+1}^{(k)} + \delta x_{n+1}^{(k)},$$

  with iteration index k. The increment $\delta x_{n+1}^{(k)}$ is computed by solving:

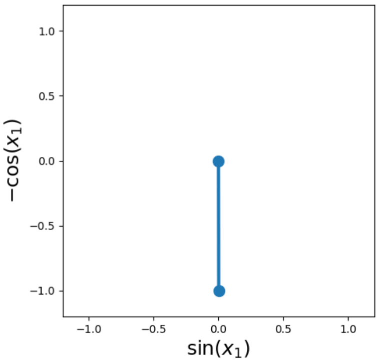
$$(\mathbb{1} - hJ_{n+1}^{(k)})\, \delta x_{n+1}^{(k)} = -r_{n+1}^{(k)}, \qquad r_{n+1}^{(k)} = x_{n+1}^{(k)} - x_n - hf(t_{n+1}, x_{n+1}^{(k)})$$

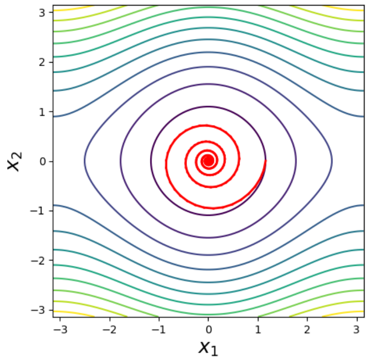
  with $J_{n+1}^{(k)} = \mathrm{D}f(t_{n+1}, x_{n+1}^{(k)})$ the Jacobian, and $r_{n+1}^{(k)}$ the residual
- the computational cost per step is larger than with the explicit Euler method.

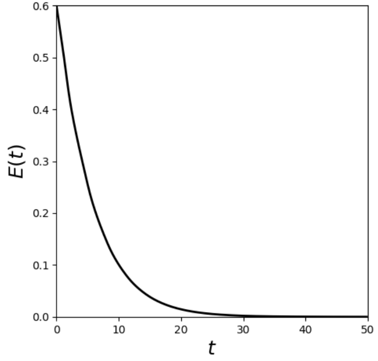
The Mathematical Pendulum with the Implicit Euler Method

- mathematical pendulum: two degrees of freedom $\{x_1, x_2\}$
- equations of motion

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\sin x_1, \qquad\qquad x_1(0) = \arccos(0.4), \qquad x_2(0) = 0$$







Runge-Kutta Methods

- use the *Fundamental Theorem of Calculus*

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} \dot{x}(t)\,dt \qquad \text{with} \qquad \dot{x}(t) = f(t, x(t))$$

- approximate the integral by some quadrature formula with s nodes c_i and corresponding weights b_i,

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i \dot{X}_{n,i}, \qquad \dot{X}_{n,i} = f(t_n + c_i h, X_{n,i}).$$

- the internal stage values $X_{n,i} \approx x(t_n + c_i h)$ are determined by another quadrature formula for the integral from 0 to c_i,

$$X_{n,i} = x_n + h \sum_{j=1}^{s} a_{ij} \dot{X}_{n,j} \approx x(t_n) + \int_{t_n}^{t_n + c_i h} \dot{x}(t)\,dt, \qquad i = 1, \dots, s,$$

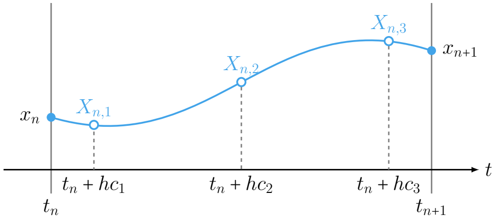
  with the same vector field values $\dot{X}_{n,j}$ used for x_{n+1}

Runge-Kutta Methods

- Runge-Kutta methods are numerical one-step methods

$$X_{n,i} = x_n + h \sum_{j=1}^{s} a_{ij}\, f(t_n + c_j h, X_{n,j}),$$

$$x_{n+1} = x_n + h \sum_{j=1}^{s} b_j\, f(t_n + c_j h, X_{n,j}),$$



defined by a set of nodes c_i, weights b_i and coefficients a_{ij}, summarised in the Butcher tableau

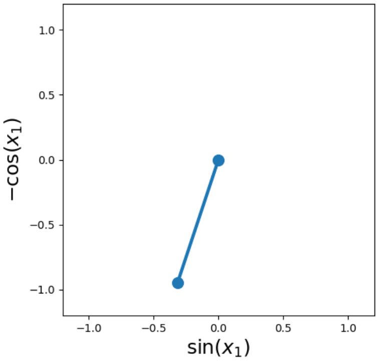
$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

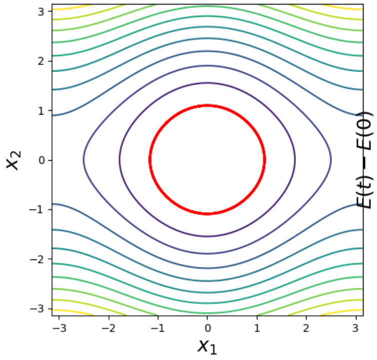
- most properties of the methods (e.g., order, stability) can be analysed just by conditions on the Butcher tableau!

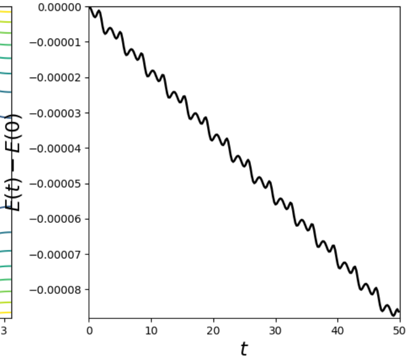
The Mathematical Pendulum with an Explicit Runge-Kutta Method (Order 4)

- mathematical pendulum: two degrees of freedom $\{x_1, x_2\}$
  (mass $m = 1$, massless rod of length $l = 1$, gravitational acceleration $g = 1$)
- equations of motion

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\sin x_1, \qquad\qquad x_1(0) = \arccos(0.4), \qquad x_2(0) = 0$$



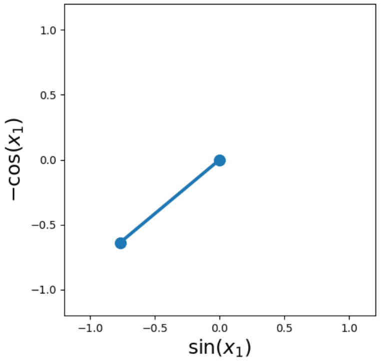


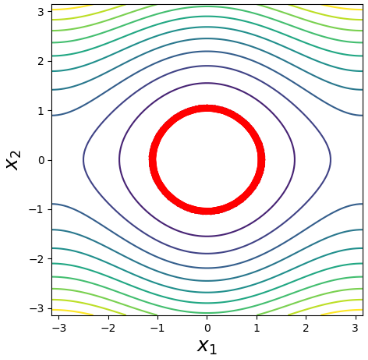


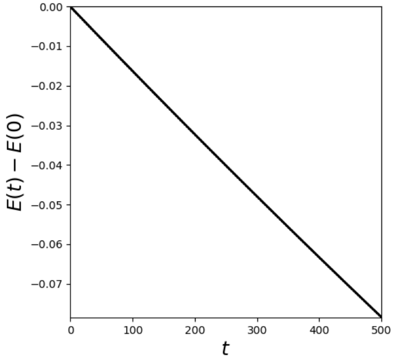
The Mathematical Pendulum with an Explicit Runge-Kutta Method (Order 4)

- mathematical pendulum: two degrees of freedom $\{x_1, x_2\}$ (mass $m = 1$, massless rod of length $l = 1$, gravitational acceleration $g = 1$)
- equations of motion

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\sin x_1, \qquad\qquad x_1(0) = \arccos(0.4), \qquad x_2(0) = 0$$







Hamiltonian Systems

Symplecticity and Hamiltonian Systems

- a canonical Hamiltonian system of ODEs in d dimensions is given by

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \qquad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}, \qquad i = 1, ..., d$$

- combining the dynamical variables into a vector $z = (q, p)$, we get

$$\Omega \dot{z} = \nabla H(z), \qquad \text{or} \qquad \dot{z} = \mathbb{J} \nabla H(z) \qquad \text{with} \qquad \mathbb{J} = \Omega^{-1}$$

where Ω and $\mathbb{J}$ are $2d \times 2d$ skew-symmetric matrices

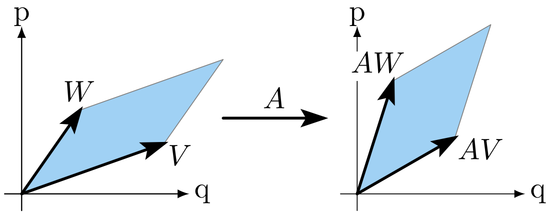
$$\Omega = \begin{pmatrix} \mathbb{0}_{d\times d} & -\mathbb{1}_{d\times d} \\ \mathbb{1}_{d\times d} & \mathbb{0}_{d\times d} \end{pmatrix}, \qquad \mathbb{J} = \begin{pmatrix} \mathbb{0}_{d\times d} & \mathbb{1}_{d\times d} \\ -\mathbb{1}_{d\times d} & \mathbb{0}_{d\times d} \end{pmatrix}$$

Symplectic Maps

- the flow ϕ_H of a Hamiltonian system is a symplectic map of the phasespace into itself

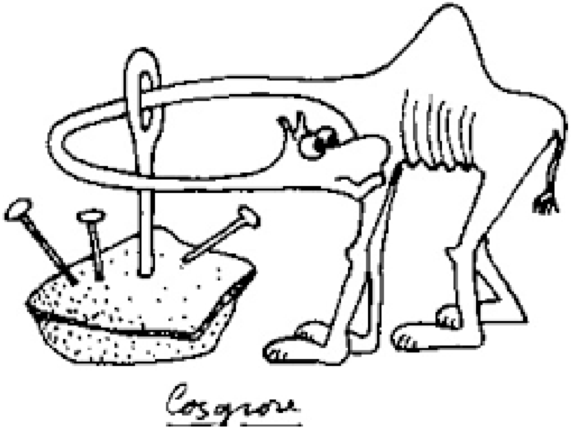
$$(p^1, q^1) = \phi_H(t_1, t_0)(p^0, q^0)$$

- a linear map $A : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ is called symplectic if $A^T \Omega A = \Omega$
- a nonlinear map $\phi : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ is called symplectic if $(\mathrm{D}\phi)^T \Omega (\mathrm{D}\phi) = \Omega$
- consequences: preservation of phasespace area (Liouville's theorem) as well as higher Poincaré integral invariants
- symplecticity dramatically restricts the number of dynamically accessible states compared to non-symplectic systems
- symplecticity is a characteristic property of Hamiltonian (and Lagrangian) systems

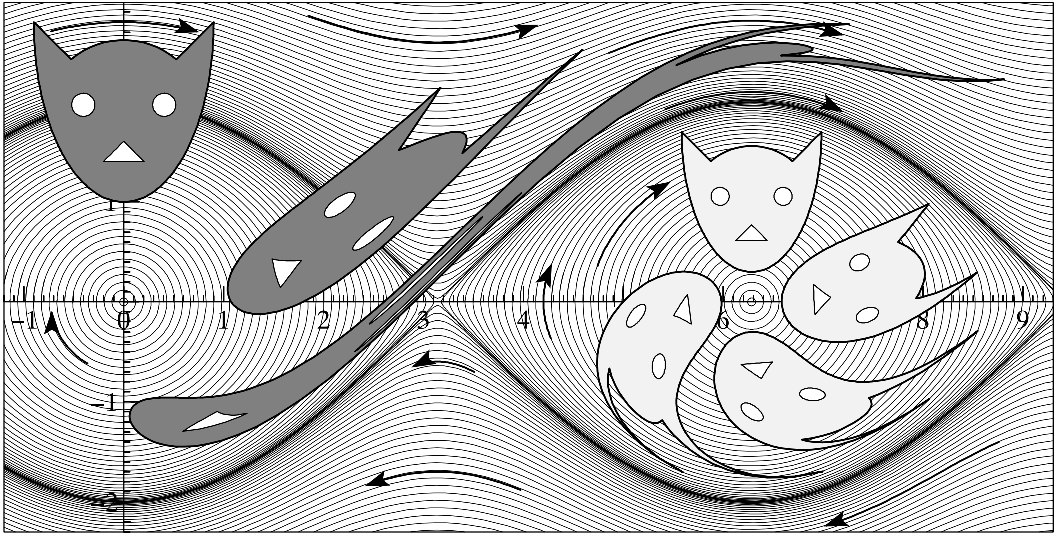


Gromov's non-squeezing theorem

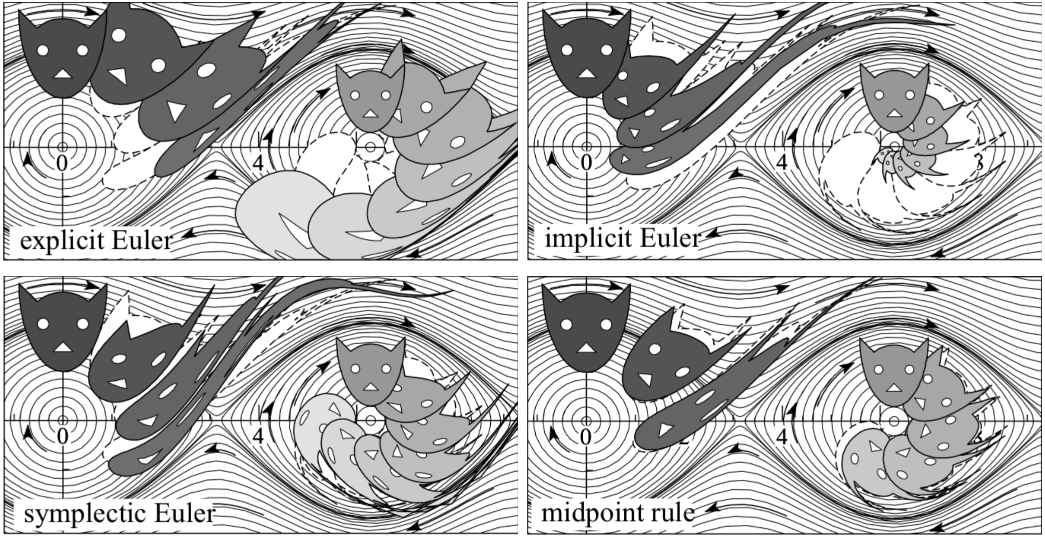
Theorem
A symplectic transformation cannot map a ball $B(R) = \{z \in \mathbb{R}^{2n} : ||z|| < R\}$ *into a cylinder* $Z(r) = \{z \in \mathbb{R}^{2n} : x_1^2 + y_1^2 < r^2\}$ *if* $r < R$.



Symplecticity and Hamiltonian Systems



Symplectic Integrators for Hamiltonian Systems

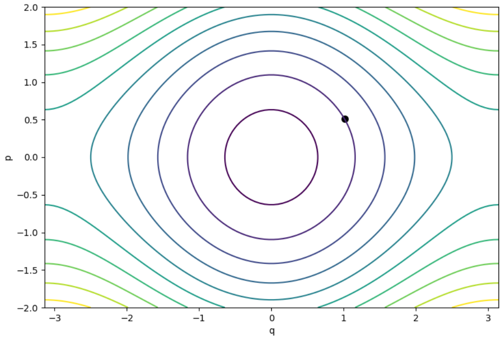


The Mathematical Pendulum as a Hamiltonian System

- Hamiltonian and equations of motion

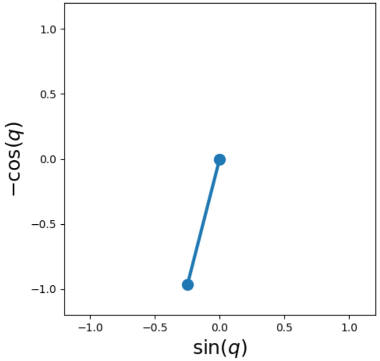
$$H(q,p) = \tfrac{1}{2}p^2 - \cos q, \qquad \dot{q} = \frac{\partial H}{\partial p}(q,p) = p, \qquad \dot{p} = -\frac{\partial H}{\partial q}(q,p) = -\sin q$$

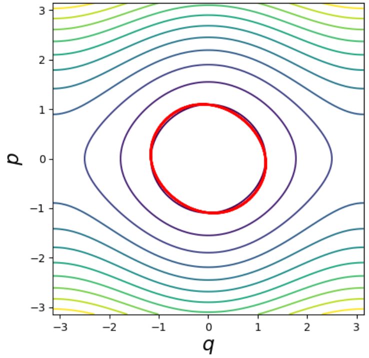
- solutions follow contour lines of the Hamiltonian (phase diagram)

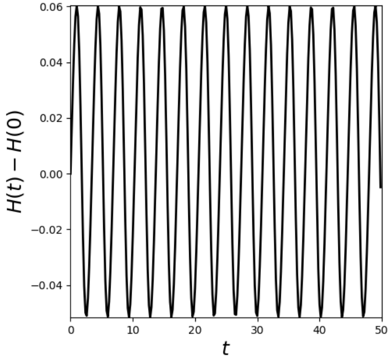


The Mathematical Pendulum (Symplectic Euler)

$$\frac{q_{k+1} - q_k}{h} = \frac{\partial H}{\partial p}(q_{k+1}, p_k), \qquad \frac{p_{k+1} - p_k}{h} = -\frac{\partial H}{\partial q}(q_{k+1}, p_k)$$

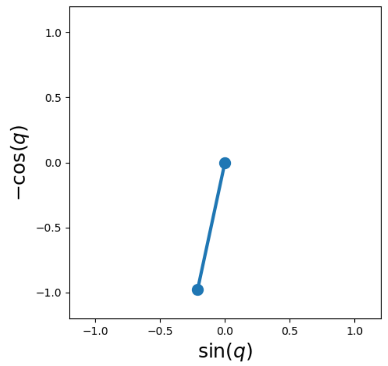


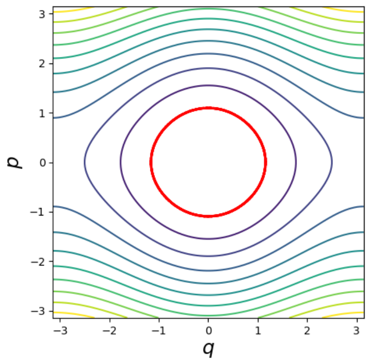


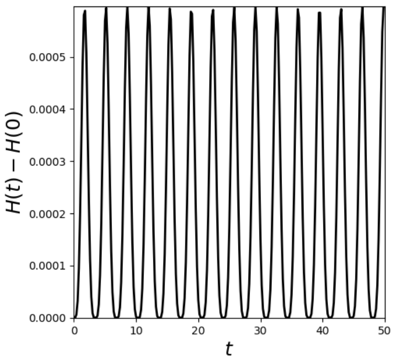


The Mathematical Pendulum (Implicit Midpoint)

$$\frac{q_{k+1} - q_k}{h} = \frac{\partial H}{\partial p}\left(\frac{q_k + q_{k+1}}{2}, \frac{p_k + p_{k+1}}{2}\right), \qquad \frac{p_{k+1} - p_k}{h} = -\frac{\partial H}{\partial q}\left(\frac{q_k + q_{k+1}}{2}, \frac{p_k + p_{k+1}}{2}\right)$$







Symplectic Integrators

Symplectic Integrators

- **Definition (Symplectic Integrator):** A numerical one-step method

  $$z_{n+1} = \phi_h(z_n)$$

  is called *symplectic* if, when applied to a smooth Hamiltonian system,

  $$\frac{dq^i}{dt} = \frac{\partial H}{\partial p_i}, \qquad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q^i}, \qquad i = 1, ..., d,$$

  the discrete flow

  $$z \mapsto \phi_h(z), \qquad z = (q, p),$$

  is a symplectic transformation for all sufficiently small step sizes h.
- symplectic integrators ϕ_h preserve the symplectic structure $\omega = \mathsf{d}q^i \wedge \mathsf{d}p_i = \frac{1}{2}\Omega_{ij}\, \mathsf{d}z^i \wedge \mathsf{d}z^j$, i.e.,

  $$(\mathrm{D}\phi_h)^T\, \Omega\, (\mathrm{D}\phi_h) = \Omega, \qquad \phi_h^* \omega = \omega, \qquad \mathsf{d}q_{n+1} \wedge \mathsf{d}p_{n+1} = \mathsf{d}q_n \wedge \mathsf{d}p_n$$
- backward error analysis:
  - there exists a nearby Hamiltonian $\bar{H}$ that is exactly preserved by the symplectic integrator
  - the difference between H and $\bar{H}$, i.e., the energy error, is bounded for exponential long times

Symplectic Integrators for Hamiltonian Systems

- Symplectic Euler A

$$\begin{aligned} q_{n+1} &= q_n + h\,H_p(q_{n+1}, p_n) && \text{(implicit Euler for } q\text{)}, \\ p_{n+1} &= p_n - h\,H_q(q_{n+1}, p_n) && \text{(explicit Euler for } p\text{)} \end{aligned}$$

- Symplectic Euler B

$$\begin{aligned} p_{n+1} &= p_n - h\,H_q(q_n, p_{n+1}) && \text{(implicit Euler for } p\text{)}, \\ q_{n+1} &= q_n + h\,H_p(q_n, p_{n+1}) && \text{(explicit Euler for } q\text{)} \end{aligned}$$

- explicit proof of symplecticity: verify

$$\mathsf{d}q_{n+1} \wedge \mathsf{d}p_{n+1} = \mathsf{d}q_n \wedge \mathsf{d}p_n \qquad \text{or} \qquad (\mathrm{D}\phi_h)^T\,\Omega\,(\mathrm{D}\phi_h) = \Omega$$

  by using the implicit function theorem to compute $\mathrm{D}\phi_h$,

$$\left(\frac{\partial f}{\partial z_{n+1}}\right)\left(\frac{\partial z_{n+1}}{\partial z_n}\right) + \left(\frac{\partial f}{\partial z_n}\right) = 0 \qquad \Rightarrow \qquad \mathrm{D}\phi_h = \left(\frac{\partial z_{n+1}}{\partial z_n}\right) = -\left(\frac{\partial f}{\partial z_{n+1}}\right)^{-1}\left(\frac{\partial f}{\partial z_n}\right)$$

Symplectic Integrators for Hamiltonian Systems

- Störmer-Verlet methods

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{h}{2} H_q(q_n, p_{n+1/2}), & q_{n+1} &= q_{n+1/2} + \frac{h}{2} H_p(q_{n+1}, p_{n+1/2}), \\ q_{n+1/2} &= q_n + \frac{h}{2} H_p(q_n, p_{n+1/2}), & p_{n+1} &= p_{n+1/2} - \frac{h}{2} H_q(q_{n+1}, p_{n+1/2}) \end{aligned}$$

and

$$\begin{aligned} q_{n+1/2} &= q_n + \frac{h}{2} H_p(q_{n+1/2}, p_n), & p_{n+1} &= p_{n+1/2} - \frac{h}{2} H_q(q_{n+1/2}, p_{n+1}), \\ p_{n+1/2} &= p_n - \frac{h}{2} H_q(q_{n+1/2}, p_n), & q_{n+1} &= q_{n+1/2} + \frac{h}{2} H_p(q_{n+1/2}, p_{n+1}) \end{aligned}$$

- symmetric, symplectic methods of order 2, explicit for separable Hamiltonians
- compositions of the two symplectic Euler methods $\varphi^A_{h/2} \circ \varphi^B_{h/2}$ and $\varphi^B_{h/2} \circ \varphi^A_{h/2}$

Symplectic Integrators for Hamiltonian Systems

- Runge-Kutta method with s internal stages $(Q_{n,i}, P_{n,i})$ for a Hamiltonian system

$$
\begin{aligned}
\dot{Q}_{n,i} &= \frac{\partial H}{\partial p}(Q_{n,i}, \dot{Q}_{n,i}), & Q_{n,i} &= q_n + h\sum_{j=1}^{s} a_{ij}\dot{Q}_{n,j}, & q_{n+1} &= q_n + h\sum_{j=1}^{s} b_j\dot{Q}_{n,j}, \\
\dot{P}_{n,i} &= -\frac{\partial H}{\partial q}(Q_{n,i}, \dot{Q}_{n,i}), & P_{n,i} &= p_n + h\sum_{j=1}^{s} a_{ij}\dot{P}_{n,j}, & p_{n+1} &= p_n + h\sum_{j=1}^{s} b_j\dot{P}_{n,j}
\end{aligned}
$$

→ the method is symplectic if the coefficients satisfy $b_i a_{ij} + b_j a_{ji} = b_i b_j$ (caveat: always implicit).

Reduced Complexity Modelling

Motivation: Parametric PDEs and Solution Manifolds

- multi-query contexts (optimisation, inverse problems, control, ...) require the repeated solution of parametric partial differential equations
- denote the parameter space by $\mathbb{P} \subset \mathbb{R}^p$ and the solution space by V
- parametrised ODE problem (e.g. semi-discretised PDE) for $z \in V$ and $\mu \in \mathbb{P}$

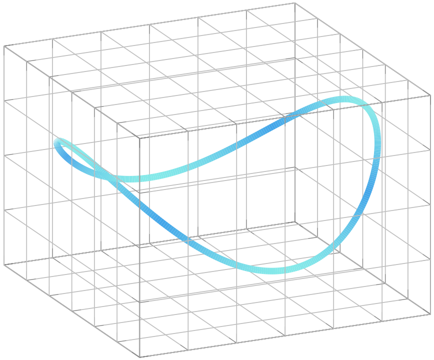
$$F(z(\mu); \mu) = 0$$

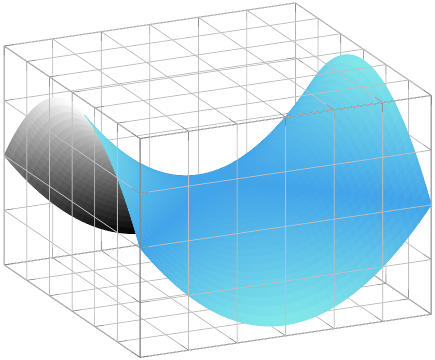
- numerical algorithms seek approximate solutions $z_h \approx z$ in finite-dimensional spaces $V_h \approx V$; typically z_h is represented by a degree-of-freedom vector $\hat{z} \in \mathbb{R}^{N_h} \simeq V_h$ where $N_h = \dim V_h$
- with traditional numerical methods, the space V_h is typically not adapted to the problem and therefore needs to be rather large, resulting in high computational costs
- the actual solution manifold $\mathcal{M}$ is a much smaller space

$$\mathcal{M} = \{ z(\mu) \in V : F(z(\mu); \mu) = 0, \ \mu \in \mathbb{P} \} \subset V_h$$

Motivation: Parametric PDEs and Solution Manifolds

- the solution manifold $\mathcal{M} = \{z(\mu) \in V : F(z(\mu);\mu) = 0,\ \mu \in \mathbb{P}\} \subset V$ is a very small space





- goal: construct an approximation of the solution manifold $\mathcal{M}_h$ and the embedding map $\mathcal{M}_h \hookrightarrow V_h$

Data-driven Model Order Reduction

- Strategy: Learn a low-dimensional representation of a system that captures relevant physical properties
- from a dataset M of solutions $\hat{z}(\mu)$ for different values of the parameter μ construct
  - a mapping $\mathcal{P}$ from V_h to the low-dimensional space V_r (reduction)
  - a mapping $\mathcal{R}$ from the low-dimensional space V_r to V_h (reconstruction)
  - a reduced representation $\tilde{z} \in V_r$ such that $\mathcal{R}\tilde{z}(\mu) \approx \hat{z}(\mu)$ and $\dim(V_r) \ll \dim(V_h)$
  - a reduced system of equations $\tilde{F}(\tilde{z}(\mu); \mu) = 0$
- the mappings $\mathcal{P}$ and $\mathcal{R}$ are chosen such that they minimize the reconstruction error:

$$\min_{\mathcal{P},\mathcal{R}} \frac{1}{2} ||\mathsf{M} - \mathcal{R} \circ \mathcal{P}(\mathsf{M})||^2$$

- in order to obtain accurate reduced oder models, important properties of the high order model, such as symplecticity or conservation of invariants, need to be accounted for in the construction of $\mathcal{P}$, $\mathcal{R}$ and $\tilde{F}$

Reduced Basis Methods

- find a small set of reduced basis functions $\{\zeta_i\}_{i=1}^{n}$ and write reduced representation of solutions as

$$\tilde{z}(\mu) = \sum_{i=1}^{n} \tilde{z}_i(\mu)\, \zeta_i$$

→ How can we construct such a set of reduced basis vectors?

  - Proper orthogonal decomposition selects the eigenvectors of the empirical correlation operator of solution snapshots for different values of the parameters μ obtained from a high fidelity integrator
  - Offline phase: limited number of simulations with high fidelity method and computation of reduced basis
  - Online phase: many (cheap) simulations with reduced basis

→ Other approaches:

  - Autoencoders, a special type of neural network architecture, are designed to map a high dimensional space to a low dimensional feature space (intrinsic manifold)
  - ...

Proper Orthogonal Decomposition

- collect snapshots $\{\hat{z}^{(j)} = \hat{z}(\mu_j)\}_{j=1}^{n_s} \subset V_h$ of solutions for $\mu_j \in \mathbb{P}$ and compose a snapshot matrix

$$\mathsf{S} = \left[\hat{z}^{(1)} \mid \ldots \mid \hat{z}^{(n_s)}\right] \in \mathbb{R}^{N_h \times n_s}$$

- singular value decomposition of the snapshot matrix $\mathsf{S} = \mathsf{V}\Sigma\mathsf{Z}^T$ yields orthonormal ζ_i as columns of V
- discrete solutions are approximated as linear combinations of the first n eigenvectors ζ_i

$$\tilde{z}(\mu) = \sum_{i=1}^{n} \tilde{z}_i(\mu)\,\zeta_i, \qquad \mathsf{V}^T = \begin{pmatrix} \zeta_{1,1} & \cdots & \zeta_{1,n} \\ \vdots & & \vdots \\ \zeta_{n,1} & \cdots & \zeta_{n,n} \end{pmatrix}, \qquad \zeta_i = \begin{pmatrix} \zeta_{i,1} \\ \vdots \\ \zeta_{i,n} \end{pmatrix}$$

- truncating $\mathsf{V} = \left[\zeta_1 \mid \ldots \mid \zeta_n\right]$ yields the reduced basis as well as the reconstruction and reduction operators $\mathcal{R} = \mathsf{V}$ and $\mathcal{P} = \mathsf{V}^T$ such that the reconstruction error satisfies

$$\sum_{i=1}^{n_s} \frac{1}{2} ||z^{(i)} - \mathsf{V}\mathsf{V}^T z^{(i)}||^2 = \text{minimum among all } n\text{-dimensional orthogonal bases}$$

Galerkin Projection

- recall the abstract form of Hamilton's equations

$$\dot{z} = \mathbb{J} \nabla H(z) \qquad \text{with} \qquad \mathbb{J} = \begin{pmatrix} \mathbb{0}_{d \times d} & \mathbb{1}_{d \times d} \\ -\mathbb{1}_{d \times d} & \mathbb{0}_{d \times d} \end{pmatrix}$$

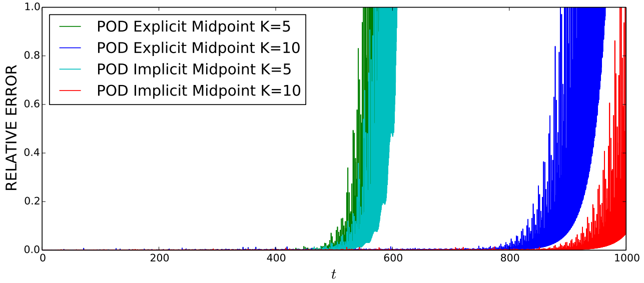
- replacing $\hat{z} = (\hat{q}, \hat{p})^T \in \mathbb{R}^{2N_p}$ with the reduced basis representation $\mathsf{V}\tilde{z} \in \mathbb{R}^{2n}$ yields a system of $2N_p$ equations for $2n$ degrees-of-freedom with $n \ll N_p$

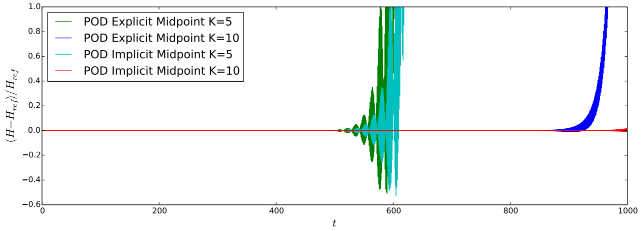
$$\mathsf{V}\dot{\tilde{z}} = \mathbb{J} \nabla H(\mathsf{V}\tilde{z})$$

- Galerkin projection with V^T yields a (dense) system of $2n$ equations (note that $\mathsf{V}^T\mathsf{V} = \mathbb{I}$)

$$\dot{\tilde{z}} = \mathsf{V}^T \mathbb{J} \nabla H(\mathsf{V}\tilde{z})$$

Proper Orthogonal Decomposition





Proper Symplectic Decomposition

- Proper Symplectic Decomposition constraints the possible matrices to a subset of the symplectic lifts

$$\min_{\mathsf{V}} \frac{1}{2} \left|\left|\mathsf{S} - \mathsf{V}\mathsf{V}^T\mathsf{S}\right|\right|^2 \qquad \text{with} \qquad \mathsf{V} = \begin{pmatrix} \mathsf{A} & \mathbb{0} \\ \mathbb{0} & \mathsf{A} \end{pmatrix}, \qquad \mathsf{S} = \left[\mathsf{S}_q \,\middle|\, \mathsf{S}_p\right]$$

→ A consists of the first n columns of V for $\left[\mathsf{S}_q \,\middle|\, \mathsf{S}_p\right] = \mathsf{V}\Sigma\mathsf{Z}^T$

- Galerkin projection with the symplectic inverse $\mathsf{V}^+ = \mathbb{J}_{2n}\mathsf{V}^T\mathbb{J}_{2N}^T$ again yields a Hamiltonian system

$$\dot{\tilde{z}} = \mathbb{J}_{2n}\nabla\tilde{H}(\tilde{z}) \qquad \text{with} \qquad \tilde{H}(\tilde{z}) = H(\mathsf{V}\tilde{z})$$

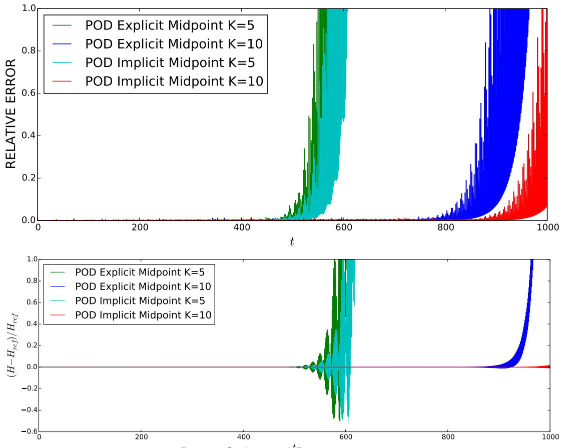
in detail:

$$\dot{\tilde{z}} = \mathsf{V}^+\mathbb{J}_{2N}\nabla H(\mathsf{V}\tilde{z}) = \mathbb{J}_{2n}\mathsf{V}^T\mathbb{J}_{2N}^T\mathbb{J}_{2N}\nabla H(\mathsf{V}\tilde{z}) = \mathbb{J}_{2n}\mathsf{V}^T\nabla H(\mathsf{V}\tilde{z}) = \mathbb{J}_{2n}\nabla\tilde{H}(\tilde{z})$$

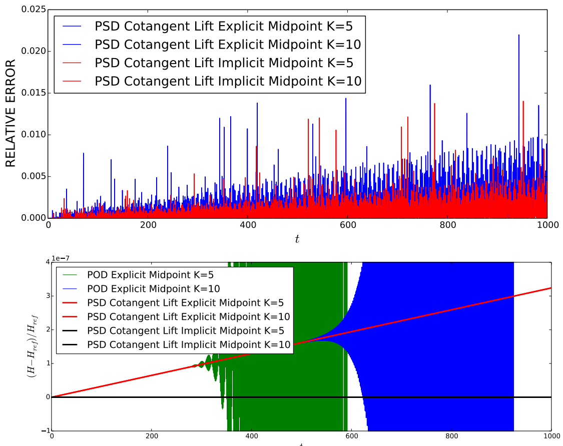
- applying a symplectic integrator on the low-dimensional PSD system yields a discrete symplectic flow

H. Peng, N. Song, Z. Kan. Data-driven model order reduction with proper symplectic decomposition for flexible multibody system. Nonlinear Dynamics 107.1, pp. 173–203, 2022.

POD vs. PSD



Proper Orthogonal Decomposition



Proper Symplectic Decomposition

Structure-preserving Hyper-reduction

- challenge: structure-preserving reduction of nonlinear operators (hyper-reduction)
- evaluation of the Hamiltonian H is expensive due to the reconstruction of the high-order solution

$$\dot{\tilde{z}} = \mathbb{J}_{2n}\mathsf{V}^T\nabla\tilde{H}(\tilde{z}) \qquad \text{with e.g.} \qquad \tilde{H}(\tilde{z}) = H(\mathsf{V}\tilde{z}) = \frac{1}{2}\tilde{p}^T\tilde{\mathsf{M}}\tilde{p} + \phi(\mathsf{V}\tilde{q}), \qquad \tilde{\mathsf{M}} = \mathsf{V}^T\mathsf{M}\mathsf{V}$$

- standard hyper-reduction methods like Discrete Empirical Interpolation Method (DEIM) or Dynamic Mode Decomposition (DMD) do not account for symplectic structure of the vector field

$$\dot{\tilde{z}} = \mathbb{J}_{2n}\mathsf{V}^T\Pi_{\mathsf{DEIM}}\nabla H(\mathsf{I}_{\mathsf{DEIM}}\mathsf{V}\tilde{z})$$

- possible solutions:
  - do not perform hyper-reduction of the vector field but on the Hamiltonian

$$\tilde{H}_{\mathsf{DEIM}}(\tilde{z}) = H(\Pi_{\mathsf{DEIM}}^T\mathsf{V}\tilde{z}) \qquad \text{i.e.} \qquad \mathsf{I}_{\mathsf{DEIM}} = \Pi_{\mathsf{DEIM}}^T$$

  - approximate $H(\mathsf{V}\tilde{z})$ by a Hamiltonian Neural Network $H_{\mathsf{HNN}}(\tilde{z})$ or SINDy Hamiltonian $H_{\mathsf{SINDy}}(\tilde{z})$
  - replace standard symplectic integrator on reduced vector field with a SympNet

Scientific Machine Learning

Neural Networks

- neural networks are nothing more than a specific nonlinear map

$$\Psi(x) = \psi_n \circ \ldots \circ \psi_2 \circ \psi_1(x)$$

- given an input x, the action of a feed-forward neural network can be summarised as

$$\begin{aligned} a^{[1]} &= x \in \mathbb{R}^{n_1}, \\ a^{[l]} &= \sigma\left(W^{[l]} a^{[l-1]} + b^{[l]}\right) \in \mathbb{R}^{n_l}, \qquad \text{for } l = 2, 3, ..., L, \\ y &\approx a^{[L]} \in \mathbb{R}^{n_L} \end{aligned}$$

- the activation function σ is a nonlinear function with simple derivative, e.g., sigmoid or $\tanh$
- the parameter $\theta = (W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]})$ are determined by minimising a cost function of the form

$$\mathcal{L}_{NN} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| y(x^{\{i\}}) - a^{[L]}(x^{\{i\}}) \right\|_2^2$$

e.g. using some version of gradient descent

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta) \qquad \text{with } \eta \text{ the learning rate}$$

Autoencoders

- autoencoders are a special type of neural network architectures, designed to map a high dimensional space (the data) to a low dimensional feature space (intrinsic manifold)
- they consist of an encoder $\Psi^{\mathrm{enc}}_{\theta_1} : \mathbb{R}^N \to \mathbb{R}^n$ and a decoder $\Psi^{\mathrm{dec}}_{\theta_2} : \mathbb{R}^n \to \mathbb{R}^N$, both of which are neural networks, parametrized by θ_1 and θ_2 respectively
- both are trained simultaneously on a data set M to minimize the reconstruction error

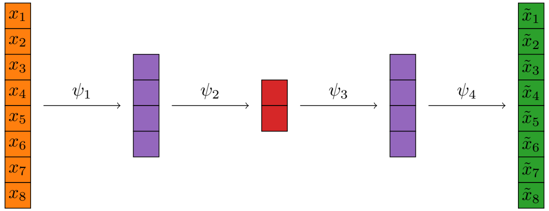
$$\mathcal{L}_{\mathsf{AE}}(\theta) := \frac{1}{2}||\mathsf{M} - \Psi^{\mathrm{dec}}_{\theta_2} \circ \Psi^{\mathrm{enc}}_{\theta_1}(\mathsf{M})||^2$$

- application: autoencoders can be used for model reduction in a similar fashion as reduced basis methods providing nonlinear solution spaces
- symplectic autoencoders can be constructed similar to Proper Symplectic Decomposition

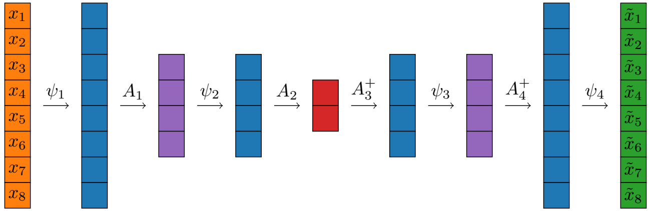
S. Fresca, L. Dede, A. Manzoni (2021). A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. Journal of Scientific Computing 87.2, pp. 1–36.

K. Lee and K. T. Carlberg (2020). Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. Journal of Computational Physics 404, p. 108973.

Symplectic Autoencoders: Architecture



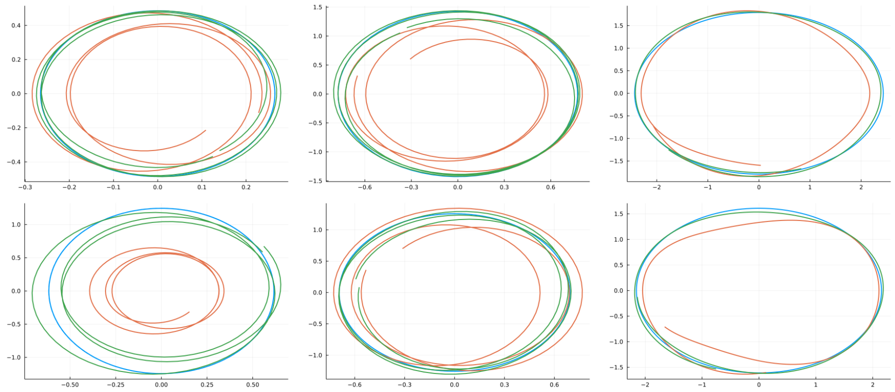
Standard Autoencoder:
ψ_i are arbitrary neural network layers that change dimension but do not respect the structure of the system.



Symplectic Autoencoder:
ψ_i are nonlinear dimension-preserving layers and A_i are linear PSD-like maps. All of these building blocks are symplectic maps.

Symplectic Autoencoders: Example

- Reduction of a 12-dimensional systems of coupled oscillators to 4d



Analytic solution, Proper Symplectic Decomposition, Symplectic Autoencoder.

Each plot shows the reconstructed solution of one oscillator.

Hamiltonian and Lagrangian Neural Networks

- when classical Neural Networks are used to fit the solution of a differential equation, stability cannot be ensured and errors grow exponentially fast (similar to non-structure-preserving reduced basis methods)
- alternative approach: instead of the vector field learn Hamiltonian and use the trained network H_θ in conjunction with symplectic integrators (HNNs)

$$\mathcal{L}_{\text{HNN}} = \left\| \frac{\partial H_\theta}{\partial p} - \dot{q}^t \right\|_2 + \left\| \frac{\partial H_\theta}{\partial q} + \dot{p}^t \right\|_2$$

- if vector field data $(\dot{q}^t, \dot{p}^t)$ is not available a symplectic integrator (e.g. symplectic Euler) on time series of the solution (q^t, p^t) can be used instead

$$\mathcal{L}_{\text{HNN}} = \sum_t \sum_{n=1}^{N} \left\| \frac{\partial H_\theta}{\partial p}\left(q_n^t, p_{n+1}^t\right) - \frac{q_{n+1}^t - q_n^t}{\Delta t} \right\|_2 + \sum_t \sum_{n=1}^{N} \left\| \frac{\partial H_\theta}{\partial q}\left(q_n^t, p_{n+1}^t\right) + \frac{p_{n+1}^t - p_n^t}{\Delta t} \right\|_2$$

- application: symplectic integrators applied to HNNs provide non-intrusive, structure-preserving time integration methods for reduced systems

S. Greydanus, M. Dzamba, J. Yosinski. Hamiltonian Neural Networks. 33rd Conference on Neural Information Processing Systems, NeurIPS 2019.

Sparse Identification of Nonlinear Dynamics (SINDy)

- parametrise the Hamiltonian by a dictionary of basis functions φ_i (e.g., monomials, trigonometric functions) of the dynamical variables

$$H_\theta(q,p) = \sum_i \theta_i \, \varphi_i(q,p)$$

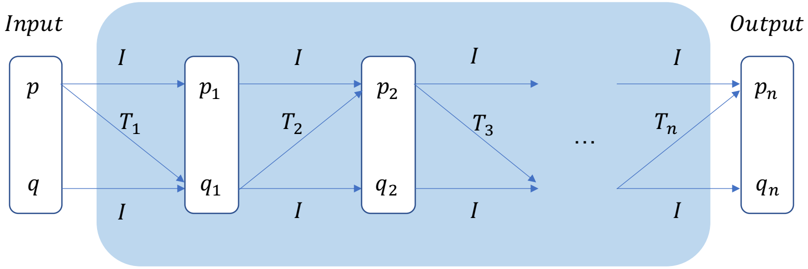
$$\mathcal{L}_{\mathsf{HSINDy}} = \sum_t \sum_{n=1}^{N} \left\| \frac{\partial H_\theta}{\partial p}\left(\frac{q_n^t + q_{n+1}^t}{2}, \frac{p_n^t + p_{n+1}^t}{2}\right) - \frac{q_{n+1}^t - q_n^t}{\Delta t} \right\|_2 + \sum_t \sum_{n=1}^{N} \left\| \frac{\partial H_\theta}{\partial q}\left(\frac{q_n^t + q_{n+1}^t}{2}, \frac{p_n^t + p_{n+1}^t}{2}\right) + \frac{p_{n+1}^t - p_n^t}{\Delta t} \right\|_2$$

- sparsification: multiple optimization cycles, after each of which parameters below a certain threshold are removed
- application: symplectic applied to SINDy Hamiltonians provide non-intrusive, structure-preserving time integration methods for reduced systems

Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems, PNAS, 2016.

SympNets

- SympNets can approximate arbitrary canonical symplectic maps
- universal model representing the solution of Hamiltonian systems
- no traditional symplectic integration methods required (in contrast to HNNs)
- the SympNet itself provides a time-stepping method for a Hamiltonian system

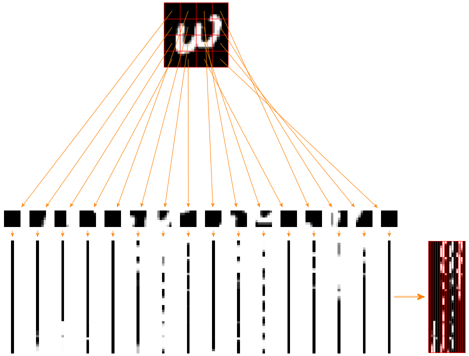


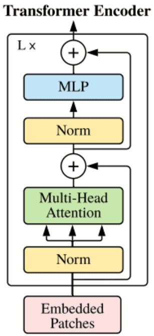
- application: non-intrusive, structure-preserving time integration of reduced systems

P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. E. Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. Neural Networks 132, pp. 166-179, 2020.

Transformers

- Transformers were originally devised for Natural Language Processing (NLP).
- They have gradually replaces LSTMs and other recurrent neural network architectures for these kinds of data and are competing with CNNs in image processing.



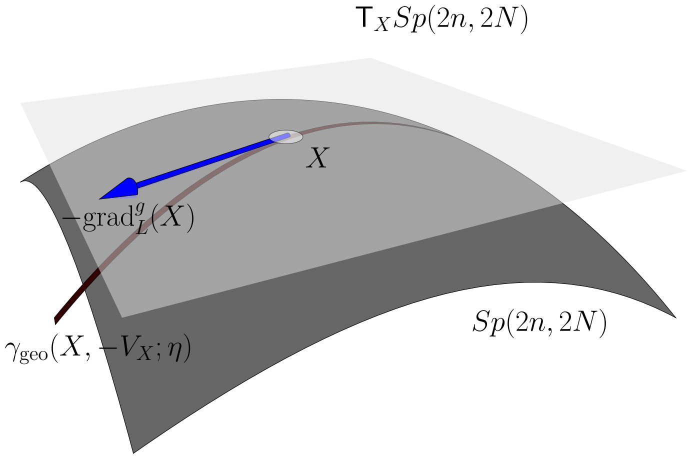


Multihead Attention

- Can find correlations in the input data.
- $\mathrm{MHA} : X \mapsto [P_1^V X \sigma((P_1^K X)^T P_1^Q X), \ldots, P_{N_h}^V X \sigma((P_{N_h}^K X)^T P_{N_h}^Q X)]$, where the P^V, P^K and P^Q are *projection matrices*.
- More concisely a MHA layer performs N_h single head attention operations: $\mathrm{SHA} : (V, K, Q) \mapsto V\sigma(K^T Q)$, where $\sigma = \mathrm{softmax}$. Therefore:

$$\mathrm{MHA}(X) = [\mathrm{SHA}(X_1^V, X_1^K, X_1^Q), \ldots, \mathrm{SHA}(X_{N_h}^V, X_{N_h}^K, X_{N_h}^Q)] \text{ and } X_i^V = P_i^V X \text{ etc.}$$

Optimization on Manifolds


$\mathsf{T}_X Sp(2n, 2N)$
X
$-\mathrm{grad}^g_L(X)$
$\gamma_{\mathrm{geo}}(X, -V_X; \eta)$
$Sp(2n, 2N)$

Summary

Summary

Preservation of structure and physics constraints is crucial for stability, robustness and correctness of numerical integrators and reduced models

Solution Space

- high fidelity
- reduced basis
- symplectic reduced basis
- autoencoder
- symplectic autoencoder

Time Integration

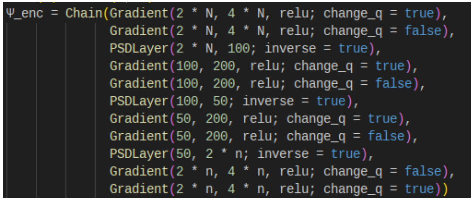
- symplectic or variational integrator
- HNNs / LNNs / SINDy for nonlinear operators
- HNNs / LNNs / SINDy for collective dynamics
- SympNets / Symplectic Transformer NNs

Julia packages

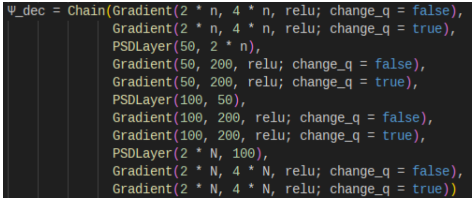
- `GeometricIntegrators`
  `https://github.com/JuliaGNI/GeometricIntegrators.jl`,
- `GeometricMachineLearning`
  `https://github.com/JuliaGNI/GeometricMachineLearning.jl`
- `ReducedBasisMethods`
  `https://github.com/JuliaRCM/ReducedBasisMethods.jl`

GeometricIntegrators

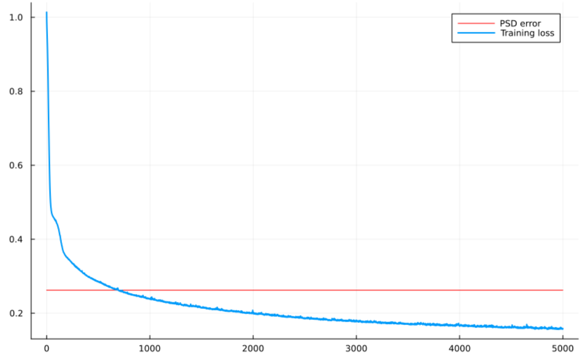
GeometricMachineLearning



```
Ψ_enc = Chain(Gradient(2 * N, 4 * N, relu; change_q = true),
              Gradient(2 * N, 4 * N, relu; change_q = false),
              PSDLayer(2 * N, 100; inverse = true),
              Gradient(100, 200, relu; change_q = true),
              Gradient(100, 200, relu; change_q = false),
              PSDLayer(100, 50; inverse = true),
              Gradient(50, 200, relu; change_q = true),
              Gradient(50, 200, relu; change_q = false),
              PSDLayer(50, 2 * n; inverse = true),
              Gradient(2 * n, 4 * n, relu; change_q = false),
              Gradient(2 * n, 4 * n, relu; change_q = true))
```

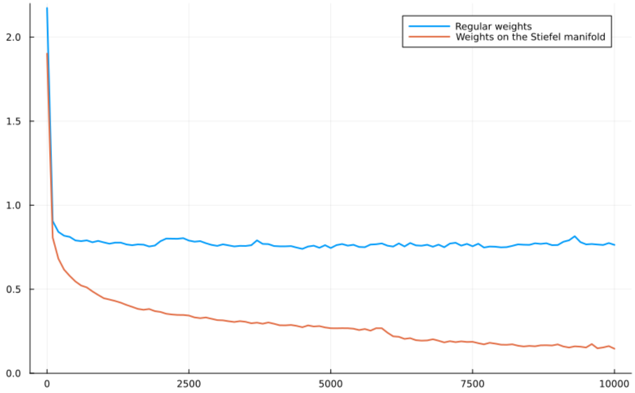


```
Ψ_dec = Chain(Gradient(2 * n, 4 * n, relu; change_q = false),
              Gradient(2 * n, 4 * n, relu; change_q = true),
              PSDLayer(50, 2 * n),
              Gradient(50, 200, relu; change_q = false),
              Gradient(50, 200, relu; change_q = true),
              PSDLayer(100, 50),
              Gradient(100, 200, relu; change_q = false),
              Gradient(100, 200, relu; change_q = true),
              PSDLayer(2 * N, 100),
              Gradient(2 * N, 4 * N, relu; change_q = false),
              Gradient(2 * N, 4 * N, relu; change_q = true))
```



GeometricMachineLearning II

1. model = Lux.Chain(Transformer(patch_length2, n_heads, L, Stiefel = false), Classification(patch_length2, 10)),
2. model = Lux.Chain(Transformer(patch_length2, n_heads, L, Stiefel = true), Classification(patch_length2, 10))


Regular weights
Weights on the Stiefel manifold