THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 14: Physics-Informed Neural Networks (PINNs)

Sergei V. Kalinin

Physics-Informed Neural Networks

We have:

- a differential equation $g(x, y) = 0$,
- some data $\{x_j, y_j\}$ and
- a neural network $f(x \mid \theta)$ that approximates y.

For a PINN, we would get a loss function that looks like the following,

$$Loss_{PINN} = \underbrace{\frac{1}{N}\sum_{j}^{N} ||f(x_j|\theta) - y_j||_2^2}_{\text{Data loss}} + \lambda \underbrace{\frac{1}{M}\sum_{i}^{M} ||g(x_i, f(x_i, |\theta))||_2^2}_{\text{Physics loss}}$$

- Here x_i are *collocation* points. These can be any value we want them to be, usually you would want them to be in the range of values we are interested in.
- The x_j and y_j are our data.
- We can also add a parameter controlling the relative strength of the data loss function and the physics loss function, here we use λ.
- And then just train as you would any other neural network.

https://medium.com/@theo.wolf/physics-informed-neural-networks-a-simple-tutorial-with-pytorch-f28a890b874a

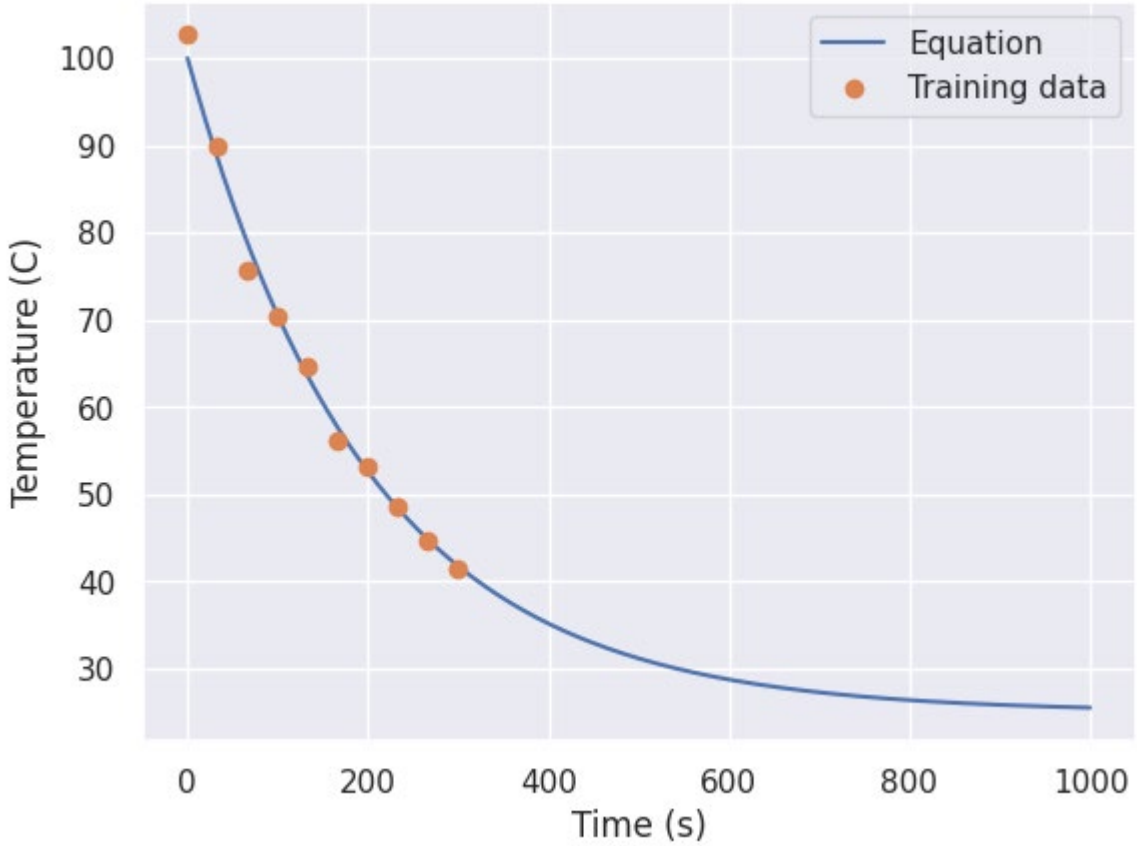
NNs and PINNs for a simple cooling problem

$$\frac{dT(t)}{dt} = r(T_{env} - T(t))$$

$T(t)$: temperature

T_{env} : temperature of the environment

r : cooling rate



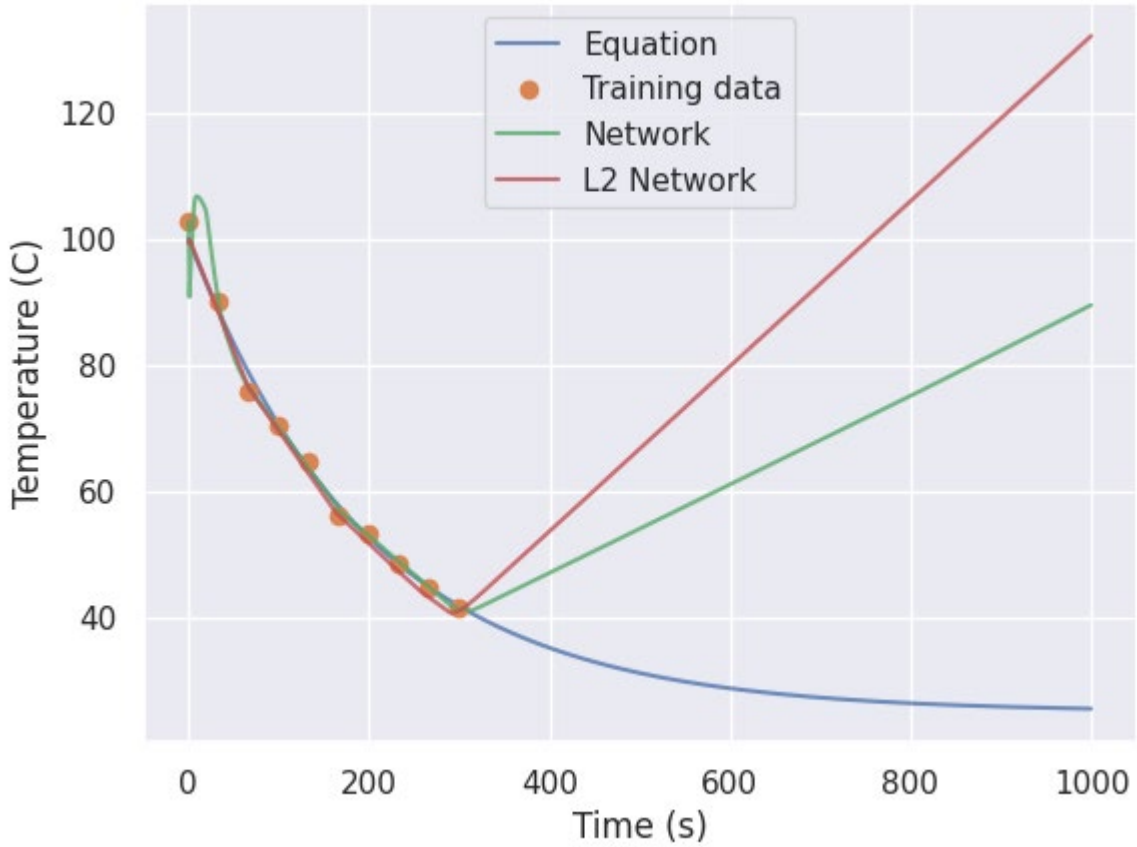
NNs Solution

$$\frac{dT(t)}{dt} = r(T_{env} - T(t))$$

$T(t)$: temperature

T_{env} : temperature of the environment

r : cooling rate



Setting up PINN

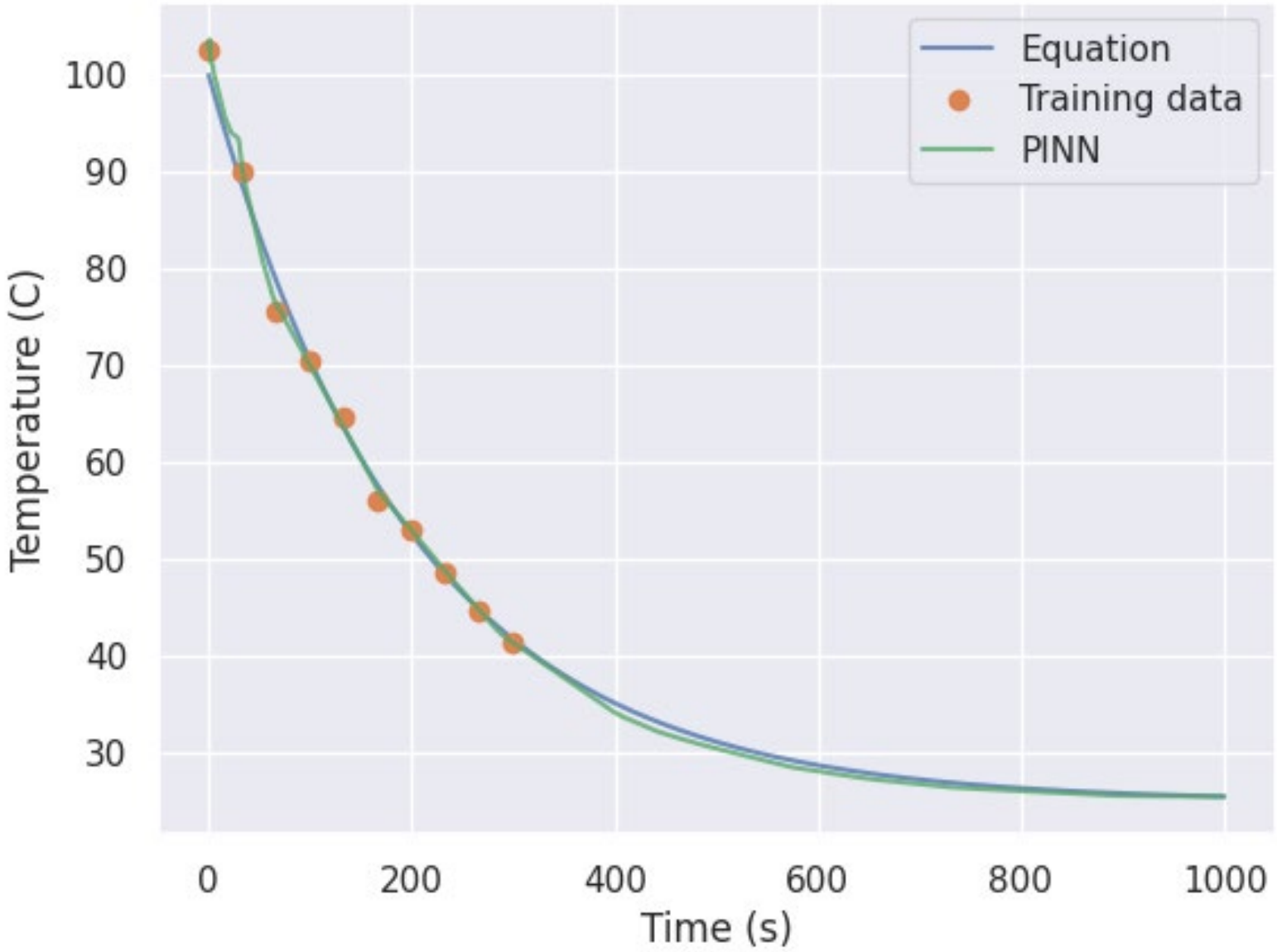
$$g(t,T) = \frac{dT(t)}{dt} - r(T_{env} - T(t)) = 0$$

$$g(t, f(t|\theta)) = \frac{df(t|\theta)}{dt} - r(T_{env} - f(t|\theta))$$

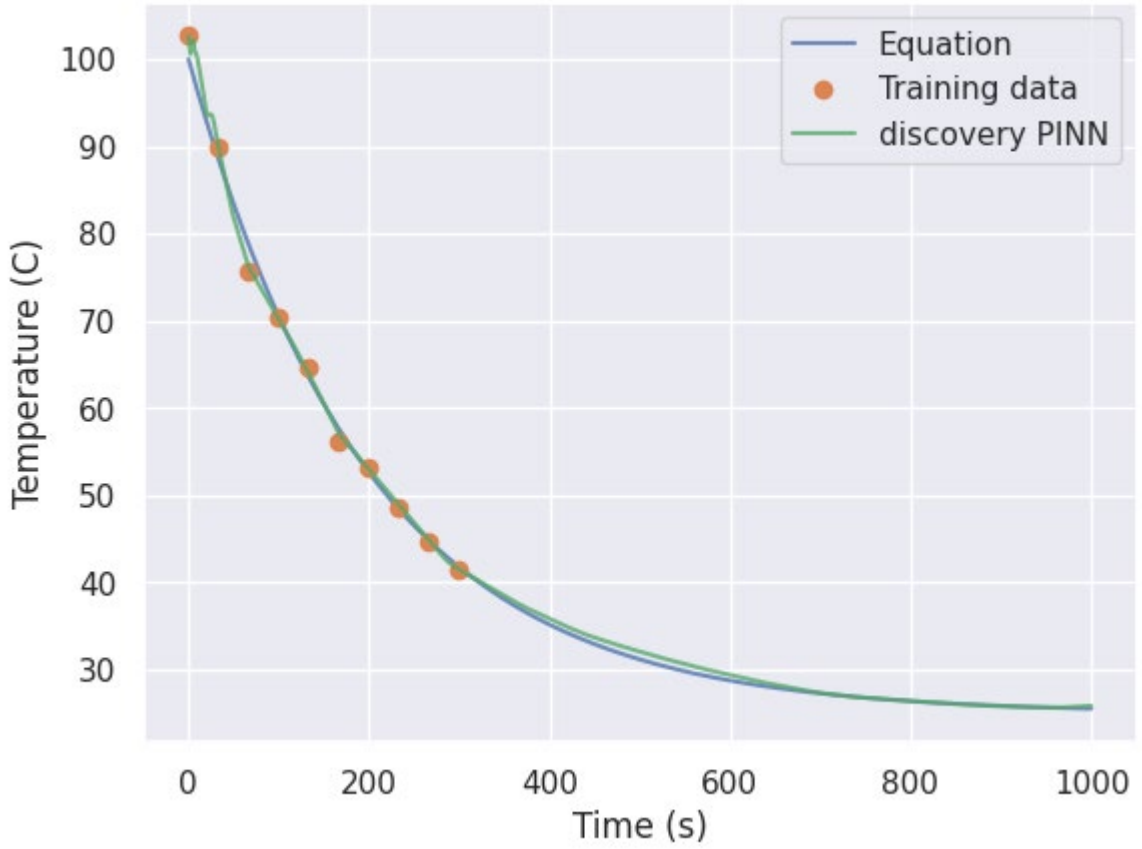
$$Loss_{PINN} = \underbrace{\frac{1}{10}\sum_{j}^{10}(f(t_j|\theta) - T_j)^2}_{\text{data loss}} + \underbrace{\lambda\frac{1}{M}\sum_{i}^{M}\left(\frac{df(t_i|\theta)}{dt_i} - r(T_{env} - f(t_i|\theta))\right)^2}_{\text{physics loss}}$$

To take the derivative of your neural network, *torch.autograd* module has a function called *grad()* which does exactly that (you can even take higher order derivatives). Just ensure that *create_graph* is set to True

PINN for known cooling rate



But what if the cooling rate is unknown?



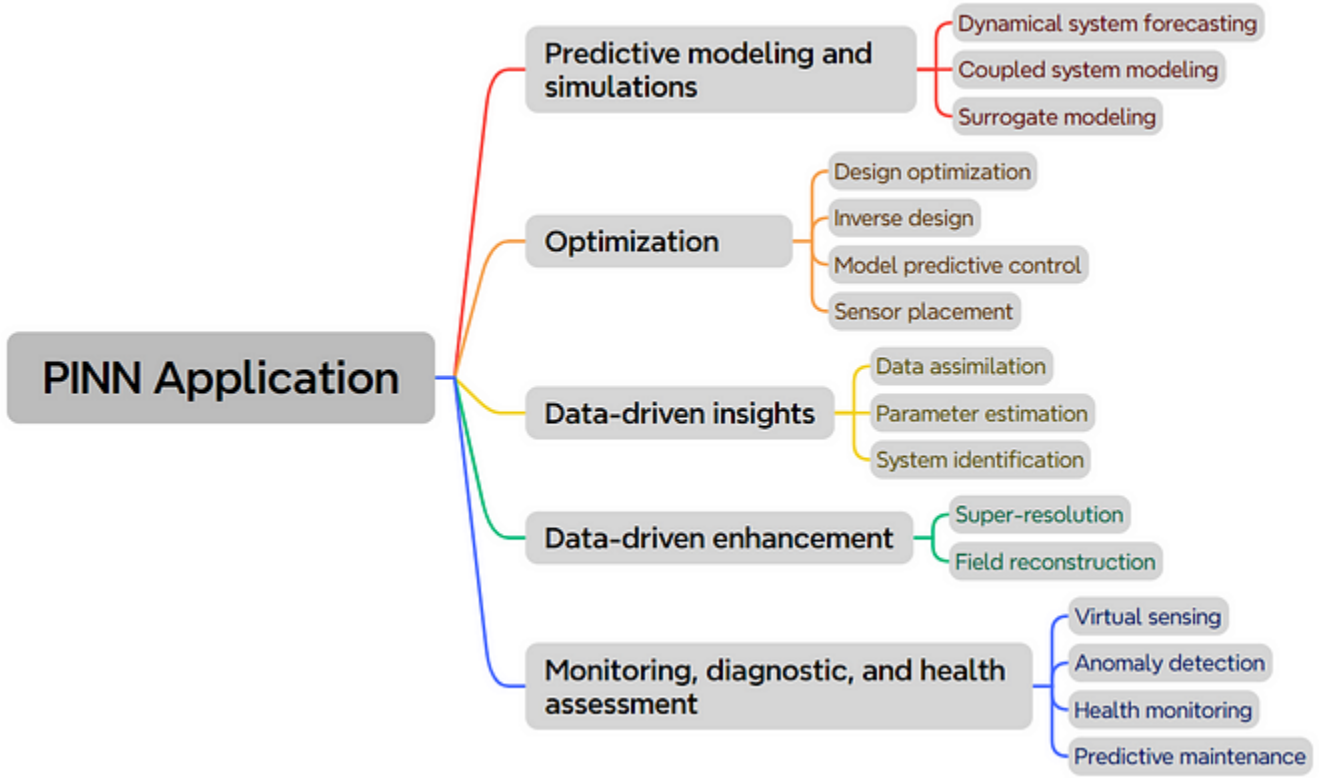
Our differential equation is then g(t, T | r) = 0 where r is unknown. Thanks to PyTorch, all we need to do is just one small change: add r as a differentiable parameter.

PINNs also work for PDEs!

Colab

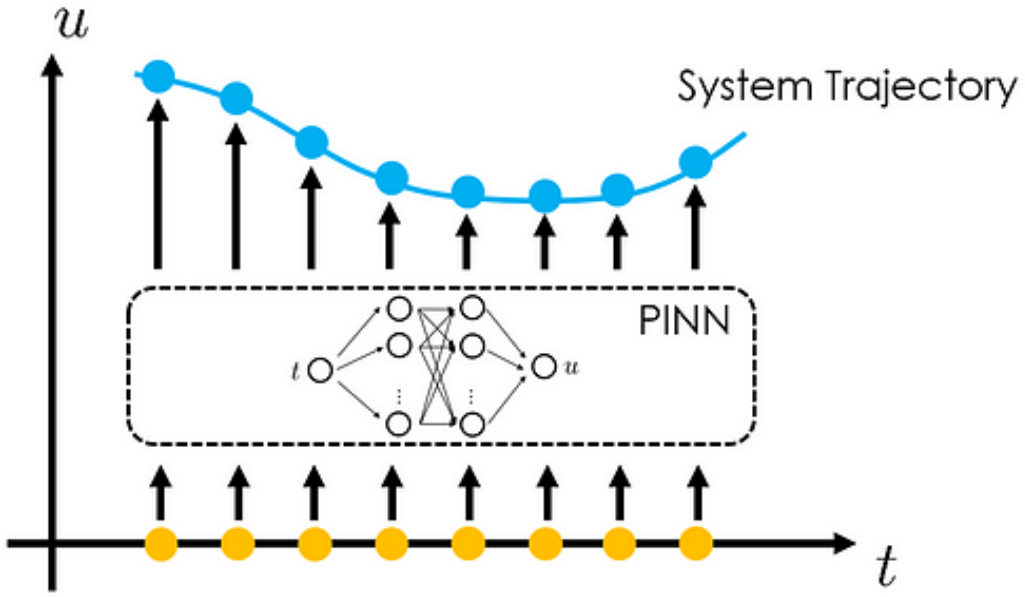
What PINNs can are used for?

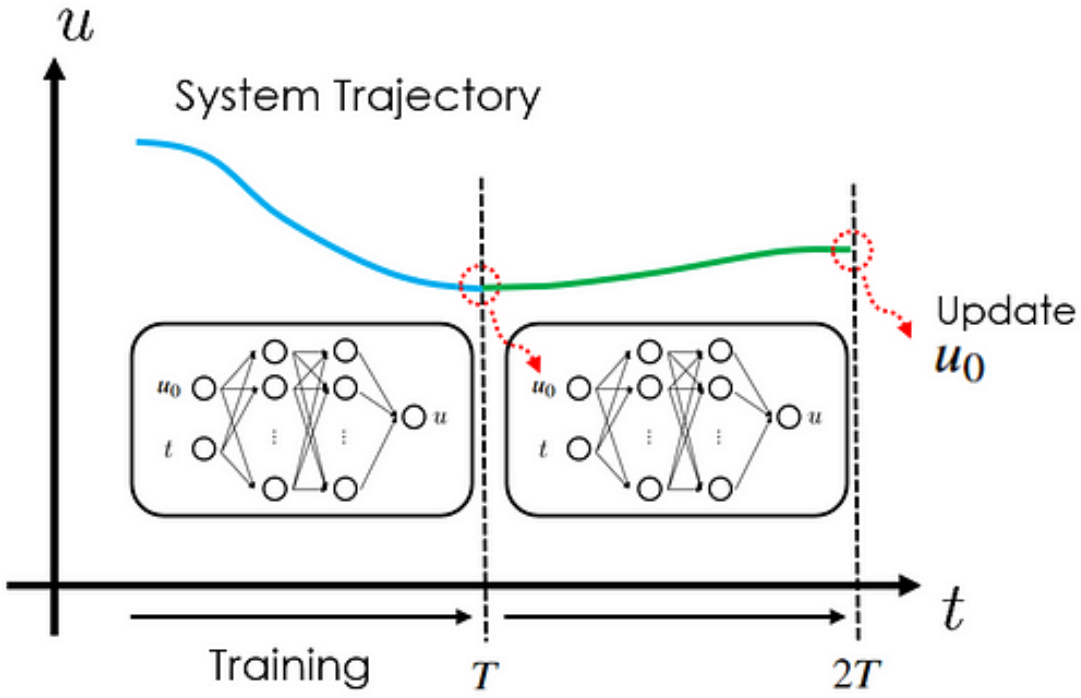
- Predictive modeling and simulations
- Optimization
- Data-driven insights
- Data-driven enhancement
- Monitoring, diagnostic, and health assessment



https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

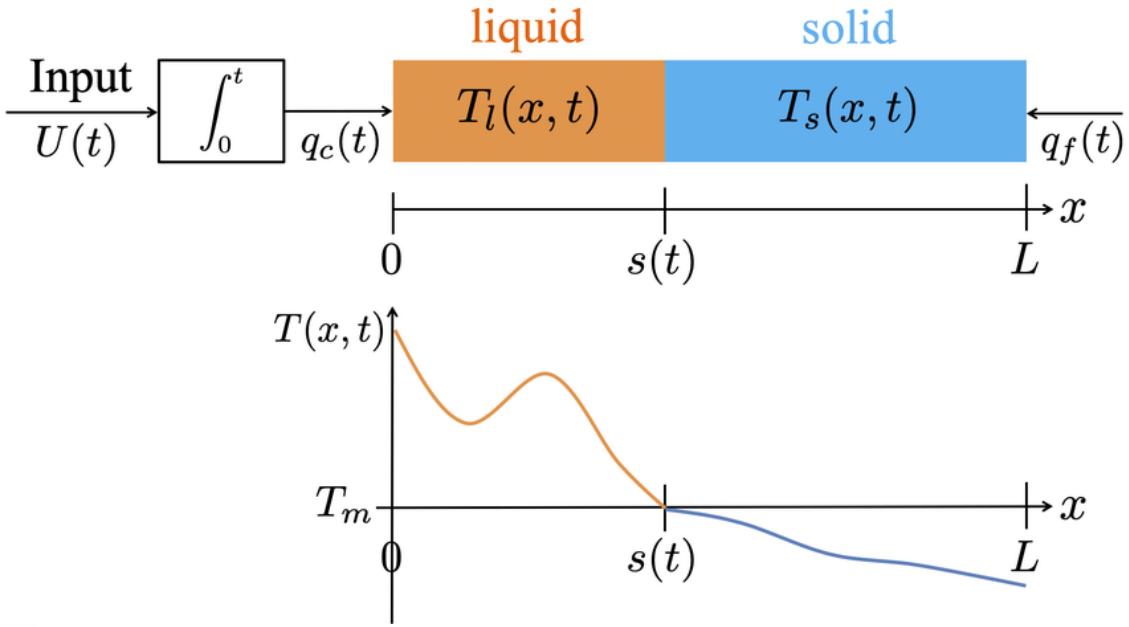
Dynamic System Forecasting





https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

Solving Coupled Problems: Stefan Problem



https://nypost.com/2013/09/17/stunning-images-of-live-lava-from-hawaiian-volcano/

https://www.researchgate.net/publication/355872792_Safe_PDE_Backstepping_QP_Control_with_High_Relative_Degree_CBFs_Stefan_Model_with_Actuator_Dynamics/figures?lo=1

https://brilliant.org/wiki/fractals/

From: Physics-Informed Neural Networks for Heat Transfer Problems

J. Heat Transfer. 2021;143(6). doi:10.1115/1.4050542

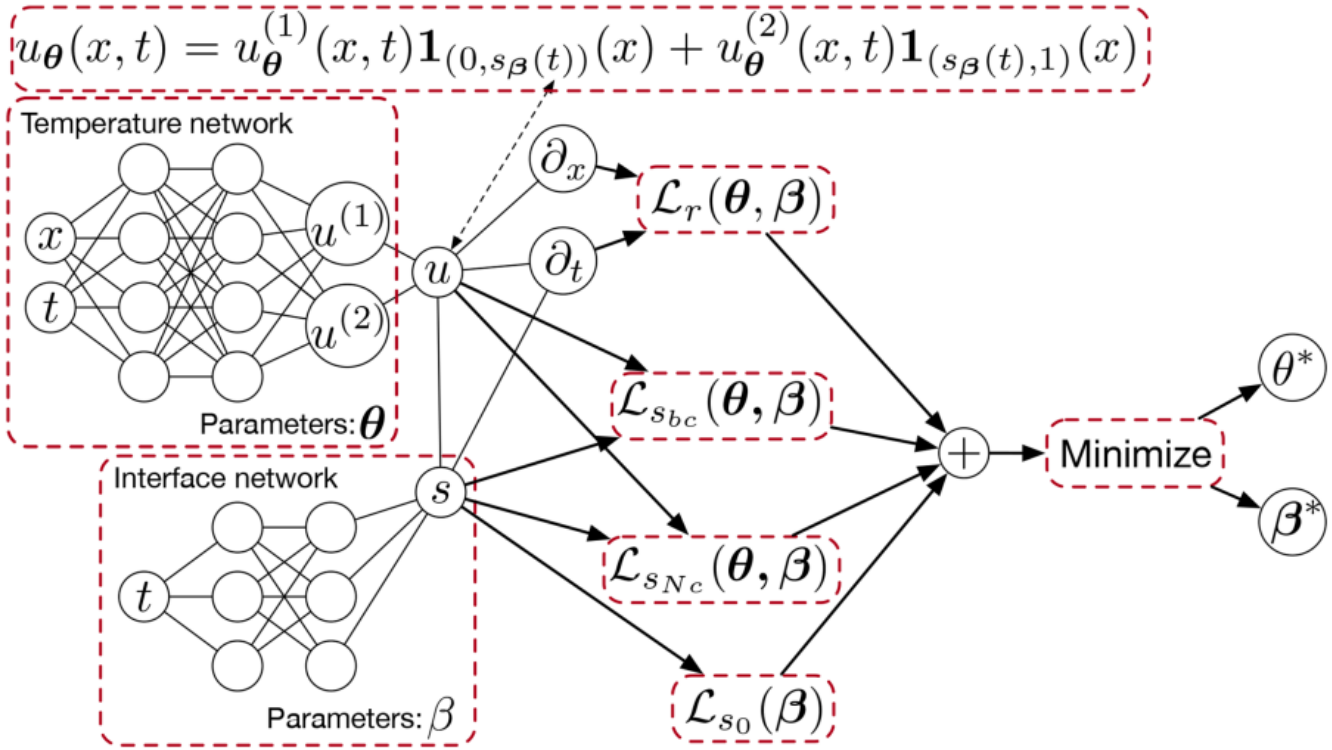
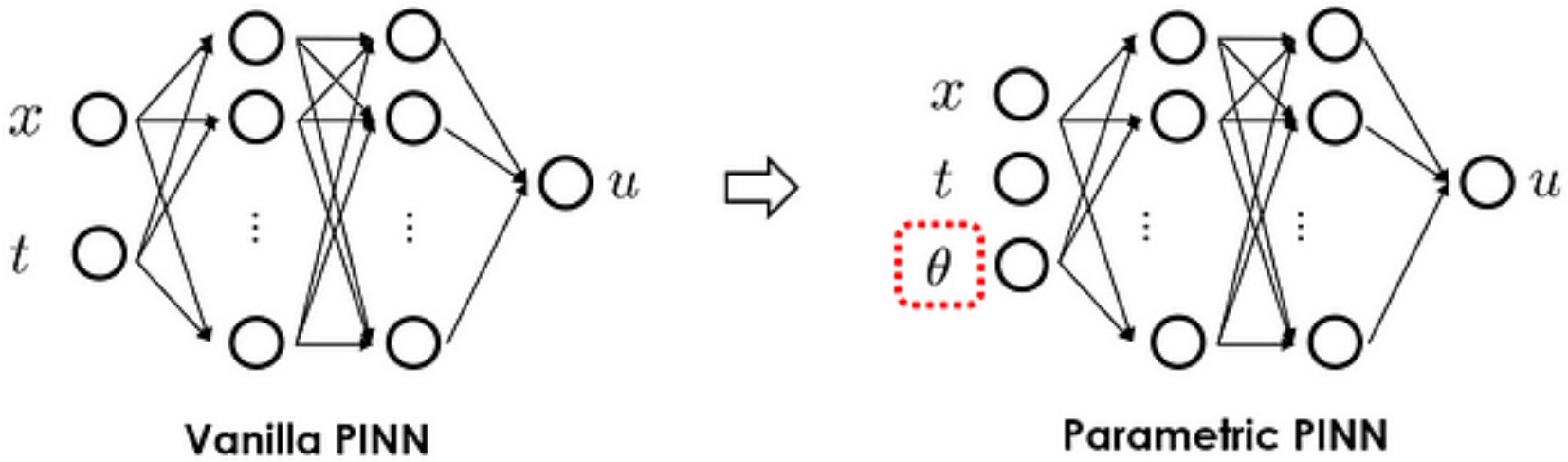


Figure Legend:

Two-phase Stefan problem: PINN architecture for inferring the latent temperature fields $u_1(x,t)$, $u_2(x,t)$, and phase-transition interface $s(t)$, from scattered noisy observations of temperature

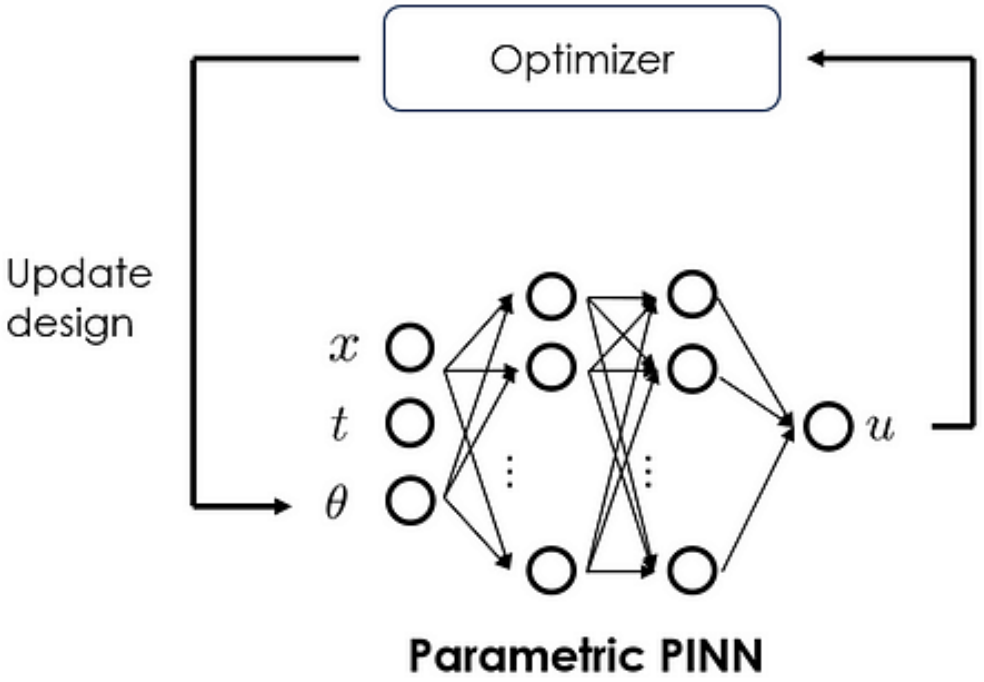
Surrogate Models



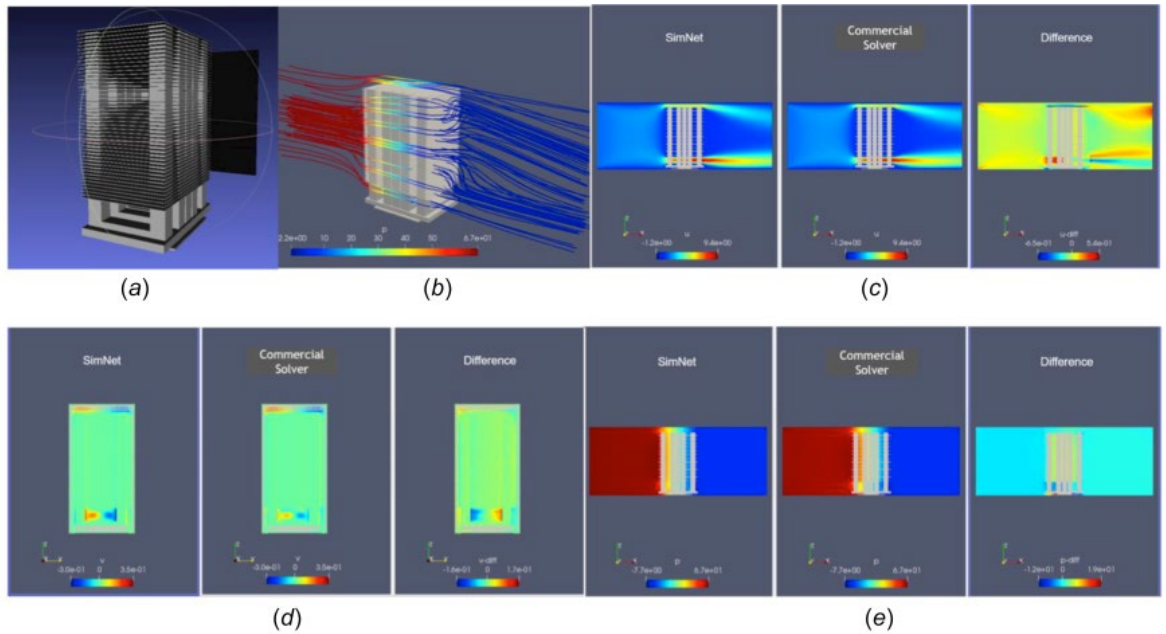
- For predictive modeling and simulations, we would like to ask "what-if" questions to examine the system's response given different sets of, e.g., boundary conditions, initial conditions, material properties, etc.
- Traditionally, we would need to run simulations many times where each run corresponds to one specific setting (i.e., BC, IC, material properties, etc.). Very time-consuming.
- Surrogate modeling refers to the practice of building machine learning models to approximate the input-output relationship conveyed by the simulator.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

Optimization



https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02



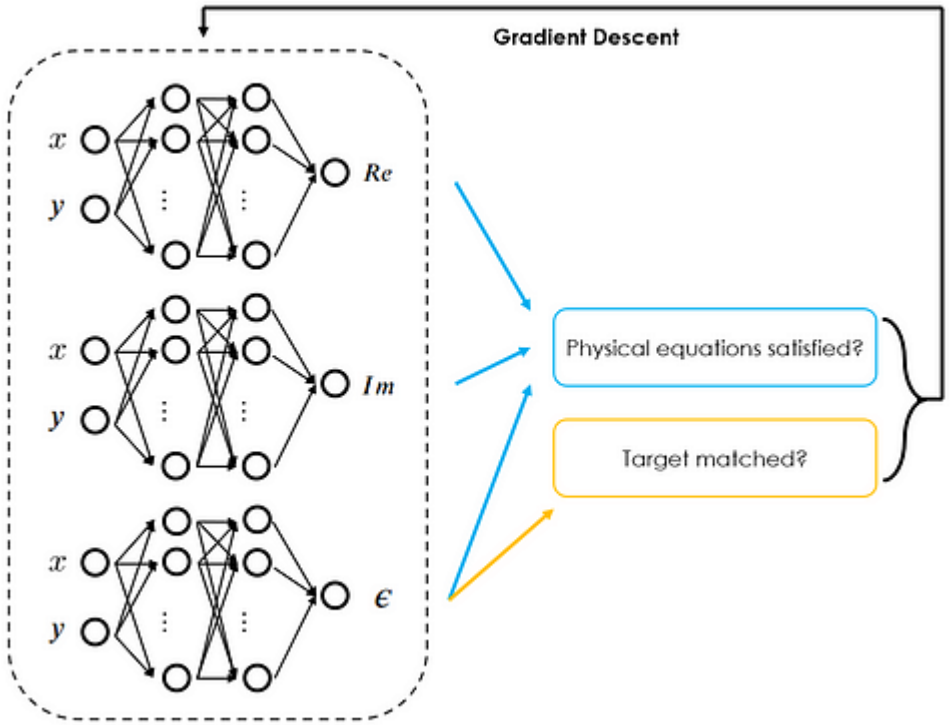
Turbulent Flow (Re=20,687)		
	Temperature	Pressure Drop
SimNet - Fourier Network	43.1 °C	3.56
Commercial Solver	43.5 °C	3.6
OpenFOAM	41.6 °C	4.58

Computational Times (10 parameters, 3 values per parameter)	
SimNet	1000 V100 GPU hrs.
Traditional Solver (OpenFOAM) 59,049 separate runs (26 wall hours on 12 CPU cores)	18.4M CPU core hrs.

From: Physics-Informed Neural Networks for Heat Transfer Problems

J. Heat Transfer. 2021;143(6). doi:10.1115/1.4050542

Inverse Design



- Inverse design refers to the scenario where the desired system performance is specified first, and then the system configurations that achieve this outcome are determined. Usually involves applying optimization algorithms to search for the optimal design parameters that meet pre-defined targets.

- Addressing inverse design problems relies heavily on simulations of how different designs will perform. However, as exploring a very large design space is usually required, traditional numerical ODE/PDE simulators that are slow to run may be inadequate to efficiently identify the optimal design.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

Inverse Design

PINNs are well-suited for solving the inverse design problem:

- PINNs can significantly speed up simulations compared to traditional numerical methods.
- PINNs are capable of efficiently handling high-dimensional problems and nonlinear dynamics, which is often the case in inverse design scenarios.
- PINNs are fully differentiable, meaning that we can easily compute the gradients of some objective with respect to the input parameters (e.g., using TensorFlow's tf.GradientTape). This opens the door for gradient-based optimization algorithms, which may facilitate more efficient searching in the design space.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

Inverse Design

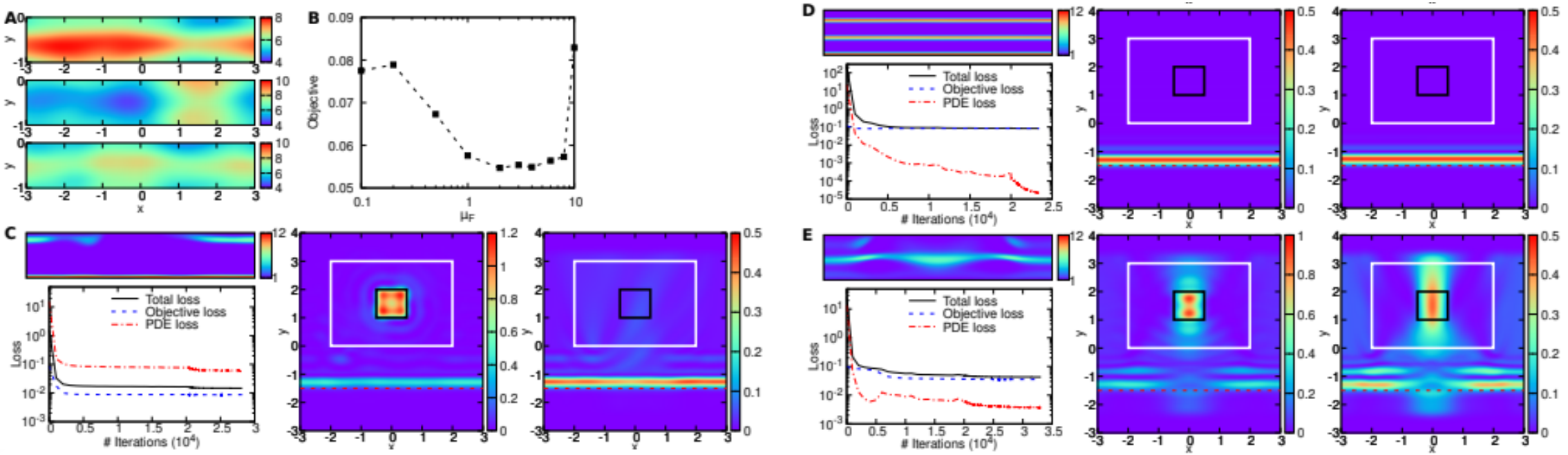
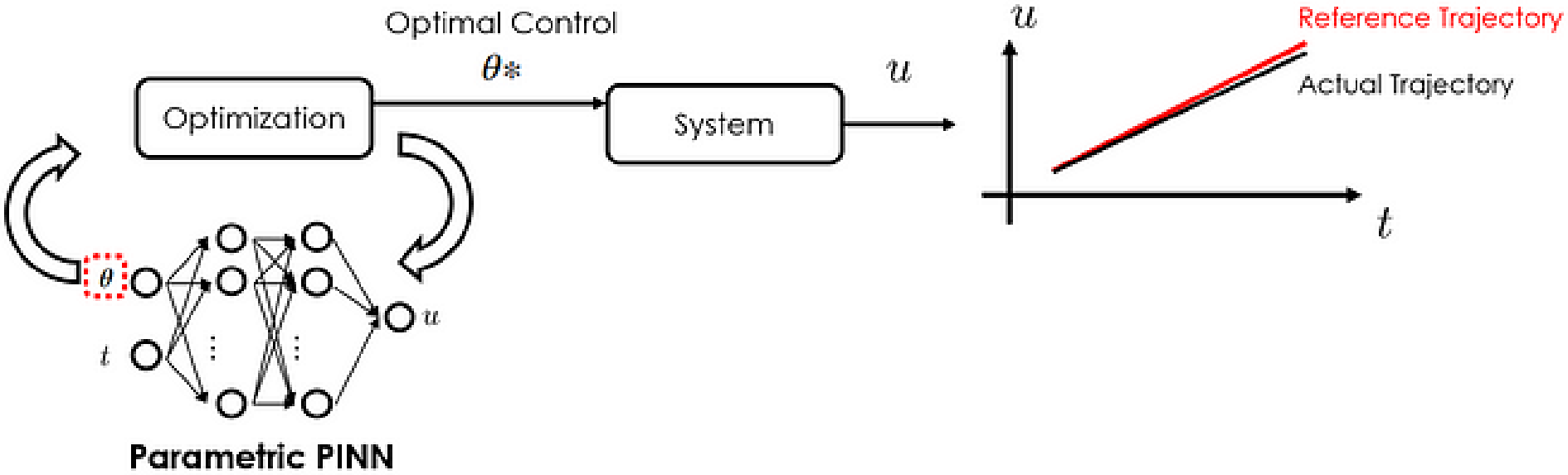


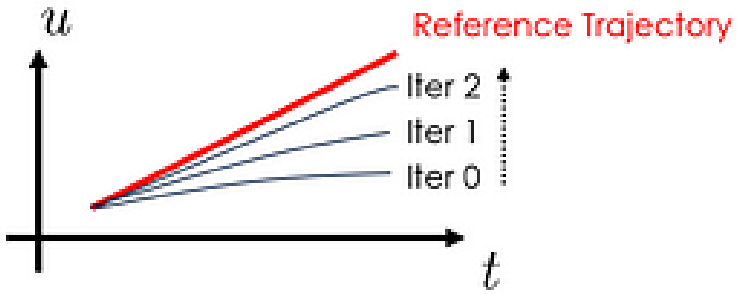
FIG. 4. ***hPINN for the inverse design of holography via the approach of soft constraints.*** *(**A**) Three examples of random initialization of ε. (**B**) The objective $\mathcal{J}$ of different designs obtained from hPINNs with different $\mu_{\mathcal{F}}$. (**C**, **D**, and **E**) The (left top) permittivity ε, (left bottom) training trajectory, (center) hPINN solution $|E|^2$, and (right) the reference $|E|^2$ obtained from FDFD for (C) $\mu_{\mathcal{F}} = 0.1$, (D) $\mu_{\mathcal{F}} = 10$, and (E) $\mu_{\mathcal{F}} = 2$.*

https://arxiv.org/pdf/2102.04626.pdf

Model predictive control



Parametric PINN



Can also be used in an active learning methods – e.g. select where to place sensors

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

Data Enhancement

- Field reconstruction
- Super resolution

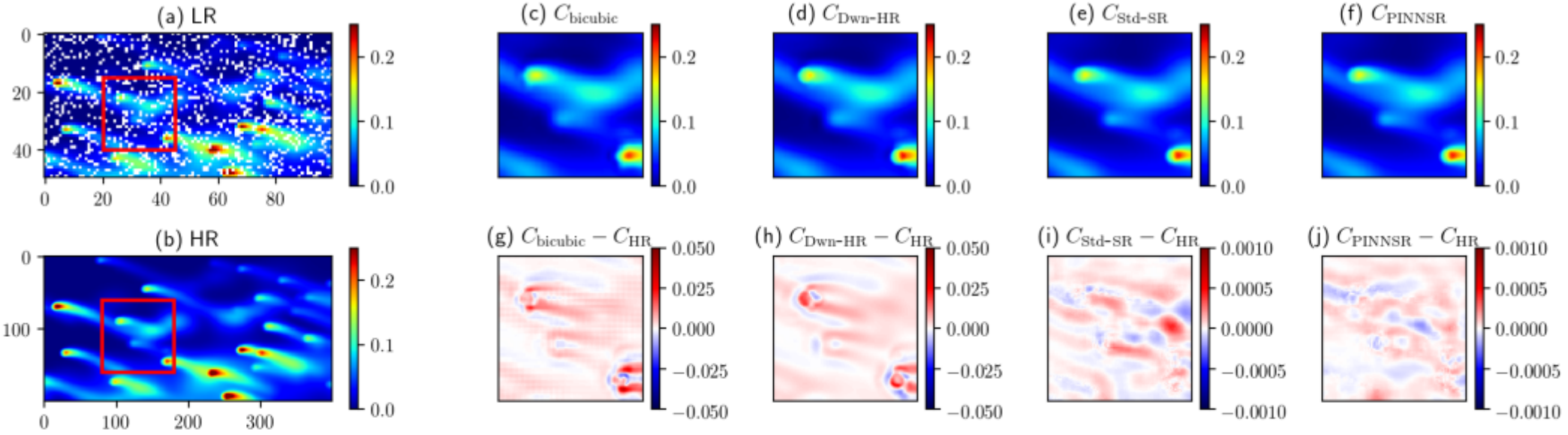


Figure 3: Qualitative comparison between different SR models from the test set when 20% of the pixels are dropped. (a) LR input; (b) ground truth HR output; (c) - (f) SR generated by bicubic, Dwn-HR, Std-SR, and PINNSR; (g) - (j) the corresponding pixel residual calculated from $C_{\text{SR}} - C_{\text{HR}}$, the PINNSR clearly out-performs other models. Additional visualizations can be found in Appendix C.

https://arxiv.org/pdf/2011.02519.pdf

• **Sensing**: PINNs can provide accurate and efficient predictions of the system states. By complementing the field measurements, PINNs can greatly enhance the data quantity and quality for better system observability.

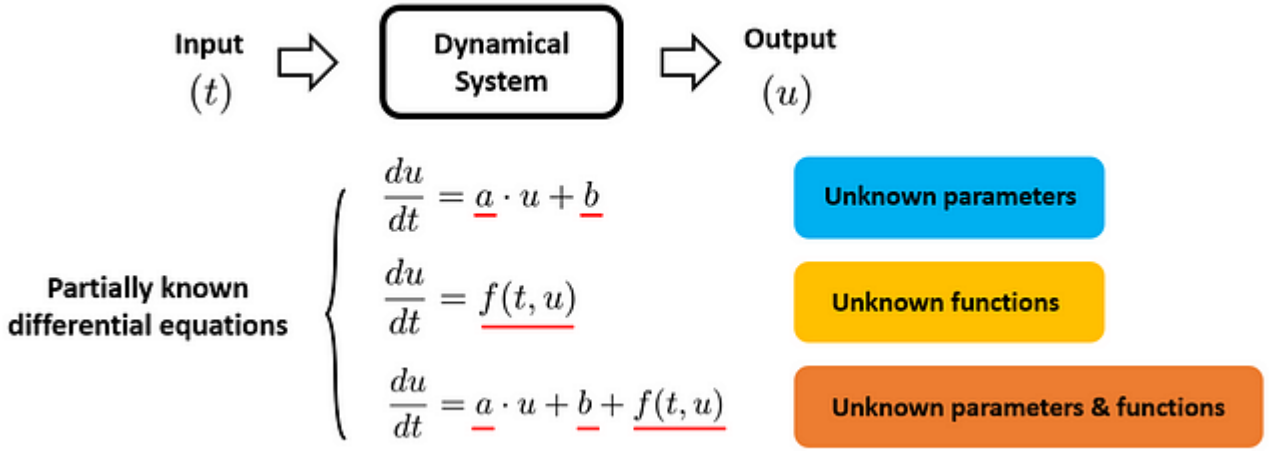
• **Monitoring**: PINNs can infer in real time the values of system health indicators from the sparse observational data, thus facilitating tracking and assessing the system's condition to detect signs of deterioration or failure.

• **Diagnosing**: PINNs can also serve as an anomaly detector to examine if the system displays unusual behaviors that deviate from the norm. Compared to the traditional anomaly detection strategies that only assess the data pattern consistency, PINNs additionally assess the physics consistency of the data, thus improving the robustness and accuracy of the anomaly detection.

•**Forecasting**: By fusing physical principles with observational data, PINNs can accurately model the system degradation process and estimate the system's remaining useful life until failure, thus enabling effective predictive maintenance.

https://towardsdatascience.com/physics-informed-neural-networks-an-application-centric-guide-dc1013526b02

PINN Problems



- The **parameters** of the differential equation are unknown. For example, the governing equations of fluid dynamics are well-established, but the coefficients are highly uncertain.
- The **functional forms** of the differential equations are unknown. For instance, in chemical engineering, the exact functional form of the rate equations may not be fully understood due to the uncertainties in rate-determining steps and reaction pathways.
- Both **functional forms** and **parameters** are unknown. Example is battery state modeling, where the commonly used equivalent circuit model only partially captures the current-voltage relationship (the functional form of the missing physics is therefore unknown). The model itself contains unknown parameters (i.e., resistance and capacitance values).

Discovering Differential Equations with Physics-Informed Neural Networks and Symbolic Regression | by Shuai Guo | Towards Data Science