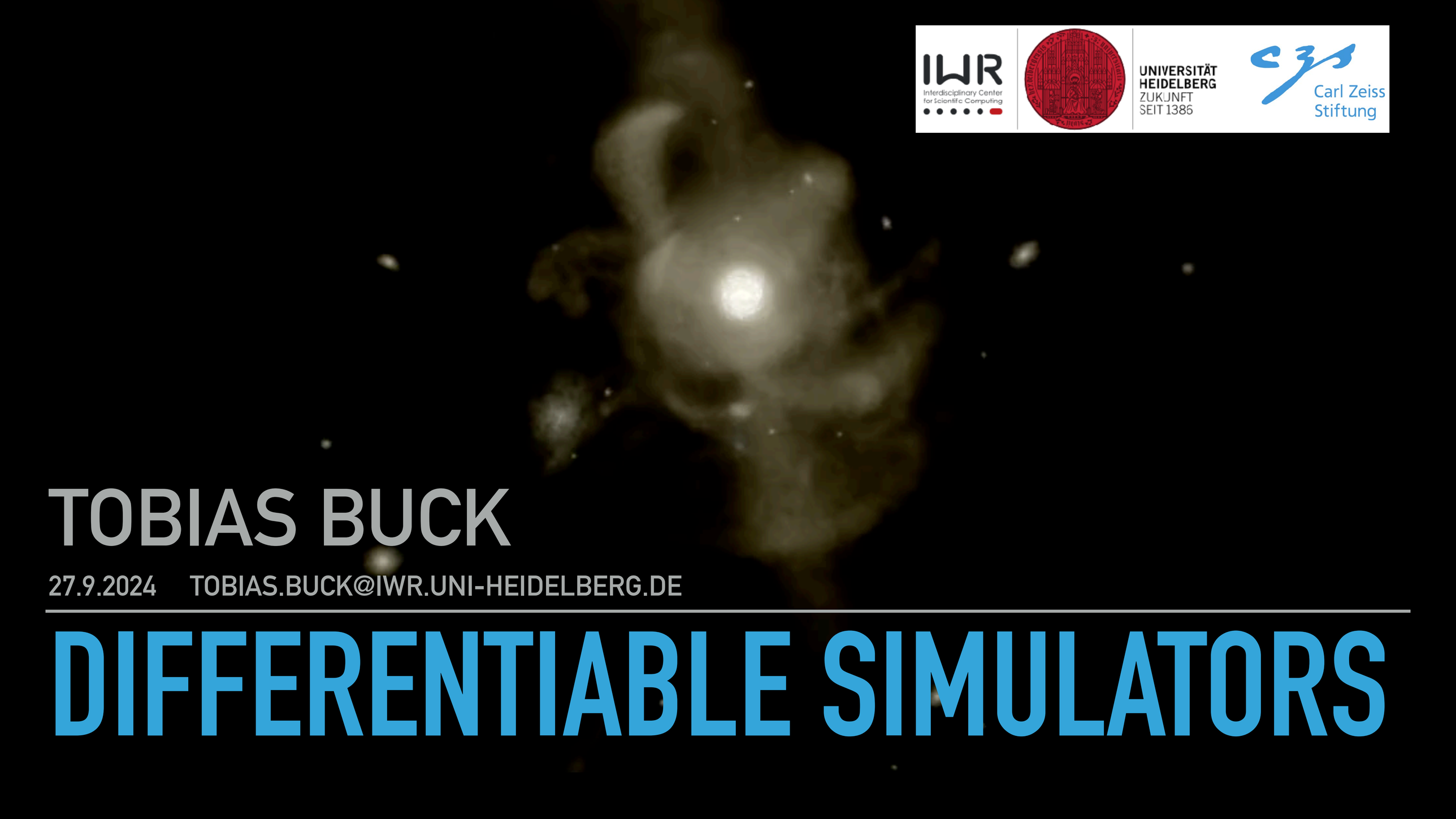


TOBIAS BUCK

27.9.2024 TOBIAS.BUCK@IWR.UNI-HEIDELBERG.DE

DIFFERENTIABLE SIMULATORS

OUTLINE OF TODAY

1. Day 5: Differentiable physics simulators cont.
   A. Introduction to Differentiable Physics
   B. Gradient based optimisation of computer programs
   C. Training NNs with „the simulator in the loop“
   D. reducing numerical errors with deep learning
   E. solving inverse problems with NNs

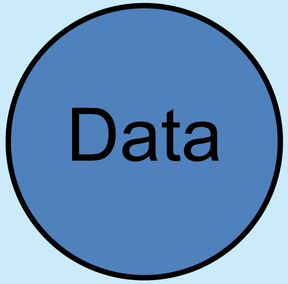
WHERE TO FIND THE LECTURE MATERIAL

https://github.com/TobiBu/graddays

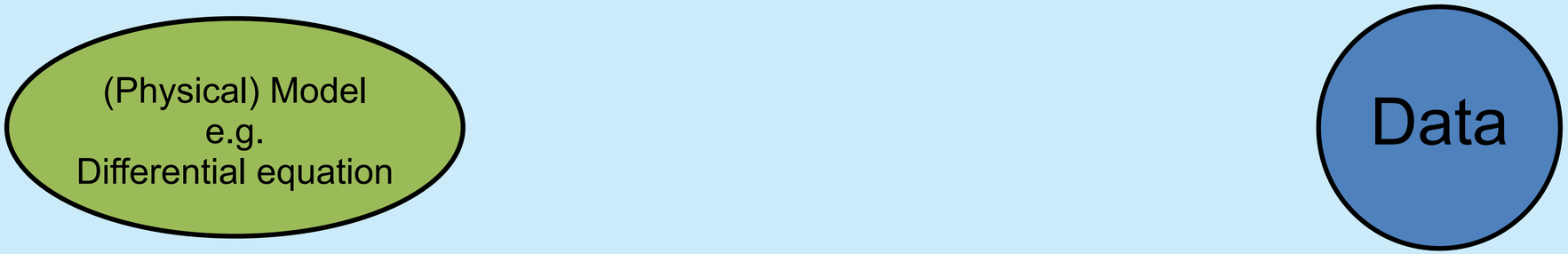
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?

For Fun!

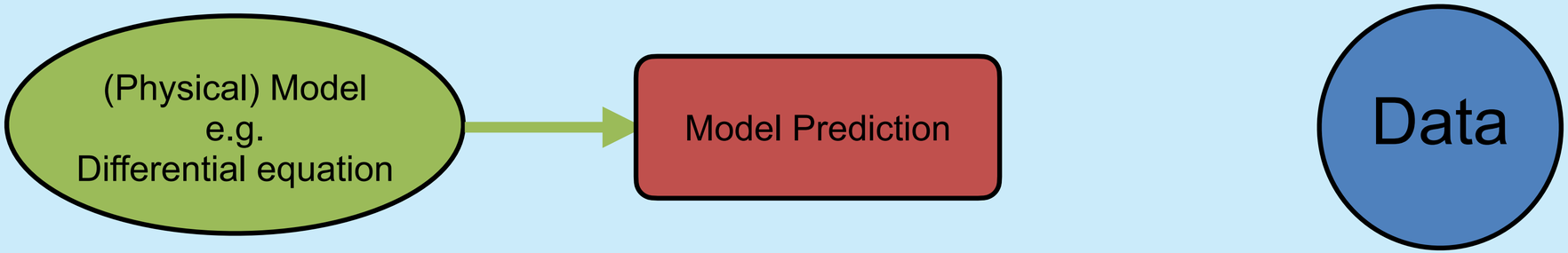
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?



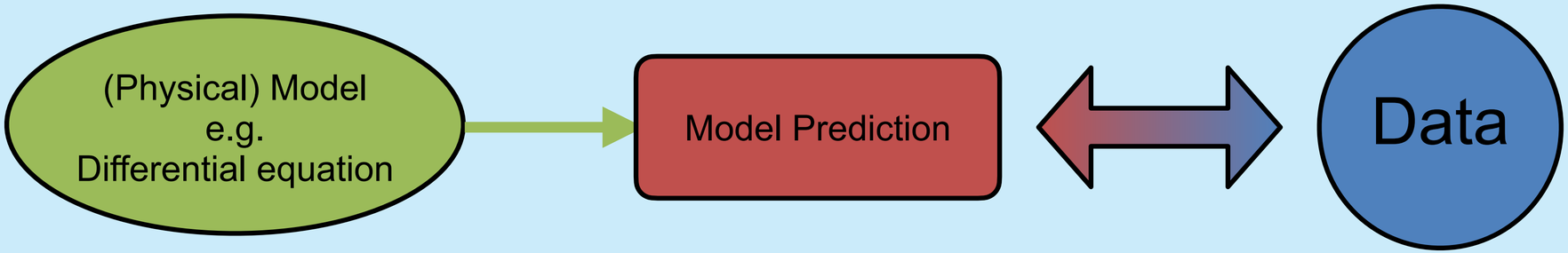
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?



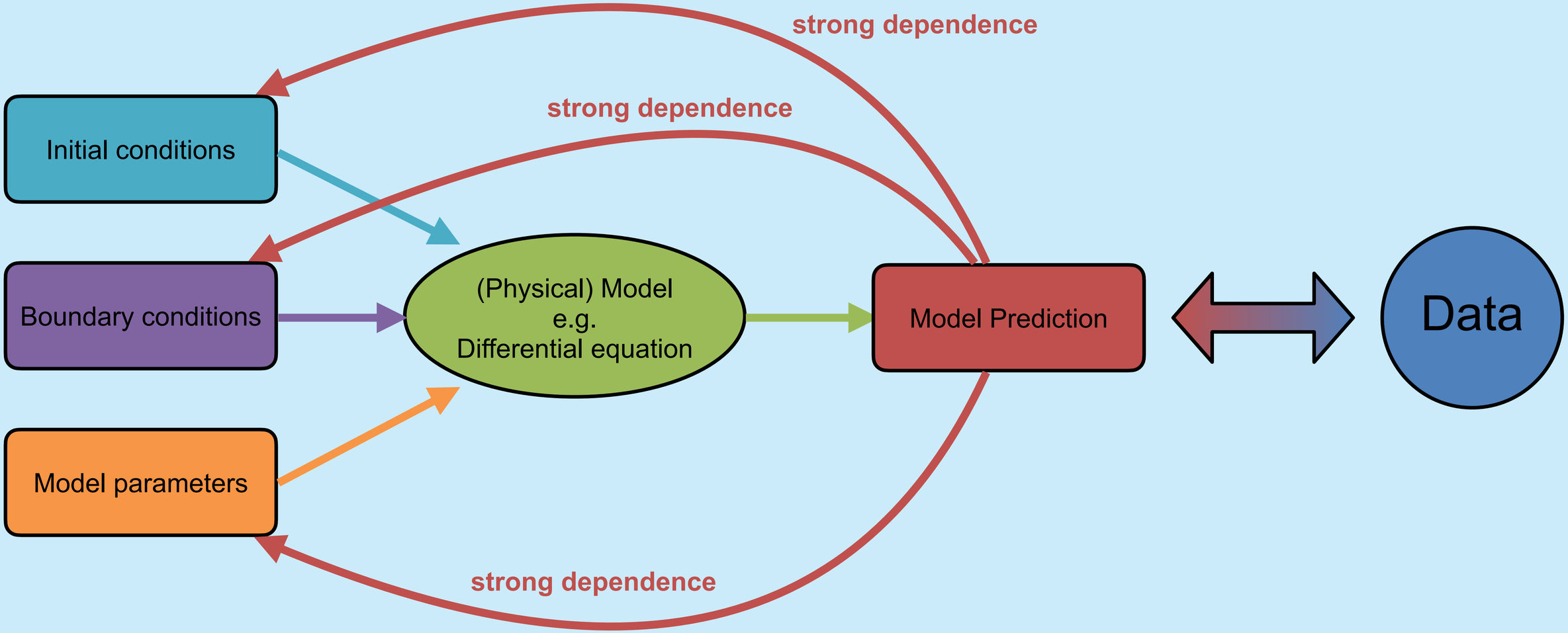
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?



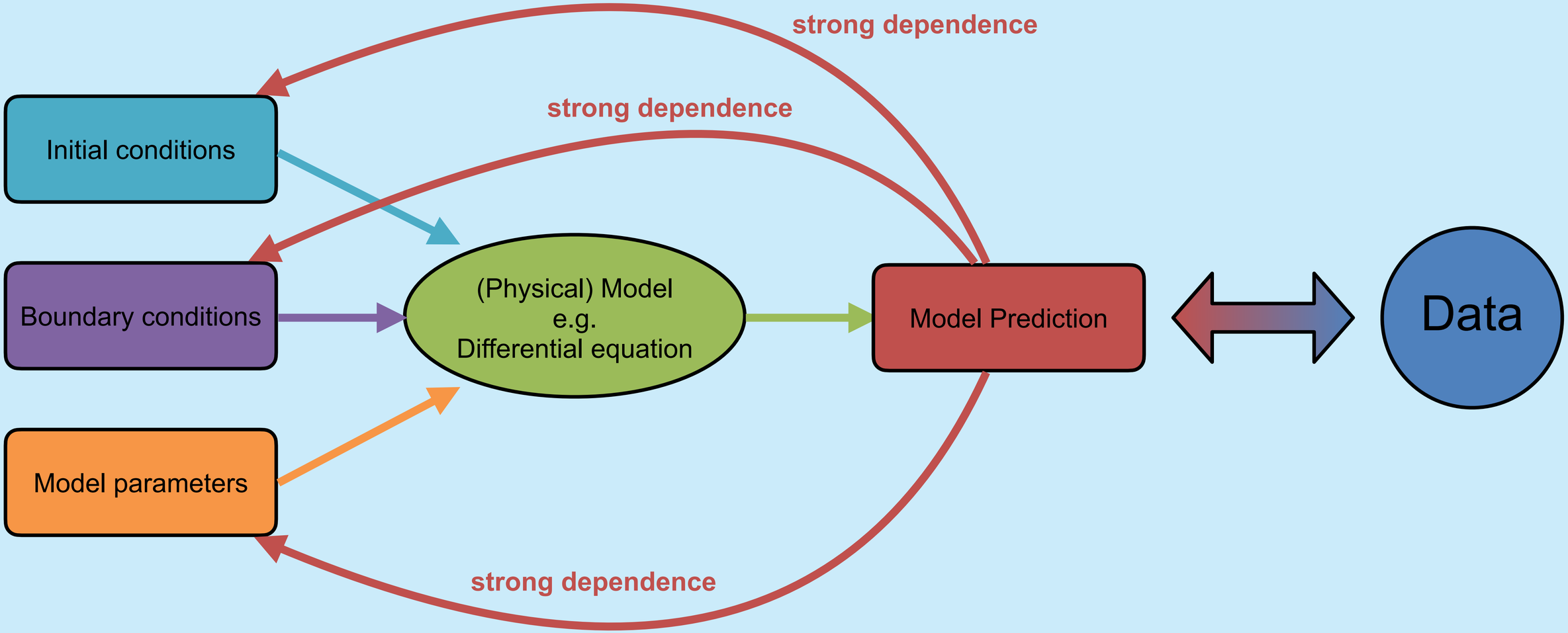
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?



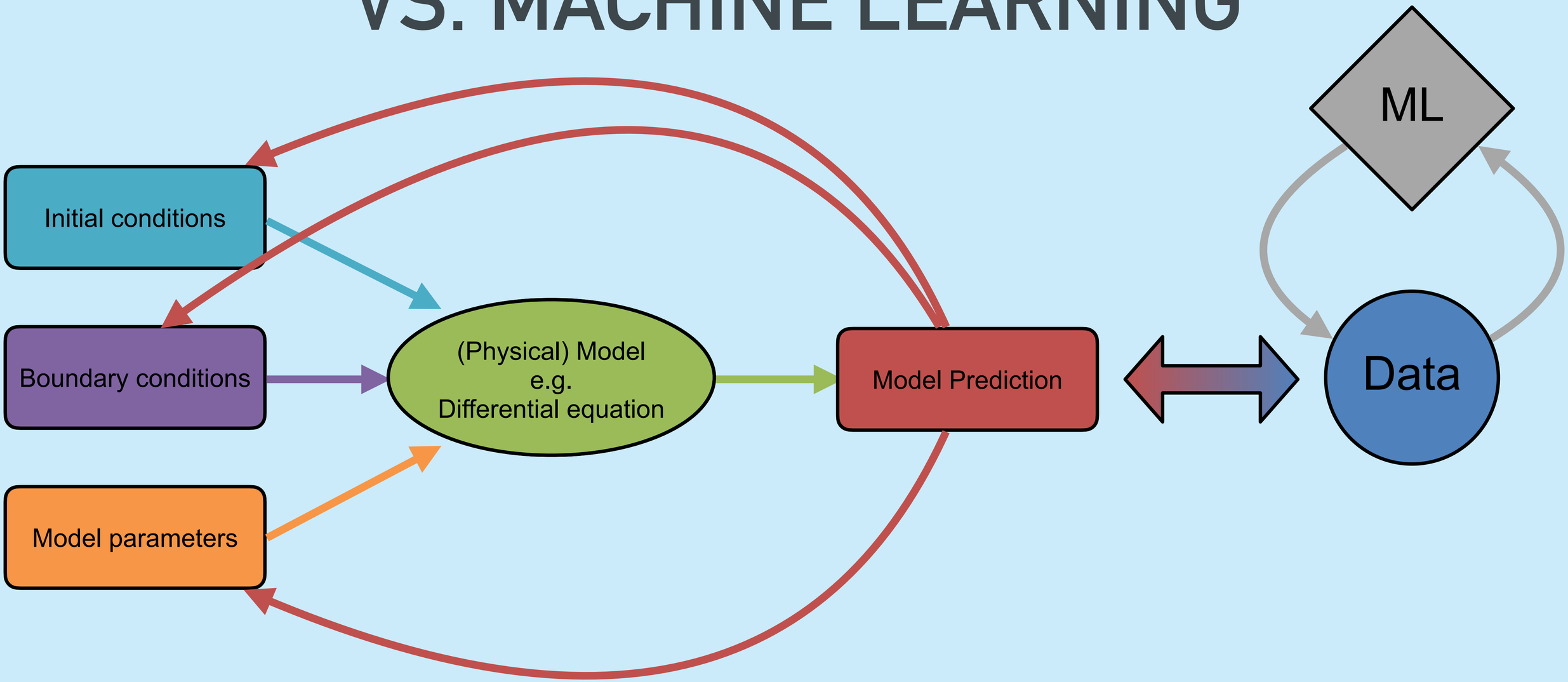
WHY DO WE WANT A DIFFERENTIABLE PHYSICS SIMULATOR?



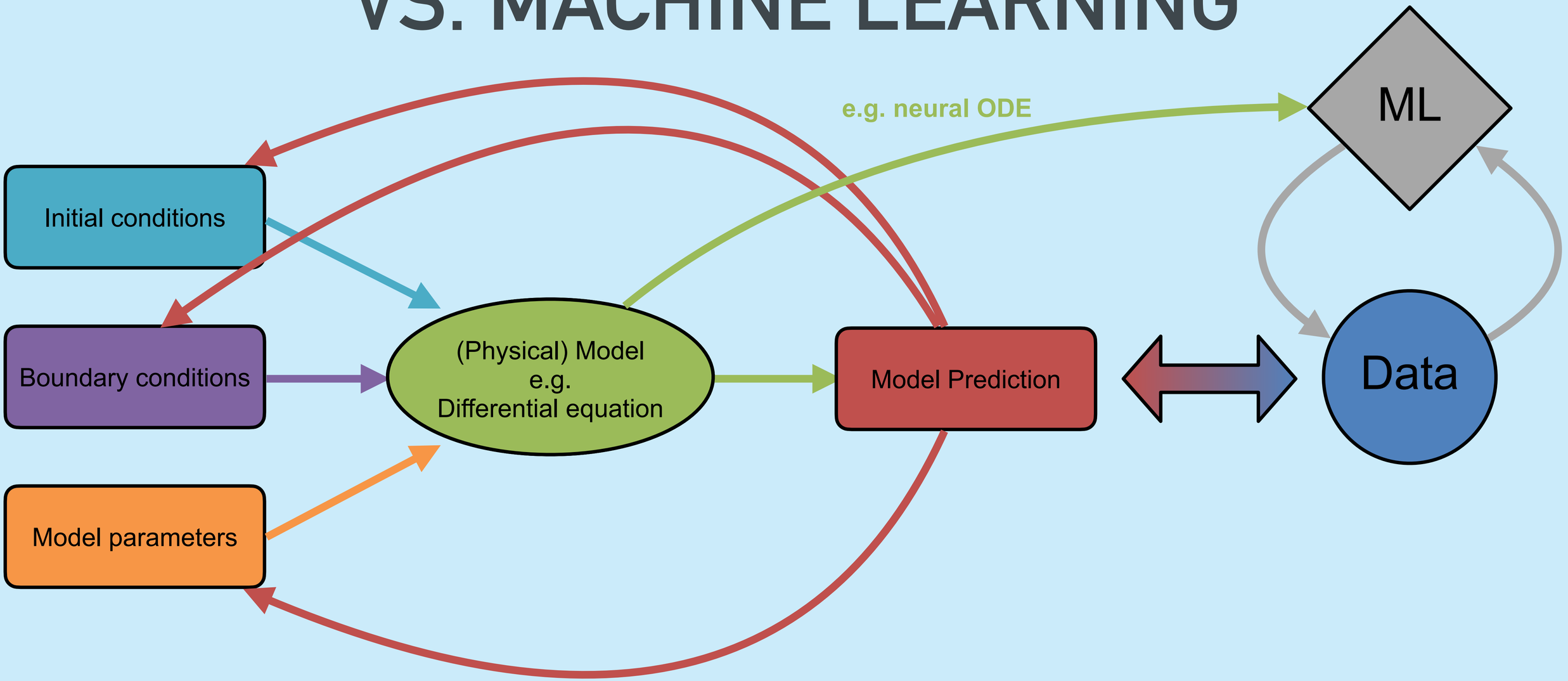
CAN WE INVERT SOLVERS TO OPTIMISE ICS, BCS OR FIND OPTIMAL PARAMETERS?



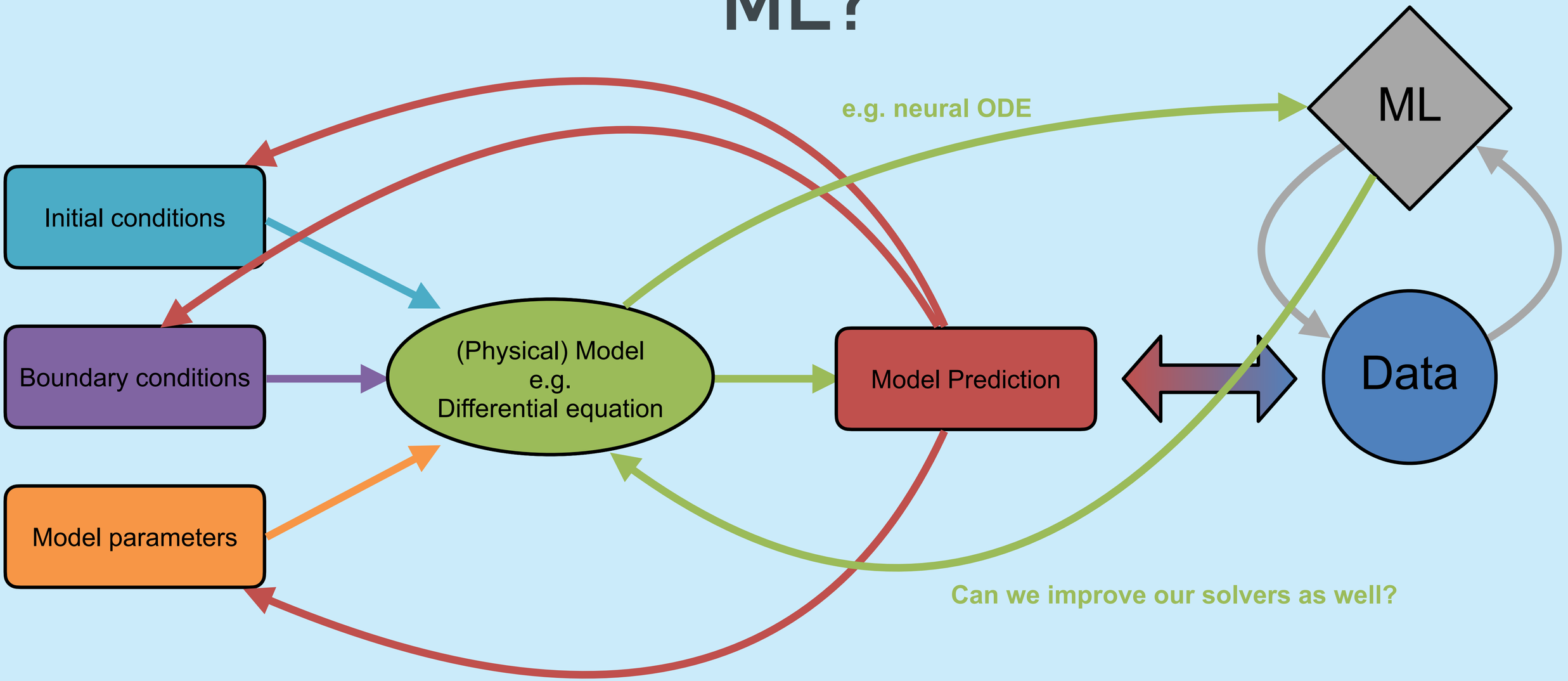
DIFFERENTIABLE PHYSICS SIMULATOR VS. MACHINE LEARNING

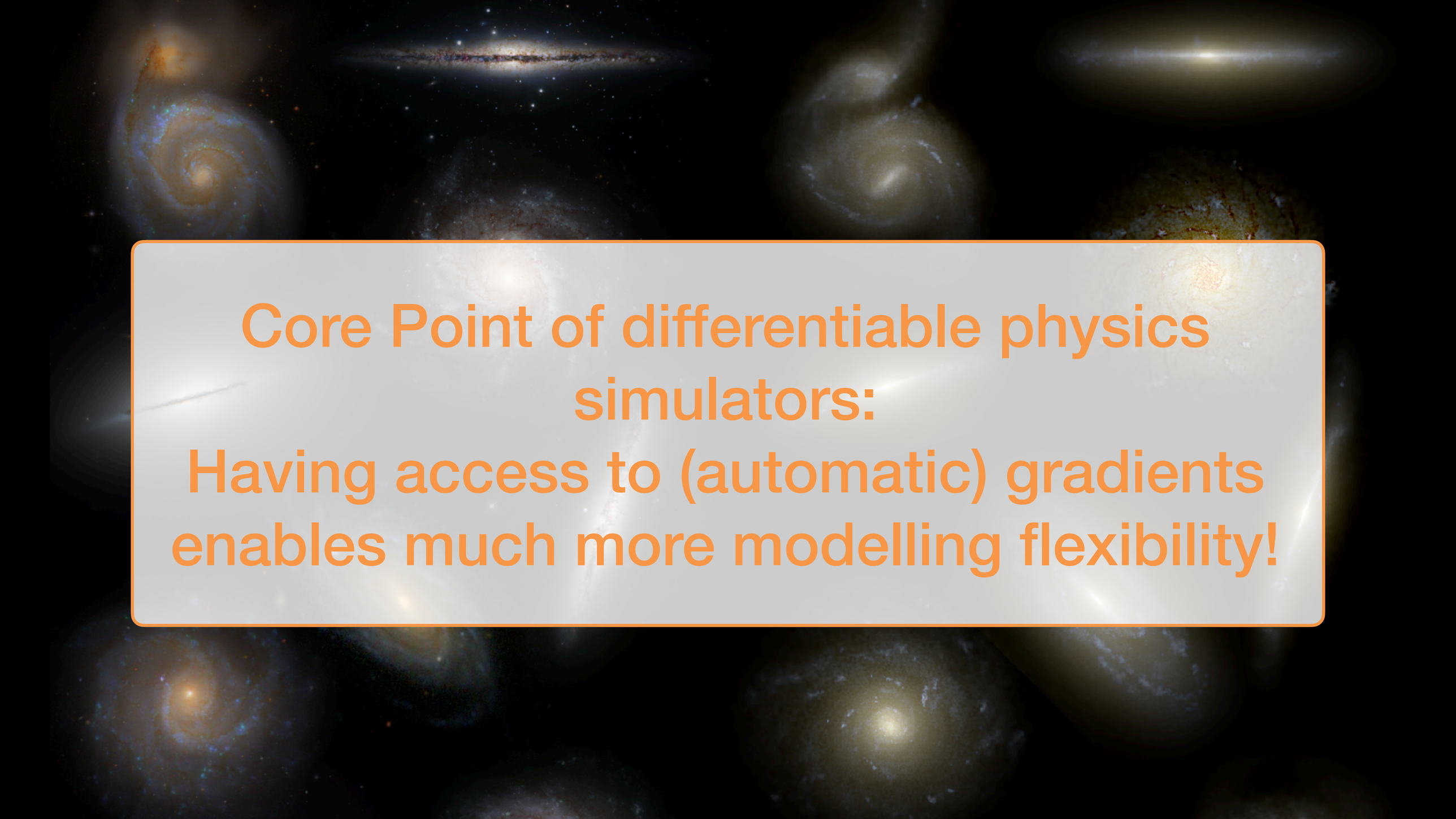


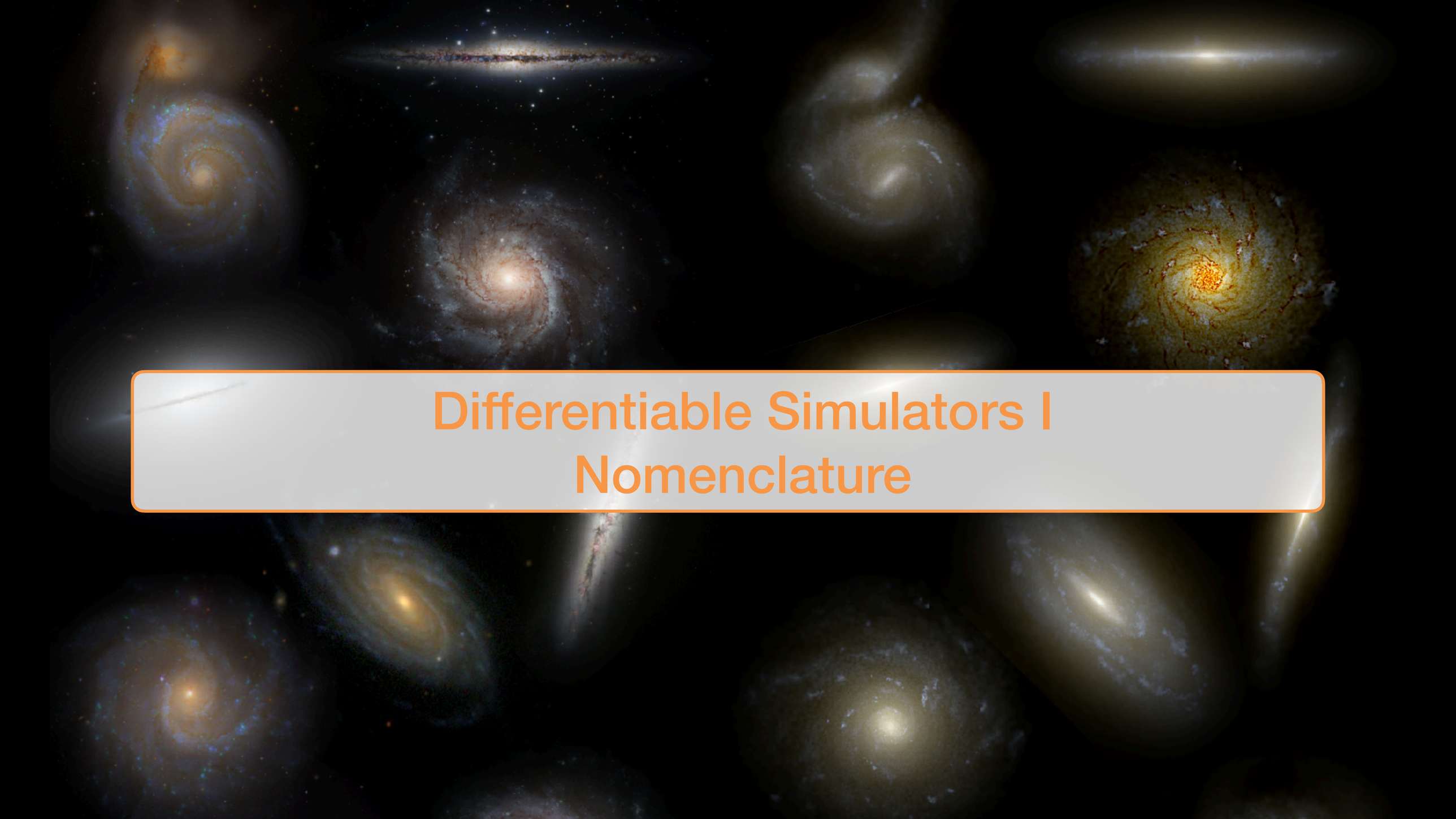
DIFFERENTIABLE PHYSICS SIMULATOR VS. MACHINE LEARNING



CAN WE IMPROVE OUR SOLVERS WITH ML?




Core Point of differentiable physics simulators:
Having access to (automatic) gradients enables much more modelling flexibility!



Differentiable Simulators I
Nomenclature

HOW DO WE BUILD DIFFERENTIABLE SIMULATORS?

remember autodiff from monday?

For all practical purposes, just rely on packages such as JAX, pytorch, Julia, phiflow, …

and make use of their autodiff frameworks.

however, if you want to go deeper:

differentiable operators and implicit function theorem

https://physicsbaseddeeplearning.org/diffphys.html
Thürey et al. 2022

https://arxiv.org/pdf/2406.09699
Sapienza et al. 2024

WHAT DO WE WANT TO DIFFERENTIATE?

In the context of ML: A loss function

$$L(\theta) = \frac{1}{2} \left\| u(t_1; \theta) - u^{\text{target}}(t_1) \right\|_2^2$$

In the context of statistics or physics:
The likelihood function $\mathscr{L}$ or the posterior distribution

Having gradients of $\mathscr{L}$ enables more efficient sampling methods such as Hamilton Monte Carlo sampling instead of MCMC and even improves SBI

Simply a quantity of interest:
quantifying how the solution of the differential equation changes as we vary the parameter values

SOME MORE NOMENCLATURE

$$L(\theta) = \frac{1}{2} \left\| u(t_1; \theta) - u^{\text{target}}(t_1) \right\|_2^2$$

Gradient based optimisation:

Find θ such that $\mathscr{L}(\theta)$ is minimised (e.g. using ADAM)

$$\theta^{m+1} = \theta^m - \alpha_m \frac{dL}{d\theta}(\theta^m),$$

SOME MORE NOMENCLATURE

$$L(\theta) = \frac{1}{2} \left\| u(t_1; \theta) - u^{\text{target}}(t_1) \right\|_2^2$$

Sensitivity matrix:

gradient of loss is typically given by

$$\frac{dL}{d\theta} = \frac{\partial L}{\partial u} \frac{\partial u}{\partial \theta} + \frac{\partial L}{\partial \theta}.$$

which results in

$$\frac{\partial L}{\partial u} = u - u^{\text{target}}(t_1) \qquad \frac{\partial L}{\partial \theta} = 0.$$

since L does not depend on θ directly.

complicated derivative is $\frac{\partial u}{\partial \theta}$, also called sensitivity.

SOME MORE NOMENCLATURE

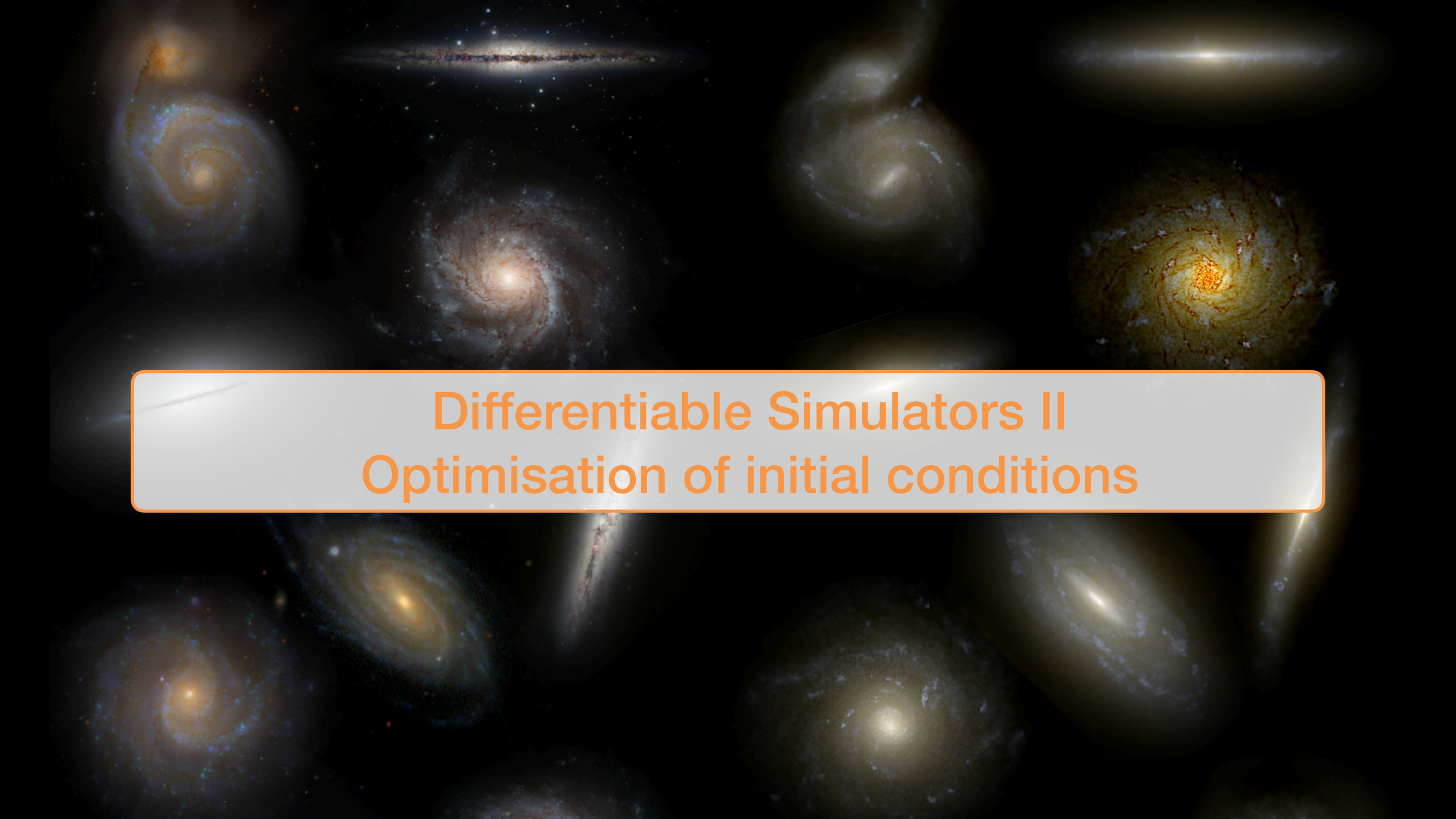
$$\text{sensitivity } s = \frac{\partial u}{\partial \theta}$$

$$s = \frac{\partial u}{\partial \theta} = \begin{bmatrix} \frac{\partial u_1}{\partial \theta_1} & \cdots & \frac{\partial u_1}{\partial \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_n}{\partial \theta_1} & \cdots & \frac{\partial u_n}{\partial \theta_p} \end{bmatrix} \in \mathbb{R}^{n \times p}.$$

sensitivity s is a Jacobian, the matrix of first derivatives of a vector-valued function

product sv, with $v \in \mathbb{R}^p$, is the directional derivative of the function $u(\theta)$ in the direction v

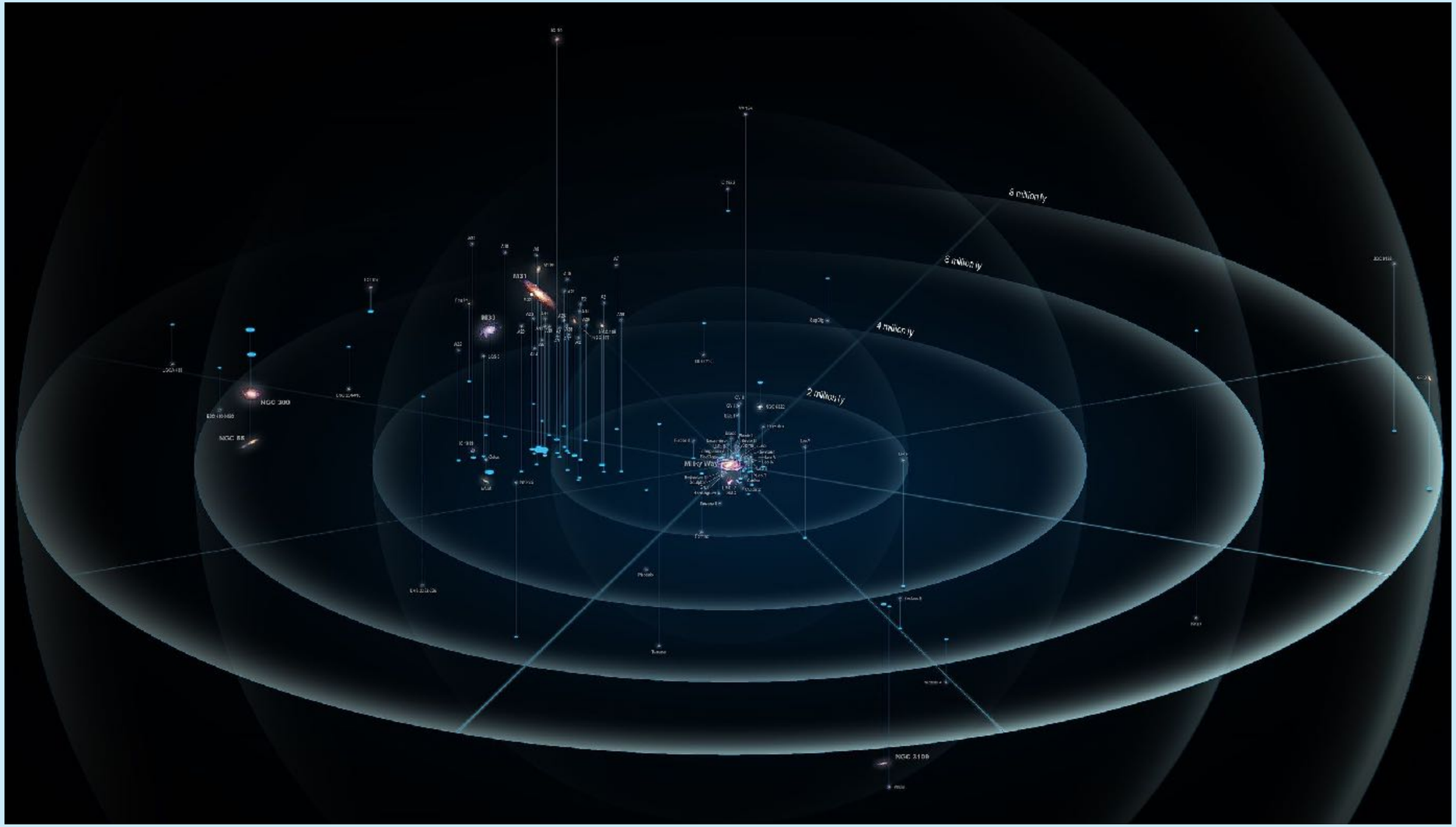
$$sv = \frac{\partial u}{\partial \theta} v = \lim_{h \to 0} \frac{u(\theta + hv) - u(\theta)}{h},$$



Differentiable Simulators II
Optimisation of initial conditions

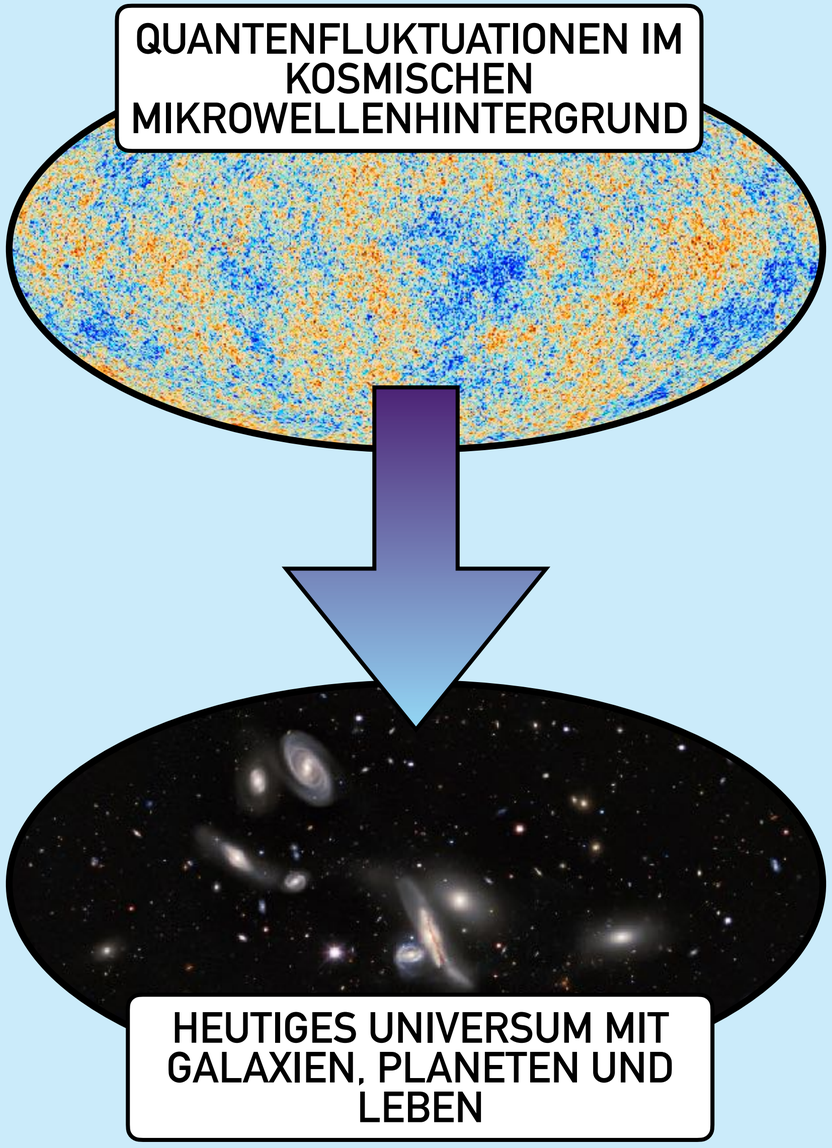
OPTIMISING INITIAL CONDITIONS

One important question in cosmology: what are the ICs of the local group?



OPTIMISING INITIAL CONDITIONS

One important question in cosmology: what are the ICs of the local group?

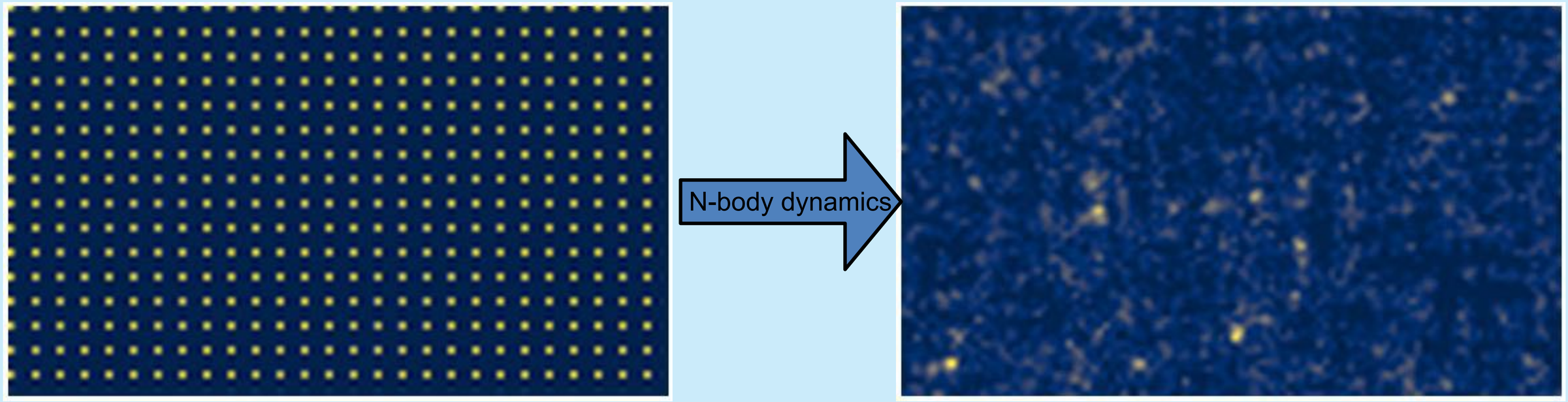


PMWD: A COSMOLOGICAL DIFFERENTIABLE GRAVITY SOLVER

https://github.com/eelregit/pmwd/tree/master?tab=readme-ov-file
written in JAX

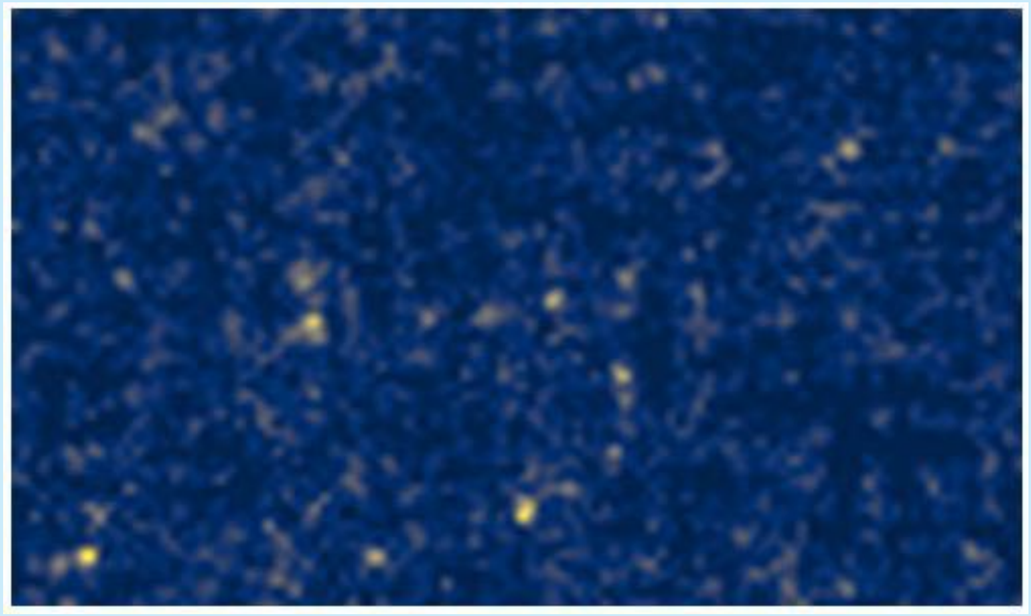
PMWD: A COSMOLOGICAL DIFFERENTIABLE GRAVITY SOLVER

Example:
16×27×16=6912 particles, in a 160 × 270 × 160 Mpc^3/h^3 box, evolving with a 2x mesh (32×54×32) from a=1/64 to a=1 to match the 2D projection to the target image below at 4x resolution (64×108)



PMWD: LOSS FUNCTION

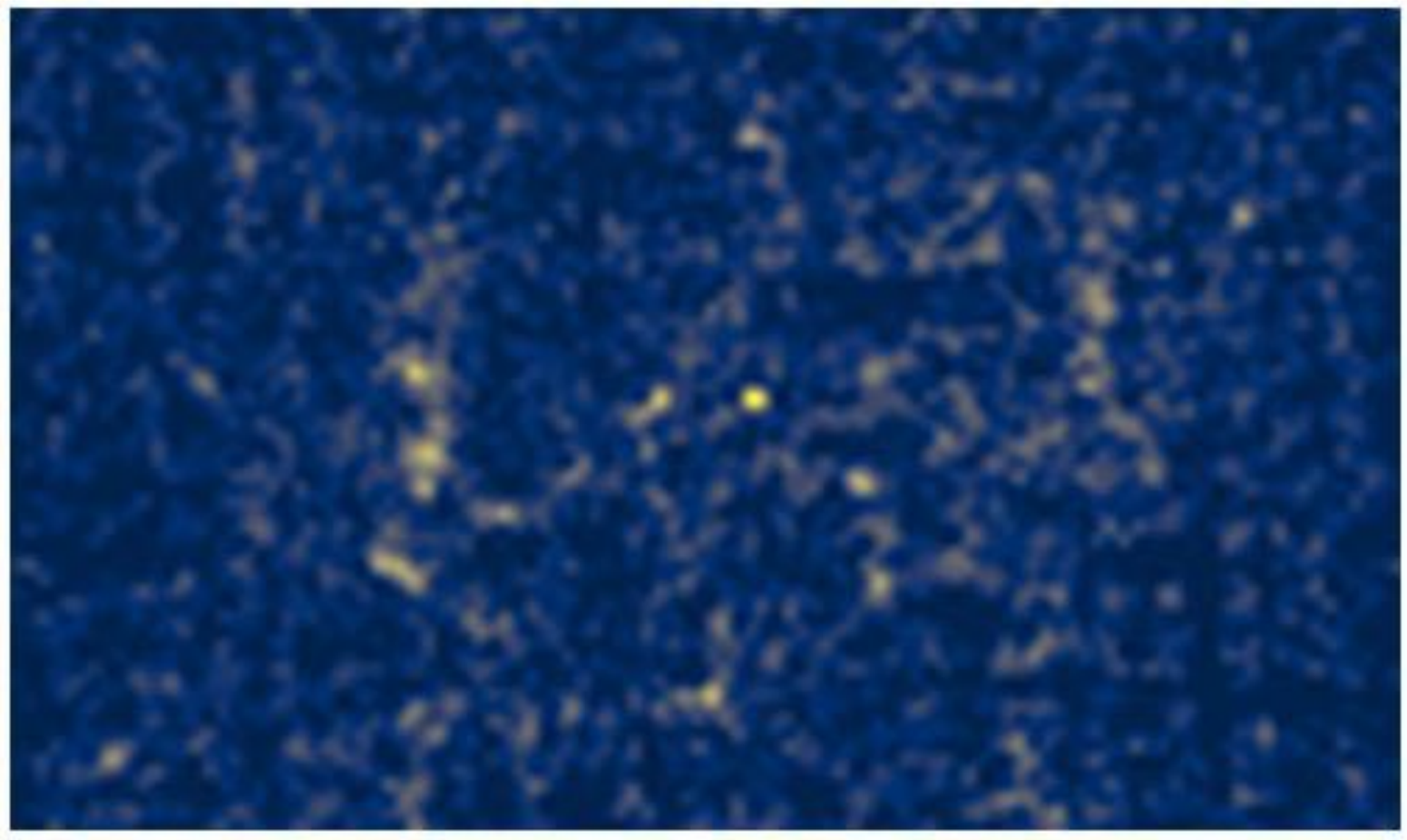
Example:
16×27×16=6912 particles, in a 160 × 270 × 160 Mpc^3/h^3 box, evolving with a 2x mesh (32×54×32) from a=1/64 to a=1 to match the 2D projection to the target image below at 4x resolution (64×108)



$$- \boxed{pmwd} = 0$$

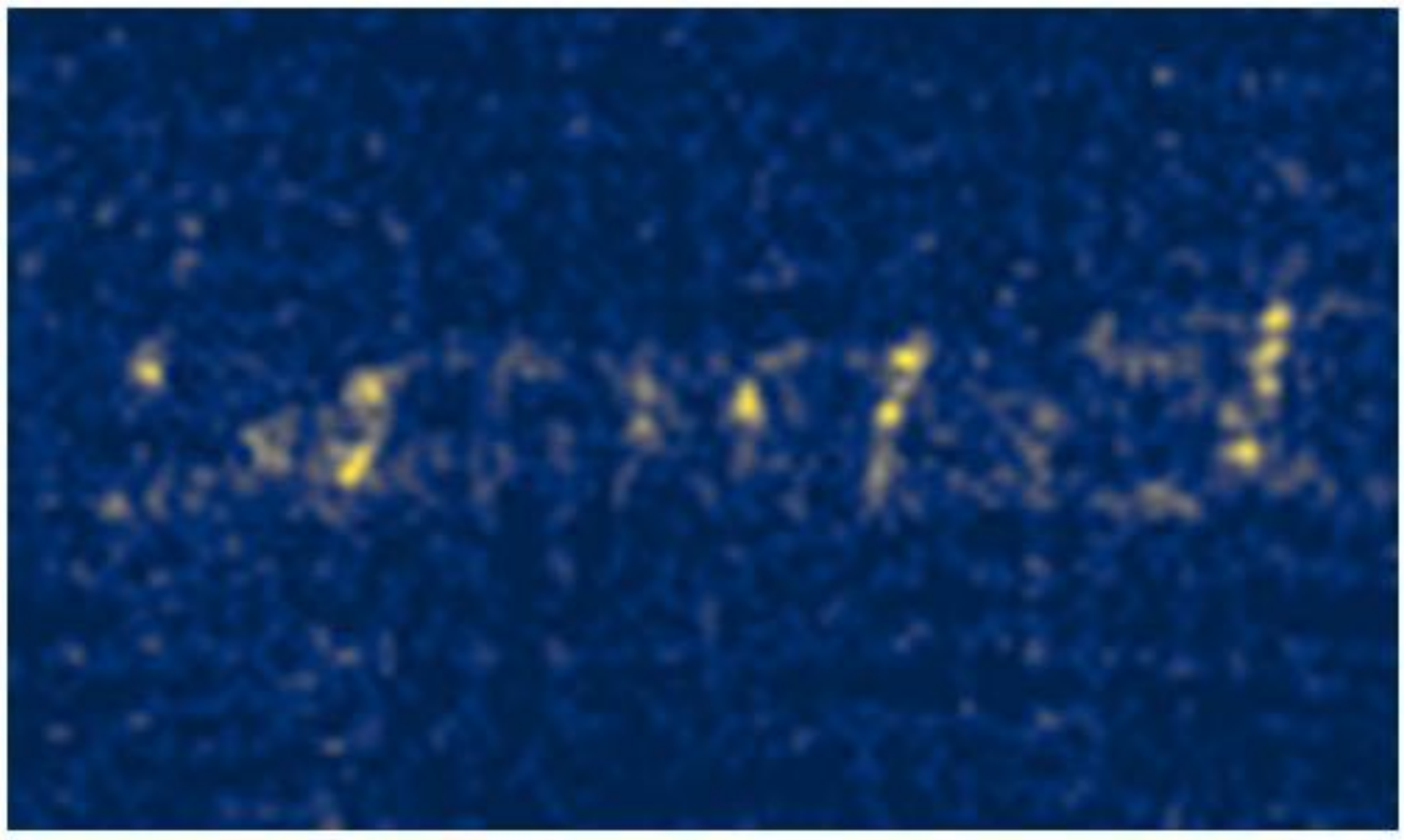
PMWD: ONE GRADIENT STEP

pull gradient through gravity solve, update ICs and rerun simulation!



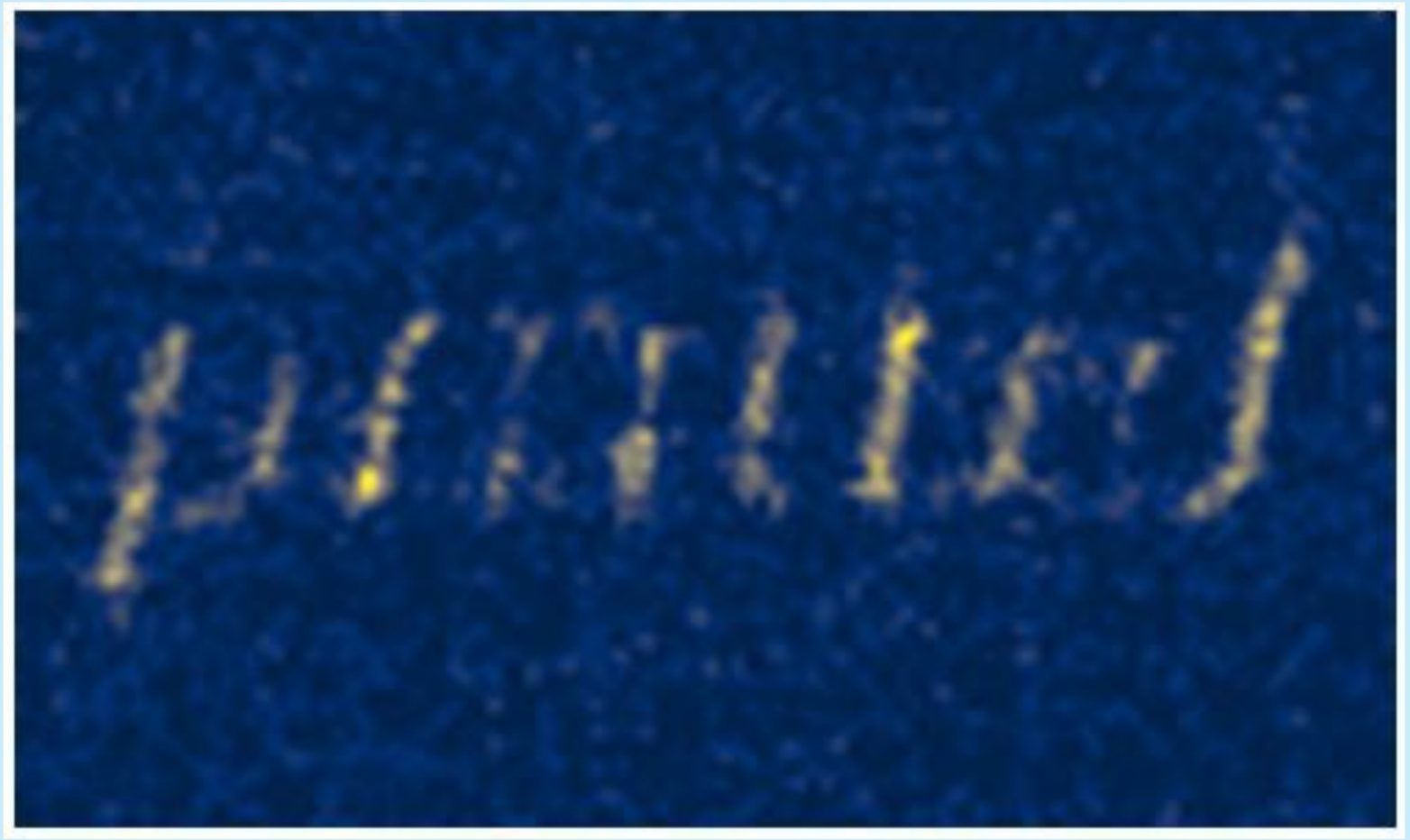
PMWD: TEN GRADIENT STEPS

pull gradient through gravity solve, update ICs and rerun simulation!



PMWD: 30 GRADIENT STEPS

pull gradient through gravity solve, update ICs and rerun simulation!



PMWD: 100 GRADIENT STEPS

pull gradient through gravity solve, update ICs and rerun simulation!

PMWD: 300 GRADIENT STEPS

pull gradient through gravity solve, update ICs and rerun simulation!

PMWD: 1000 GRADIENT STEPS

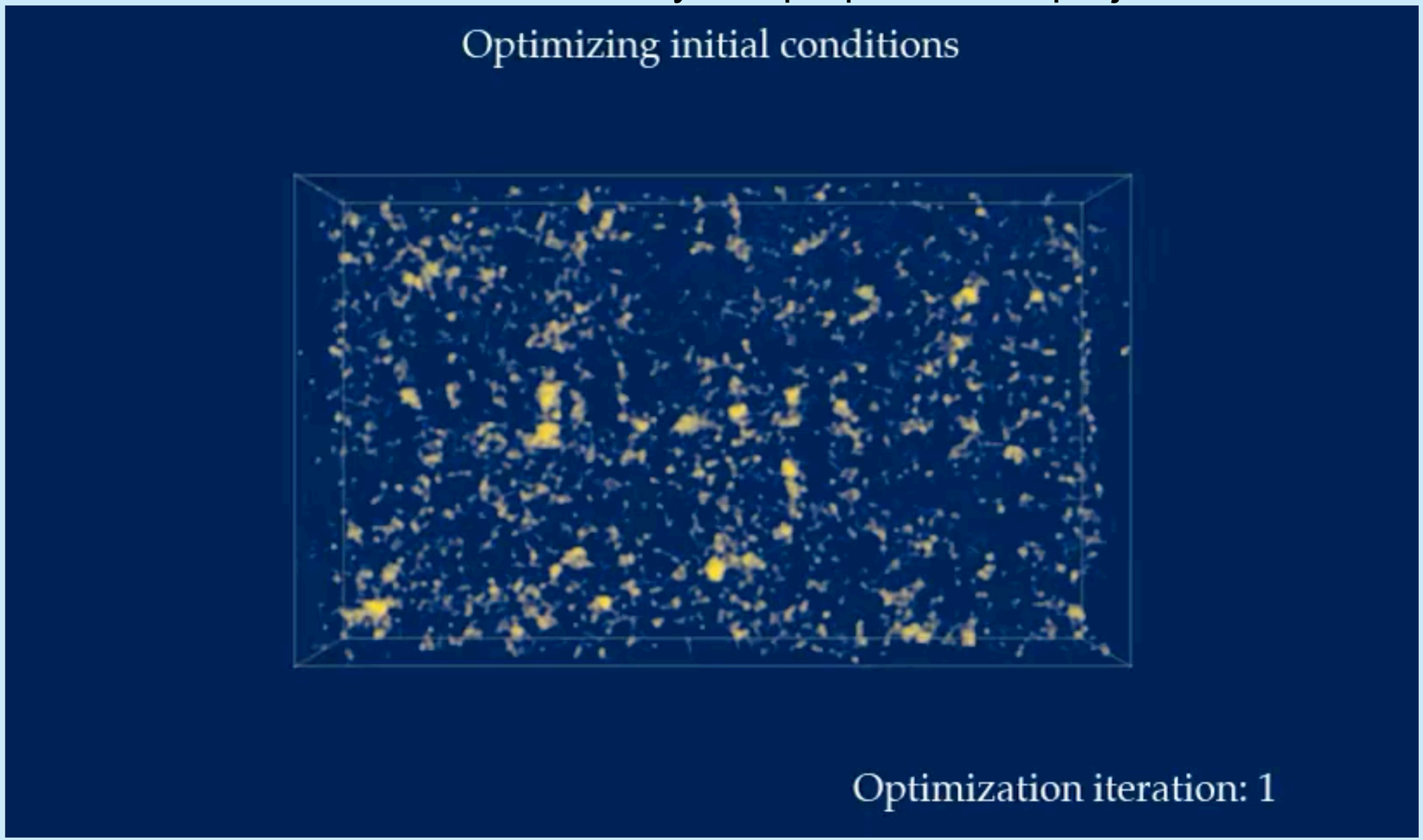
pull gradient through gravity solve, update ICs and rerun simulation!

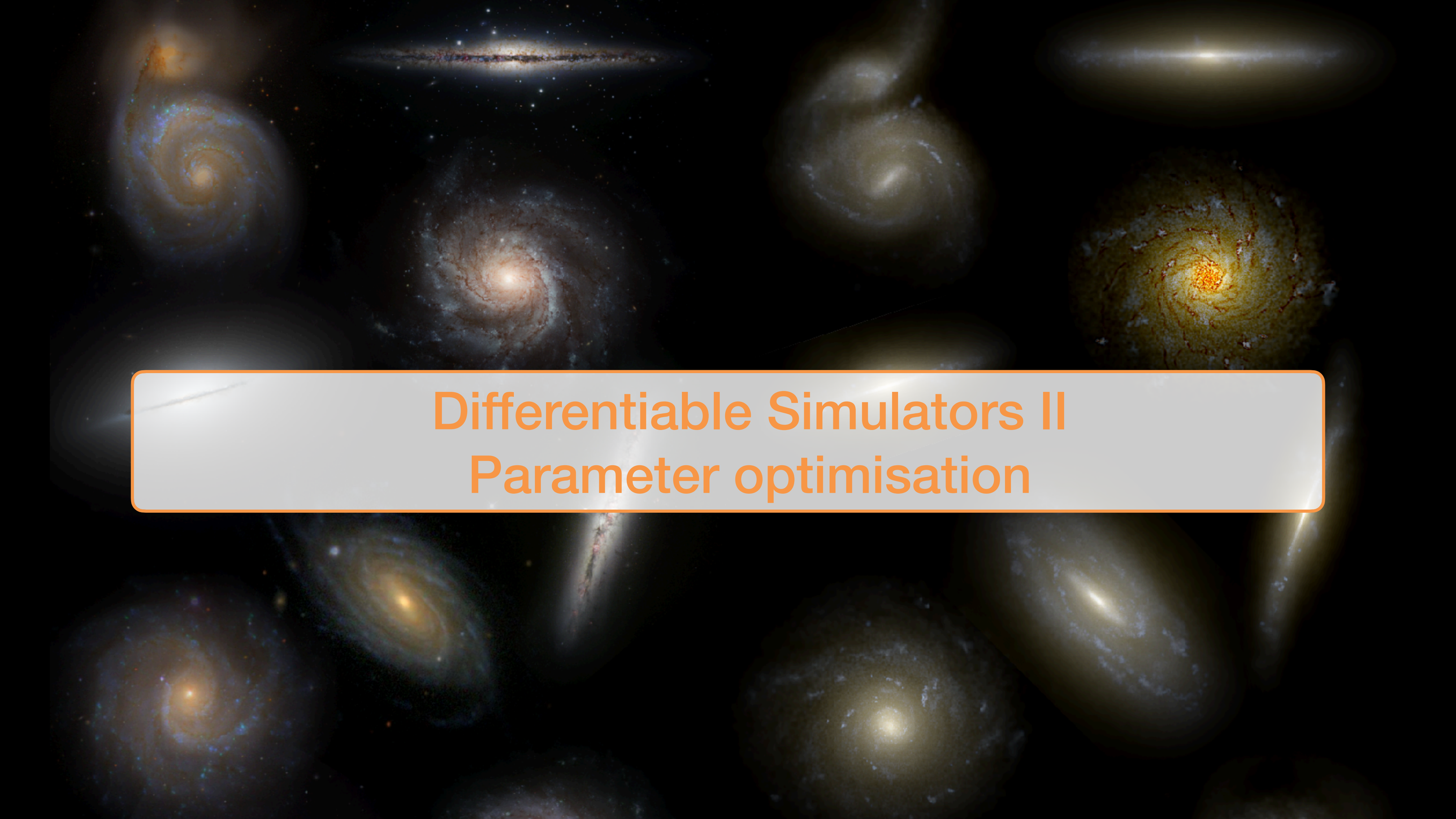
PMWD: 3000 GRADIENT STEPS

pull gradient through gravity solve, update ICs and rerun simulation!

PMWD: TWO PERPENDICULAR CONSTRAINTS!

let the ICs be constrained by two perpendicular projections.





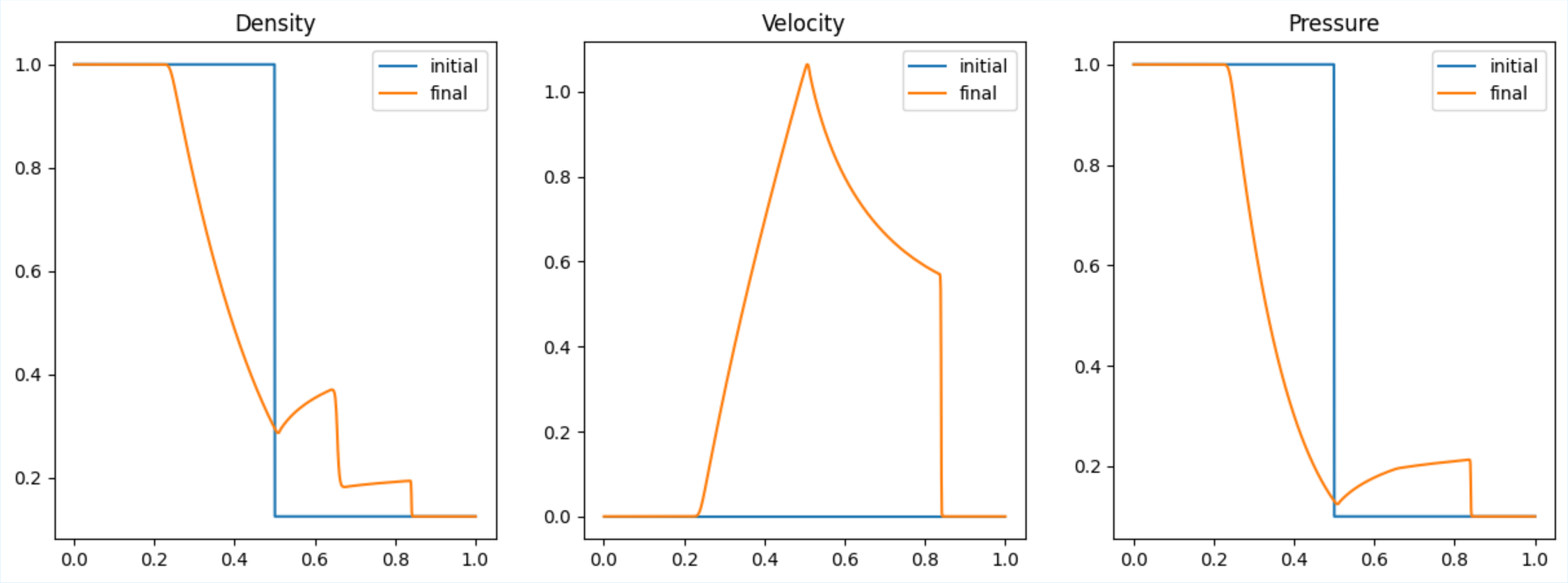
Differentiable Simulators II
Parameter optimisation

JF1UIDS: A 1D RADIAL FLUID SOLVER WRITTEN IN JAX

https://github.com/leo1200/jf1uids/tree/dev

Leonard Storcks

simple shock example



JF1UIDS: A 1D RADIAL FLUID SOLVER WRITTEN IN JAX

https://github.com/leo1200/jf1uids/tree/dev

Leonard Storcks

simple stellar wind example

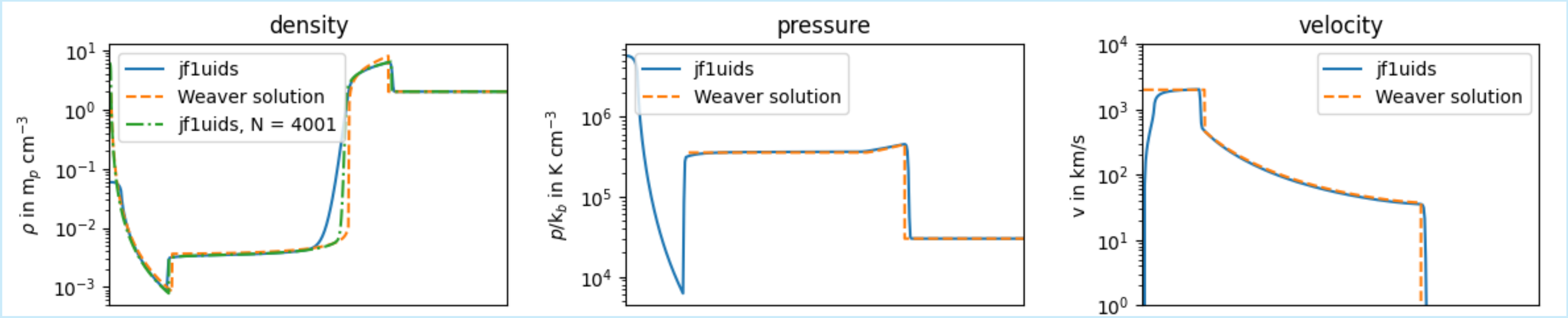
JF1UIDS: A 1D RADIAL FLUID SOLVER WRITTEN IN JAX

https://github.com/leo1200/jf1uids/tree/dev

Leonard Storcks

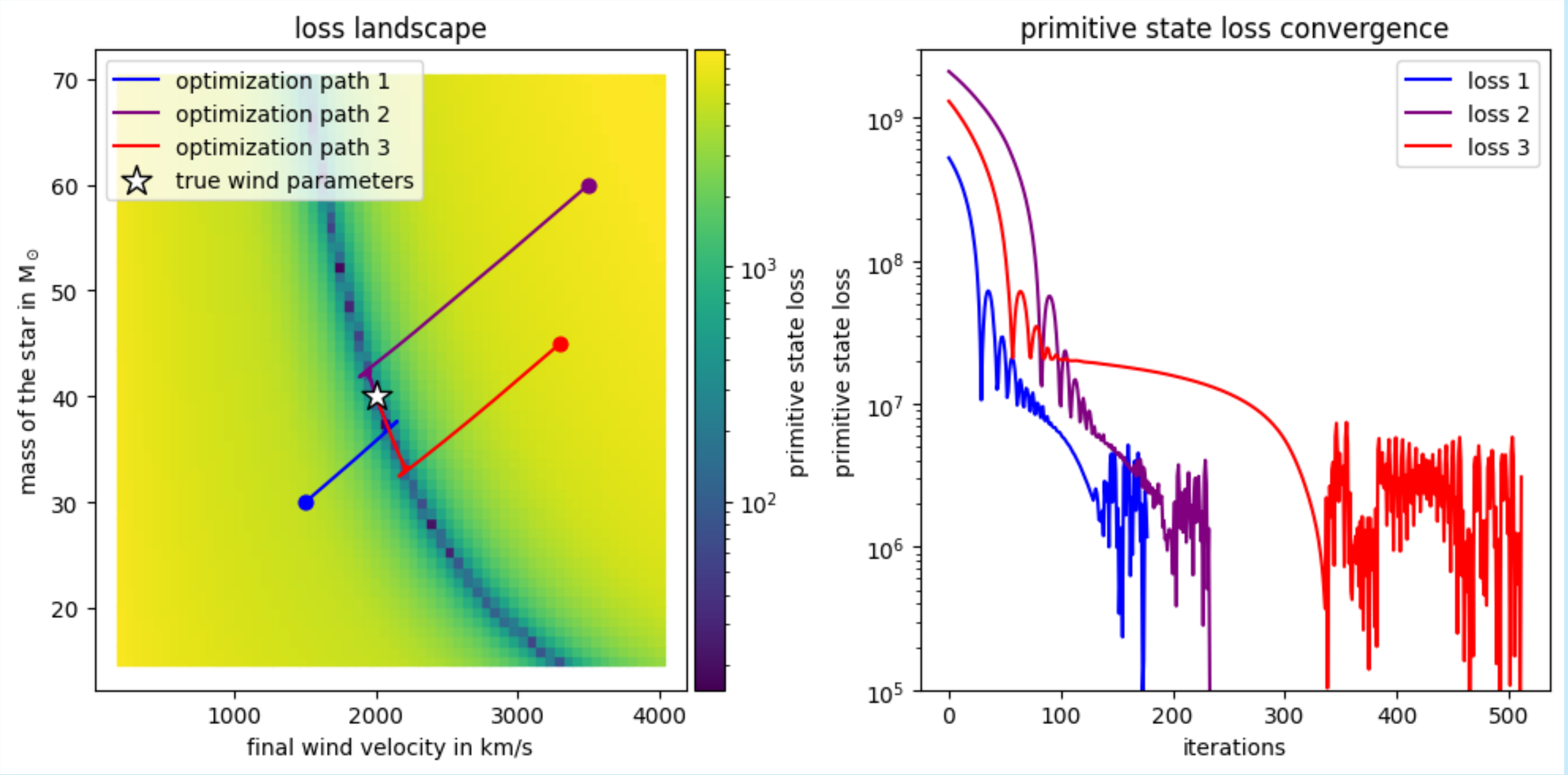
simple stellar wind example

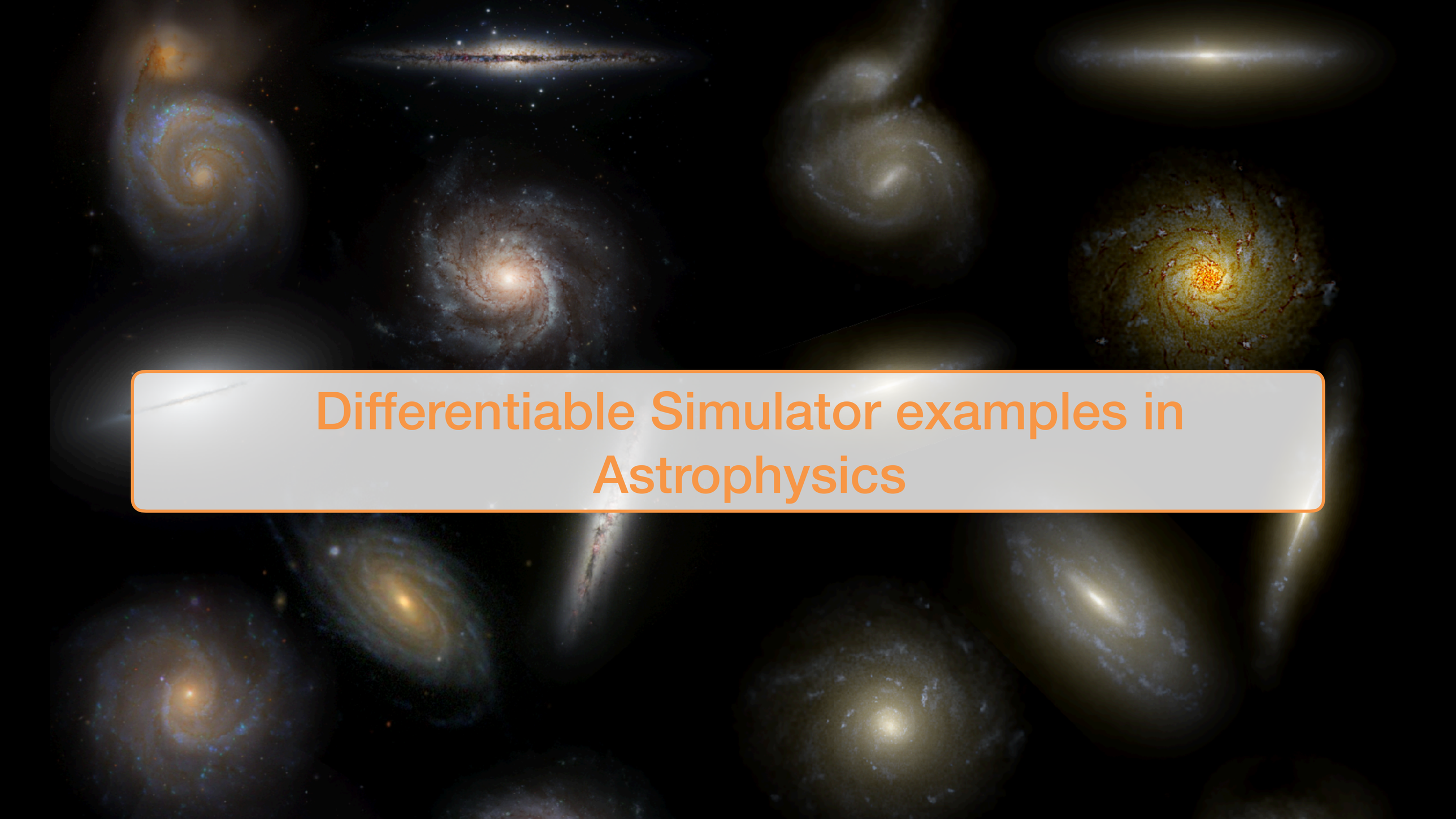
homogeneous medium of 2 particle per ccm
40 Msun star with 2000 km/s terminal wind velocity and mass loss rate of ~0.1 Msun/Myr



JF1UIDS: WIND PARAMETER OPTIMISATION

objective:
start from unknown stellar wind parameters and recover true parameters just from final simulation state



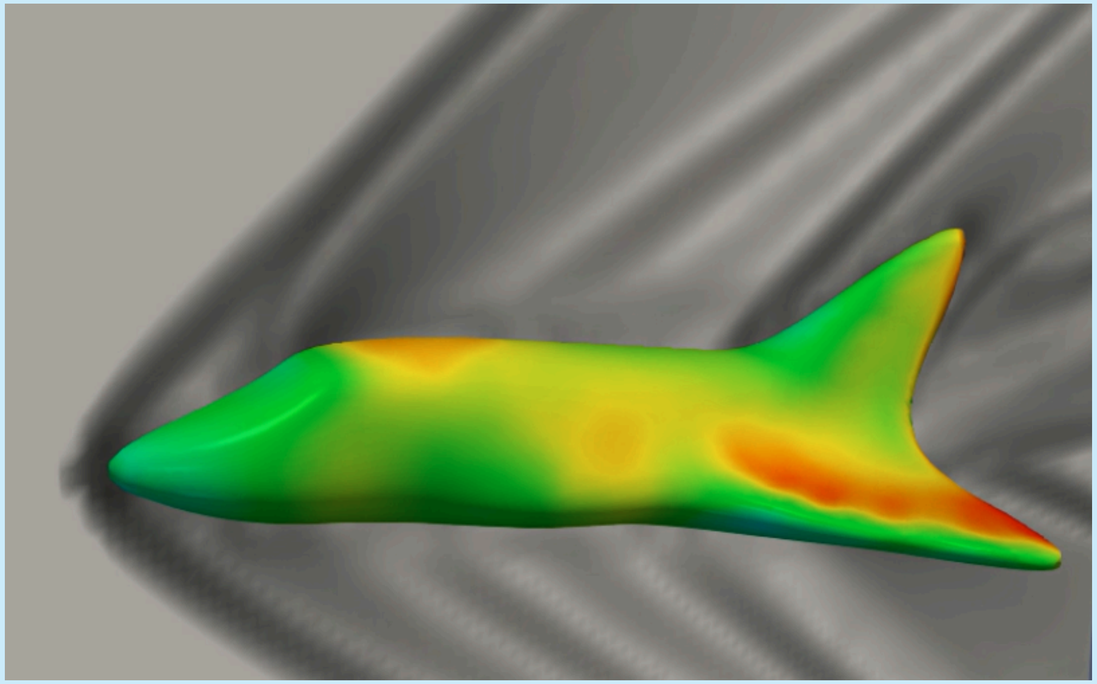


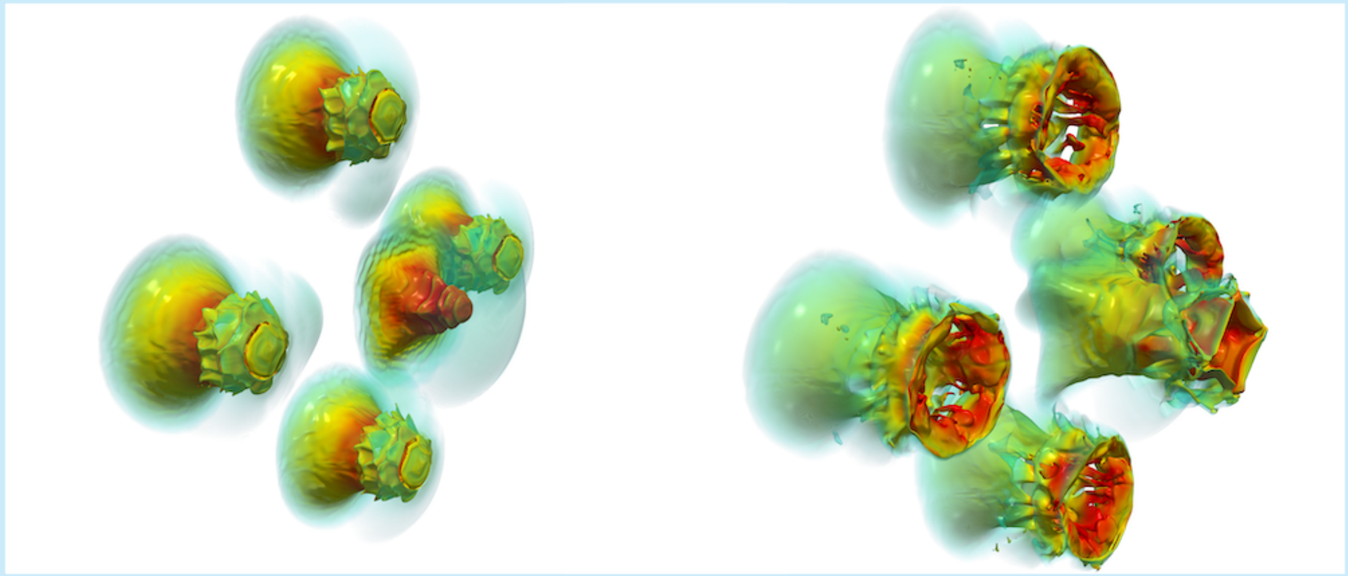
Differentiable Simulator examples in Astrophysics

JAXFLUIDS

https://github.com/tumaer/JAXFLUIDS?tab=readme-ov-file

general purpose fluid solver written in jax



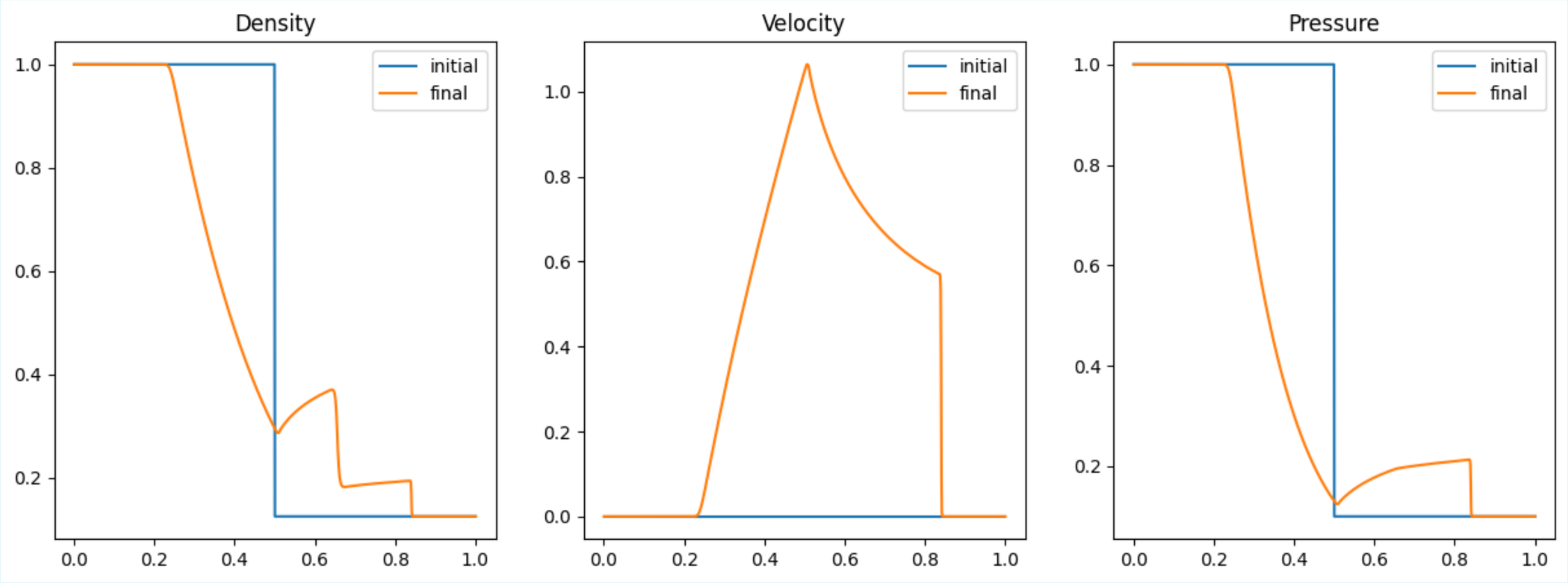


JF1UIDS: A 1D RADIAL FLUID SOLVER WRITTEN IN JAX

https://github.com/leo1200/jf1uids/tree/dev

Leonard Storcks

simple shock example



PMWD: A COSMOLOGICAL DIFFERENTIABLE GRAVITY SOLVER

https://github.com/eelregit/pmwd/tree/master?tab=readme-ov-file
written in JAX

JAX-COSMO

https://github.com/DifferentiableUniverseInitiative/jax_cosmo

differentiable cosmology library

DIFFSTAR

https://github.com/ArgonneCPAC/diffstar

differentiable library for simple stellar population synthesis

DIFFMAH

https://github.com/ArgonneCPAC/diffmah

differentiable model for mass assembly history of dark matter halos

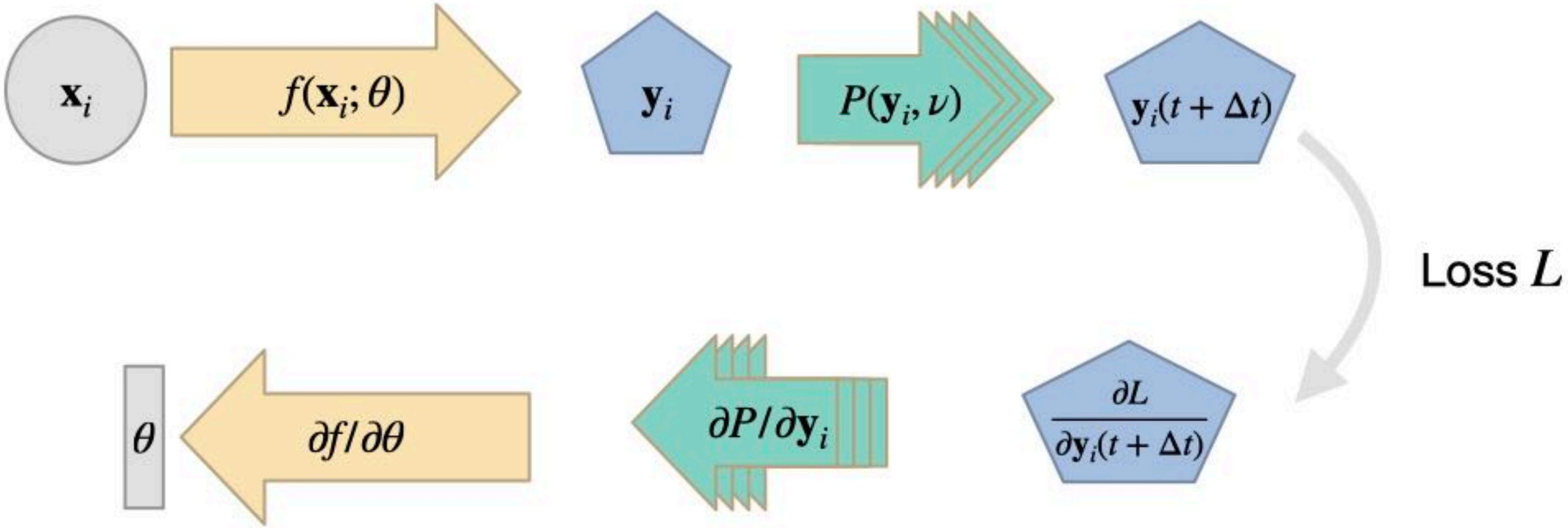
RUBIX

Differentiable simulator for Integral field spectroscopic observations of galaxies
(Çakir, Schaible, Buck + 2024)

Differentiable Simulators III
Combining DP and NN training

DIFFERENTIABLE SIMULATORS WITH NN

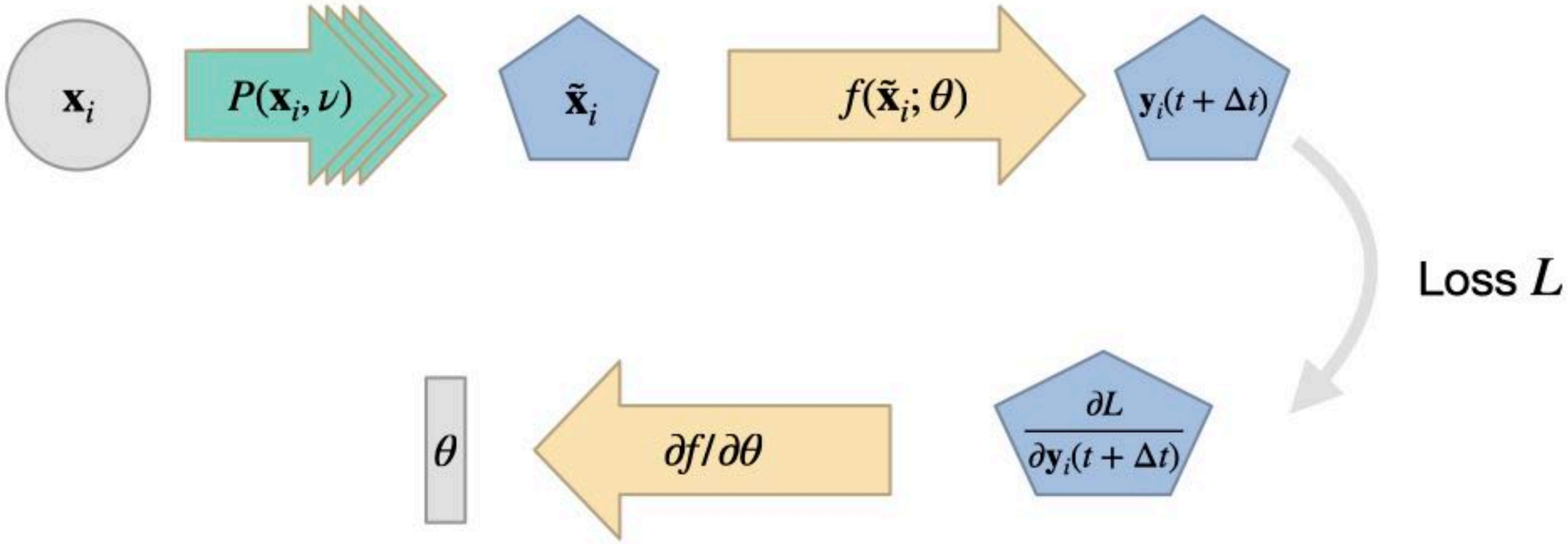
DP approach I: A network produces an input to a PDE solver $\mathscr{P}$, which provides a gradient for training during the backpropagation step



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

DIFFERENTIABLE SIMULATORS WITH NN

A PDE solver produces an output which is processed by an NN

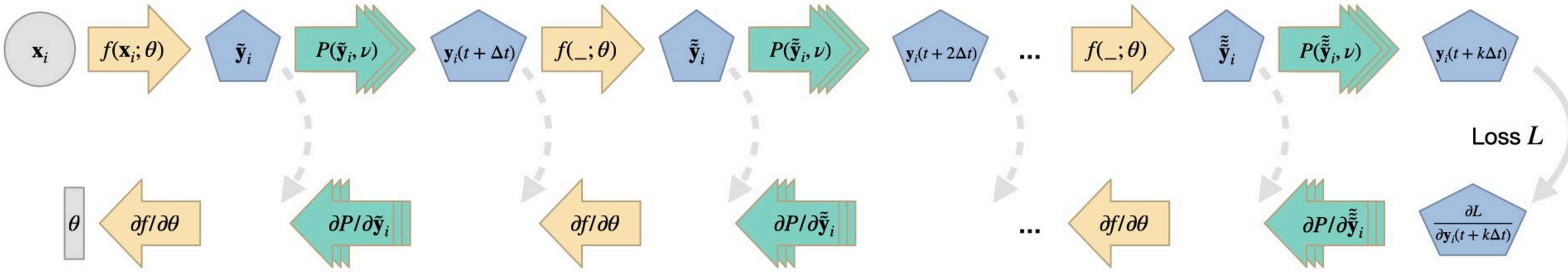


taken from https://physicsbaseddeeplearning.org/diffphys-examples.html

Thürey+2022

DIFFERENTIABLE SIMULATORS WITH NN

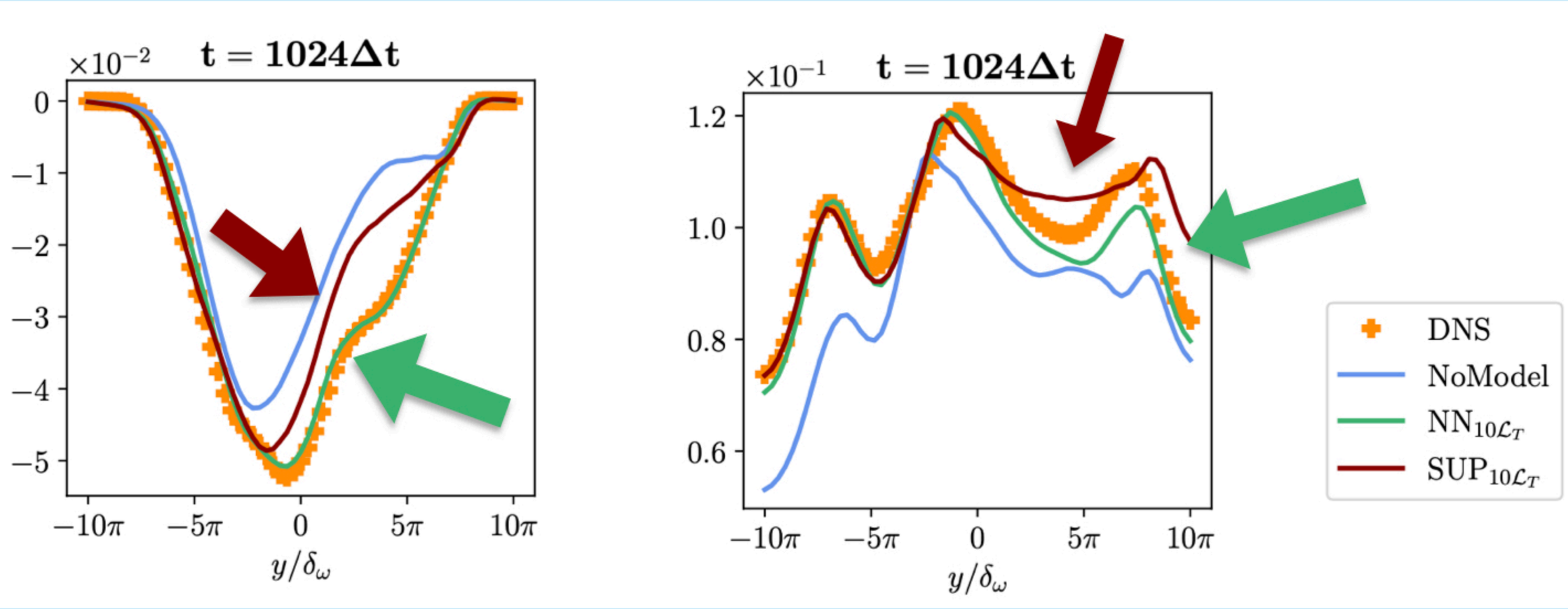
Time stepping with interleaved DP and NN operations for k solver iterations. The dashed gray arrows indicate optional intermediate evaluations of loss terms (similar to the solid gray arrow for the last step k), and intermediate outputs of the NN are indicated with a tilde



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

IS COMBINING A NN AND A DP ANY USEFUL?

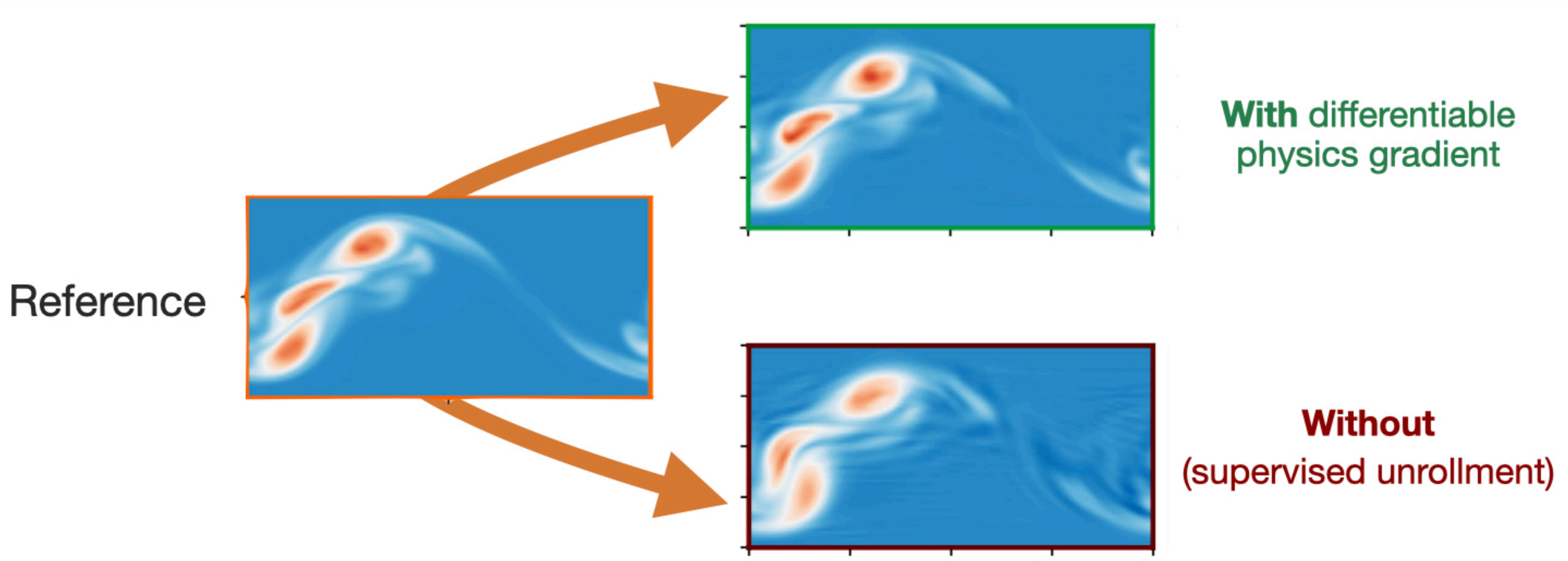
Short answer: Quite a lot!
Unrollment over 10 time steps adds long time information into training



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022, List+2022

IS COMBINING A NN AND A DP ANY USEFUL?

Short answer: Quite a lot!



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022


Reducing numerical errors with deep learning

MOTIVATION: DISCRETIZATION ERRORS

Pretty much all numerical methods contain some form of iterative process: repeated updates over time for explicit solvers, or within a single update step for implicit solvers.

simplest example: Discretisation in space and time

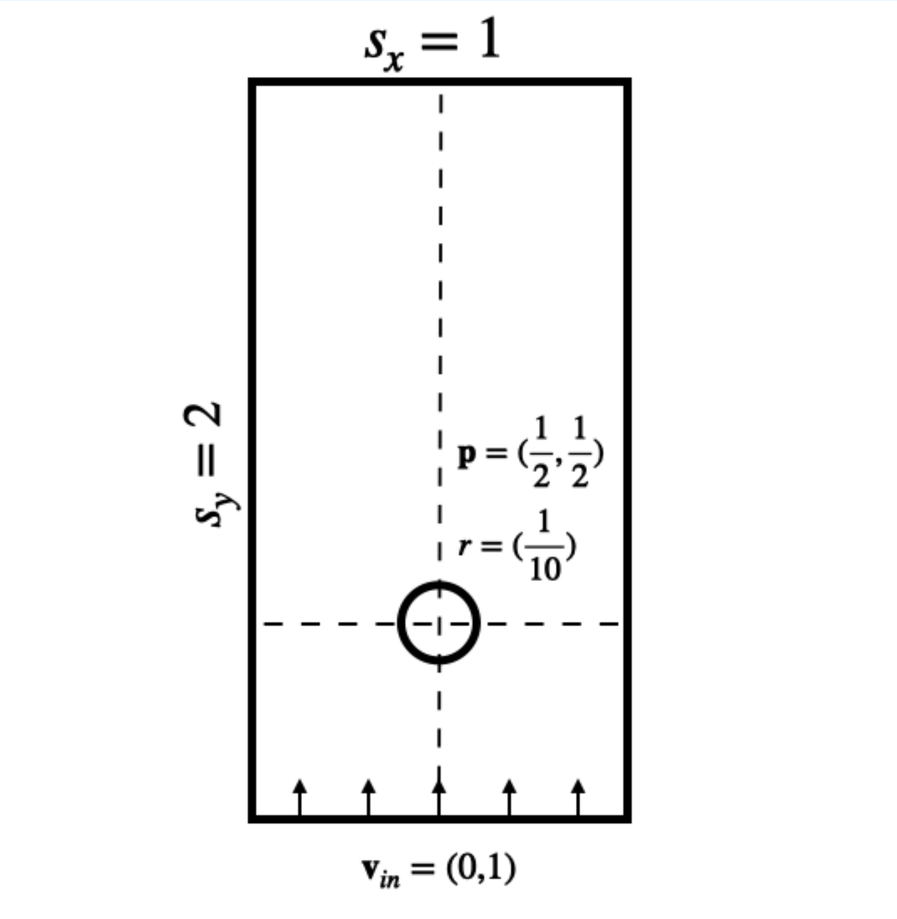
Let’s consider two different discretizations of the same PDE
(e.g. two different spatial discretizations)

easy to imagine that higher spatial resolution will result in better accuracy
similar for time discretisation

but also: fixed grid vs. mesh refinement vs. square or rectangular or unstructured mesh

taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

DISCRETIZATION ERRORS IN KARMANFLOW



for this advanced example there is a collar notebook here: https://colab.research.google.com/github/tum-pbs/pbdl-book/blob/main/diffphys-code-sol.ipynb

taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

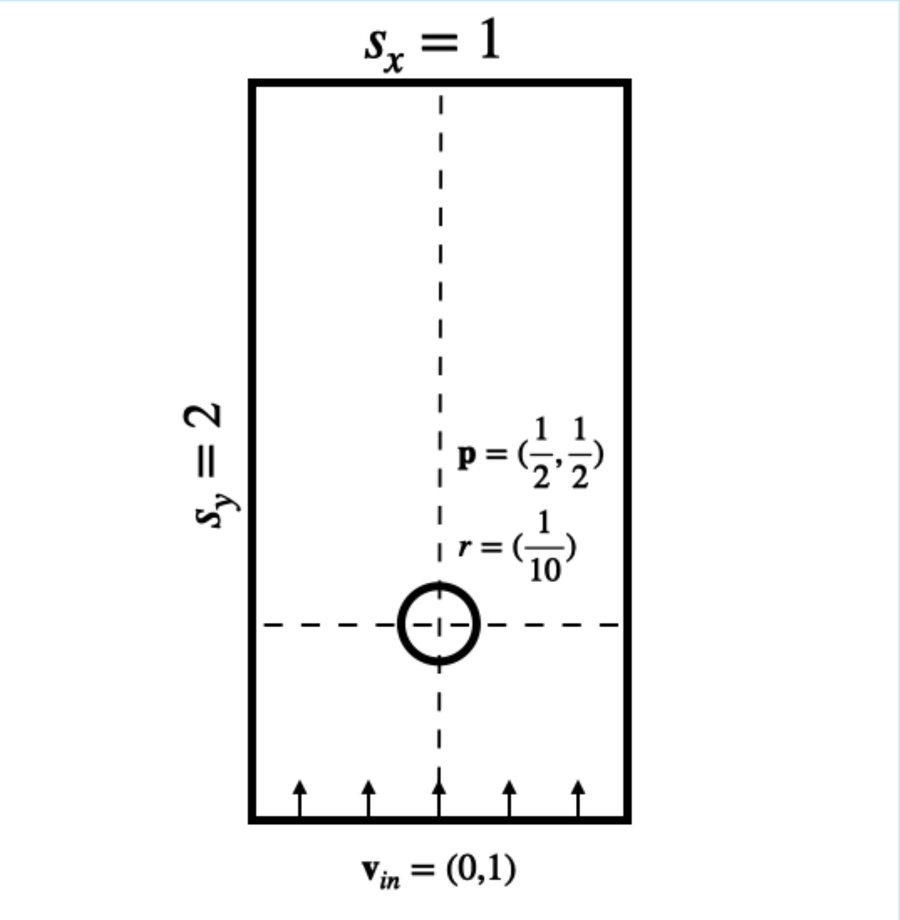
DISCRETIZATION ERRORS IN KARMANFLOW

Problem setup:

ground truth high-res simulation

lower resolution solver: source

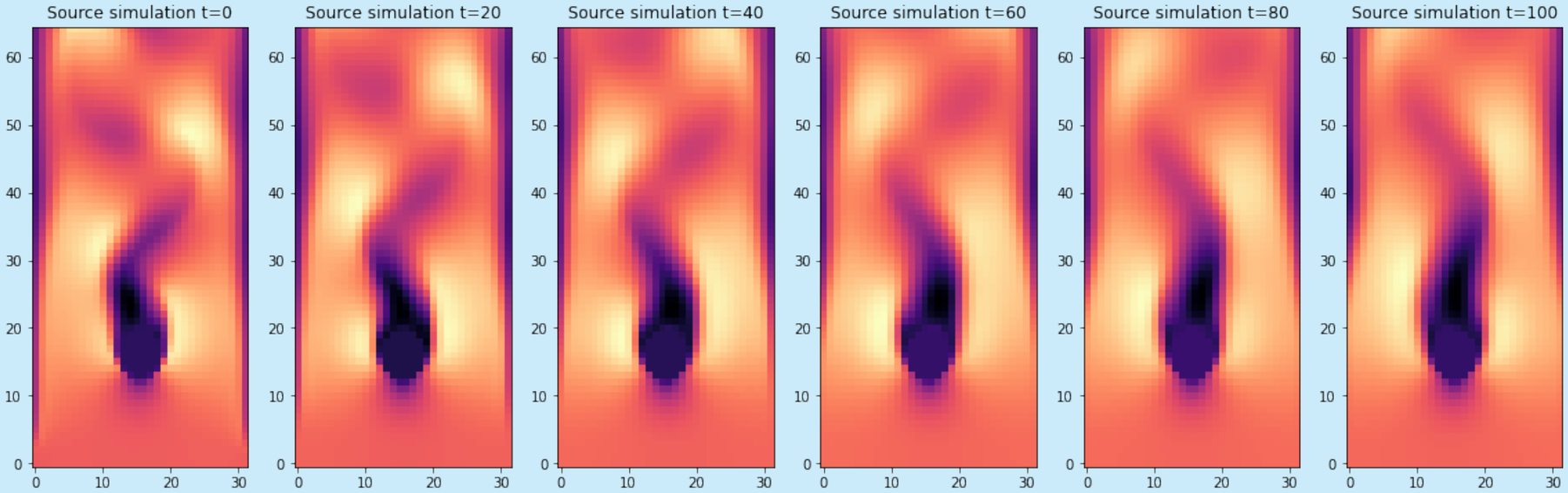
NN to „correct“ the source



for this advanced example there is a collar notebook here: https://colab.research.google.com/github/tum-pbs/pbdl-book/blob/main/diffphys-code-sol.ipynb

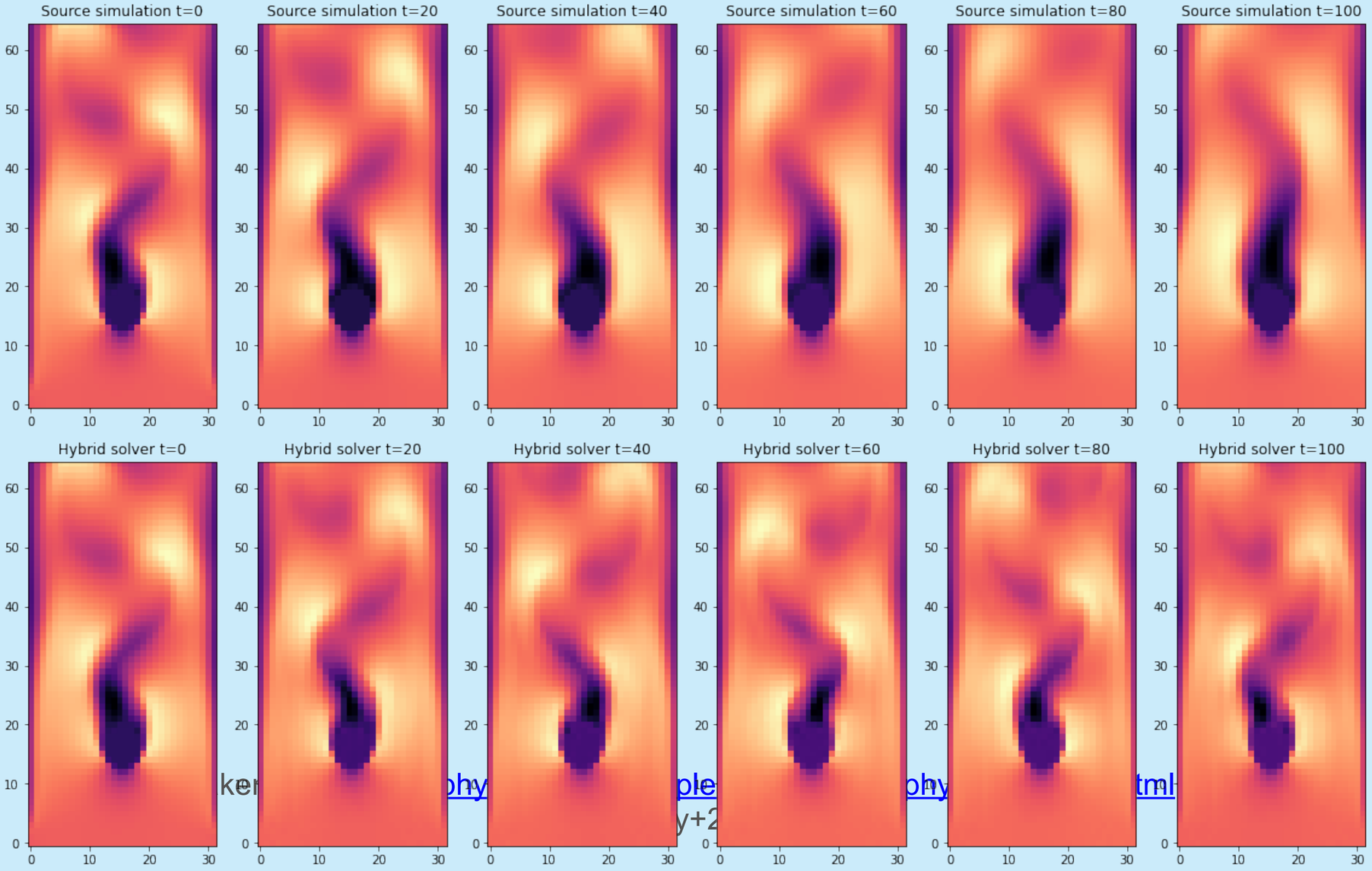
taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

DISCRETIZATION ERRORS IN KARMANFLOW: SOURCE SOLVER RESULT

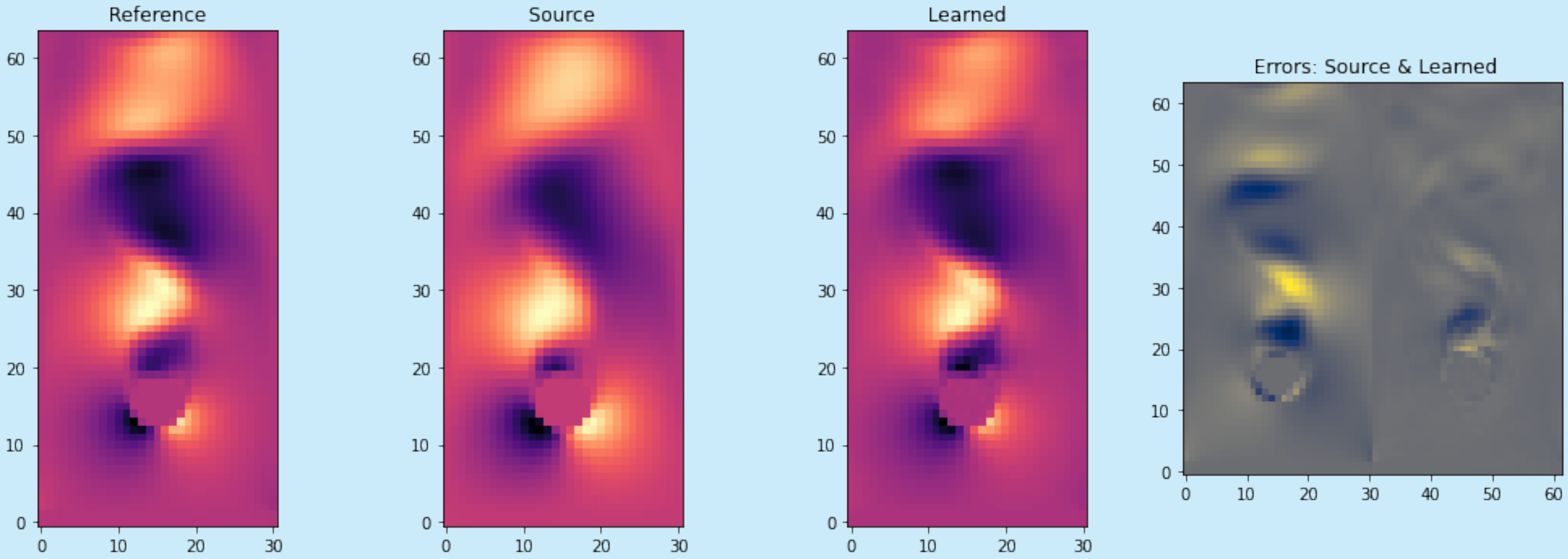


taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

DISCRETIZATION ERRORS IN KARMANFLOW: HYBRID SOLVER RESULT

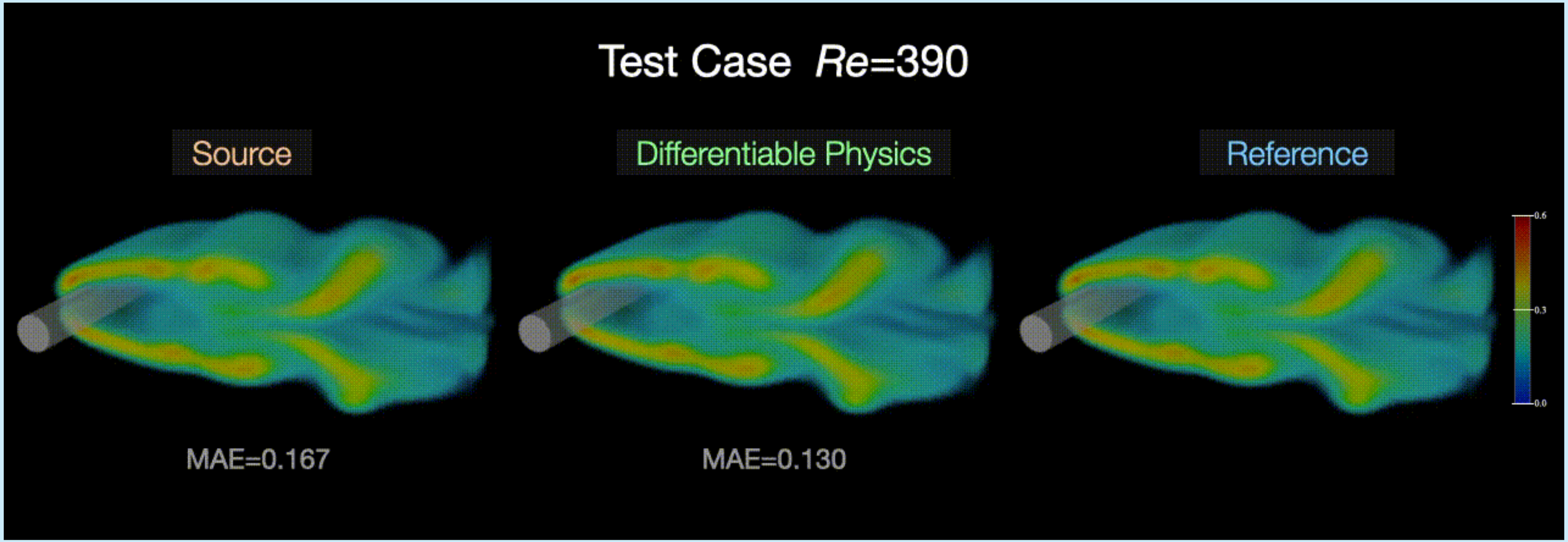


DISCRETIZATION ERRORS IN KARMANFLOW: COMPARISON AFTER 50 STEPS



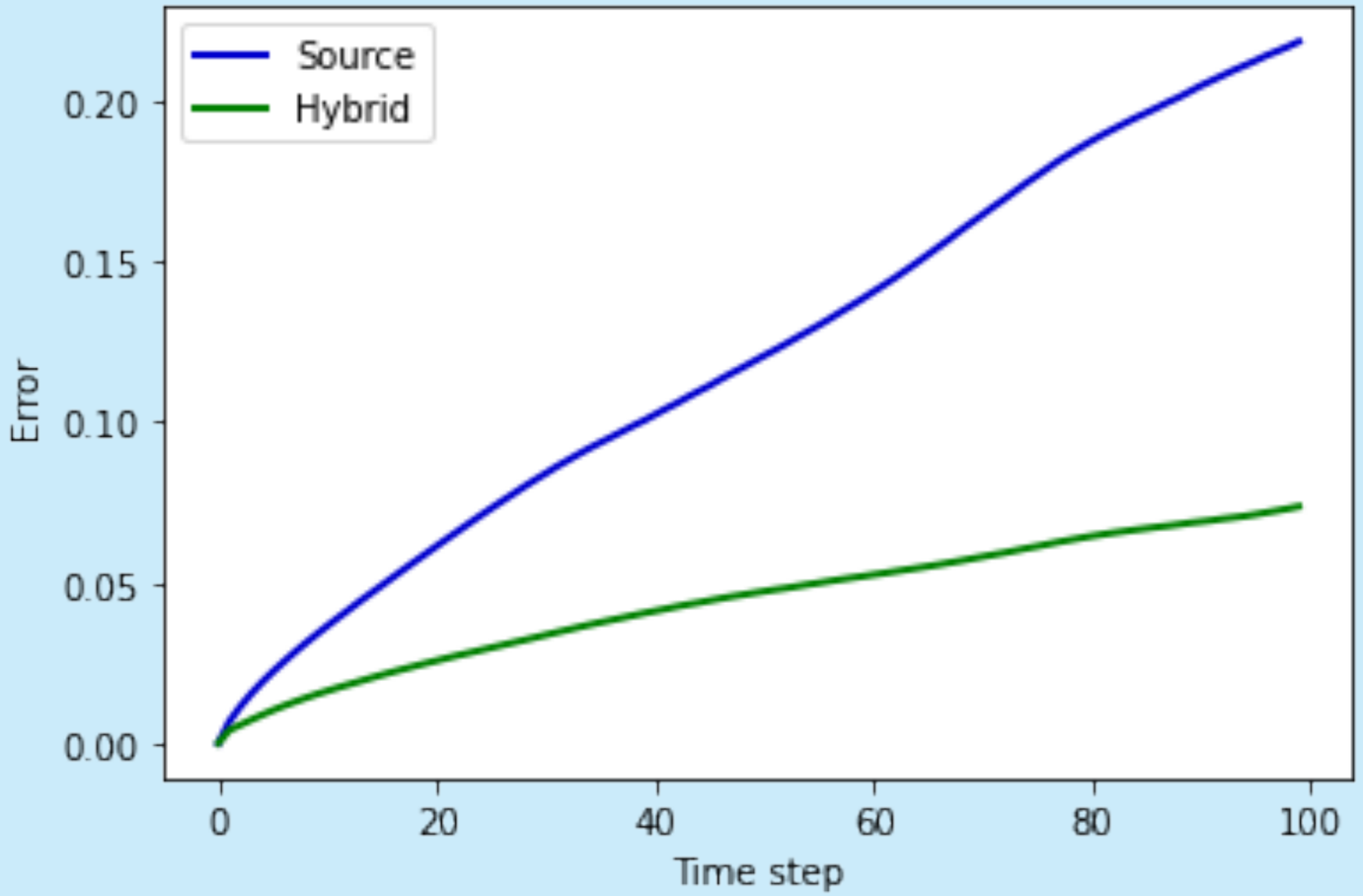
taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

DISCRETIZATION ERRORS IN KARMANFLOW: COMPARISON AFTER 50 STEPS



taken from https://github.com/tum-pbs/Solver-in-the-Loop
Um+2020

DISCRETIZATION ERRORS IN KARMANFLOW: SOURCE SOLVER RESULT



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

Solving Inverse Problems with NNs

INVERSE PROBLEM FOR PREDICTOR-CORRECTOR SETUPS

Problem setup:

computing a high dimensional control function that exerts forces over the full course of an incompressible fluid simulation in order to reach a desired goal state for a passively advected marker in the fluid

incompressible Navier-Stokes equations in two dimensions

$$\frac{\partial u_x}{\partial t} + \mathbf{u} \cdot \nabla u_x = -\frac{1}{\rho} \nabla p$$
$$\frac{\partial u_y}{\partial t} + \mathbf{u} \cdot \nabla u_y = -\frac{1}{\rho} \nabla p$$
$$\text{s.t.} \quad \nabla \cdot \mathbf{u} = 0,$$

Plus additional equation for marker density

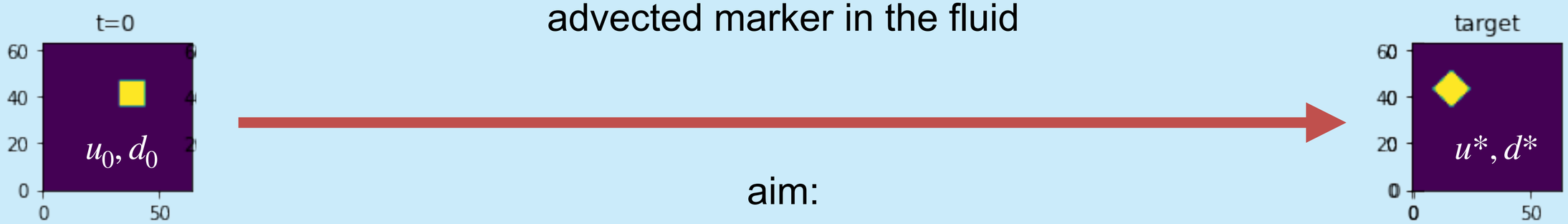
$$\frac{\partial d}{\partial t} + \mathbf{u} \cdot \nabla d = 0.$$

for this advanced example there is a collar notebook here: https://colab.research.google.com/github/tum-pbs/pbdl-book/blob/main/diffphys-code-control.ipynb

INVERSE PROBLEM FOR PREDICTOR-CORRECTOR SETUPS

Problem setup:

computing a high dimensional control function that exerts forces over the full course of an incompressible fluid simulation in order to reach a desired goal state for a passively advected marker in the fluid



aim:

train two networks OP and CFE such that

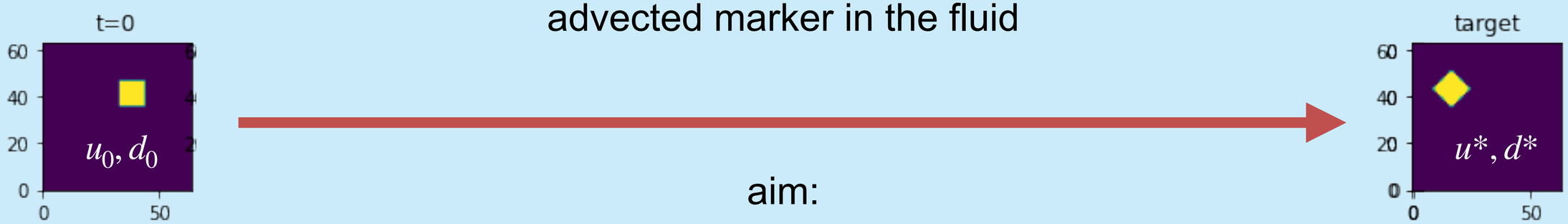
$$\mathbf{u}_n, d_n = \mathcal{P}(\mathrm{CFE}(\,\mathcal{P}(\mathrm{CFE}(\cdots\mathcal{P}(\mathrm{CFE}(\mathbf{u}_0, d_0, d_{OP}))\cdots)))) = (\mathcal{P}\ \mathrm{CFE})^n(\mathbf{u}_0, d_0, d_{OP}).$$

minimises the loss.

INVERSE PROBLEM FOR PREDICTOR-CORRECTOR SETUPS

Problem setup:

computing a high dimensional control function that exerts forces over the full course of an incompressible fluid simulation in order to reach a desired goal state for a passively advected marker in the fluid



aim:

train two networks OP and CFE such that

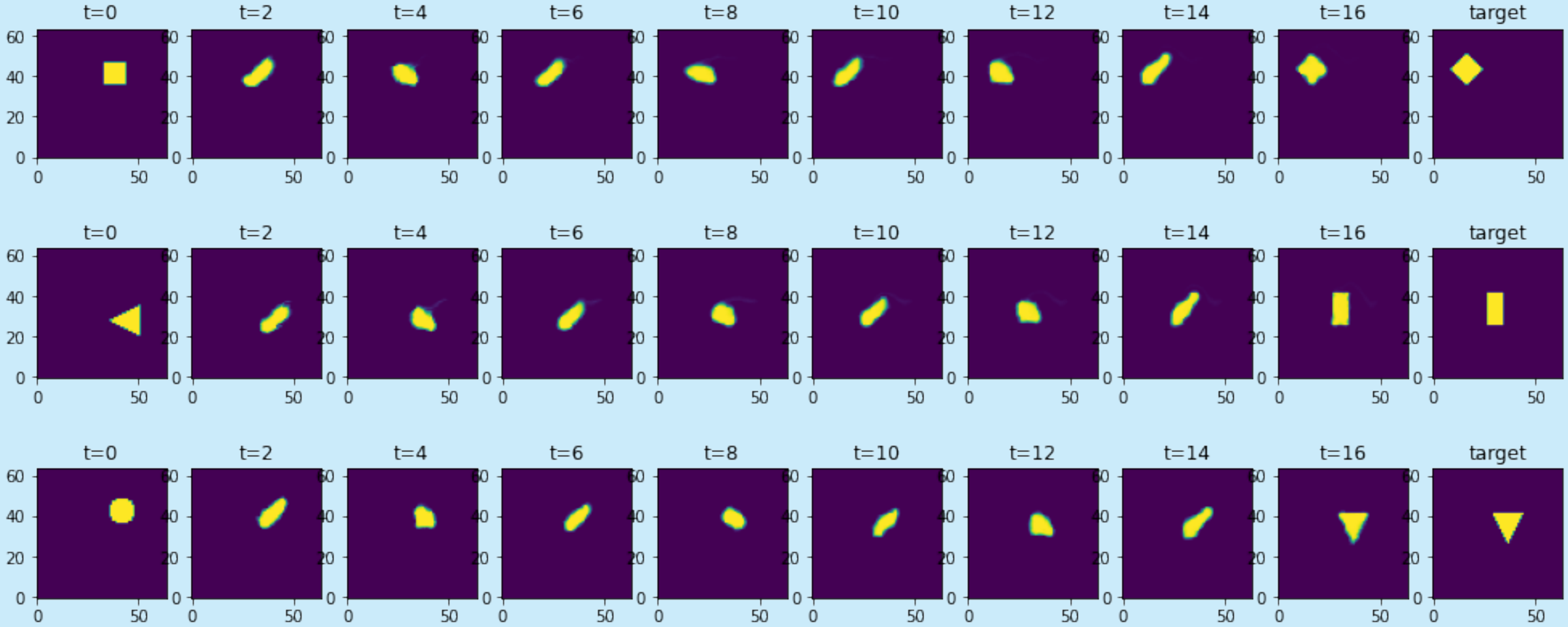
$$\mathbf{u}_n, d_n = \mathcal{P}(\mathrm{CFE}(\,\mathcal{P}(\mathrm{CFE}(\cdots\mathcal{P}(\mathrm{CFE}(\mathbf{u}_0, d_0, d_{OP}))\cdots)))) = (\mathcal{P}\ \mathrm{CFE})^n(\mathbf{u}_0, d_0, d_{OP}).$$

minimises the loss.

$$d_{OP} = \mathrm{OP}(d, d^*) = f_{\mathrm{OP}}(d, d^*; \theta_{\mathrm{OP}}).$$

$$\mathbf{u} + f_{\mathrm{CFE}}(\mathbf{u}, d, f_{\mathrm{OP}}(d, d^*; \theta_{\mathrm{OP}}); \theta_{\mathrm{CFE}})$$

INVERSE PROBLEM FOR PREDICTOR-CORRECTOR SETUPS



taken from https://physicsbaseddeeplearning.org/diffphys-examples.html
Thürey+2022

WHERE TO FIND THE LECTURE MATERIAL

https://github.com/TobiBu/graddays

SUMMARY & CONCLUSION

Main take away:

Differentiable physics simulations offer a new versatile approach to scientific modelling.

Let’s try and explore it in the context of Astrophysics!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS