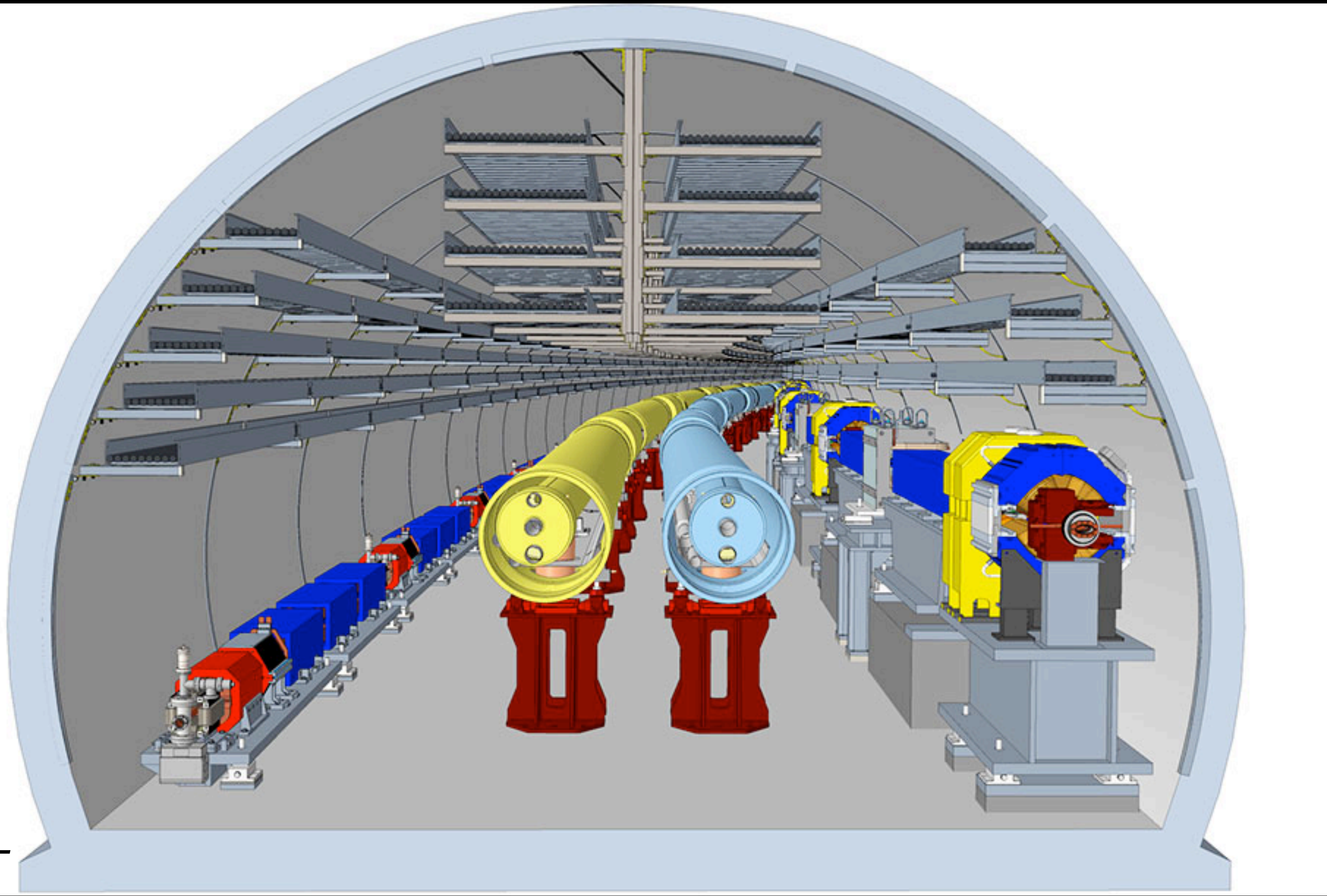
Uncertainty in digital twins from imperfect system information

Nathan Urban
nurban@bnl.gov
Applied Mathematics, Computing & Data Sciences
Brookhaven National Laboratory

September 17, 2024



BNL

What is a digital twin?

- For the purposes of this talk:
  - A predictive computational model of a specific, real-world system
    - Can be physical simulation, AI, anything else
    - Incorporates decision making (control, planning, design optimization, …)
    - Automatically “ingests" data about the system and updates its representation of reality, following these decisions
    - Learn → Predict → Act → Learn → …
- Beyond this, I am not going to open the terminology can of worms!

What is a predictive digital twin, computationally?

- A **mechanistic ("physical") simulation**
  - Constructed to emulate a specific system
  - Initialized with data (state estimation), or "tuned" to data (parameter estimation)
- A **data-driven model** (usually statistical/machine learning)
  - Data as inputs to a prediction ("state estimation"), or trained to data ("parameter estimation")
- **Both of these**
  - A ML model that uses both simulated and real-world data
  - A ML model of data from a simulation that has been tuned to the real world
  - A mechanistic simulation that contains AI components or submodels

What is a predictive digital twin, computationally?

- A **mechanistic (“physical”) simulation**
  - Constructed to emulate a specific system
  - Initialized with data (state estimation), or *“tuned” to data (parameter estimation)*
- A **data-driven model** (usually statistical/machine learning)
  - Data as inputs to a prediction (“state estimation”), or *trained to data (“parameter estimation”)*
- **Both of these**
  - A ML model that uses both simulated and real-world data
  - *A ML model of data from a simulation that has been tuned to the real world*
  - *A mechanistic simulation that contains AI components or submodels*

Uncertainty in digital twins

- Much uncertainty quantification (UQ) for DTs focuses on *state estimation*: the DT updates its state representation with data from the measured system
- But digital twins are not identical twins: they imperfectly model the true system
- “All models are wrong, but some are useful” –G. Box

Uncertainty in digital twins

- Much uncertainty quantification (UQ) for DTs focuses on *state estimation*: the DT updates its state representation with data from the measured system
- But digital twins are not identical twins: they imperfectly model the true system
- “All models are wrong, but some are useful” –G. Box; *“But they’re still wrong” –me*

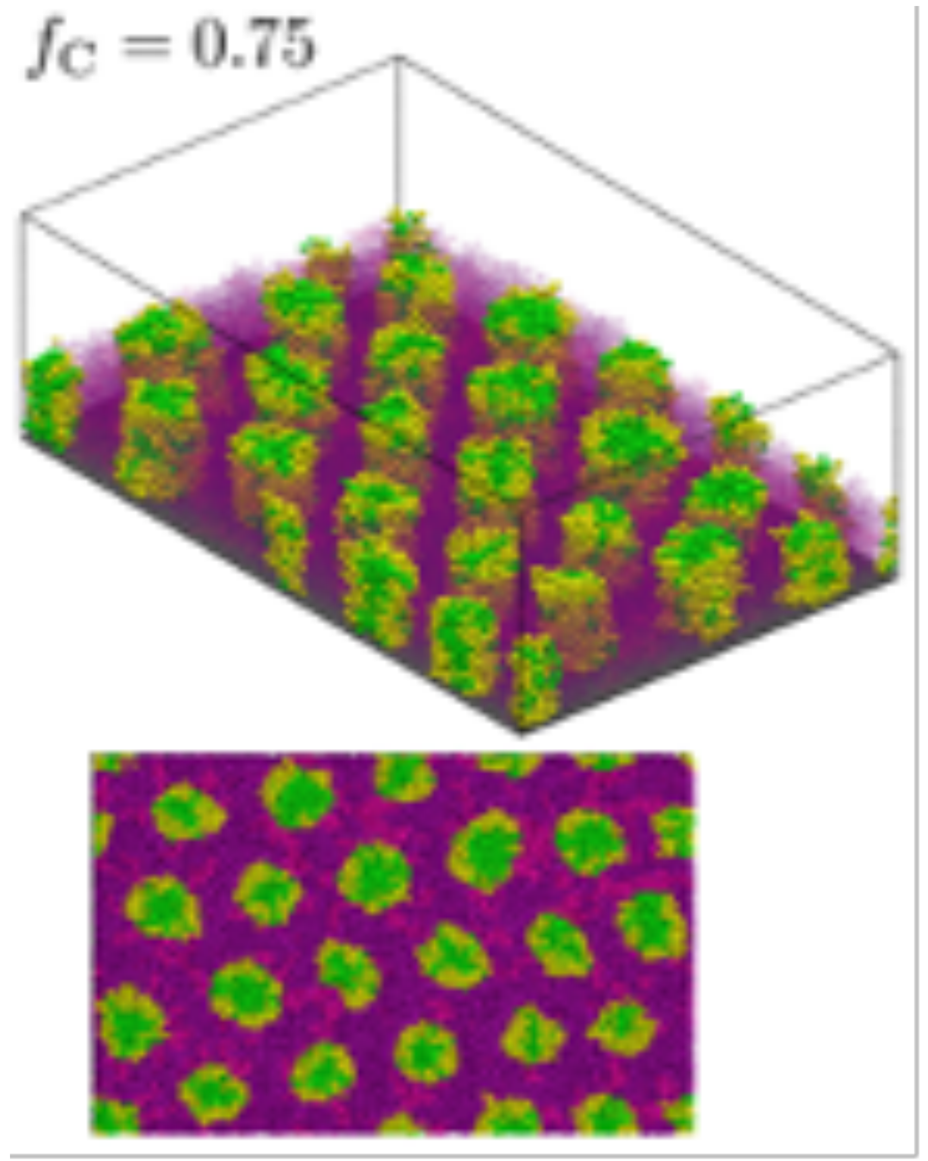
Uncertainty in digital twins

- Much uncertainty quantification (UQ) for DTs focuses on *state estimation*: the DT updates its state representation with data from the measured system
- But digital twins are not identical twins: they imperfectly model the true system
- "All models are wrong, but some are useful" –G. Box; *"But they're still wrong" –me*
    - Learn and quantify errors in their representation of the system dynamics
    - This could be *parameter estimation*, or *system identification*, or *equation learning*
- **Data driven-models:**
    - <u>Pro:</u> based on the observed, real world
    - <u>Con:</u> hard to make predictions outside the scope of the observed data
- **Physical simulation:**
    - <u>Pro:</u> can make predictions under new, unseen conditions (extrapolate)
    - <u>Con:</u> some physics is unknown/intractable, simplified theory approximations

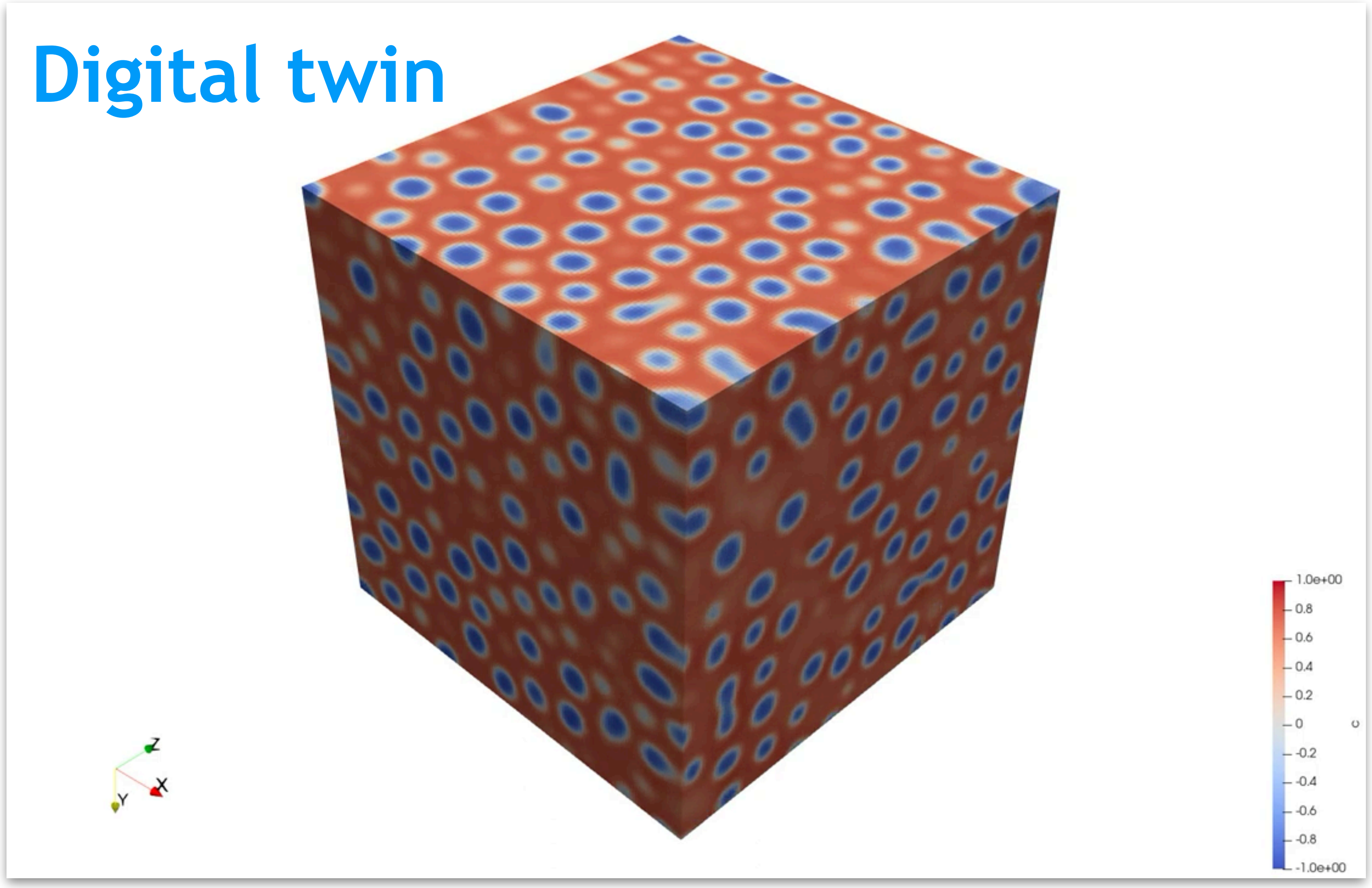
Parameter uncertainty in simulations

- Example: nanomaterial self-assembly
  - Phase separation dynamics in block copolymer systems
  - Applications to batteries, photovoltaics, etc.
- "Digital twin" is a simulation of the binary nonlocal Cahn-Hilliard equation
  - Partial differential equation (PDE) system: $\partial c/\partial t = \nabla^2(-\epsilon^2\nabla^2 c) + \nabla^2(c^3 - c) - \sigma(c - \bar{c})$
- Bayesian parameter estimation: probabilistically fit the PDE coefficients to data

Data

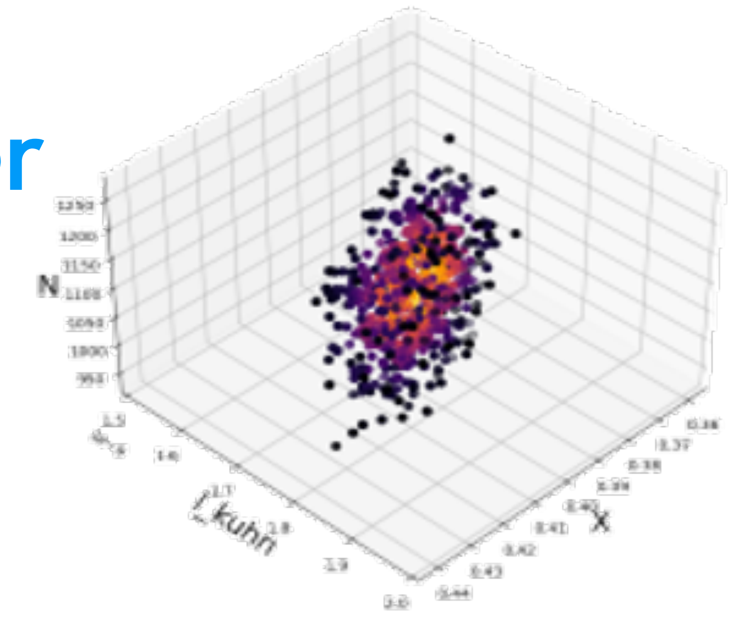


K. Yager

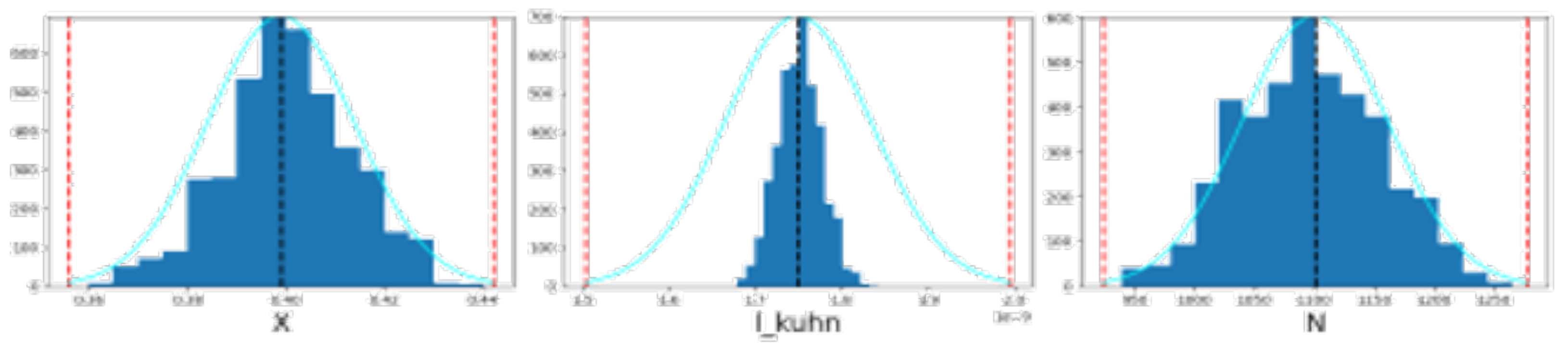


DT parameter estimates

A. DeGennaro



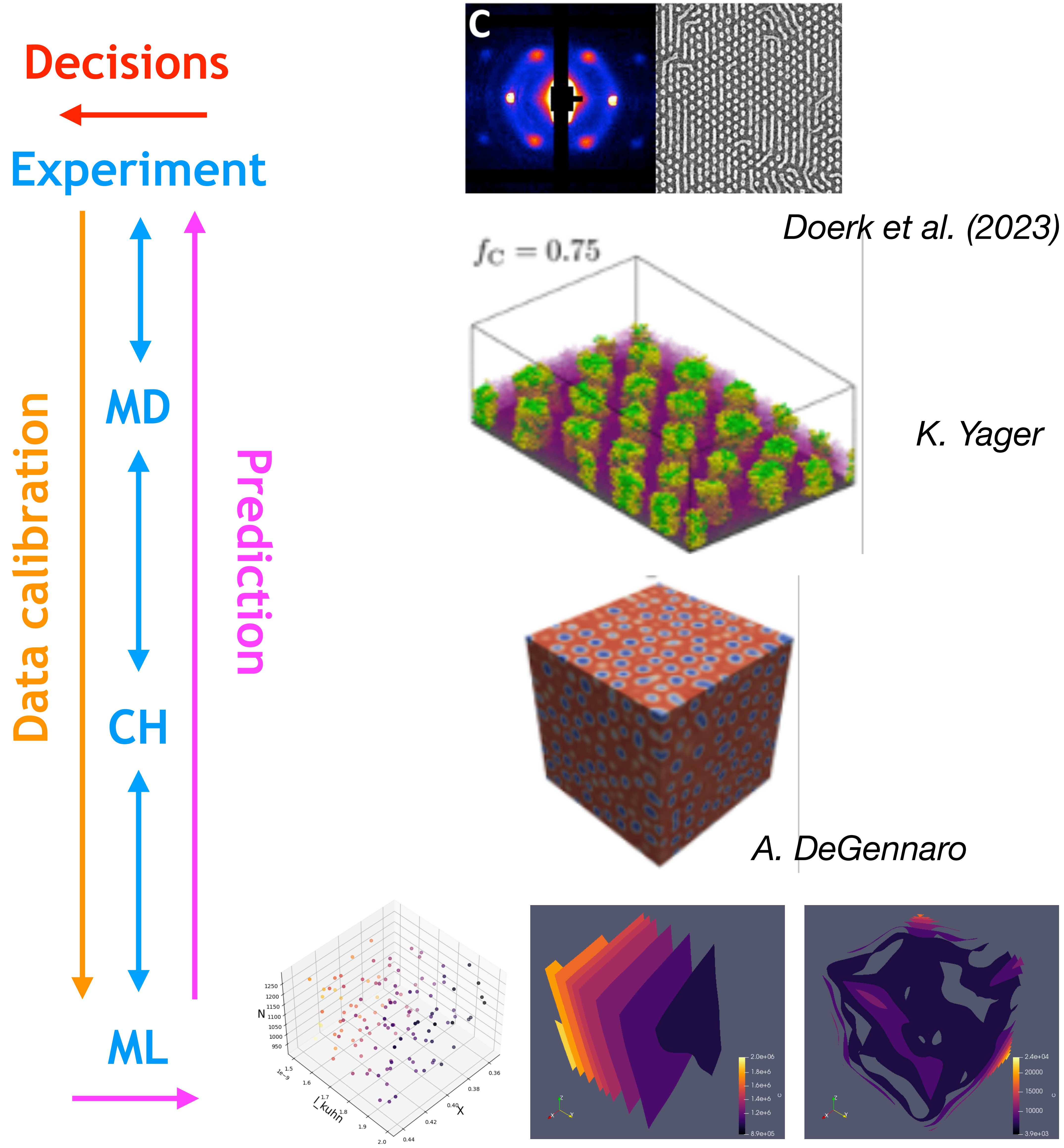
(a) Posterior samples.



(b) Individual parameter posterior distributions.

A probabilistic hierarchy of digital twins

- Tradeoffs between DT accuracy and speed
  - Molecular dynamics (MD): gold standard but slow (thousands of hours)
  - Cahn-Hilliard PDE (CH): captures coarse-scale dynamics (fraction of an hour)
  - ML: fraction of a second, but little training data
- Build a multifidelity hierarchy of twins
  - Real world = MD + MD error
  - MD = CH(parameters) + CH error
  - CH = ML(parameters) + ML error
- Uncertainty in parameters and errors at each level
- Use in a DT decision loop: calibrate → predict → select next experiment → calibrate

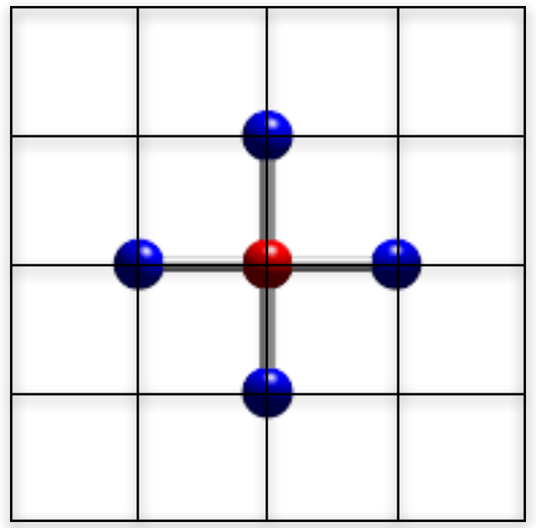


(a) Training points. (b) GP mean. (c) GP std dev.

Learning a simulation-based digital twin

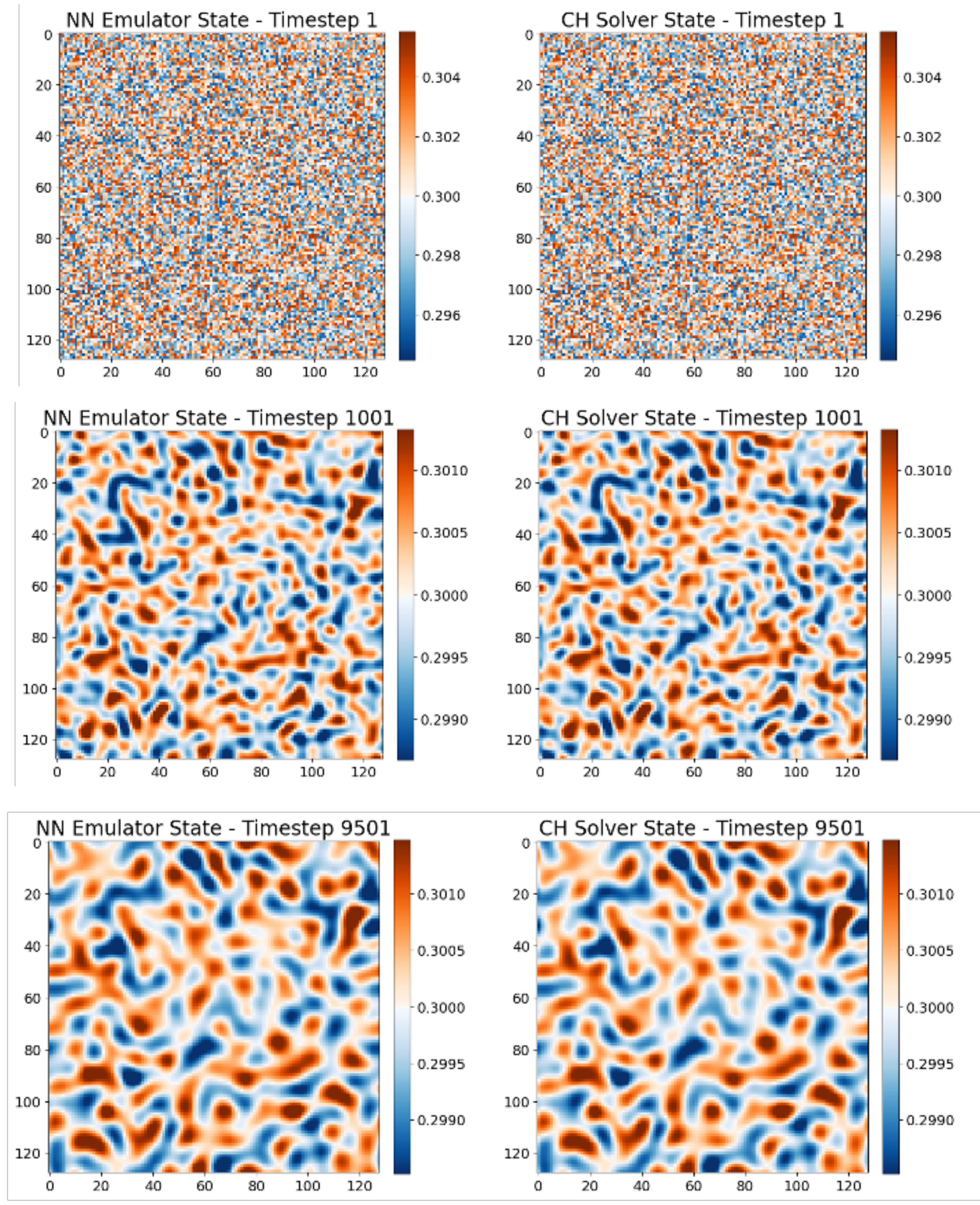
- We know the Cahn-Hilliard PDE describes structure formation in a simple set of cases
- We don’t know what effective PDE describes the coarse-grained dynamics of harder cases
  - We may be able to derive it by hand, but this is laborious for each new system
  - Can we learn the PDE? ($\partial c/\partial t = ???$)
- Treat right hand side (RHS) as unknown function
  - cell neighbor states → next cell state
  - Insert neural net into PDE solver
- Once equations have been learned, can run the twin at unseen initial or boundary conditions (unlike ML approaches that train on input/output pairs)

PDE stencil operator



$\boxed{\mathbf{u}_i^{t+1} = N(\mathbf{u}_i^t, \mathbf{u}_{i+1}^t, \mathbf{u}_{i-1}^t)}$

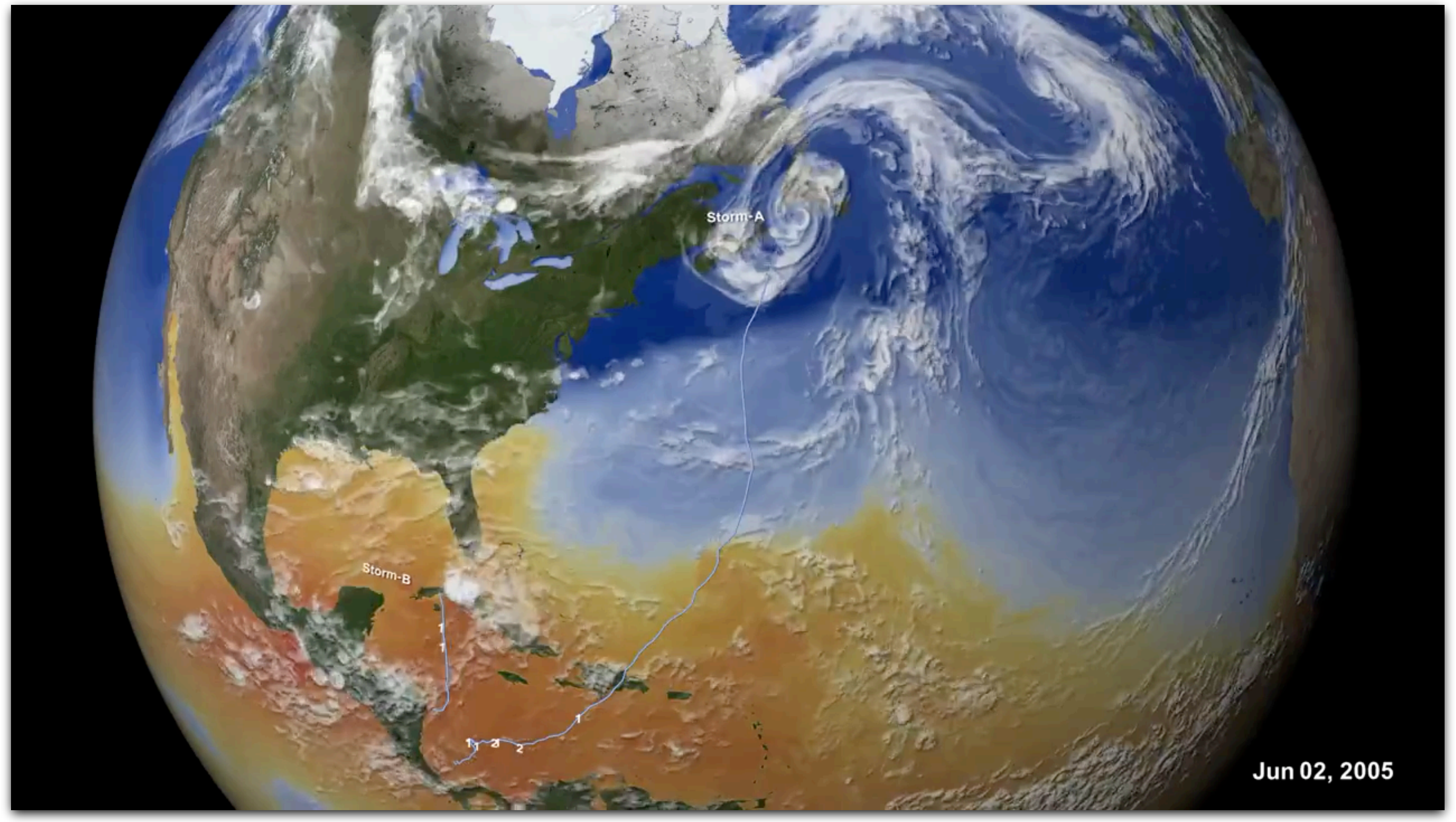
CH solver   Neural PDE



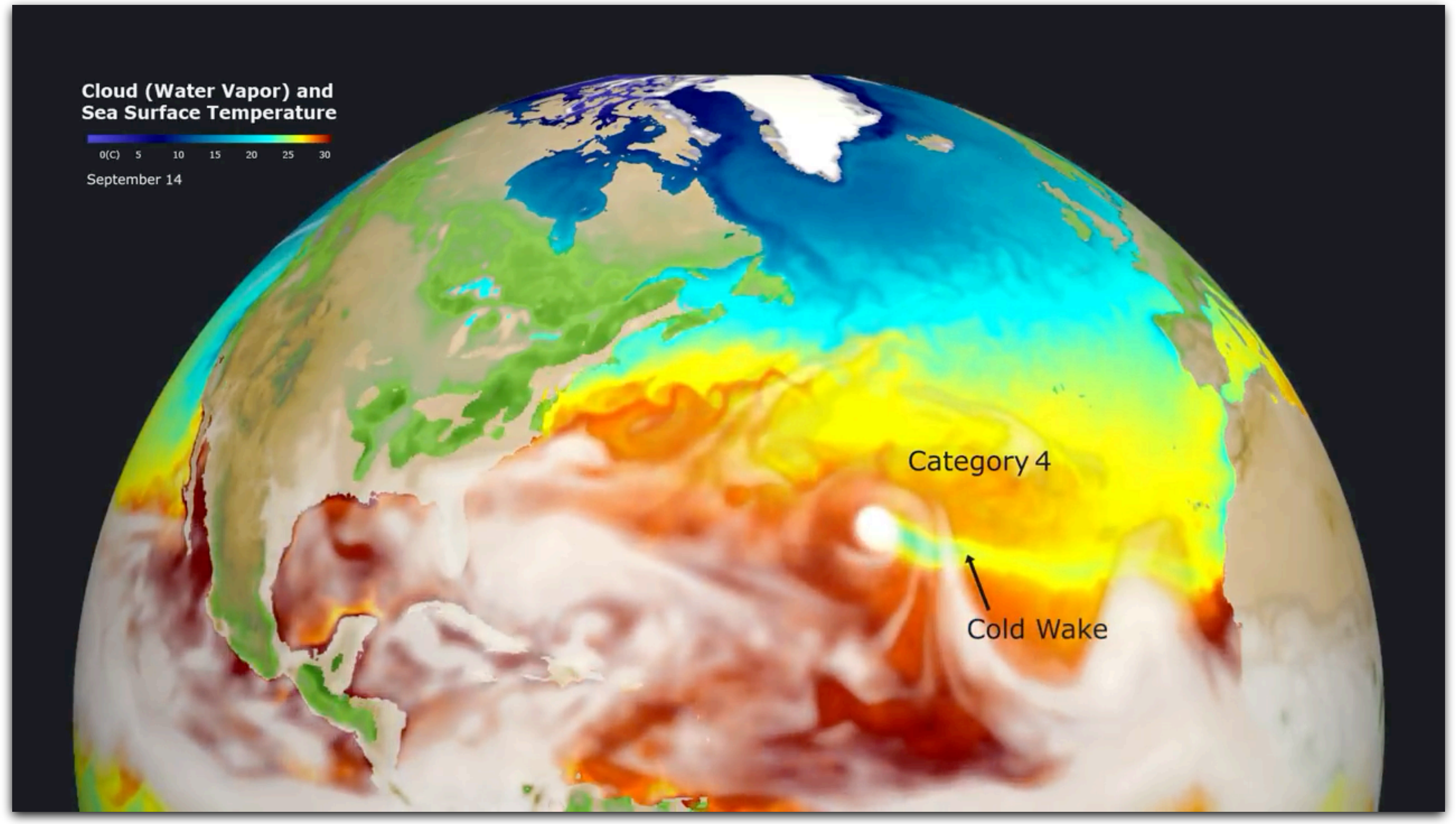
Jantre et al.

Inconvenient truth of modeling: *approximations*

- Climate example: models differ mostly due to their representations of cloud dynamics
- Clouds are approximated (too small to simulate)
- Many choices involved in approximations:
  - numerical time and space discretization schemes
  - closure models
  - other unresolved sub-grid approximations
  - choice of processes to include
- In a climate model, **~10%** of the code is the PDEs you're solving, and **~90%** is approximations to all the physics you're *not* solving from first principles
- True for many complex systems



NASA GEOS-5 global cloud resolving model (1.5 km resolution)



DOE E3SM global climate model (25 km resolution)

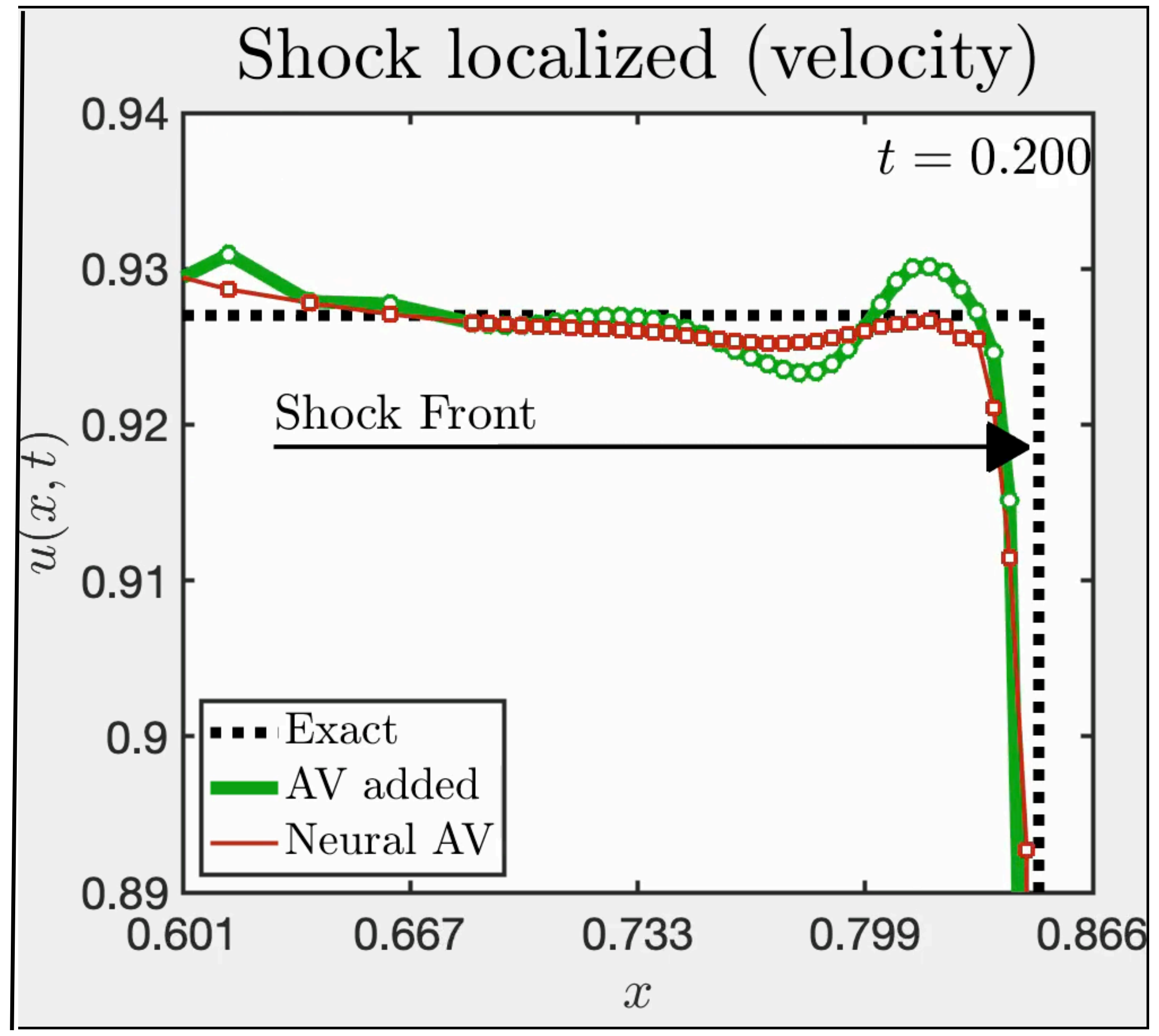
Hybrid physics+ML digital twins

- Can we build a "hybrid" digital twin that makes use of physics we know and trust, but learns physics we don't know from data?
- Example: partial differential equation system
  - $\partial \mathbf{u}/\partial t = f_{PDE}(\mathbf{u}(x,t);p) + g_{ML}(\mathbf{u}(x,t);w)$
  - *f*(·) embodies "known physics" equations (e.g. heat or wave equations …)
  - *g*(·) embodies "unknown physics" that can be learned from data
    - Can represent correction terms, closure schemes, missing processes, …
- The uncertainties are now *functional* (what should go on the PDE's right hand side?), rather than *parametric*
  - Formally an infinite-dimensional space
  - Can parameterize it with neural net weights, but still high-dimensional

Hybrid physics+ML digital twins

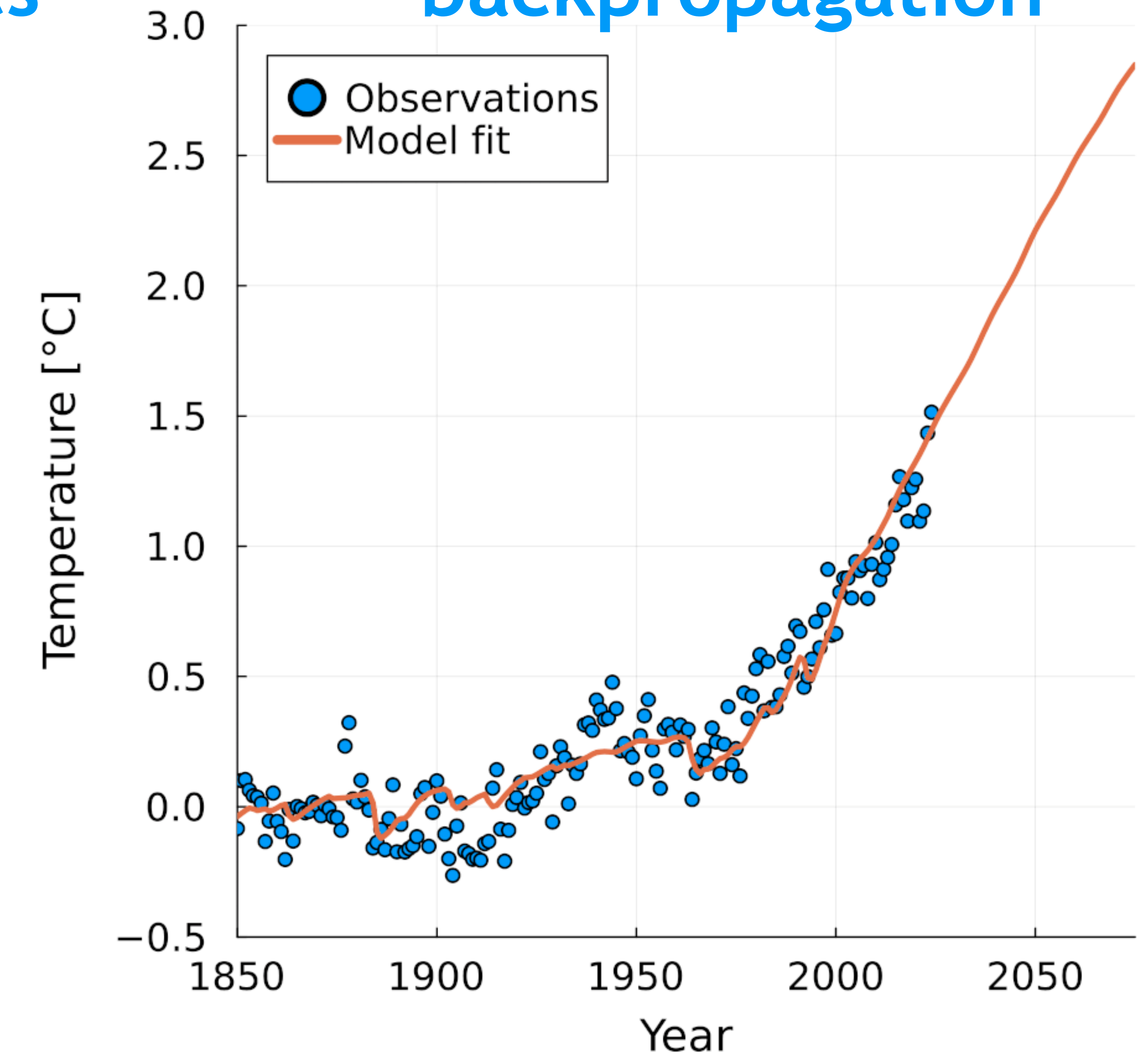
- "Offline learning": have measurements of the unknown physics, and we fit a function to it
- "Online learning": don't know what the function should look like, but we can guess a functional form, run the hybrid model, and see what it predicts
  - **Differentiable programming:** backpropagate DT prediction errors through both ML component *and* the simulation solver (adjoint model)

Learning an artificial viscosity scheme in shock hydrodynamics

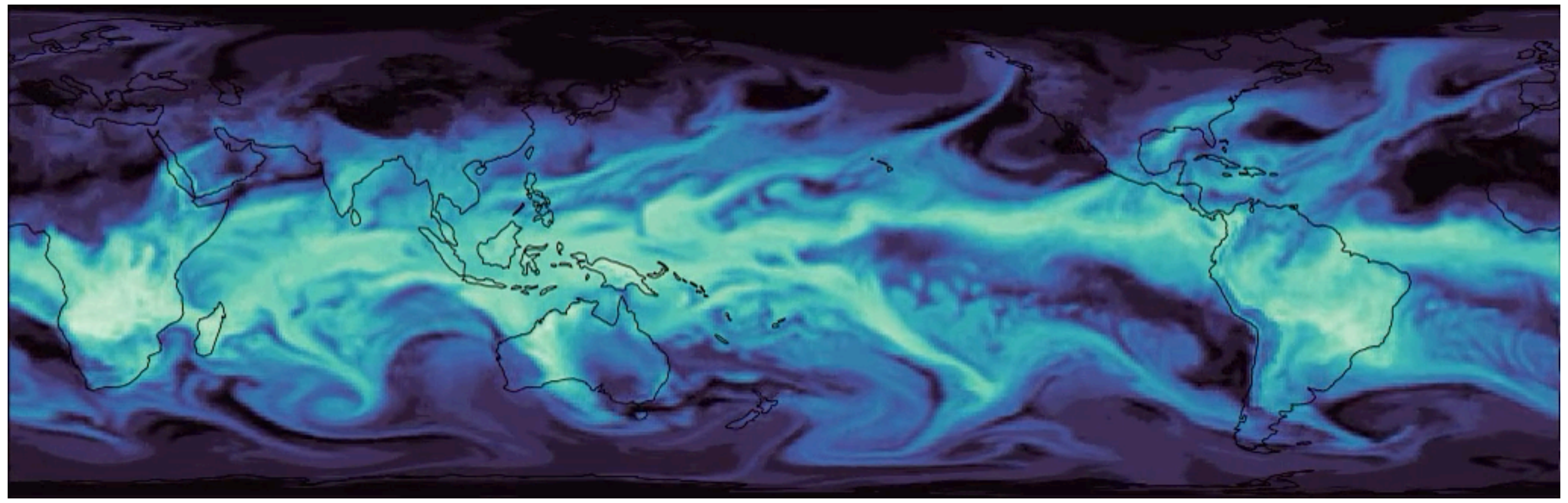


Melland et al. (2021)

Fitting a climate model by backpropagation



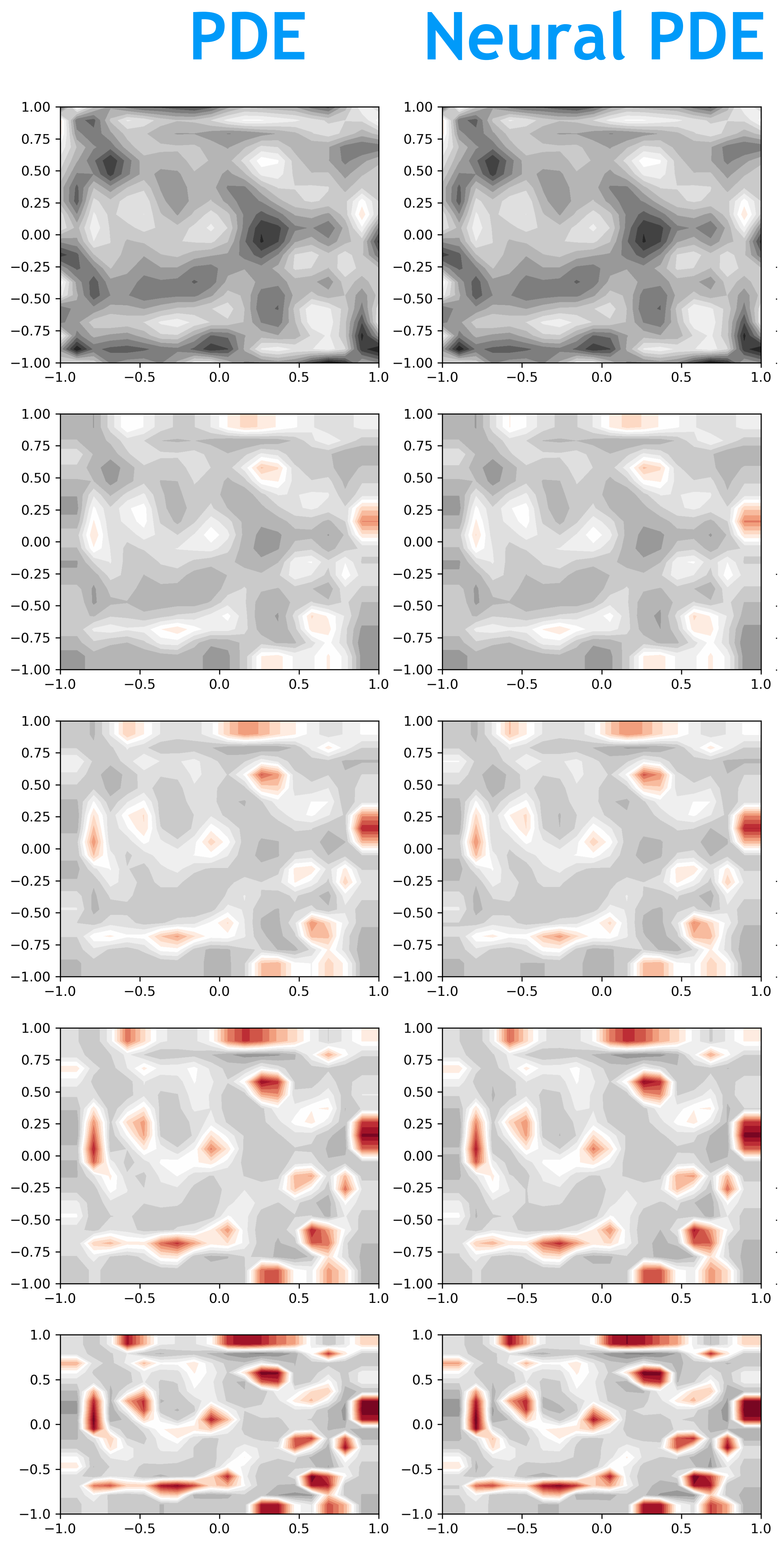
Learning unresolved cloud physics in a climate model



Kochkov et al. (2024)

Hybrid physics+ML digital twins

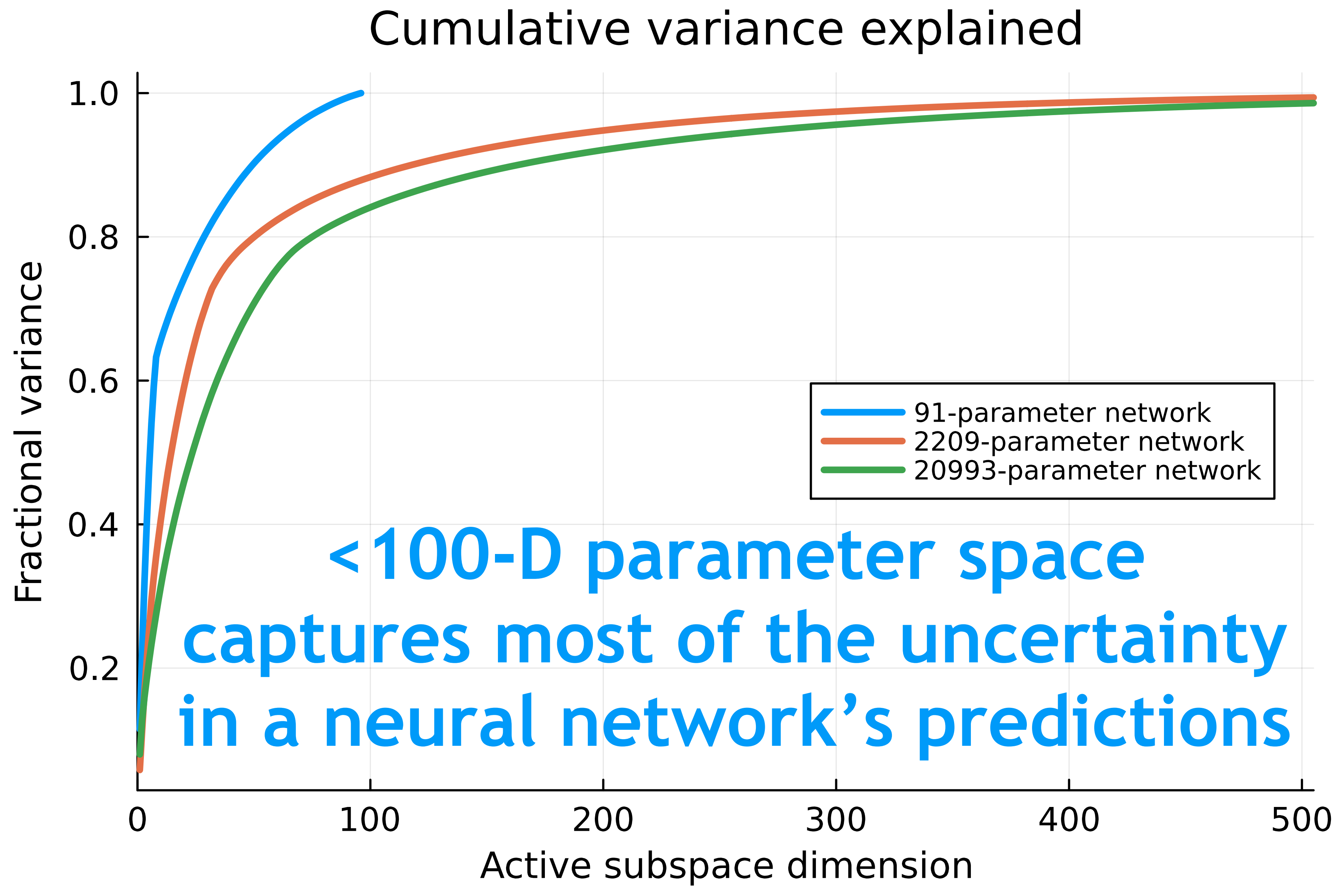
- Example: reaction-diffusion PDE
    - $\partial v_1/\partial t = D_1 \nabla^2 v_1 + v_1 - v_1^3 - v_2 - 0.005$
    - $\partial v_2/\partial t = D_2 \nabla^2 v_2 + g_{ML}(v_1, v_2)$
- Can we recover the unknown function $g(\cdot)$ from solution data, $(v_1(t), v_2(t))$?
    - And propagate its uncertainty to predictions?
- How do we handle the high-dimensional uncertainty space of functions?

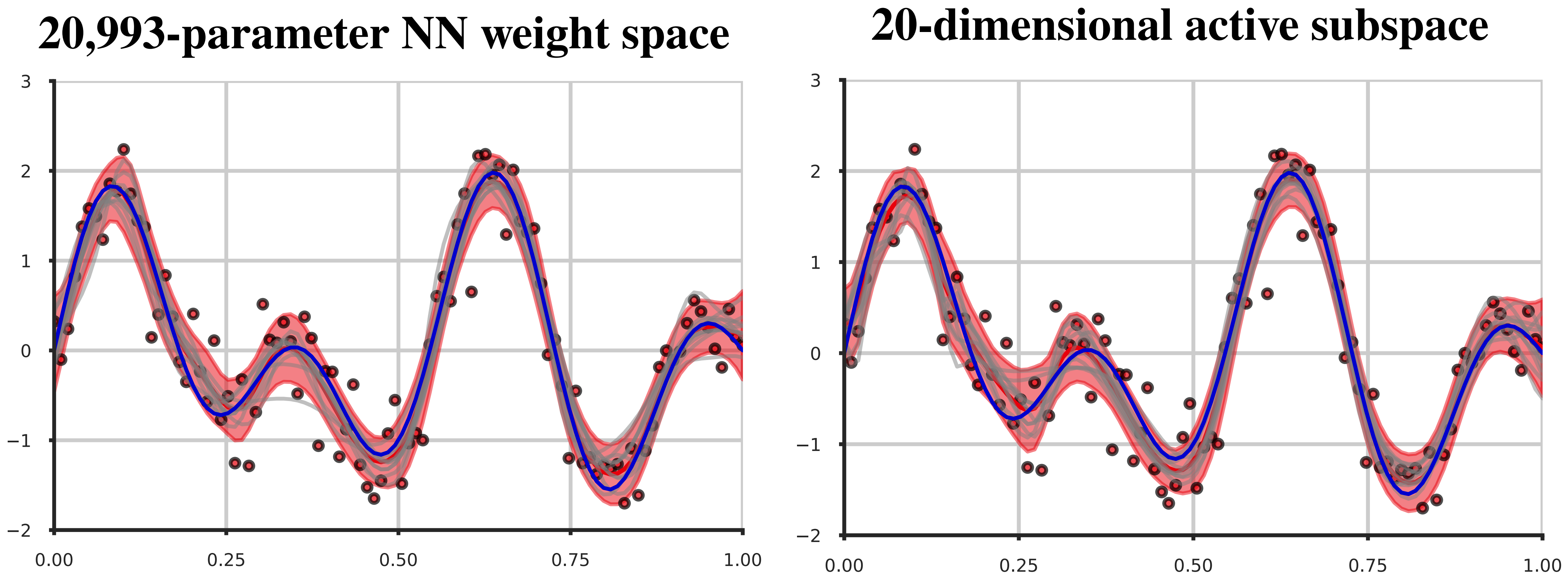


Jantre et al.
Akhare et al. (2023)

Functional subspace reduction

- Neural network (NN) weight space is too high-dimensional to explore uncertainties
  - Each function represented by a NN requires an expensive PDE solve to compute the loss
- → Find a low-dimensional parameter *subspace* that captures most of the predictive uncertainty
- **SGD-PCA subspace** (Izmailov *et al*., 2019):
  - Record weights visited during stochastic gradient descent; compute principal components in weight space
- **Active subspace** (Jantre *et al*., 2024):
  - Compute principal components of the loss function gradient with respect to weights (sampled over a prior distribution)



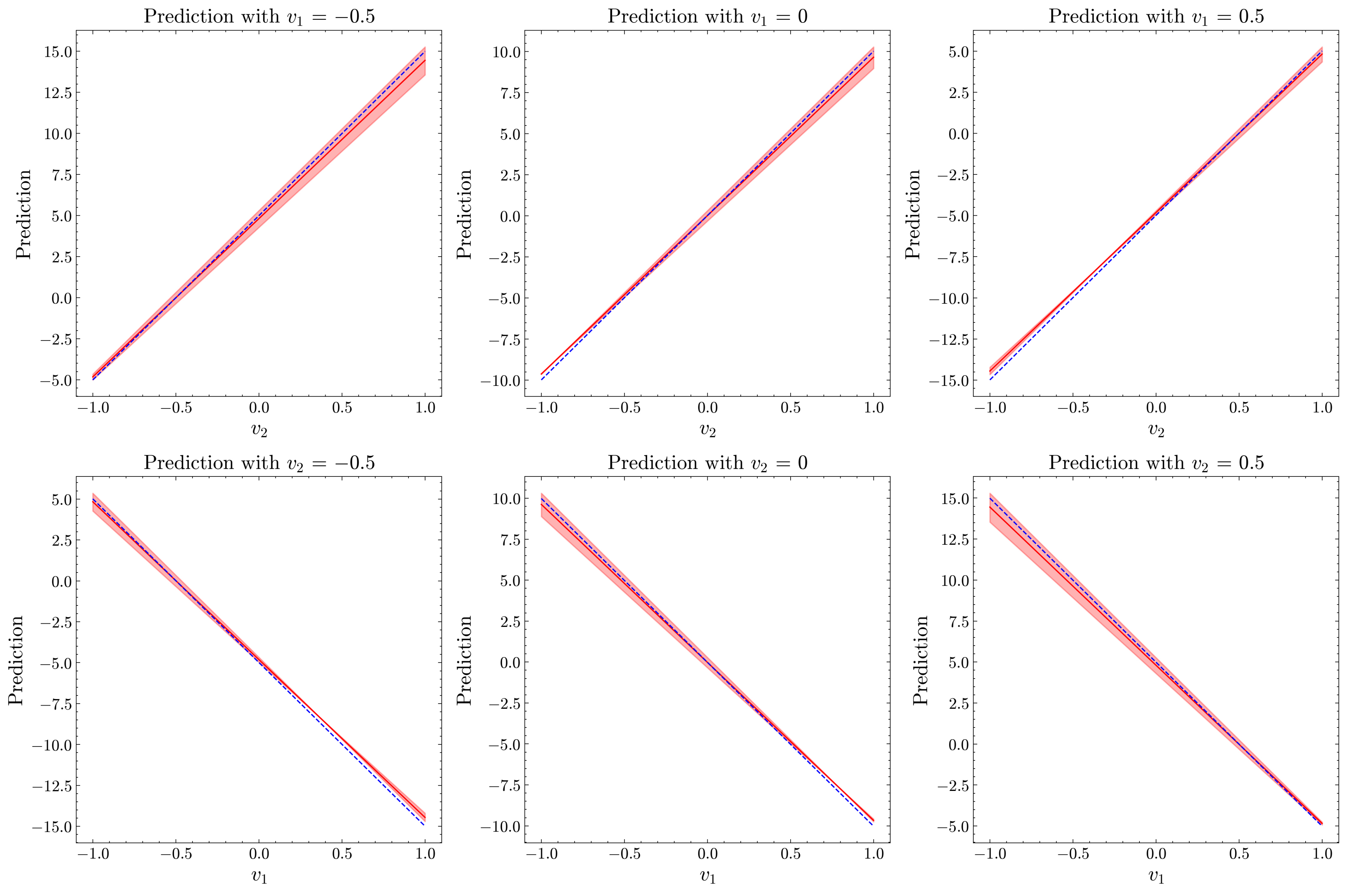


Jantre et al. (2024)

Functional subspace reduction

- Toy example: linear function $g_{ML}(v_1, v_2) = 10(v_1 - v_2)$
- Recover the correct function, assuming it's linear
  - Note that we never observe this function directly, just the PDE solutions

Function recovery (slices through the 2D function)

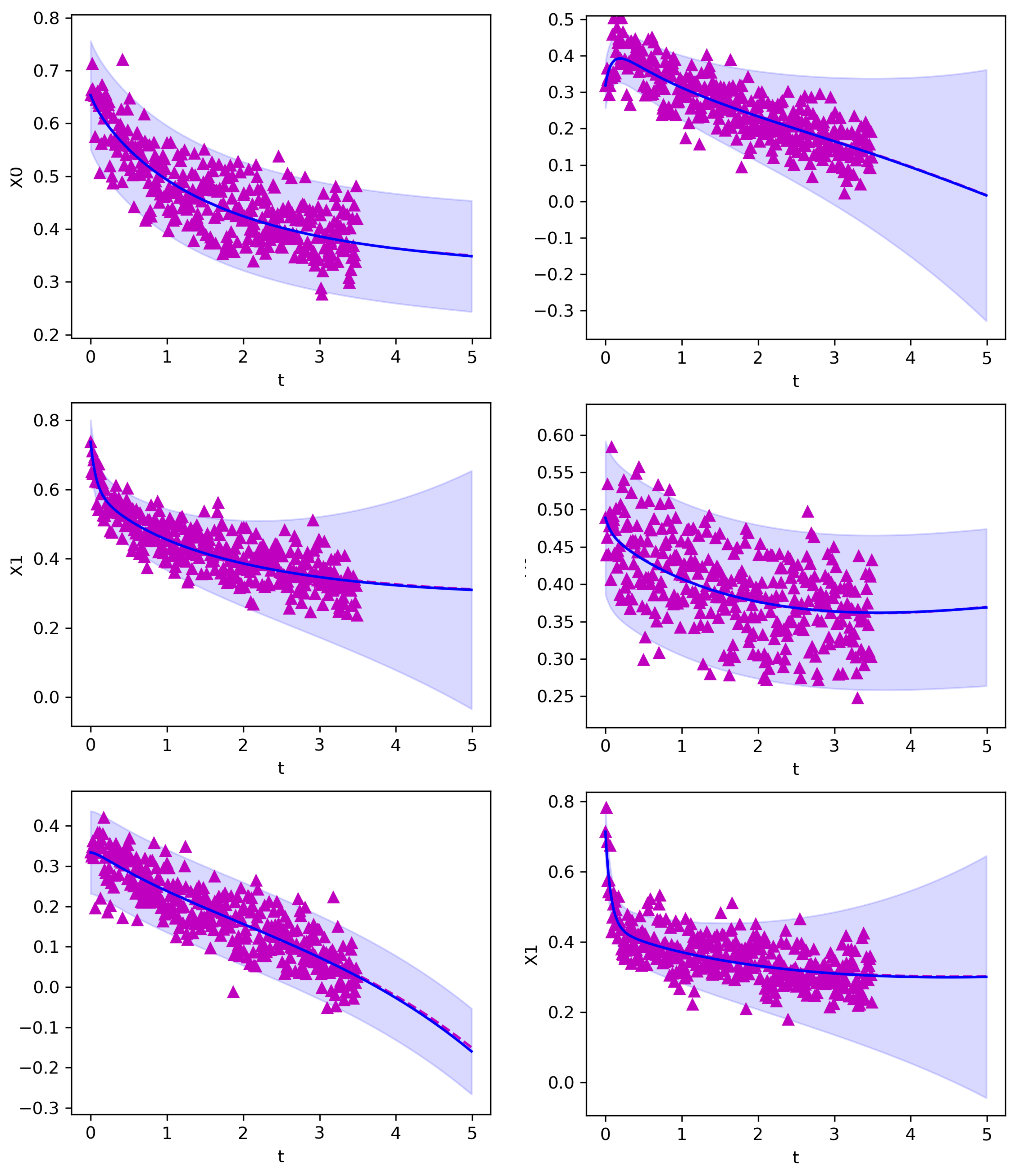


Jantre et al.

Functional subspace reduction

- Toy example: linear function $g_{ML}(v_1, v_2) = 10(v_1 - v_2)$
- Recover the correct function, assuming it's linear
  - Note that we never observe this function directly, just the PDE solutions
- Predictive uncertainties also well calibrated
- Like the Cahn-Hilliard neural PDE, we can <u>run the learned hybrid model for new initial or boundary conditions without having trained on them</u>

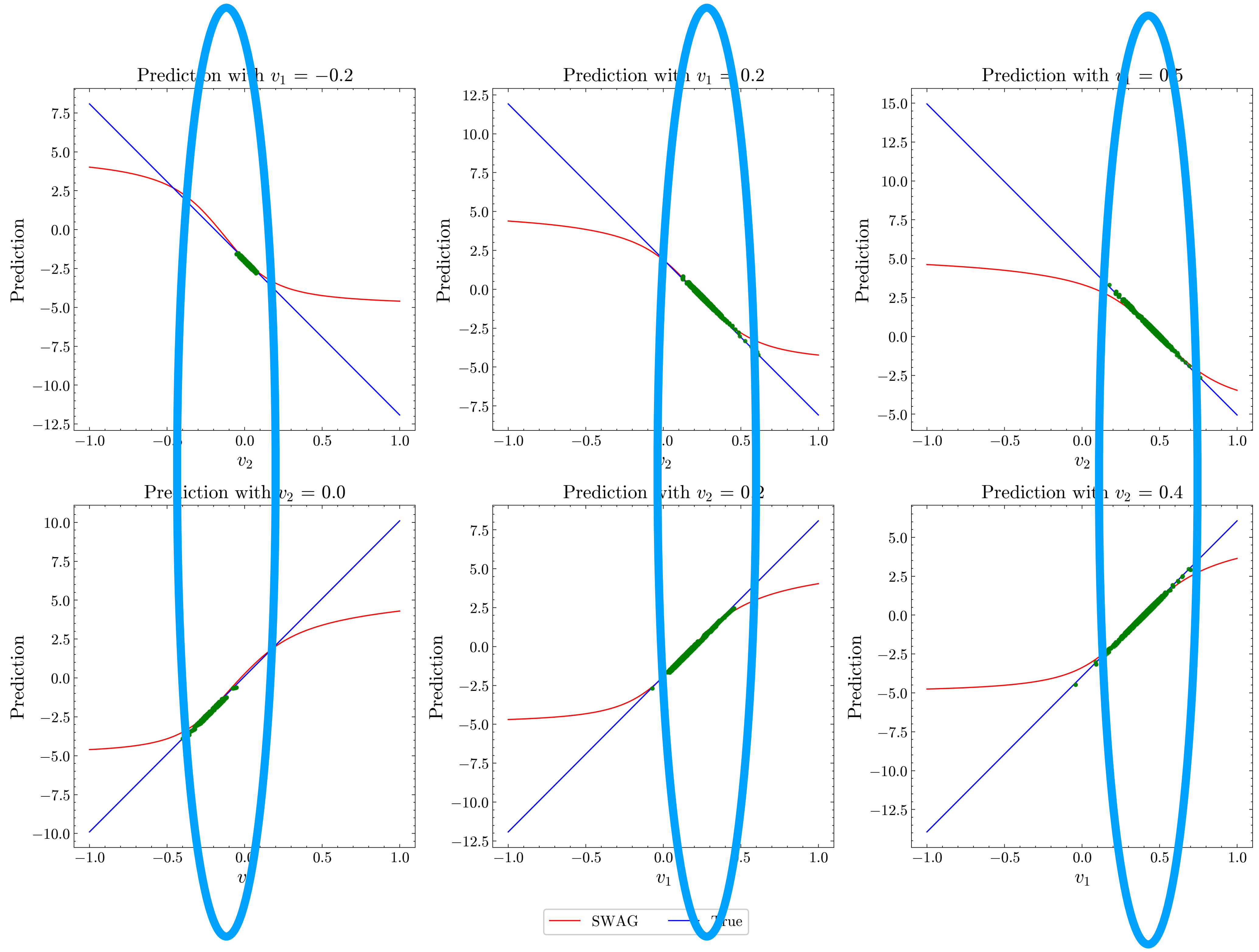
Predictive uncertainty (selected grid cells)



Jantre et al.

Functional subspace reduction

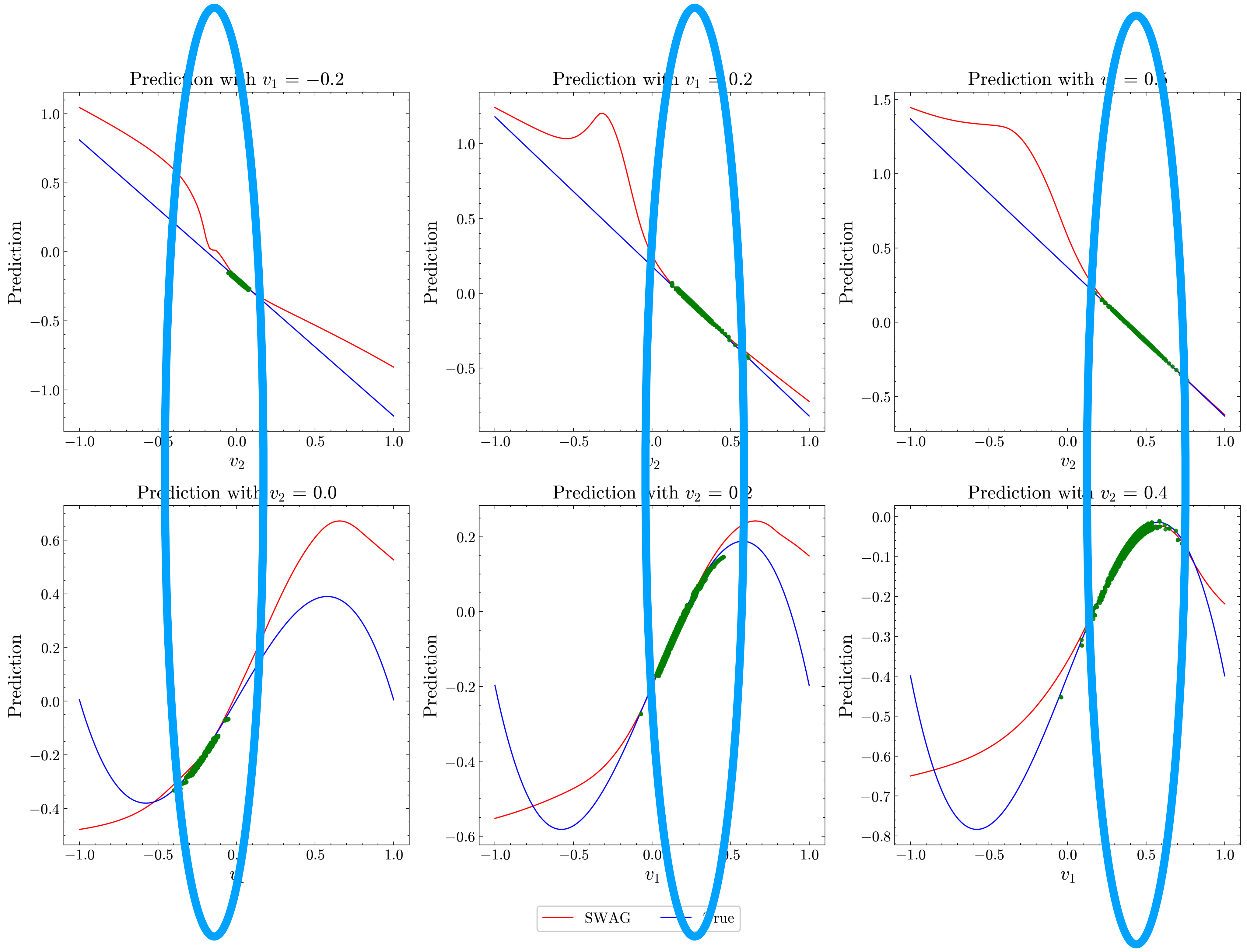
- Toy example: linear function $g_{ML}(v_1, v_2) = 10(v_1 - v_2)$
- Now try full nonlinear NN function approximation
  - Recovers linear function for the states that are highly sampled by the hybrid model
  - Reverts to constant prior outside those states
  - … but ok for prediction *when* solutions live in that region of state space
- → Data augmentation and active learning
  - Force model to sample where NN is uncertain



Jantre et al.

Functional subspace reduction

- Nonlinear function $g_{ML}(v_1, v_2) = v_1 - v_1^3 - v_2 - 0.005$
  - $\partial v_1/\partial t = D_1 \nabla^2 v_1 + g_{ML}(v_1, v_2)$
  - $\partial v_2/\partial t = D_2 \nabla^2 v_2 + 10(v_1 - v_2)$
- NN nonlinear function approximation
  - Recovers sinusoidal-linear function for states sampled by the hybrid model
  - Unconstrained outside those states
  - … but ok for prediction *when* solutions live in that region of state space
- → Data augmentation and active learning
  - Force model to sample where NN is uncertain



Jantre et al.

Model reduction to accelerate functional UQ

- Digital twins (e.g., PDE solvers) can be very computationally expensive
- Even if the space of uncertainties is reduced, it might still be computationally infeasible to sample them
- <u>Idea:</u> Automatically construct a fast surrogate model for *any* functional term in the digital twin
- <u>Approach:</u> Principal orthogonal decomposition (POD) Galerkin projection reduced order model (ROM)
  - Projects dynamics onto reduced state subspace by modal decomposition of solution data
  - Converts the <u>equations</u> of the full order model into a smaller set of <u>equations</u> that are faster to solve

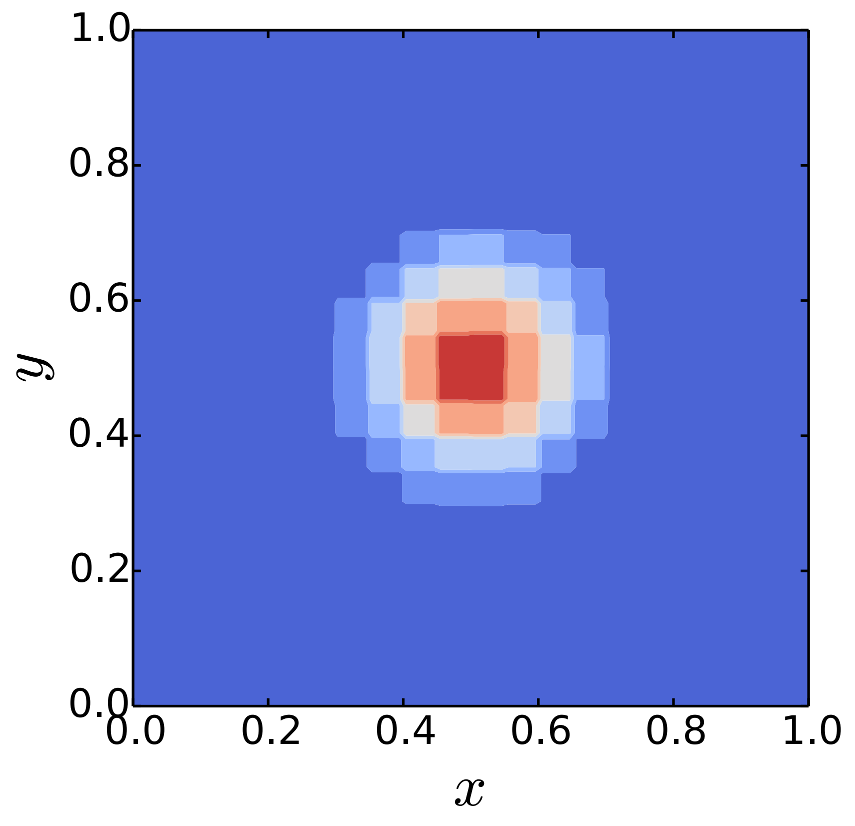
projection $\tilde{\mathbf{u}} = P\mathbf{u},$ **lifting** $\mathbf{u} = L\tilde{\mathbf{u}}$

original system

$$d\mathbf{u}/dt = N(\mathbf{u})$$

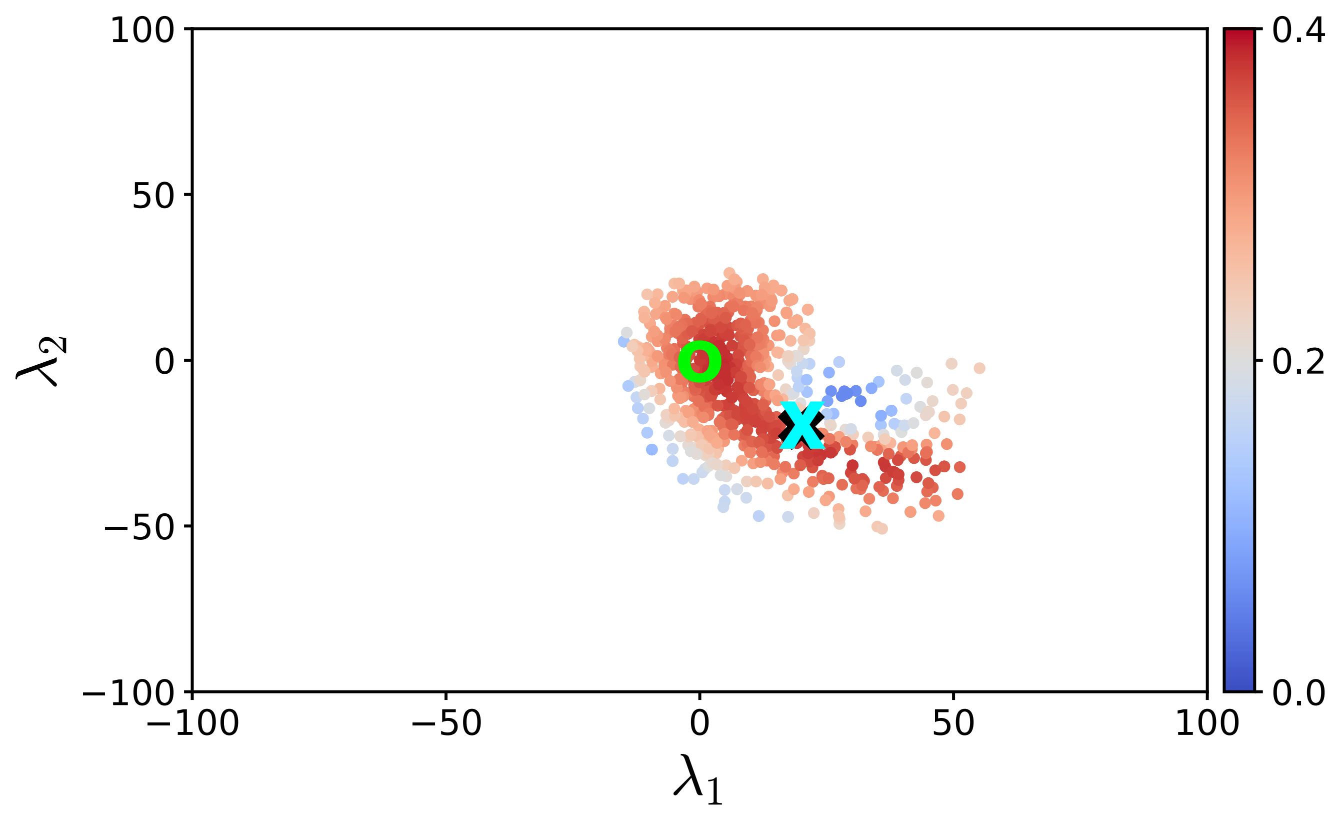
$$\Longrightarrow \quad d\tilde{\mathbf{u}}/dt = PN(L\tilde{\mathbf{u}}) \equiv \tilde{N}(\tilde{\mathbf{u}})$$

reduced system



Observed state

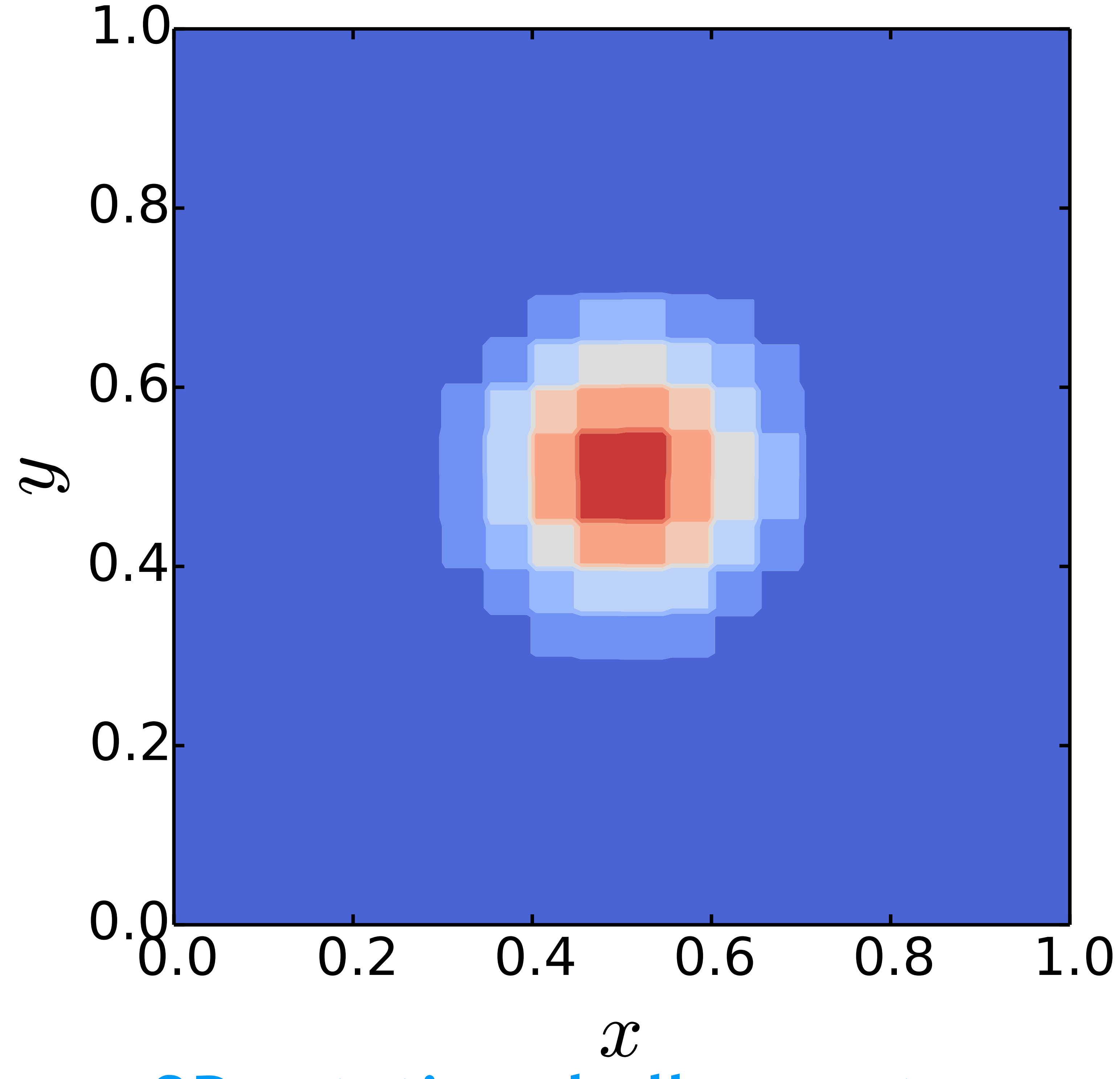
DeGennaro et al. (2019)



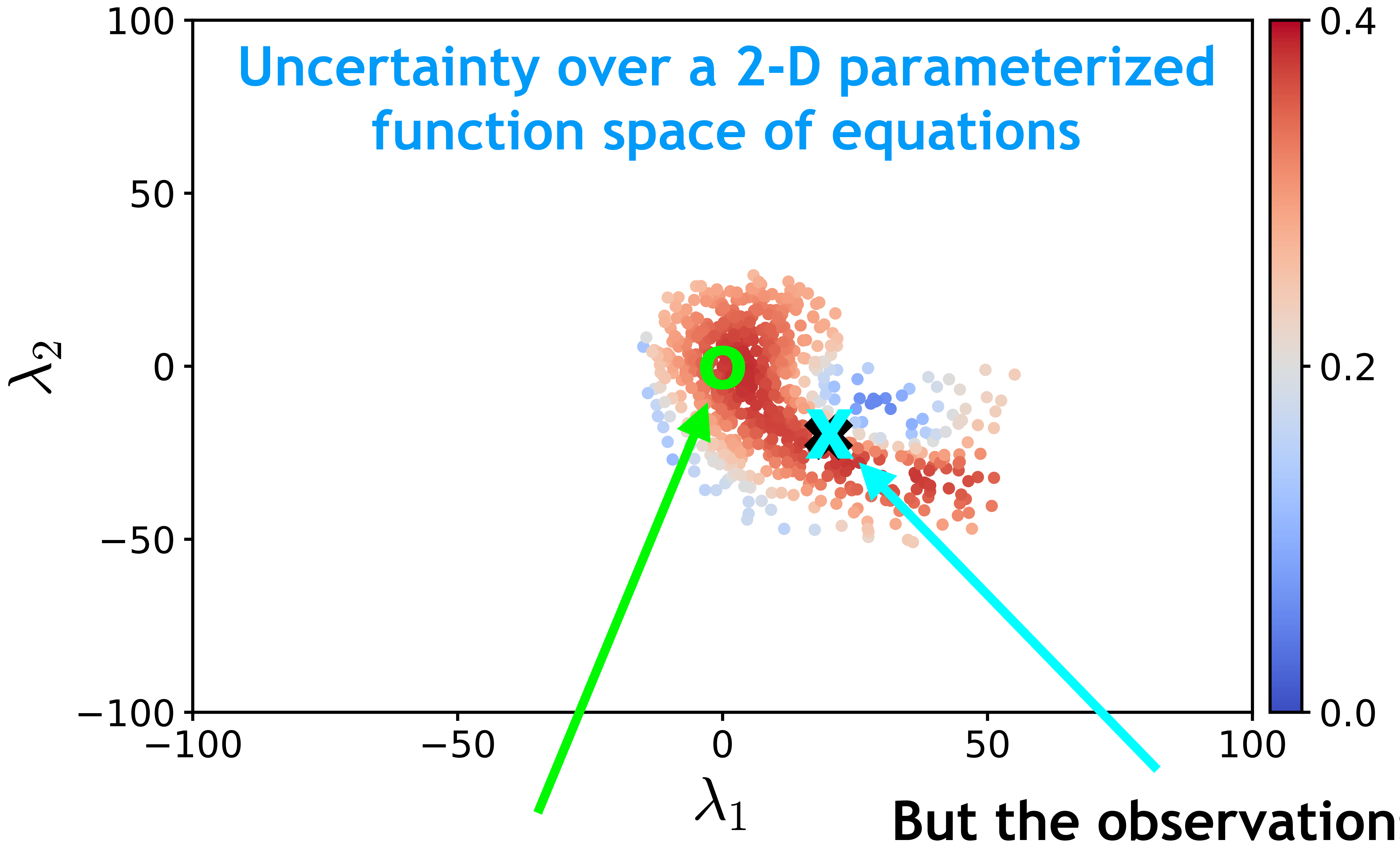
Inference over 2-parameter family of equations

“Few-shot learning”: Construct a ROM from a <u>single</u> training example, and predict the physics of a different, never-seen system

projection $\tilde{\mathbf{u}} = P\mathbf{u}$, **lifting** $\mathbf{u} = L\tilde{\mathbf{u}}$ **original system** $d\mathbf{u}/dt = N(\mathbf{u}) \implies$ **reduced system** $d\tilde{\mathbf{u}}/dt = PN(L\tilde{\mathbf{u}}) \equiv \tilde{N}(\tilde{\mathbf{u}})$



2D rotating shallow water equations: observed state



The ROM was trained with data from this equation

But the observations came from this equation … and we recover the <u>true</u> equations in our posterior distribution

DeGennaro et al. (2019)

Future directions

- Foundational mathematical research on how to construct surrogates of high-dimensional systems and functional uncertainty spaces from small training sets
- Streaming / online / realtime updating of uncertainties
- Improved sample design and active learning to generate optimal training data
- Not discussed here: **Decisions!**
  - Closing the DT loop, automation
  - Experimental design: explore (reduce uncertainty) vs. exploit (optimize system)
    - Bayesian decision theory, mean objective cost of uncertainty, Bayesian optimization, dynamic programming/tree search, …
  - Other decision problems
    - Control (e.g., accelerators), design (e.g., of materials, molecules, facilities), planning (climate resilience, urban systems, Big Science campaigns)

Conclusions

- The measure-act-control loop of DTs has focused attention on ingesting state information about the system (e.g. realtime data assimilation)
- However, it is also important to improve the DT's representation of the system's governing dynamics (parameter estimation, system identification, etc.)
- This is an enormous computational challenge (high-dimensional input spaces)
- Use reduced models and system identification to avoid need for large training sets
- Approaches:
    - Multifidelity hierarchy of digital twins
    - System identification (learning unknown dynamics / missing processes)
    - Function-space uncertainty quantification and subspace reduction
    - Hybrid physics-ML models
    - Data augmentation and active learning
    - Automated model reduction