PHD ORAL DEFENSE · NATIONAL UNIVERSITY OF SINGAPORE · 2026

Machine Learning Augmented Simulation and Analysis of Dynamical Systems

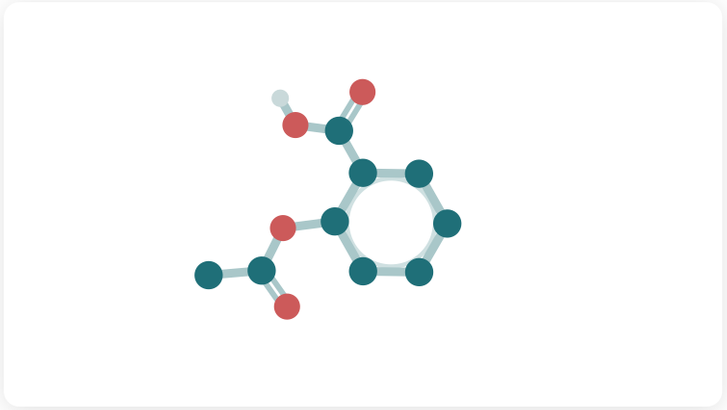
Presenter: Guo Yue

Supervisor: Li Qianxiao

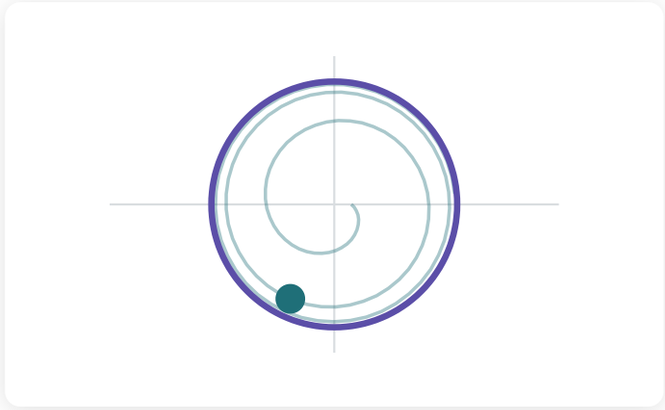
Department of Mathematics

What is a dynamical system?

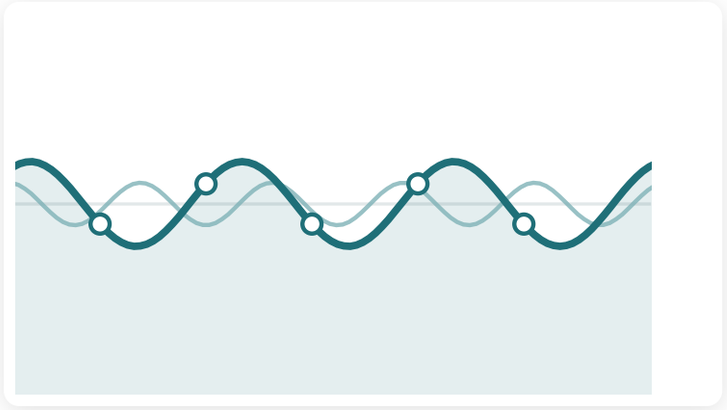
A **state that evolves in time** — four everyday examples, all in motion:



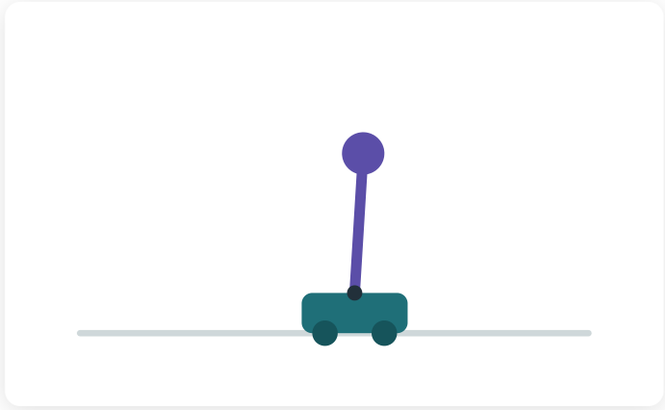
molecule · molecular dynamics



oscillator · limit cycle



wave · transport



pendulum · with control

Dynamics = choosing a computable form

molecule · oscillator · wave · pendulum · control

All are **states evolving in time**

The equation

the evolution law

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$$

The representation

coordinates
we track in

The scheme

step size & solver

Right form → the trajectory computes **accurately, stably, cheaply**;
Wrong form → it drifts or blows up.

Classically, every dial is set by hand

Each form built **from first principles, with a theorem attached**:

The equation

first principles

$$m\ddot{\boldsymbol{x}} = -\nabla V$$

exact · interpretable

The representation

human insight

normal modes

decoupled

The scheme

hand-designed

Euler · RK · symplectic

order p · stable

Strict and guaranteed: hand-derivable,unknown, or too expensive.

When the form is not easily hand-writable, we learn it

Only data align with trajectories

we watch the system move
but the **equation is unknown**

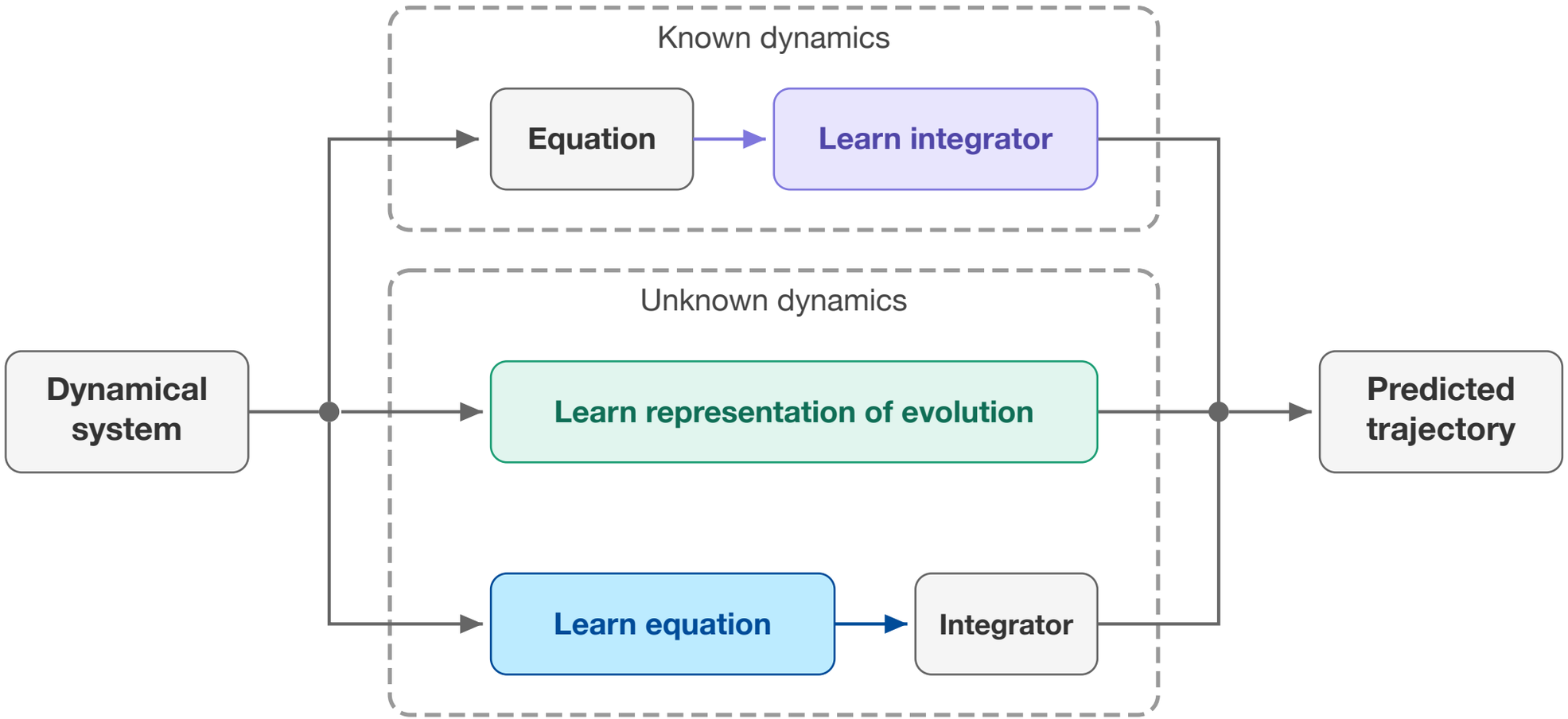
Computation costs too much

each force is a **quantum-scale calculation** (molecular dynamics)

Learning replaces a hand-designed form with
one **parameterized flexibly and fit from data**.

Bring that power into **scientific computing**, **without giving up the classical guarantees** → so **which dial do we learn?** →

Thesis viewpoint: learn the structured components



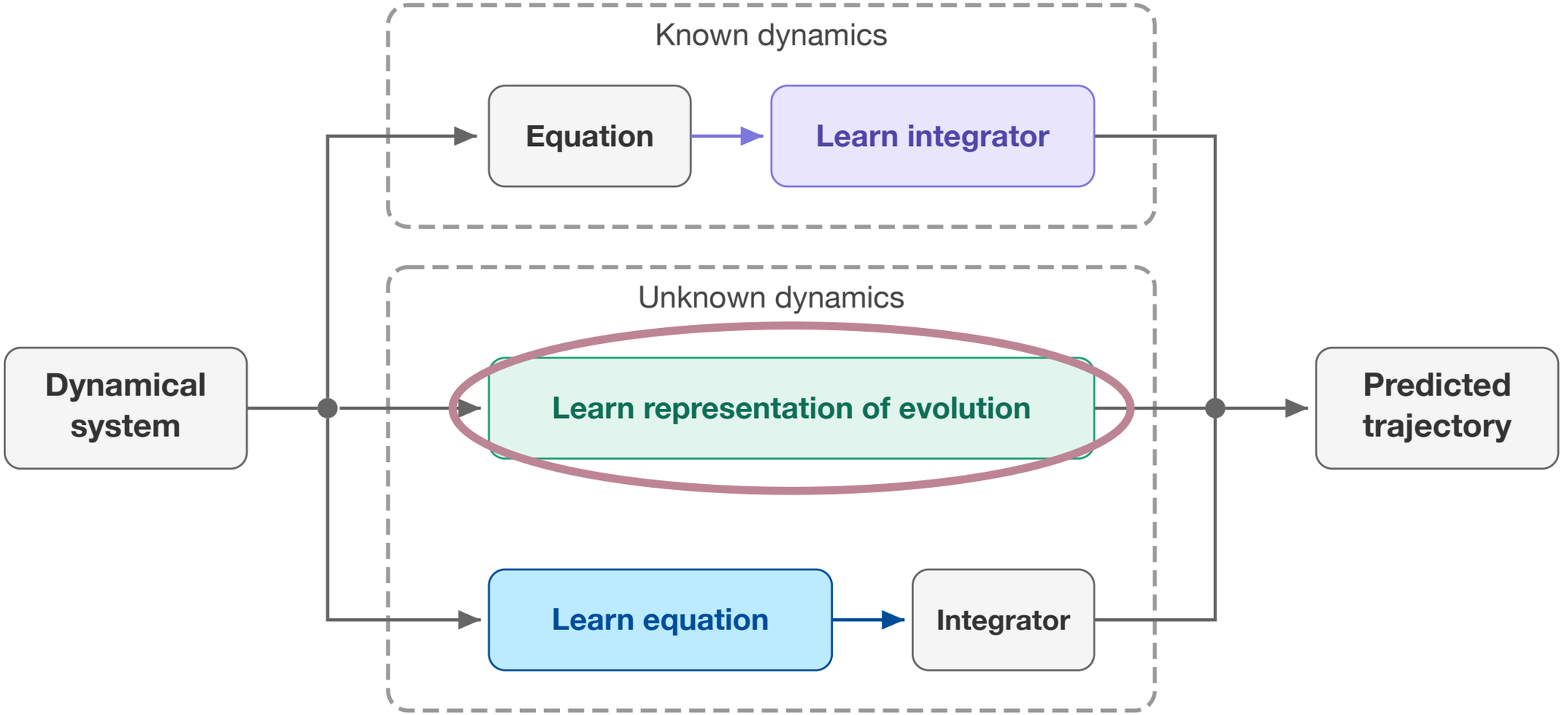
PART I · LEARNING REPRESENTATIONS

Parametric Koopman Decompositions

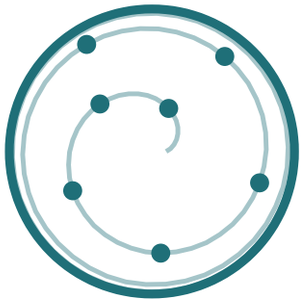
PUBLISHED AS ***“Learning Parametric Koopman Decompositions for Prediction and Control”***

in **SIAM Journal on Applied Dynamical Systems** (2025)

Yue Guo, Milan Korda, Ioannis G. Kevrekidis, Qianxiao Li


Known dynamics
Equation
Learn integrator
Unknown dynamics
Dynamical system
Learn representation of evolution
Predicted trajectory
Learn equation
Integrator

Make a nonlinear system look linear PK-NN



our system: **nonlinear**, from data

learn a representation →

IF THE SYSTEM WERE LINEAR

- spectral analysis
- forecast / prediction
- convex optimal control
- stability guarantees

Linear systems come with this whole toolbox **but only if the system is linear**.

Driving question: can we **learn a representation** in which a nonlinear system becomes **linear**?

The problem: data-driven prediction & control

We have only **trajectory data** from an **unknown** transition map $\boldsymbol{x}_{n+1} = \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{u}_n)$ — state $\boldsymbol{x} \in X$, parameter / control $\boldsymbol{u} \in U$. Two problems are posed on $\boldsymbol{f}$:

Forward prediction

Data: $\{\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{x}_{n+1}\}$

Goal: given $\boldsymbol{x}_0, \{\boldsymbol{u}_i\}$, predict $\{\boldsymbol{x}_i\}$.

Optimal control

Data: $\{\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{x}_{n+1}\}$

Goal: given $\boldsymbol{x}_0$ and a cost, find $\{\boldsymbol{u}_n\}$ minimizing it.

Two obstacles for data-driven modelling: **non-linearity** & **high dimensionality**.

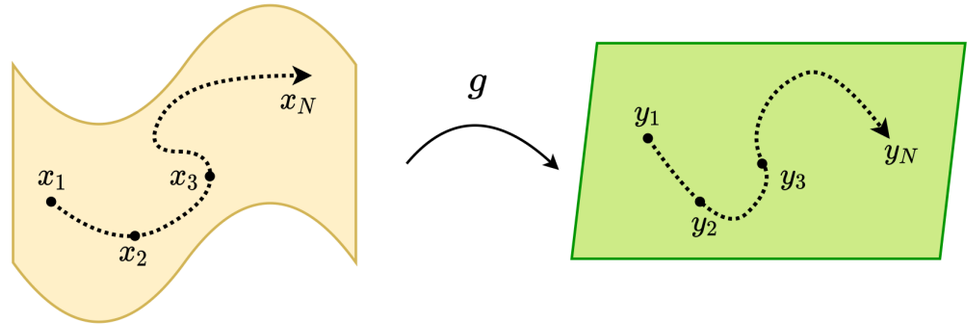
Koopman: a finite, linear invariant subspace

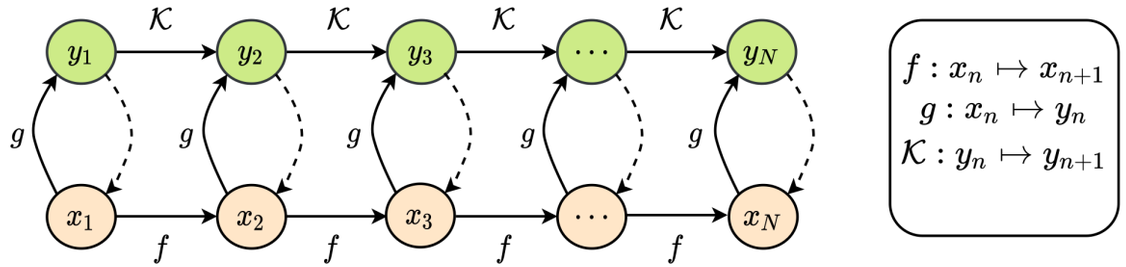
Track **observables** $g(\boldsymbol{x})$ instead of the state. The **Koopman operator** $\mathcal{K}$ advances any observable **linearly**: $g(\boldsymbol{x}_{n+1}) = (\mathcal{K}g)(\boldsymbol{x}_n)$.

We keep only a chosen dictionary $\boldsymbol{\Psi} = (\psi_1, \ldots, \psi_m)$ spanning a finite **invariant** subspace S: $\mathcal{K}\,S \subset S$.

> On S, $\mathcal{K}$ becomes a finite **matrix** K (basis $\boldsymbol{\Psi}$) — the **Koopman decomposition**:
>
> $$\boldsymbol{\Psi}(\boldsymbol{x}_{n+1}) = K\,\boldsymbol{\Psi}(\boldsymbol{x}_n)$$

finite-dim non-linear → infinite-dim **linear**; DL handles high dimensions.





Nonlinear state space → flat observable space, where $\mathcal{K}$ is linear.

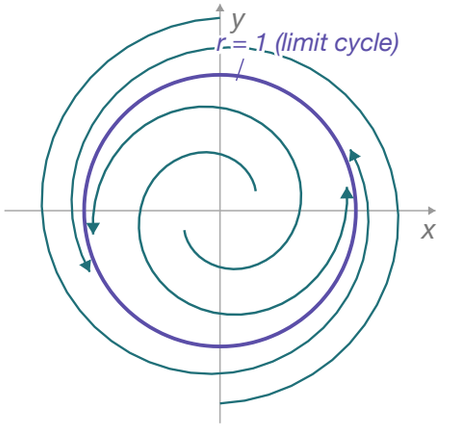
A less trivial example: a nonlinear limit cycle

A limit-cycle system (**Stuart–Landau**):

$$\dot{x} = (1 - x^2 - y^2)\, x - \omega y$$
$$\dot{y} = \omega x + (1 - x^2 - y^2)\, y$$

With $r^2 = x^2 + y^2$, the polar form decouples:

$$\dot{r} = r(1 - r^2), \qquad \dot{\theta} = \omega$$



The amplitude relaxes **nonlinearly**, while the phase rotates uniformly.

Koopman coordinates separate decay and rotation

Observables ψ_a (amplitude), (ψ_c, ψ_s) (phase):

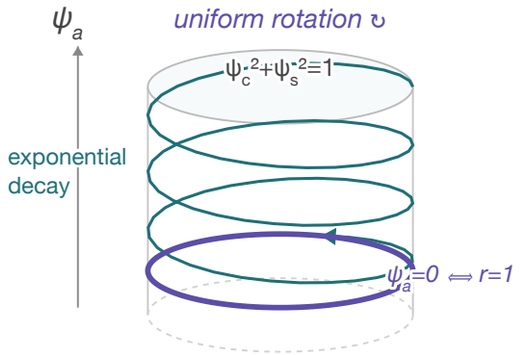
$$\psi_a = \frac{1}{r^2} - 1, \quad \psi_c = \cos\theta, \quad \psi_s = \sin\theta$$

The linear evolution on the observables:

$$\dot{\mathbf{\Psi}} = A\mathbf{\Psi}, \quad A = \begin{pmatrix} -2 & 0 & 0 \\ 0 & 0 & -\omega \\ 0 & \omega & 0 \end{pmatrix}$$

$$\mathbf{\Psi}_{n+1} = K\mathbf{\Psi}_n$$

$$K = e^{A\Delta t} = \begin{pmatrix} e^{-2\Delta t} & 0 & 0 \\ 0 & \cos\omega\Delta t & -\sin\omega\Delta t \\ 0 & \sin\omega\Delta t & \cos\omega\Delta t \end{pmatrix}$$



The lifted states still form a **two-dimensional invariant manifold**:

- No increase in intrinsic dimension.
- Linear evolution in observable space.

Finite-dimensional closure is not generic

Nonlinear **pendulum**:

$$\dot{\theta} = v, \qquad \dot{v} = -\sin\theta$$

Start from a finite dictionary:

$$\boldsymbol{\Psi} = (\theta,\ v,\ \sin\theta,\ \cos\theta)^\top$$

Differentiating keeps producing **new** observables: the dictionary **never closes**.

$$\frac{d}{dt}\sin\theta = v\cos\theta$$
$$\frac{d}{dt}\cos\theta = -v\sin\theta$$

$$\frac{d}{dt}(v\cos\theta) = -\sin\theta\cos\theta - v^2\sin\theta$$
$$\frac{d}{dt}(v\sin\theta) = -\sin^2\theta + v^2\cos\theta$$

$$\begin{aligned}\sin\theta, \cos\theta &\to v\cos\theta,\ v\sin\theta \\ &\to v^2\sin\theta,\ v^2\cos\theta,\ \sin\theta\cos\theta,\ \cdots\end{aligned}$$

The observable hierarchy keeps growing.

The Koopman operator is always **linear**, but its invariant observable space is generally **infinite-dimensional**.

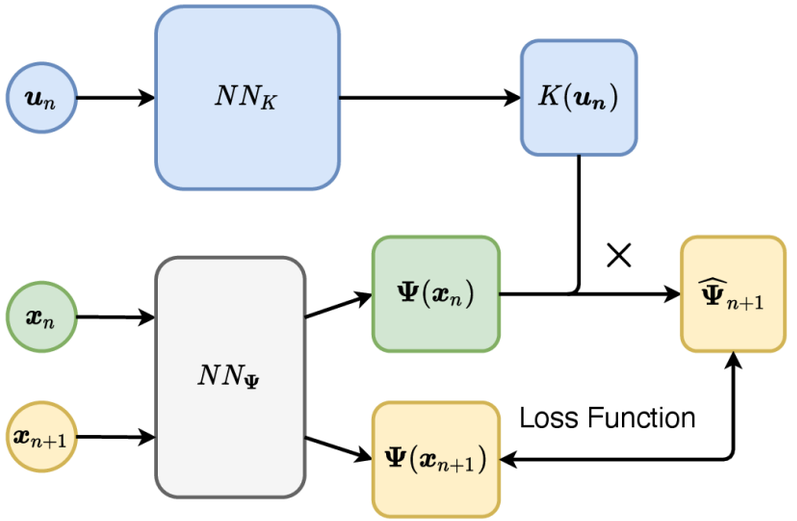
Finite-dimensional Koopman models are therefore often **approximations**, not exact closures.

PK-NN: learn Ψ and $K(u)$ jointly

For **parametric** systems $\mathcal{K}$ depends on $\boldsymbol{u}$, PK-NN learns one shared **dictionary Ψ** and a **fully nonlinear operator $K(u)$**.

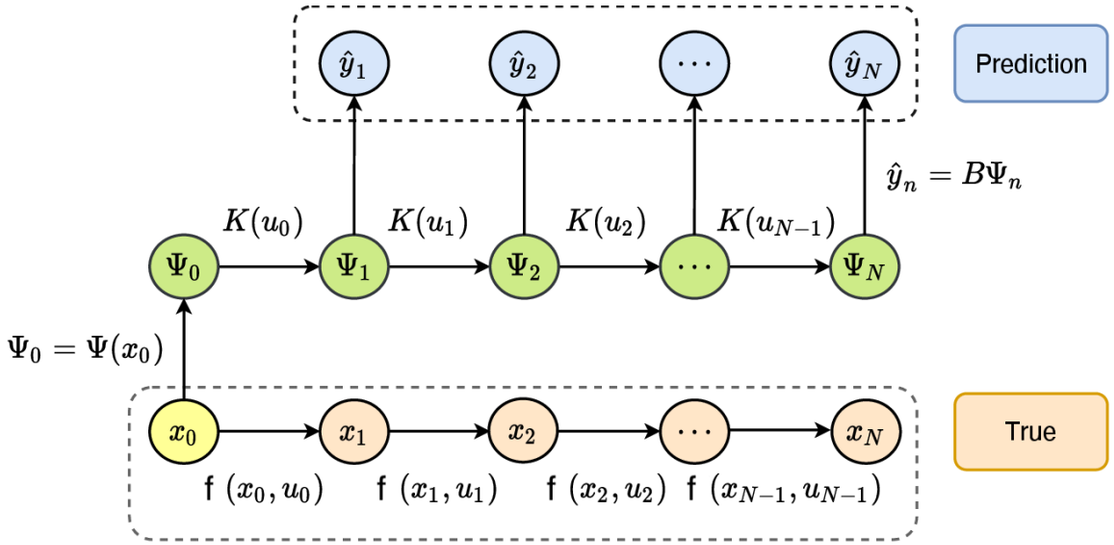
Model	Form	
Optimized DMD	$\boldsymbol{x}_{n+1} = K\boldsymbol{x}_n$	M0
EDMD / EDMD-DL	$\boldsymbol{\Psi}(\boldsymbol{x}_{n+1}) = K\,\boldsymbol{\Psi}(\boldsymbol{x}_n)$	M1
Linear control	$\boldsymbol{\Psi}_{n+1} = A\boldsymbol{\Psi}_n + B\boldsymbol{u}_n$	M2
Bilinear	$\boldsymbol{\Psi}_{n+1} = A\boldsymbol{\Psi}_n + \sum_i B_i u_{n,i}\boldsymbol{\Psi}_n$	M3
Fixed-basis param	$\boldsymbol{\Psi}_{n+1} = \sum_i h_i(\boldsymbol{u}_n)K_i\,\boldsymbol{\Psi}_n$	M4
Ours (PK-NN)	$\boldsymbol{\Psi}(\boldsymbol{x}_{n+1}) = K(\boldsymbol{u}_n)\,\boldsymbol{\Psi}(\boldsymbol{x}_n)$	**Ours**

▸ comparison

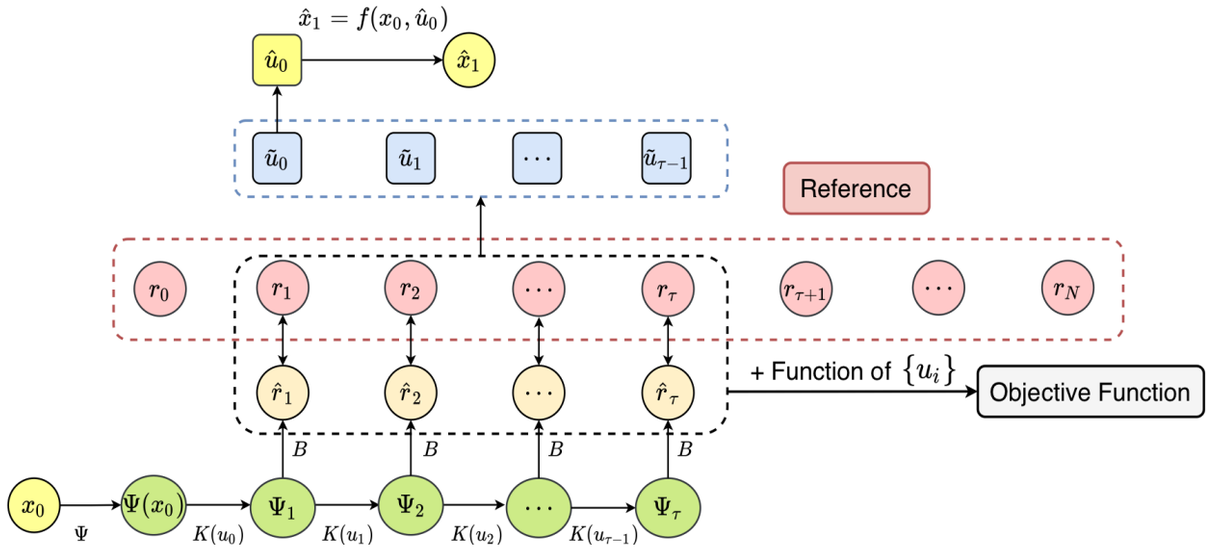


$$\min_{\theta_\Psi,\,\theta_K} \sum_{n,j} \left\| \boldsymbol{\Psi}(\boldsymbol{x}_{n+1}^{(j)};\theta_\Psi) - K(\boldsymbol{u}_n^{(j)};\theta_K)\,\boldsymbol{\Psi}(\boldsymbol{x}_n^{(j)};\theta_\Psi) \right\|^2$$

We solve both: forward prediction & optimal control

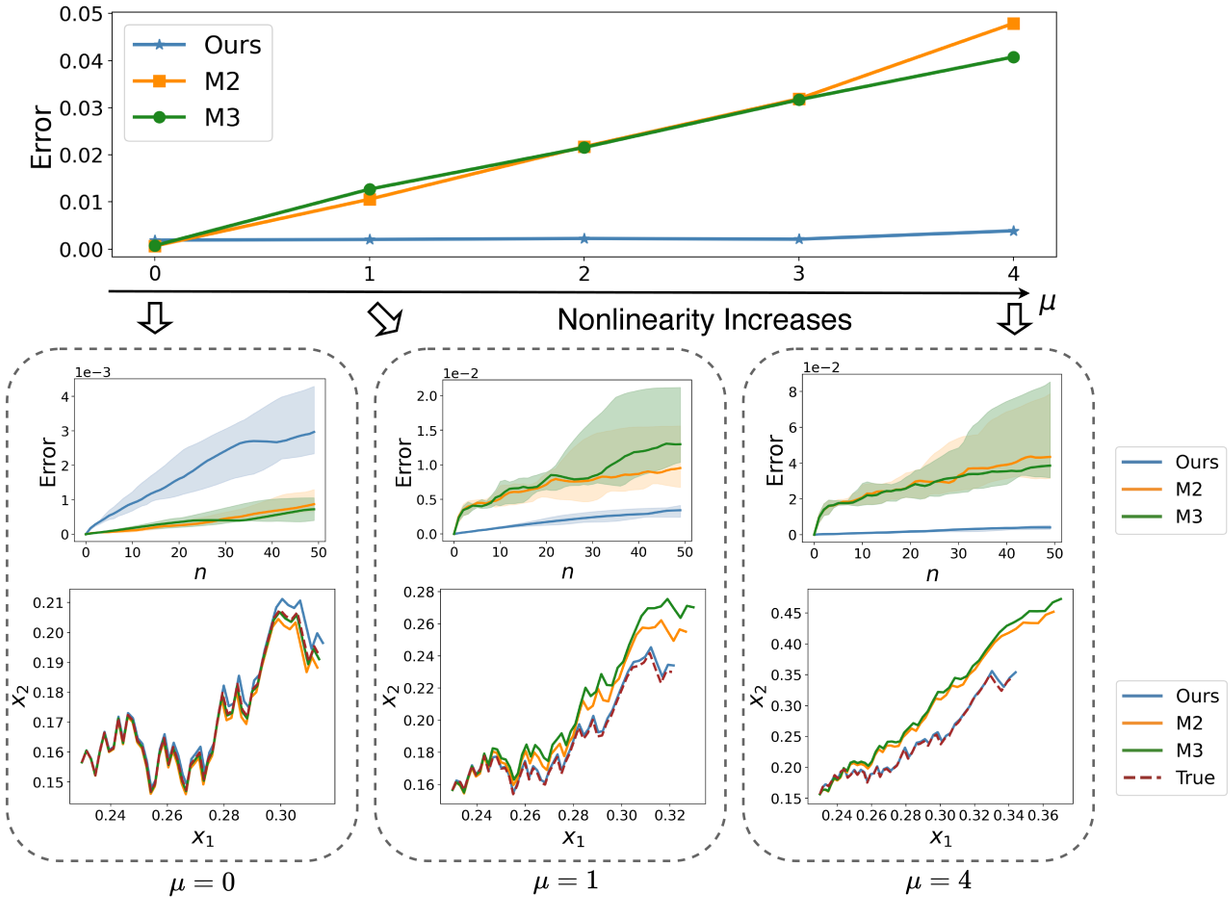


Forward prediction — lift, step with $K(\boldsymbol{u}_n)$, read out $\hat{\boldsymbol{y}} = B\boldsymbol{\Psi}$.



Optimal control (MPC) — optimize $\{\boldsymbol{u}_n\}$ to track a reference.

Prediction under strong nonlinearity in u



Van der Pol–Mathieu; μ controls how nonlinearly the control u enters:

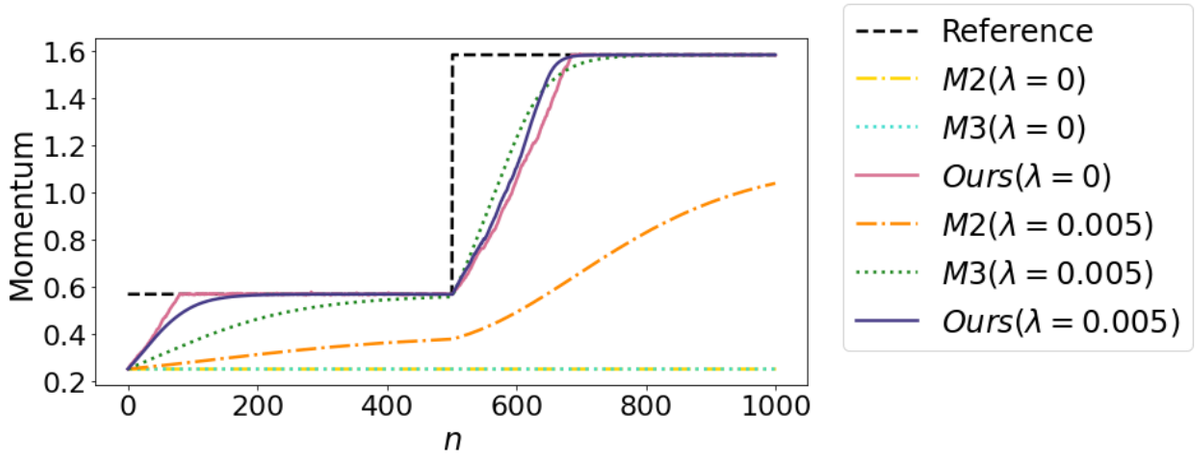
$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = (2 - 2x_1^2)\, x_2 - (1 + 2\mu\, u^2 - \mu)\, x_1 + u$$

Advantage: the fully nonlinear $K(\boldsymbol{u})$ stays accurate as μ grows and generalizes to **unseen parameters**

Van der Pol–Mathieu — error vs. step, as nonlinearity in u increases.

EXPERIMENTS · CONTROL

Optimal control — forced KdV (MPC)



Ours (solid) follows the step reference (dashed); baselines M2 / M3 lag.

Forced KdV — control u enters nonlinearly:

$$\partial_t \eta + \eta\, \partial_x \eta + \partial_{xxx} \eta = \sum_i v_i(x) \sin(\pi u_i)$$

Track a reference r_n (momentum $\int \eta^2 dx$) by **Model Predictive Control** in the lifted space:

$$\min_{\{u_n\}} \sum_n |\hat{r}_n - r_n|^2 + \lambda \|u_n\|^2$$

The model reads $\hat{r}_n = \boldsymbol{c}^\top \boldsymbol{\Psi}(\boldsymbol{x}_n)$ **linearly** off the lifted state $\boldsymbol{\Psi}_{n+1} = K(\boldsymbol{u}_n)\boldsymbol{\Psi}_n$.

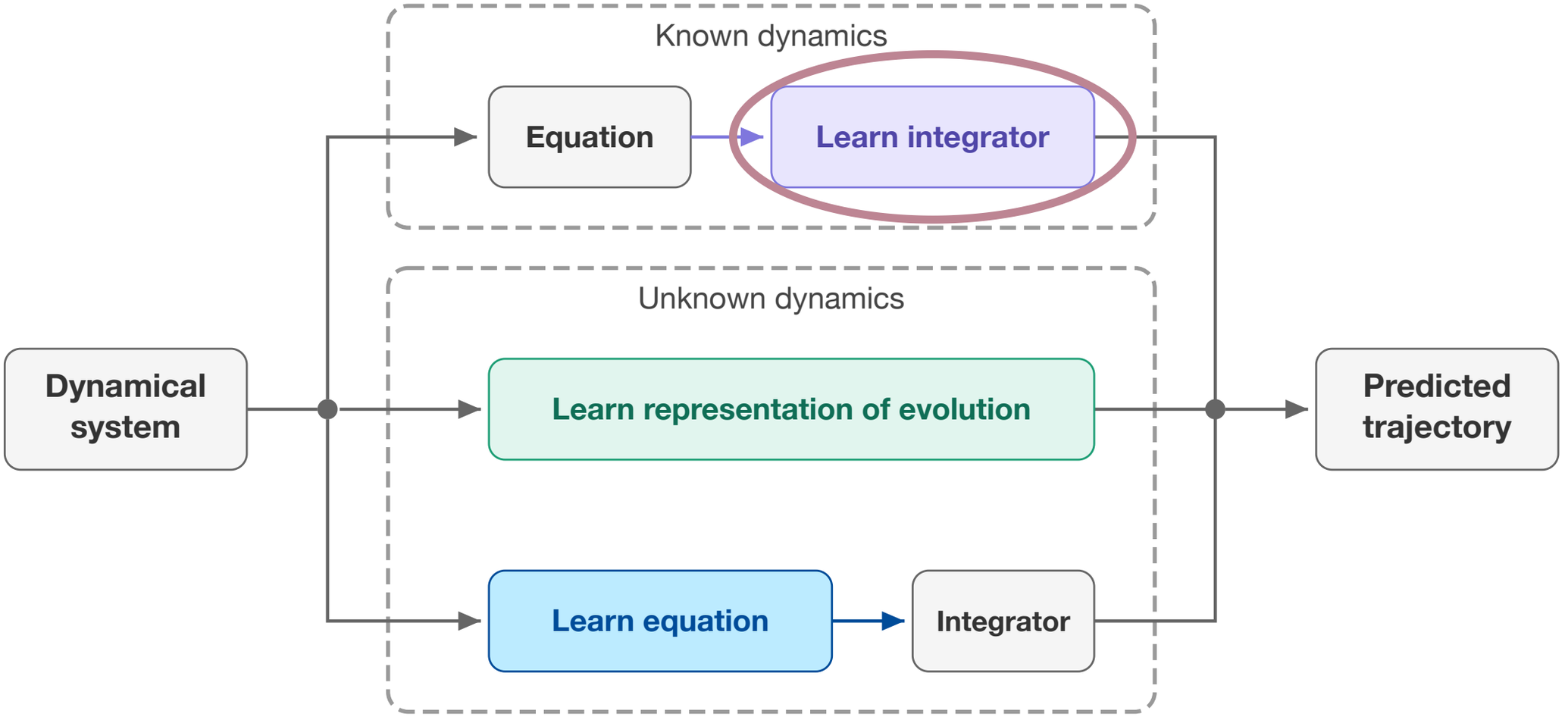
PART II · LEARNING THE NUMERICAL MECHANISM

Personalized ODE Integrators

PUBLISHED AS ***“Personalized Algorithm Generation: A Case Study in Learning ODE Integrators”***

in **SIAM Journal on Scientific Computing** (2022)

Yue Guo, Felix Dietrich, Tom Bertalan, Danimir T. Doncevic, Manuel Dahmen, Ioannis G. Kevrekidis, Qianxiao Li


Known dynamics
Equation
Learn integrator
Unknown dynamics
Dynamical system
Learn representation of evolution
Predicted trajectory
Learn equation
Integrator

An integrator built for Specific system RK-NN

Classical Solver: one recipe for all f

A **worst-case tax**: robust
to systems you never solve

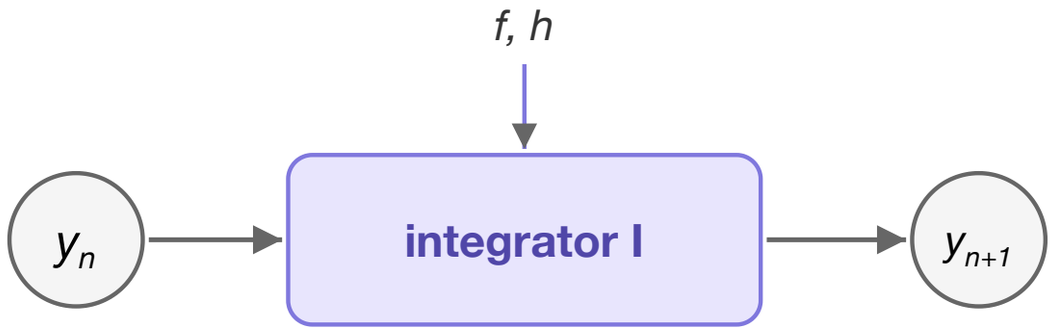
Specific family: solved many times

Worst-case coefficients are never tuned to it,
you pay that tax on every run

Driving question: can we build (learn) an integrator
specialized to the family we actually solve?

Personalized ODE integrators RK-NN

Known ODE $\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y})$. A numerical **integrator** I advances the state one step h: $\boldsymbol{y}_n \to \boldsymbol{y}_{n+1}$.



One step, using the known $\boldsymbol{f}$.

How should we **optimize** the integrator I?

Classical: I tuned for the **worst case** over a large class $\mathcal{F}$:

$$\sup_{F \in \mathcal{F}} \|L(I, F)\| = \mathcal{O}(h^{\alpha})$$

Ours: I optimized over the **family** μ we actually solve:

$$\mathbb{E}_{\mu} \|L(I, F_{\mu})\| = \mathcal{O}(h^{\alpha})$$

$\Rightarrow$ A **personalized** integrator

Classical RK: coefficients fixed by hand

A **Runge–Kutta** step $\hat{\boldsymbol{y}}_{n+1} = \boldsymbol{y}_n + h\sum_i b_i \boldsymbol{k}_i$
, $\boldsymbol{k}_i = \boldsymbol{f}(\boldsymbol{y}_n + h\sum_j a_{ij}\boldsymbol{k}_j)$, with coefficients (a_{ij}, b_i), $c_i = \sum_j a_{ij}$.

Taylor-expand the **exact** step in the **elementary differentials** (fixed coefficients):

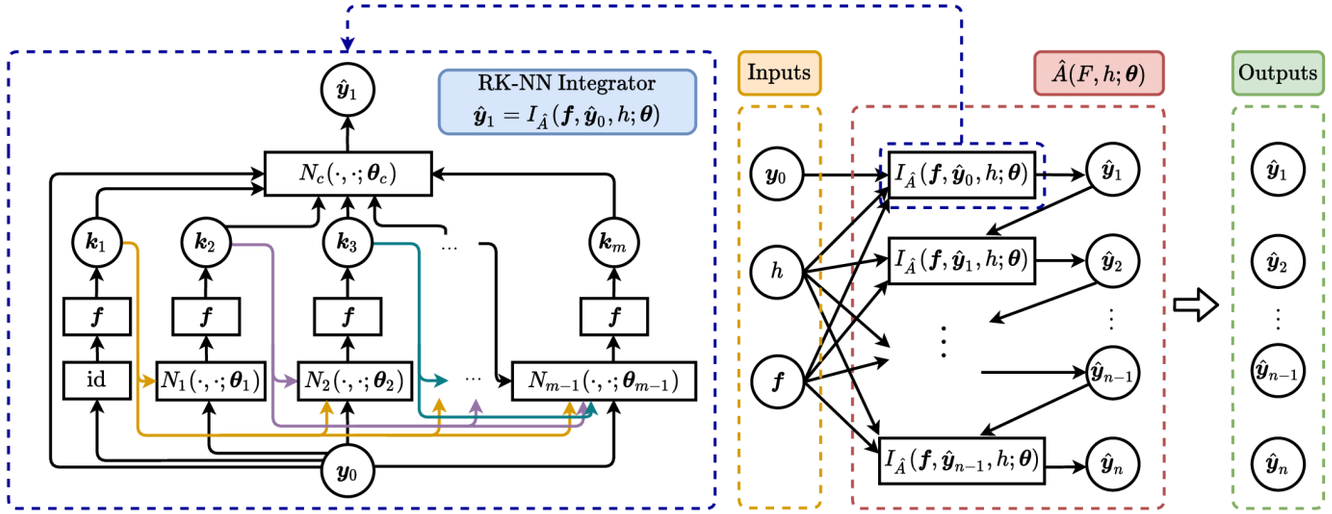
$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\,\boldsymbol{f} + \tfrac{h^2}{2}\,\boldsymbol{f}'\boldsymbol{f} + \tfrac{h^3}{6}\left(\boldsymbol{f}''\boldsymbol{f}\boldsymbol{f} + \boldsymbol{f}'\boldsymbol{f}'\boldsymbol{f}\right) + \cdots$$

RK expands in **the same** differentials — match each (for **all** $\boldsymbol{f}$) $\Rightarrow$ **one condition per term**:

term	set these two coefficients equal		⇒ order condition on (a_{ij}, b_i)
	in exact $\boldsymbol{y}_{n+1}$	in RK $\hat{\boldsymbol{y}}_{n+1}$	
$h\,\boldsymbol{f}$	1	$\sum_i b_i$	$\sum_i b_i = 1$
$h^2\,\boldsymbol{f}'\boldsymbol{f}$	$\frac{1}{2}$	$\sum_i b_i c_i$	$\sum_i b_i c_i = \frac{1}{2}$
$h^3\,\boldsymbol{f}''\boldsymbol{f}\boldsymbol{f}$	$\frac{1}{6}$	$\frac{1}{2}\sum_i b_i c_i^2$	$\frac{1}{2}\sum_i b_i c_i^2 = \frac{1}{6}$
$h^3\,\boldsymbol{f}'\boldsymbol{f}'\boldsymbol{f}$	$\frac{1}{6}$	$\sum_{ij} b_i a_{ij} c_j$	$\sum_{ij} b_i a_{ij} c_j = \frac{1}{6}$

Matched through h^3: the step differs only by the **truncation error** $\hat{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1} = O(h^4)$.

RK-NN: learn the coefficients $\boldsymbol{\theta}$



The Butcher-tableau coefficients become learnable $\boldsymbol{\theta}$ inside an m-stage RK.

Data: sample the family F_μ, initial states $\boldsymbol{y}_0$, and steps $h \in (0.01, 0.1)$; the target $\boldsymbol{y}_1 = \varphi_h(\boldsymbol{y}_0)$ is the accurate one-step solution of the known $\boldsymbol{f}$.

Train $\boldsymbol{\theta}$ on the loss:

$$\mathcal{L} = \frac{\|\boldsymbol{y}_1 - \hat{\boldsymbol{y}}_1\|^2}{\|\boldsymbol{y}_1 - \hat{\boldsymbol{y}}_1^{(\mathrm{RK})}\|^2} + \lambda \mathcal{R}$$

$$\mathcal{R} = \sum_{i=1}^{\alpha} \left\| \frac{d^i}{dh^i}\Big|_0 (\boldsymbol{y}_1 - \hat{\boldsymbol{y}}_1) \right\|^2$$

$\mathcal{L}$: one-step error, scaled by the classical RK error $\|\boldsymbol{y}_1 - \hat{\boldsymbol{y}}_1^{(\mathrm{RK})}\|$

$\mathcal{R}$: Taylor regularizer enforcing order α.

RK-NN keeps order 3 yet beats RK3

Two real 2-D nonlinear systems, state $\boldsymbol{y} = (u, v)$. A 3-stage RK-NN ($\alpha = 3$) is trained per family over $h \in (0.01, 0.1)$; outside that range it degrades — **specialized by design**.

Van der Pol

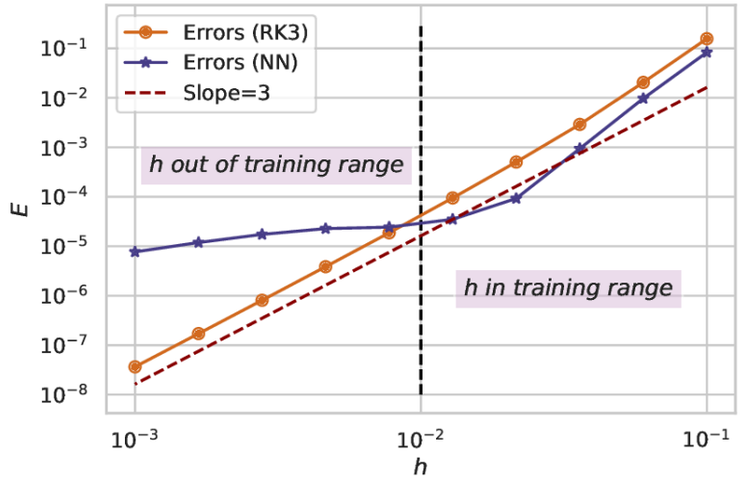
$$\dot{u} = v, \qquad \dot{v} = \mu\,(1 - u^2)\,v - u$$

limit cycle; nonlinearity set by μ.

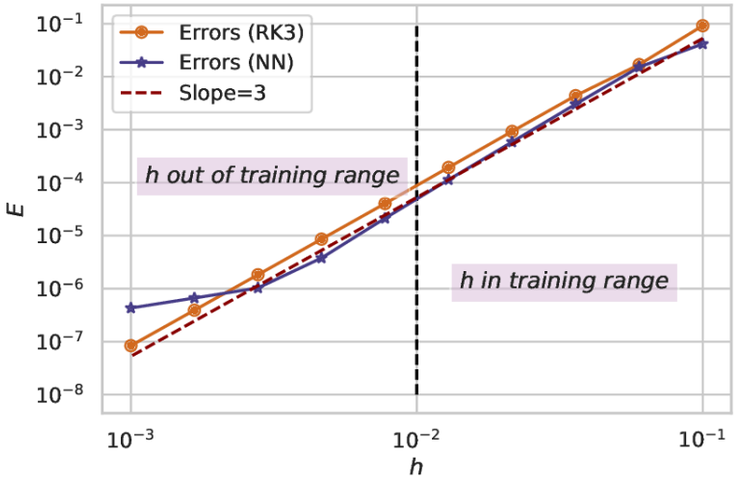
Brusselator

$$\dot{u} = A + u^2 v - (B{+}1)u, \qquad \dot{v} = Bu - u^2 v$$

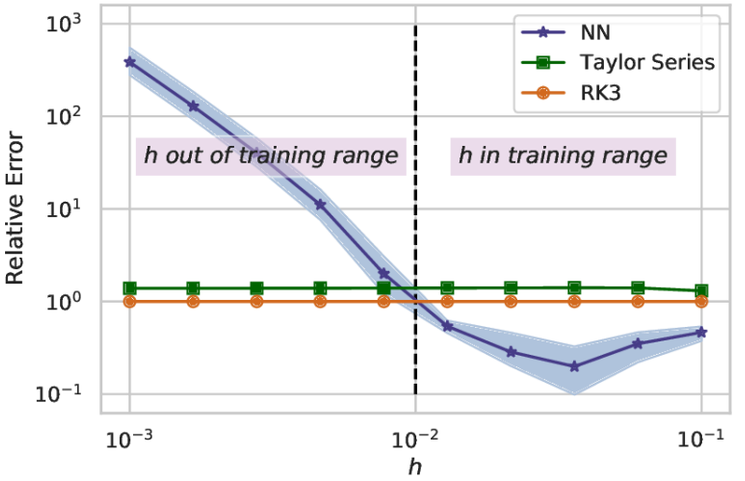
autocatalytic reaction with constants A, B.



Van der Pol — order check



Brusselator — order check



Van der Pol — relative error

Highlight: “breaking” the order barrier

2-stage scheme, $k_2 = f(y + h\theta_1 k_1)$:

$$y_{n+1} = y + h(\theta_{c1} k_1 + \theta_{c2} k_2)$$

Classical: Taylor-match → order 2:

$$\theta_{c1} + \theta_{c2} = 1, \quad 2\,\theta_1 \theta_{c2} = 1 \quad \underset{\text{(Heun)}}{\Rightarrow} \quad \theta_1 = 1,\ \theta_{c1} = \theta_{c2} = \tfrac{1}{2}$$

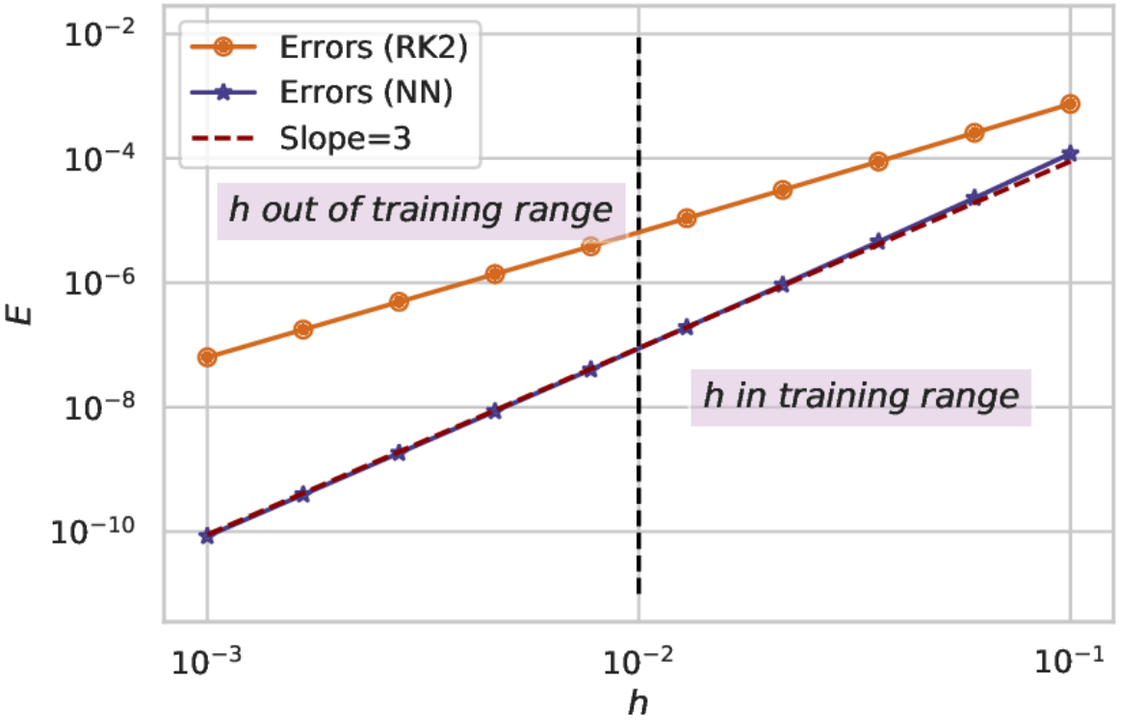
Two stages cannot meet both 3rd-order conditions → **order barrier**.

Ours: for $f = -y^2$, both 3rd-order terms are $\propto y^4$, so they merge:

$$\begin{aligned} y(t_n + h) &= y - y^2 h + y^3 h^2 - y^4\, h^3 + O(h^4) \\ \hat{y}_{n+1} &= y - (\theta_{c1} + \theta_{c2}) y^2 h + 2\theta_{c2}\theta_1\, y^3 h^2 \\ &\quad - \theta_{c2}\theta_1^2\, y^4\, h^3 + O(h^4) \end{aligned}$$

h^3 : $\theta_{c2}\theta_1^2 = 1$ (solvable); with

$\theta_{c2}\theta_1 = \tfrac{1}{2} \Rightarrow \theta_1 = 2,\ \theta_{c2} = \tfrac{1}{4},\ \theta_{c1} = \tfrac{3}{4}$.



2-stage RK-NN attains slope-3 error.

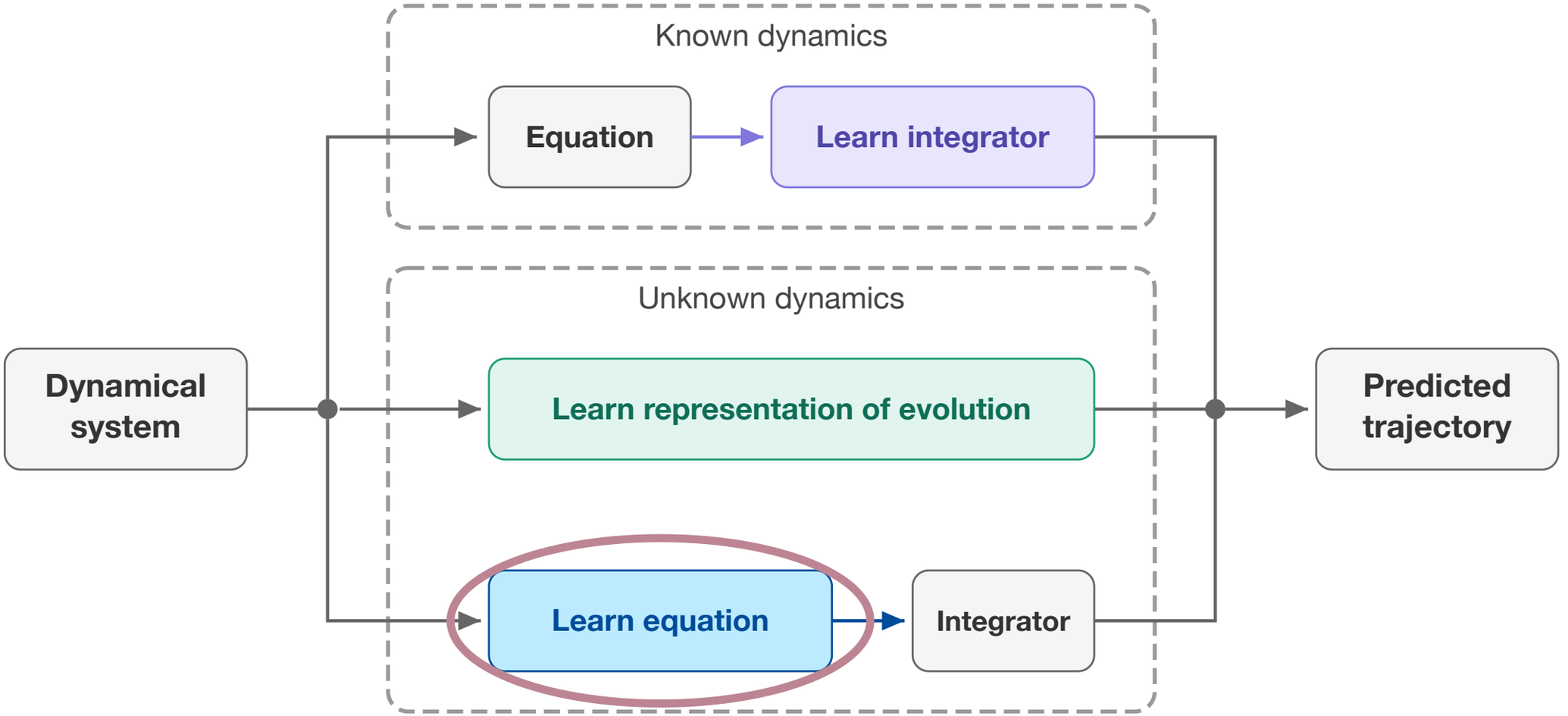
PART III · LEARNING THE CONTINUOUS DESCRIPTION

Integrator-aware Learning

MANUSCRIPT ***“Integrator-aware Learning of Modified Equations”***

in process

Yue Guo, Aiqing Zhu, Haotian Jiang, Qianxiao Li.

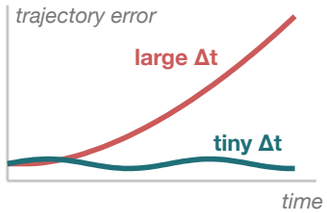

Known dynamics
Equation
Learn integrator
Unknown dynamics
Dynamical system
Learn representation of evolution
Predicted trajectory
Learn equation
Integrator

Cheap forces, but tiny steps ModEqLearn

ethanol naphthalene aspirin

ML potentials make the **force per step** cheap

Molecular motion needs a **tiny** Δt (fs), so a long run takes lots of steps.



Take a **larger** Δt to go faster, and per-step error **accumulates** over the trajectory.

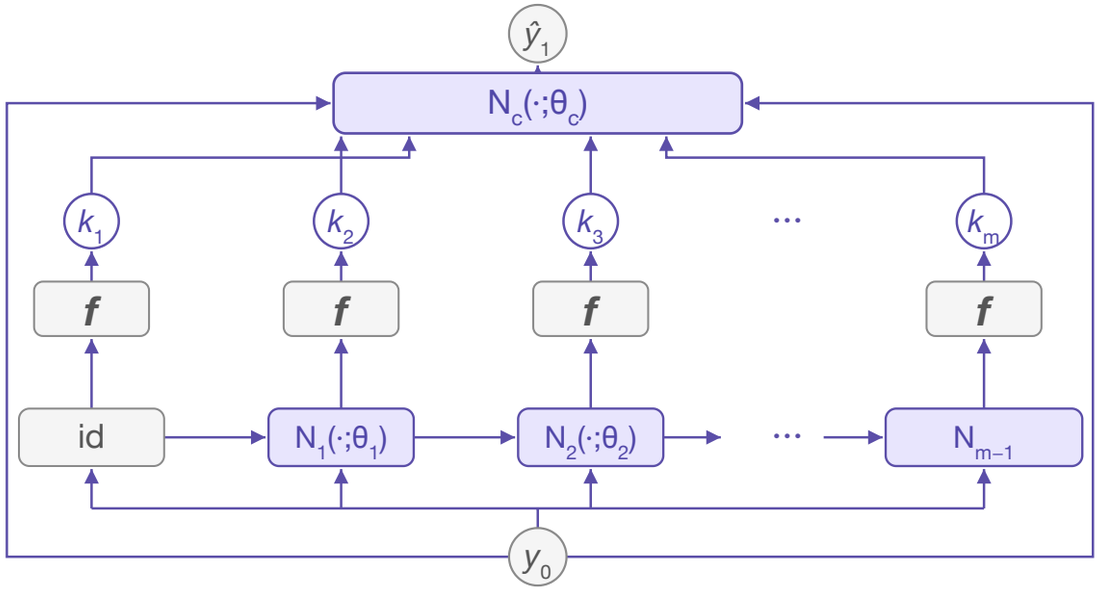
Driving question: can we learn forces accurate at a **larger** Δt?

Key idea ①: Duality with learning the integrator

Last Project

Given a system, learn the **integrator**

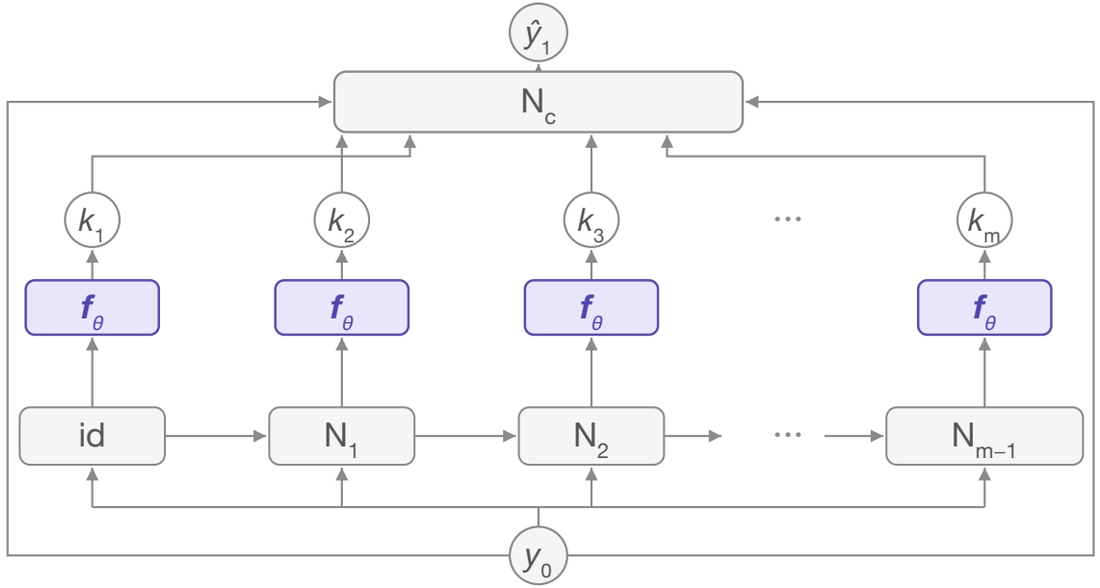
$$\min_{\Phi} \ \mathrm{err}\left(\Phi_h,\ \varphi_{h,\boldsymbol{f}_{\mathrm{true}}}\right)$$



Current Project

Given an integrator, learn the **equation**

$$\min_{\boldsymbol{f}_\theta} \ \mathrm{err}\left(\Phi_{h,\boldsymbol{f}_\theta},\ \varphi_{h,\boldsymbol{f}_{\mathrm{true}}}\right)$$



Key idea ②: modified equations

- **Backward error analysis:** a numerical integrator exactly solves a **modified** differential equation whose field depends on the integrator **and** the step size h.

$$\Phi_{h,\boldsymbol{f}_{true}}(\boldsymbol{y}) = \varphi_{h,\,\mathcal{M}_h(\boldsymbol{f}_{true})}(\boldsymbol{y}), \qquad \mathcal{M}_h(\boldsymbol{f}_{true}) = \boldsymbol{f}_{true} + h\,\boldsymbol{f}_{true}^{(1)} + h^2\,\boldsymbol{f}_{true}^{(2)} + \cdots$$

- **Inverse modified equation:** Given the true field $\boldsymbol{f}_{true}$ we want the **discrete map** to reproduce, find the field $\boldsymbol{f}_\theta$ to feed the integrator so that its numerical flow **exactly** matches the true flow.

$$\Phi_{h,\boldsymbol{f}_\theta}(\boldsymbol{y}) = \varphi_{h,\,\mathcal{M}_h(\boldsymbol{f}_\theta)}(\boldsymbol{y}) \iff \varphi_{h,\,\boldsymbol{f}_{true}}(\boldsymbol{y}) \approx \Phi_{h,\mathcal{M}_h^{-1}(\boldsymbol{f}_{true})}(\boldsymbol{y})$$

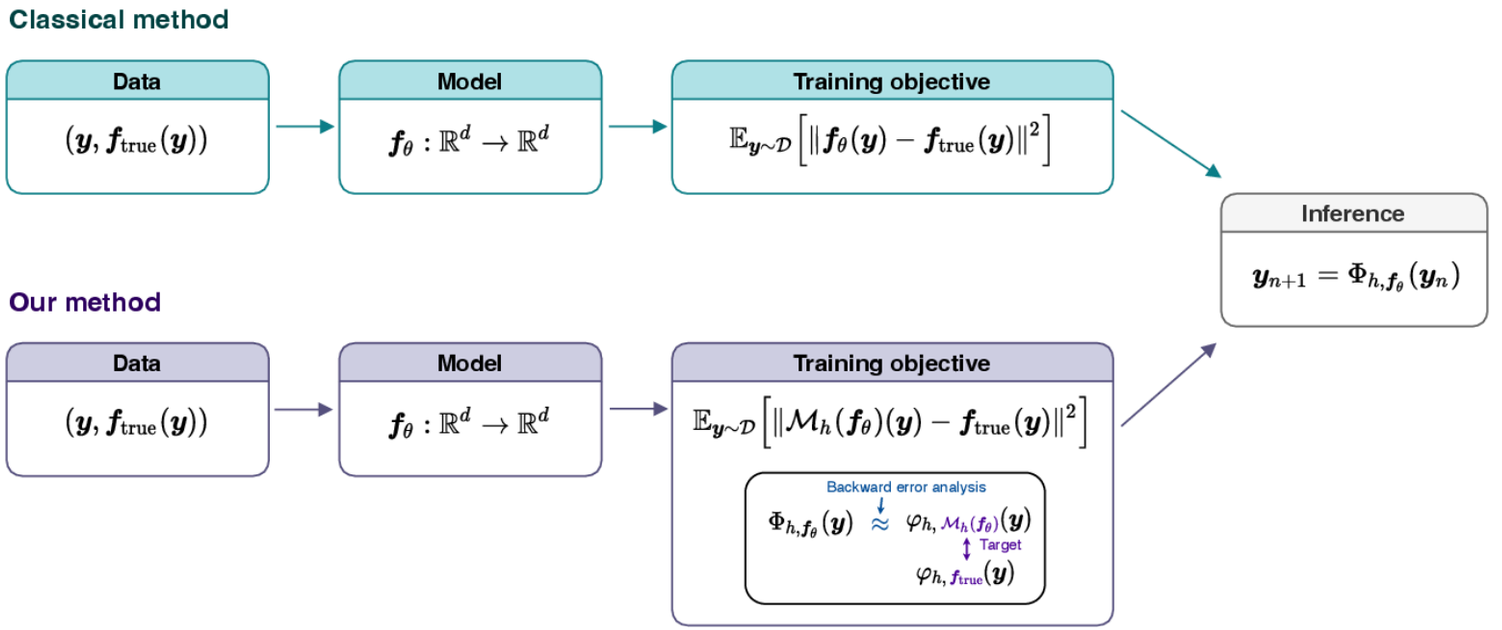
> The integrator's “true” target $\boldsymbol{f}_\theta$ is $\mathcal{M}_h^{-1}(\boldsymbol{f}_{true})$, not $\boldsymbol{f}_{true}$.

Method: ModEqLearn

Train the **modified equation** of f_θ matches f_{true}:

$$\min_\theta \ \mathbb{E}_{y\sim\mathcal{D}} \left\| \mathcal{M}_h(f_\theta)(y) - f_{\text{true}}(y) \right\|^2$$

Data: states $y \sim \mathcal{D}$ sampled over the domain, each paired with the true field $f_{\text{true}}(y)$.



Classical field matching vs. modified-equation-aware training.

Case study: molecular dynamics

Hamiltonian · leapfrog

Hamiltonian System positions $\boldsymbol{q}$, momenta $\boldsymbol{p}$dynamics from partials of H:

$$H(\boldsymbol{q},\boldsymbol{p}) = \tfrac{1}{2}\,\boldsymbol{p}^\top \boldsymbol{M}^{-1}\boldsymbol{p} + V(\boldsymbol{q})$$
$$\dot{\boldsymbol{q}} = \quad \partial_{\boldsymbol{p}} H = \boldsymbol{M}^{-1}\boldsymbol{p}$$
$$\dot{\boldsymbol{p}} = -\partial_{\boldsymbol{q}} H = -\nabla V(\boldsymbol{q})$$

Symplectic Leapfrog (Störmer–Verlet)

$$\boldsymbol{p}_{n+\frac{1}{2}} = \boldsymbol{p}_n - \tfrac{h}{2}\,\nabla V(\boldsymbol{q}_n)$$
$$\boldsymbol{q}_{n+1} = \boldsymbol{q}_n + h\,\boldsymbol{M}^{-1}\boldsymbol{p}_{n+\frac{1}{2}}$$
$$\boldsymbol{p}_{n+1} = \boldsymbol{p}_{n+\frac{1}{2}} - \tfrac{h}{2}\,\nabla V(\boldsymbol{q}_{n+1})$$

Molecular ML learns the potential V_θ, $H_\theta = T(\boldsymbol{p}) + V_\theta(\boldsymbol{q})$, field $\boldsymbol{f}_\theta = (\partial_{\boldsymbol{p}} H_\theta, -\partial_{\boldsymbol{q}} H_\theta) = (\boldsymbol{M}^{-1}\boldsymbol{p},\ -\nabla V_\theta)$:

Classical MACE: **pointwise** fit of V_θ to energies & forces:

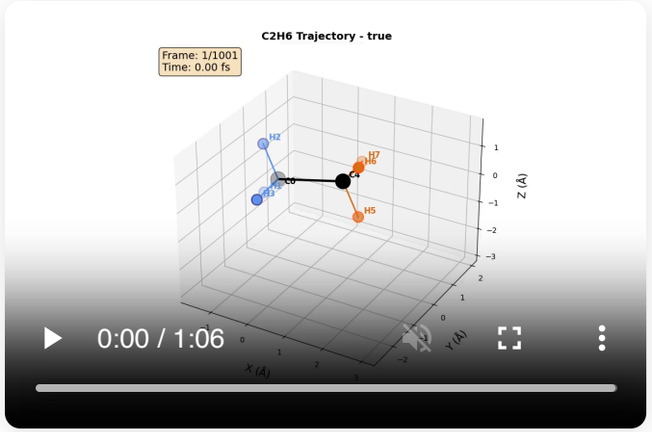
$$\min_\theta \ \sum \left(\|E - V_\theta\|^2 + \|\boldsymbol{F} + \nabla V_\theta\|^2 \right)$$

Ours (ModEqLearn): leapfrog's **modified Hamiltonian** of H_θ:

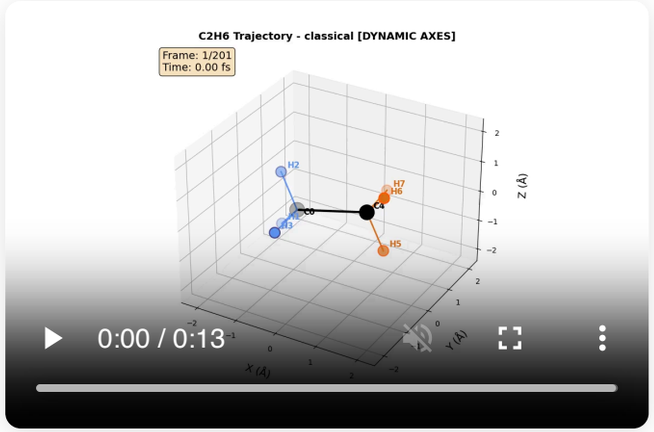
$$\tilde{H}^{(2)}_\theta = T(\boldsymbol{p}) + V_\theta + \tfrac{h^2}{24}\left(2\,\boldsymbol{v}^\top \nabla^2 V_\theta\, \boldsymbol{v} - \|\nabla V_\theta\|^2\right), \quad \boldsymbol{v} = \boldsymbol{M}^{-1}\boldsymbol{p}$$
$$\min_\theta \ \mathbb{E}\left\|\mathcal{M}_h(\boldsymbol{f}_\theta) - \boldsymbol{f}_{\text{true}}\right\|^2 = \mathbb{E}\left\|\nabla_{(\boldsymbol{q},\boldsymbol{p})}\tilde{H}^{(2)}_\theta - \nabla_{(\boldsymbol{q},\boldsymbol{p})} H_{\text{true}}\right\|^2$$

A perfect **pointwise** V_θ may **drift** in leapfrog at large step size → **the gap our work closes.**

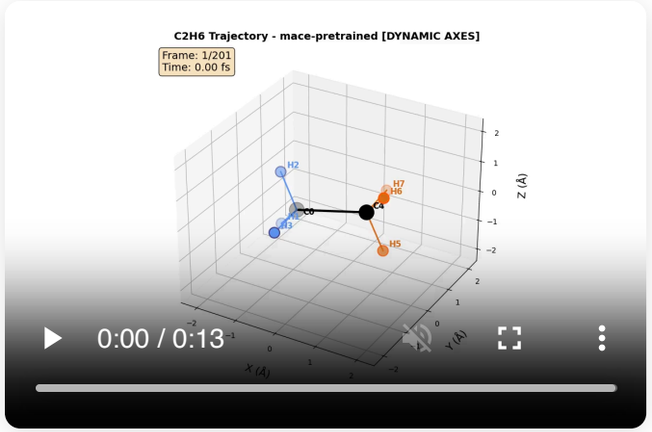
Results: Stable long-time integration



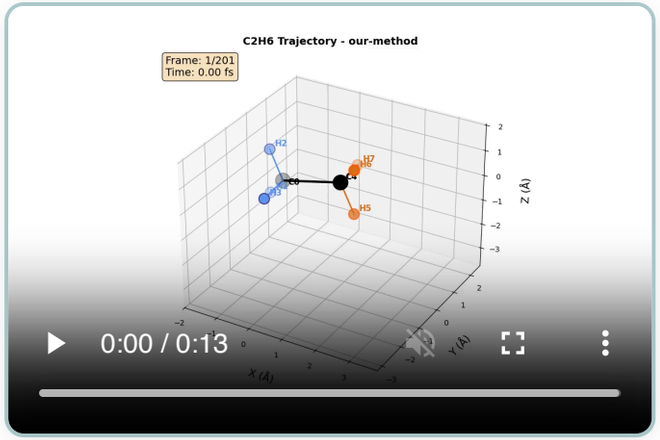
Ground truth



Classical

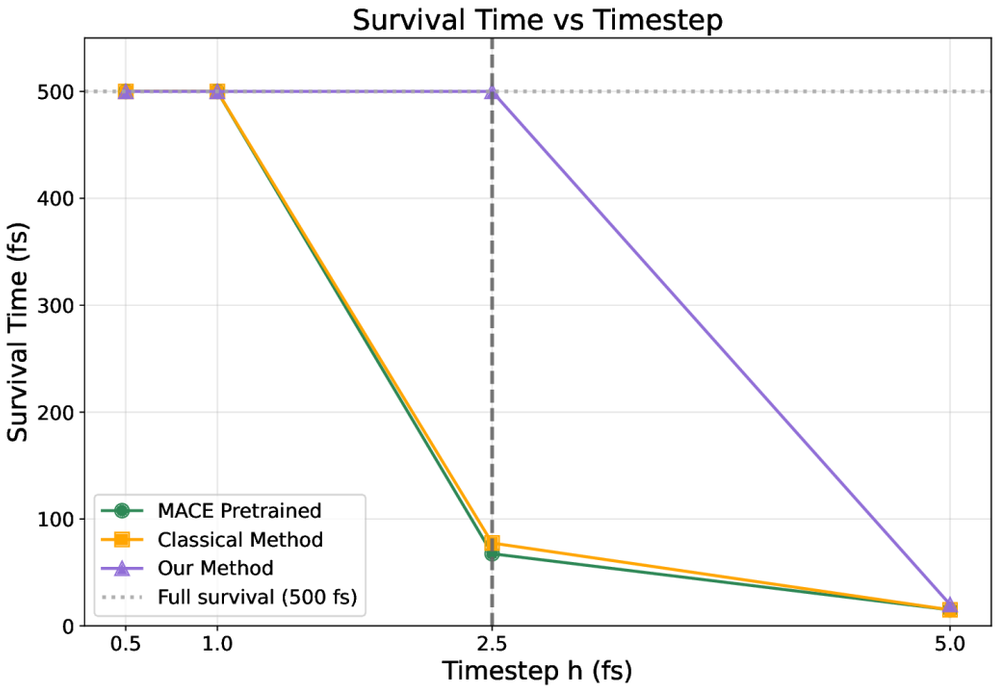


MACE pretrained



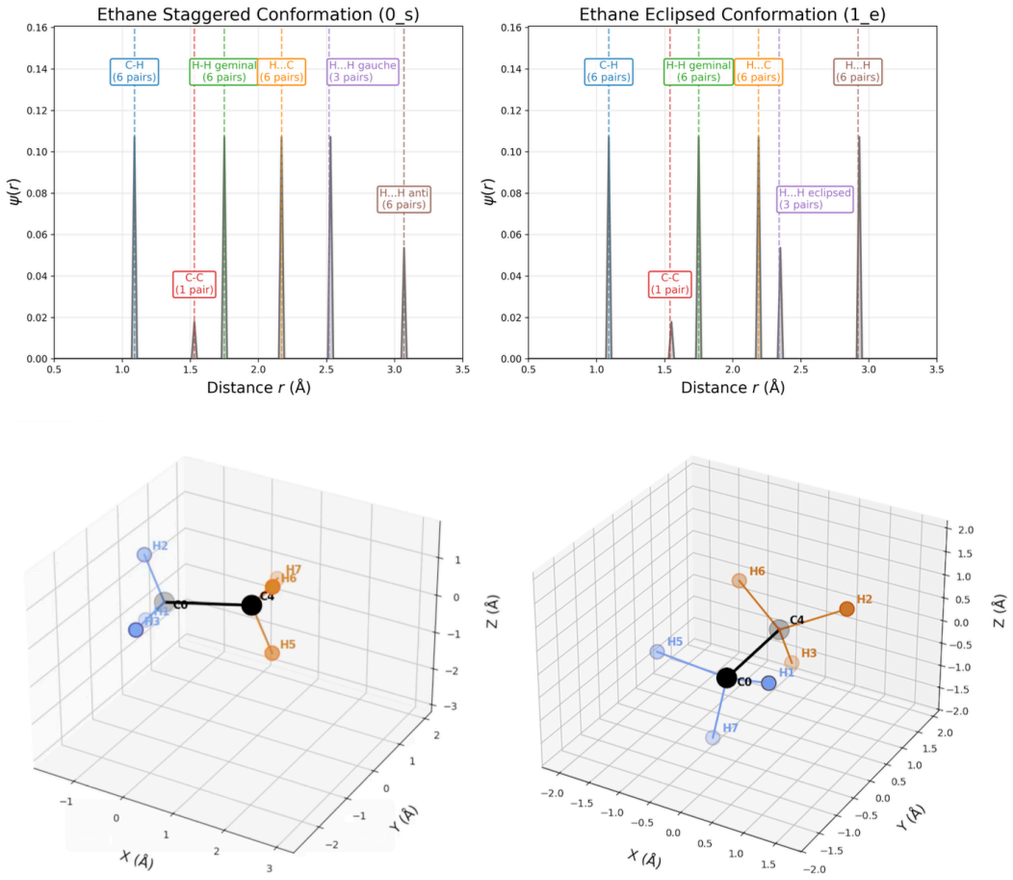
Our method

Ethane (C_2H_6): rollout trajectories at large step size.



Stable rollout survival vs. step size h.

Structural metric: bond-distance distribution



Staggered and eclipsed conformations define distinct interatomic-distance fingerprints.

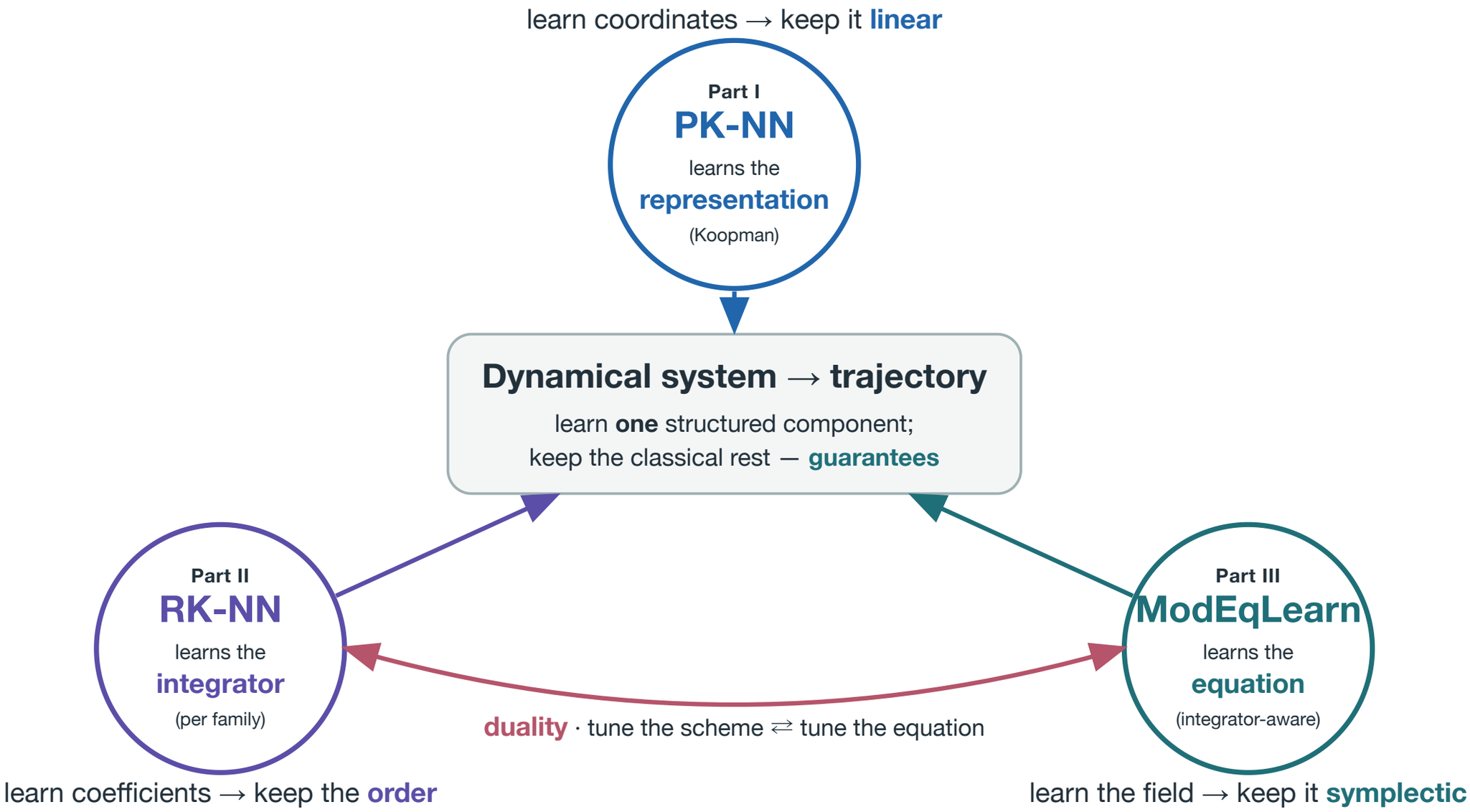
h (fs)	Method	50 fs	100 fs	200 fs	500 fs
0.5	MACE pretrained	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	0.107
	Classical	**<10^{-4}**	**<10^{-4}**	0.071	**0.071**
	Our method	0.107	0.214	0.143	0.143
1.0	MACE pretrained	**<10^{-4}**	**<10^{-4}**	**0.036**	0.071
	Classical	**<10^{-4}**	**<10^{-4}**	0.107	**<10^{-4}**
	Our method	0.107	0.214	0.107	0.143
2.5	MACE pretrained	0.071	---	---	---
	Classical	---	---	---	---
	Our method	**0.036**	**0.143**	**0.071**	**0.179**
5.0	MACE pretrained	---	---	---	---
	Classical	---	---	---	---
	Our method	---	---	---	---

Table 4.2: $\Delta\psi$ for eclipsed conformation; lower is better, --- indicates blowup.

$$\psi(r) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \delta(r - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$$

$$\Delta\psi = \int_{r_{\min}}^{r_{\max}} |\psi_{\text{pred}}(r) - \psi_{\text{true}}(r)| \, dr$$

Summary


learn coordinates → keep it linear
Part I
PK-NN
learns the
representation
(Koopman)
Dynamical system → trajectory
learn one structured component;
keep the classical rest — guarantees
Part II
RK-NN
learns the
integrator
(per family)
Part III
ModEqLearn
learns the
equation
(integrator-aware)
duality · tune the scheme ⇄ tune the equation
learn coefficients → keep the order
learn the field → keep it symplectic

Acknowledgements