AI Surrogates as Adjuncts to Traditional HPC Simulations

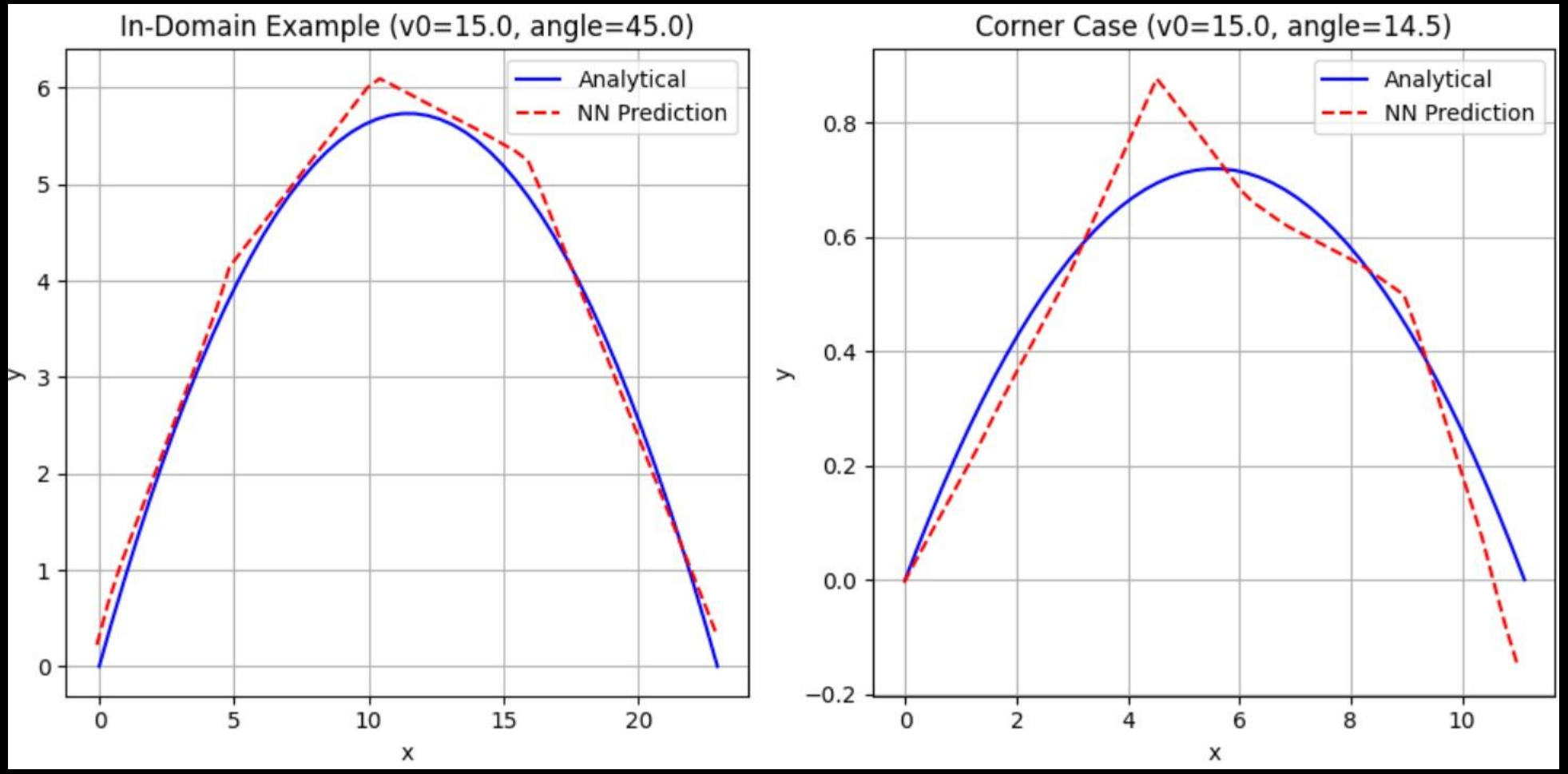
Alessandro Fanfarillo

AMD
together we advance_

Introduction

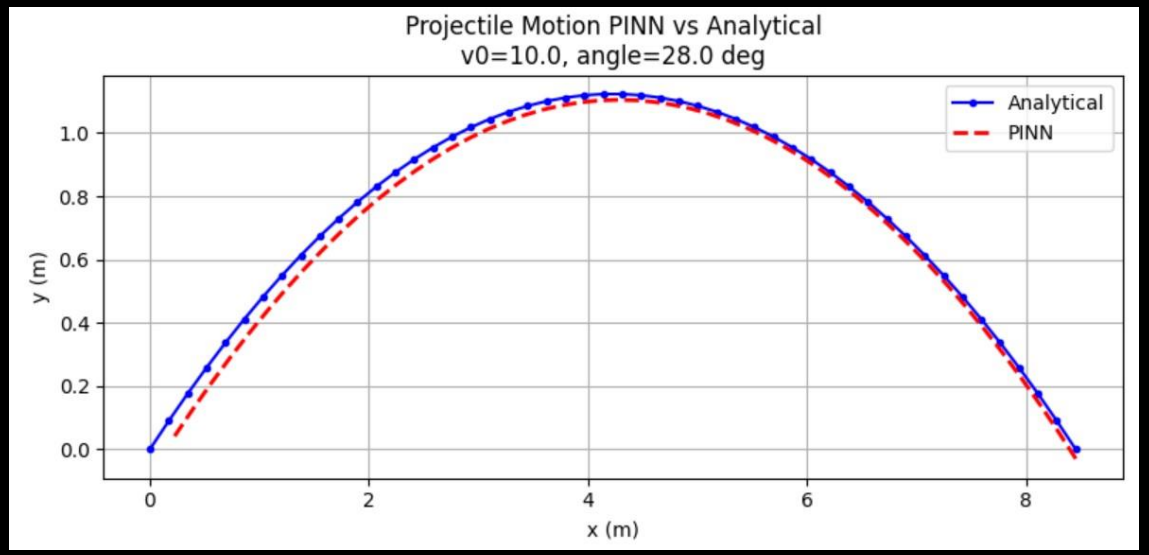
- Ideally, AI models can replace or approximate computationally intensive physics-based components
- In reality, surrogate models often struggle to generalize beyond the range of data they were trained on
- Surrogate models are often poor at predicting rare or extreme events, as these are underrepresented in the training data
- ODEs/PDEs can be solved with AI, but there is no guarantee it will always work as expected
- Physics-Informed Neural Networks (PINNs) may be a good option but difficult convergence and approximation due to unbalanced gradients
- Given these limitations, how can we use AI to improve traditional HPC applications? Some ideas:
  - Synthetic data generation, parametrization, estimation, threshold definition, decision making, dimensionality reduction, etc.
- AI is very useful as an auxiliary component to scientific applications, not as a full replacement

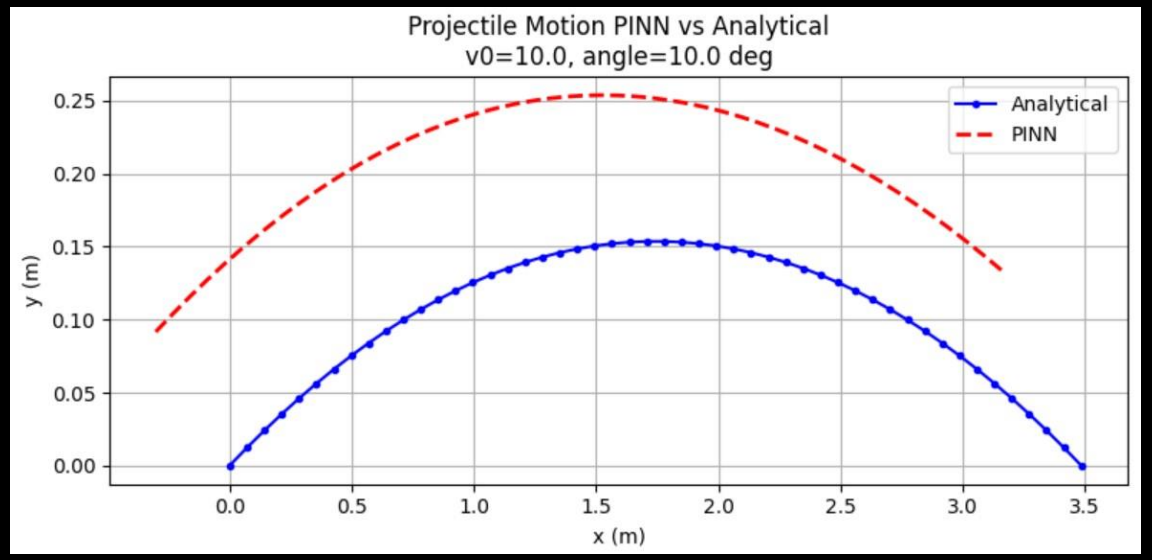
Projectile Motion: DNN



DNN trained on angles 15-75 degrees behaves ok, but not for 14.5 degrees

Projectile Motion: PINN





PINN use a more complex loss function, including laws of physics governing the phenomenon

Trained on angles 15-75 degrees behaves great, but still not perfect for out of training cases

New techniques on PINNs improved a lot the result, but not a silver bullet

Generative vs. Discriminative Models

- Generative models aim to learn the **joint probability distribution**, P(X,Y), where: X is the input data (features) and Y is the output labels (targets)
  - Model how the data is generated, compute P(Y|X) for classification and P(X), the probability of input data needed to generate new synthetic samples
  - Examples: VAE, GANs, Normalizing Flows, Diffusion Models, Bayesian generative models, etc.
- Discriminative models focus on learning the **conditional probability distribution**, P(Y|X) directly
  - Optimized to distinguish between different classes/outputs given the input features. Directly find decision boundaries
  - Examples: SVMs, DNNs, Decision Trees, Random Forests, etc.

- Generative models are more appropriate when the goal is to **understand the underlying data distribution**, **generate new data**, or **simulate complex systems** in a probabilistic way
- Discriminative models are often the best choice for surrogate modeling in HPC when the goal is to make **predictions** (e.g., regression or classification) about system behavior based on input data

Case n. 1: save storage space with a generative model

In weather and climate modelling, storing the huge amount of data produced by the models is usually the main limitation. The performance of post-processing tasks is limited by IO traffic and memory availability

Idea: train a generative model to replace an entire dataset stored on disk

Proposed approach: use a Conditional Variation Auto Encoder (CVAE) to generate new samples of temperatures for a particular city, by providing the month and day of the year as condition

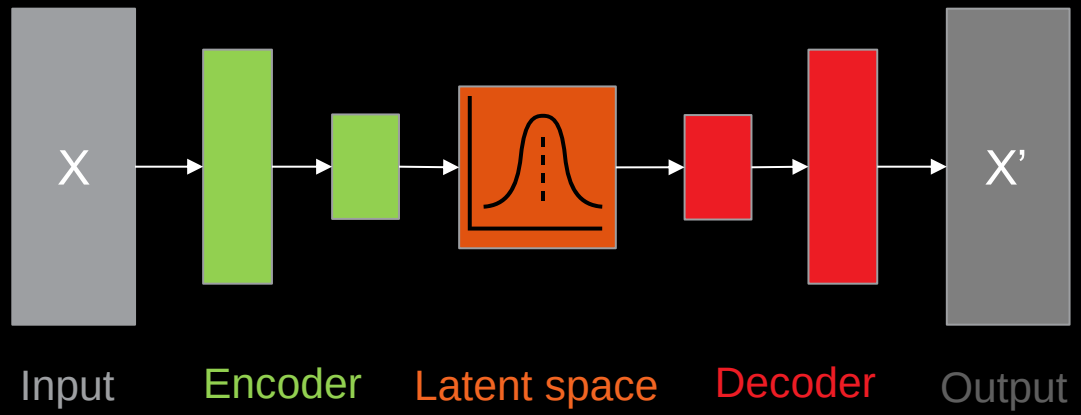
This is a probabilistic approach, so the distribution modelled by the CVAE should be very close to the real distribution

- Probabilistic Forecasting using Deep Generative Models – Fanfarillo et al. - https://arxiv.org/pdf/1909.11865
- Generative deep learning for probabilistic streamflow forecasting: Conditional variational auto-encoder - MS Jahangir et al. - https://www.sciencedirect.com/science/article/abs/pii/S0022169423014403
- Deep generative models in energy system applications: Review, challenges, and future directions – Zhang et al. - https://www.sciencedirect.com/science/article/abs/pii/S0306261924024437

Conditional Variational Autoencoder

An extension of the Variational Autoencoder (VAE) that generates data conditioned on additional information (e.g., labels or attributes). Combines deep neural networks with probabilistic modeling to capture complex data distributions

- Encoder Network: learns a conditional distribution of latent variables given the input data and condition
- Latent Space: encodes underlying features of the data in a continuous, multidimensional space
- Decoder Network: reconstructs or generates data from latent variables conditioned on the same attributes



Generate synthetic data as a dataset surrogate

Synthetic data can be generated using:

- the decoder used during training
- random samples from a Normal distribution with dimension == latent dimension

Idea: replacing an entire dataset (or part of it) by using just the weights of the decoder to generate the data needed.

Some quantities that can be generated synthetically:

- sea surface temperature
- highest temperature

The condition in the CVAE can be the day and month of interest

Training requires tuning of loss function, balancing KL divergence and reconstruction loss

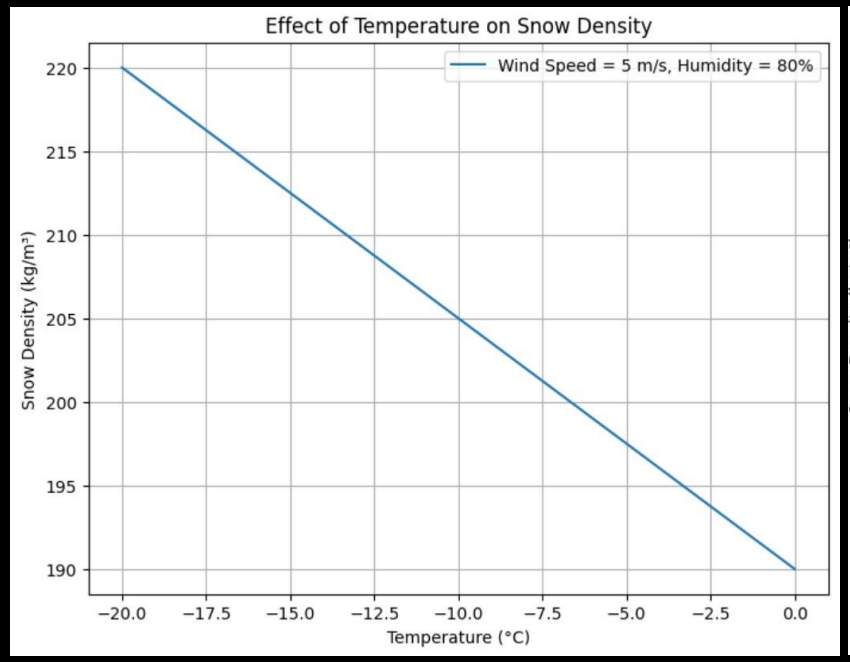
Alternative approach: use a Normalizing Flow

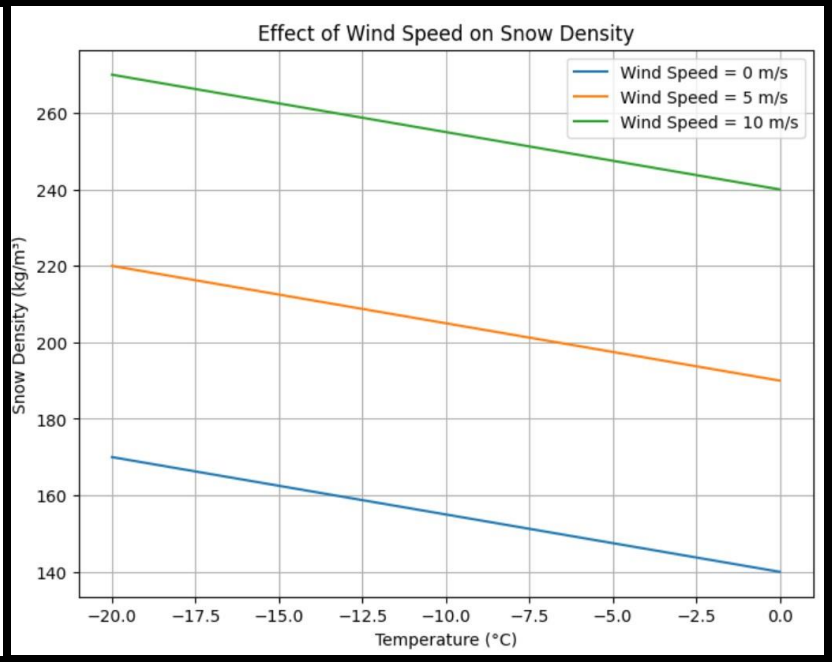
Case n. 2: use AI for efficient model parametrization

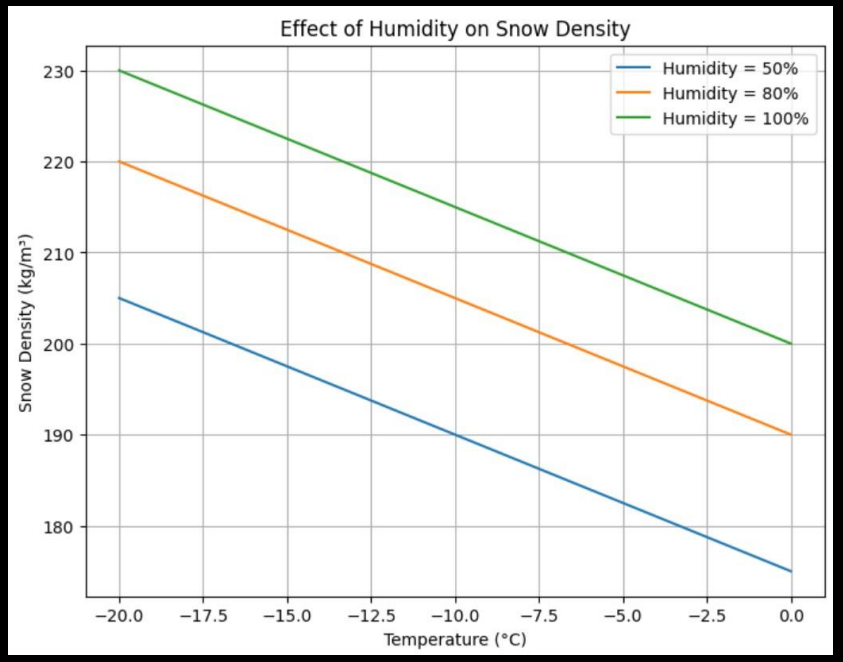
- Scientific applications rely on “set and forget” parameters to model certain phenomenon (e.g., snow density)
- An approach that dynamically sets said parameters for each case can provide much better accuracy
- There are many possible ways to select “good” parameters for each case, a particular effective way is to rely on Reinforcement Learning (RL) techniques
- In a reinforcement learning setting, usually (but not necessarily) a neural network is used to model what the optimal next action should be (policy) or what the values of the possible actions are and select the best
- Use RL to dynamically select the parameters modelling snow density at each time step

Snow Density Modelling

- Use a simple formula involving three factors: temperature, wind speed, and humidity
- Each factor weighted by fixed parameters: alpha_T, alpha_u, alpha_h
- Problem is to guess these three parameters for optimal accuracy







Case n. 3: Source-term estimation problem

- A **source-term estimation problem** is about finding the location and strength of something that is being released or emitted into an environment. Source could be a chemical, gas, pollutant, etc.
- In a scientific context, source-term estimation typically involves using measurements (e.g., sensors detecting the substance) and a mathematical model of how the substance spreads (e.g., how gas flows in the air). The goal is to work backward from the measurements to identify the source
- This is an inverse problem, where we start from the effects and try to find out the cause
- In a traditional way, inverse problems can be solved by running multiple simulations of where the source might be and assess whether the effects are consistent with the observations
- AI is very effective in solving these kind of problems
- We will use a heat diffusion problem as an example (video)

Conclusions

- AI can be used to enhance traditional HPC application but not a full replacement
- Good to make (optimal) decisions, parametrization, generate synthetic data, approximate functions
- Replace small parts of app with AI, understand if inputs may change and how AI will deal with it
- There is no single model/approach that works well for a certain problem (CVAE vs. NF vs. GANs)
- PINNs are a valid option but not a silver bullet
- Interpretability is important, covered on next talk
- Think out of the box (and in CS terms) on where AI can really improve the performance of your app!

Disclaimers and Trademark

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like.

Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information.

However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY

IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE.

IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.


AMD