Simulation AI
Optimization is all you need

Hrvoje Abraham

AI2FUTURE 2024

October 18, 2024

Agenda

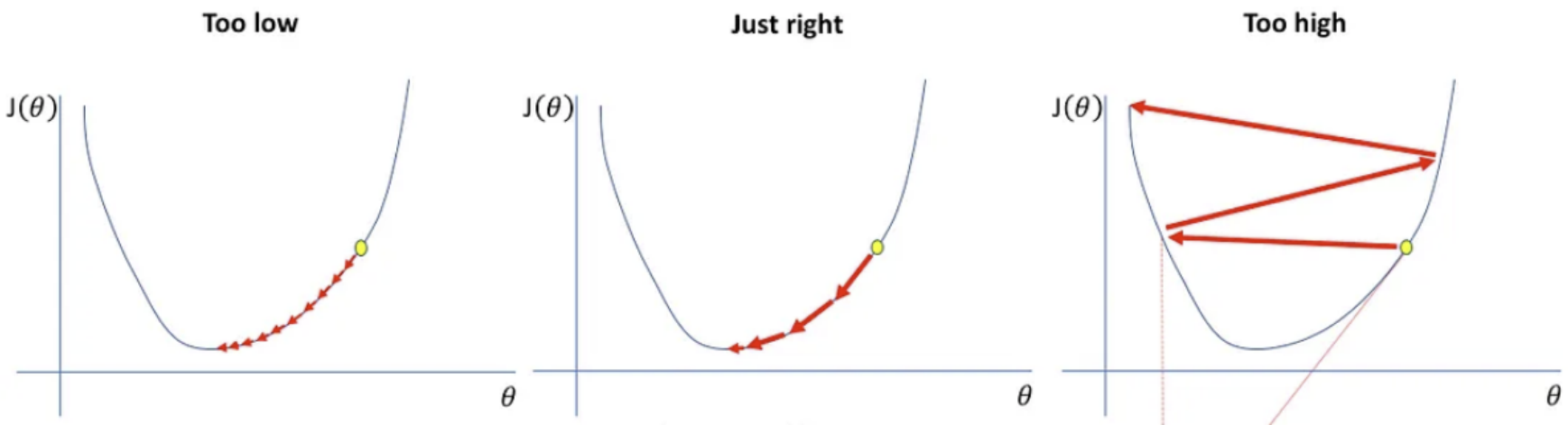
- Optimization
- Physics-informed neural networks
- Discussion

Optimization

Optimization

Given: function $f : A \to \mathbb{R}$ from some set A to the real numbers

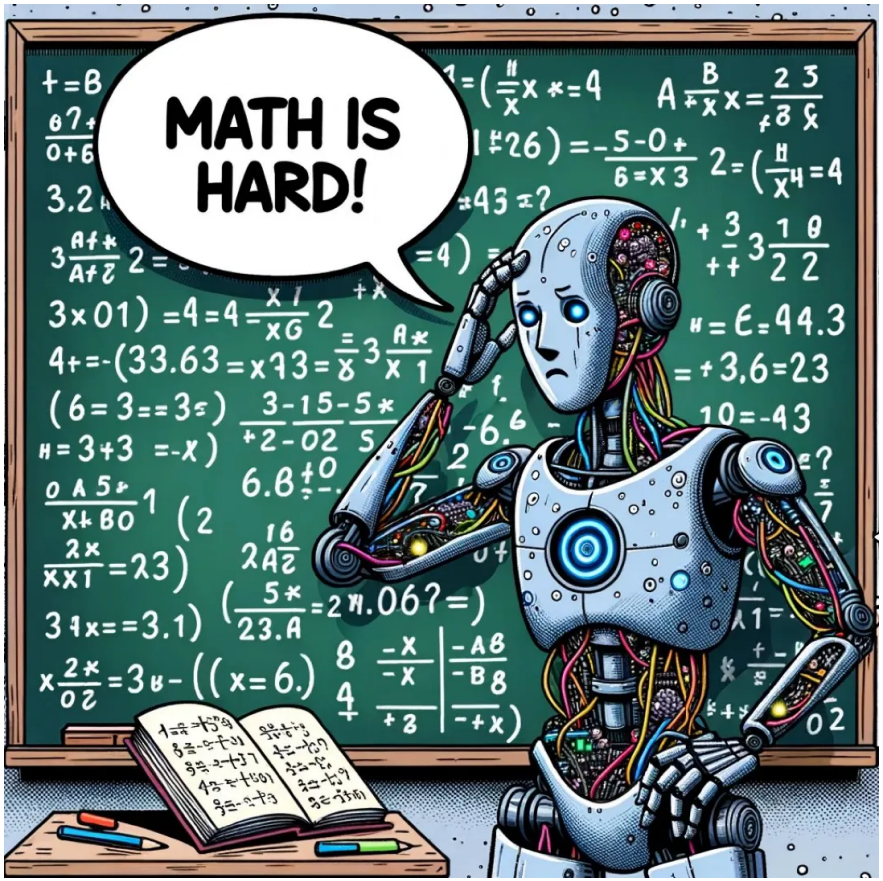
Sought: an element $x_0 \in A$ such that $f(x_0) \leq f(x)$ for all $x \in A$ ("minimization") or such that $f(x_0) \geq f(x)$ for all $x \in A$ ("maximization")



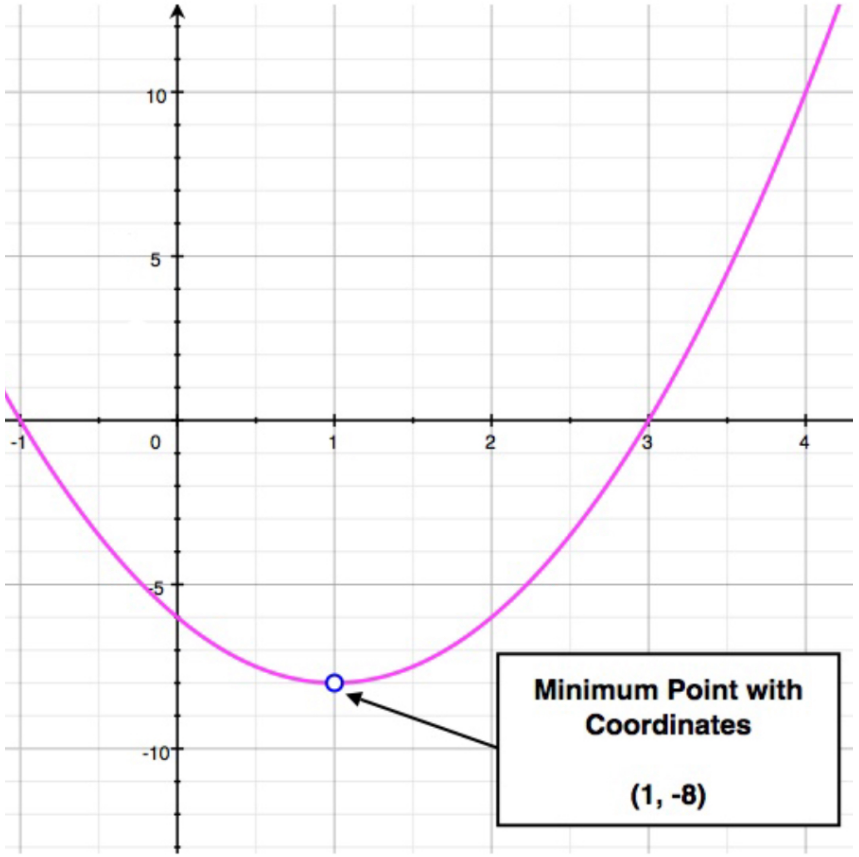
Vast history, body of research, applications, computing ecosystem...

Reduction to optimization

Problem in some specific form



As an optimization problem



Reduction to optimization - Example 1

$$x + y = 1$$

$$3x - 2y = 2$$

$$x, y = ?$$

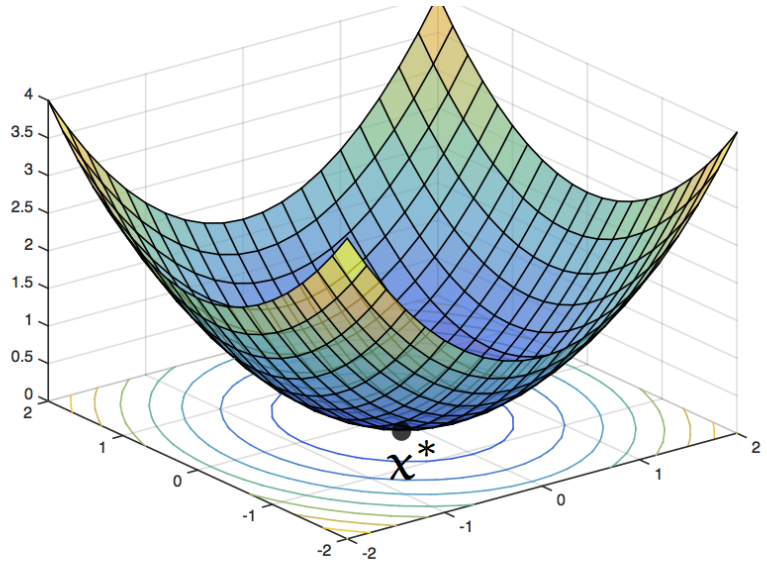
$$(x + y - 1)^2 + (3x - 2y - 2)^2 = 0$$

$$\Downarrow$$

$$f(x, y) = (x + y - 1)^2 + (3x - 2y - 2)^2$$

$$\underset{x,y}{\textit{minimize}}\ f(x, y)$$

A **reduction** from a problem A to a problem B is an algorithm for transforming inputs to problem A into inputs to problem B, such that the transformed problem has the same output as the original problem.



Reduction to optimization - Example 2

Strontium atomic clock system via a "single line" of Linear Programming.

$$
\begin{aligned}
f_r\, n_1 + f_0 &= f_1, & f_1^{min} &\le f_1 \le f_1^{max} \\
f_r\, n_2 + f_0 &= f_2, & f_2^{min} &\le f_2 \le f_2^{max} \\
f_r\, n_3 + f_0 &= f_3, & f_3^{min} &\le f_3 \le f_3^{max} \\
f_r\, n_4 + f_0 &= f_4, & f_4^{min} &\le f_4 \le f_4^{max}
\end{aligned}
$$

$$
\begin{aligned}
f_1^{min} &= f_1^r - 2\,\gamma_1, & f_1^{max} &= f_1^r - \gamma_1/2 \\
f_2^{min} &= f_2^r - \gamma_2, & f_2^{max} &= f_2^r + \gamma_2 \\
f_3^{min} &= f_3^r - 2\,\gamma_3, & f_3^{max} &= f_3^r - \gamma_3/2 \\
f_4^{min} &= f_4^r - \gamma_4, & f_4^{max} &= f_4^r + \gamma_4
\end{aligned}
$$

$$
\begin{aligned}
f_1^r &= 325\,251\,612\,993\,750.8, & \gamma_1 &= 32\,000\,000 \\
f_2^r &= 441\,331\,162\,793\,613.4, & \gamma_2 &= 1\,760\,000 \\
f_3^r &= 434\,828\,994\,151\,297.2, & \gamma_3 &= 7\,600 \\
f_4^r &= 423\,913\,370\,880\,042.4, & \gamma_4 &= 8\,960\,000
\end{aligned}
$$

$$f_r, f_0 \in \mathbb{R},\ f_r^{min} \le f_r \le f_r^{max},\ 0 \le f_0 < f_r$$

$$f_r^{min} = 100\,000\,000,\ f_r^{max} = 300\,000\,000$$

$$n_1, n_2, n_3, n_4 \in \mathbb{N}$$

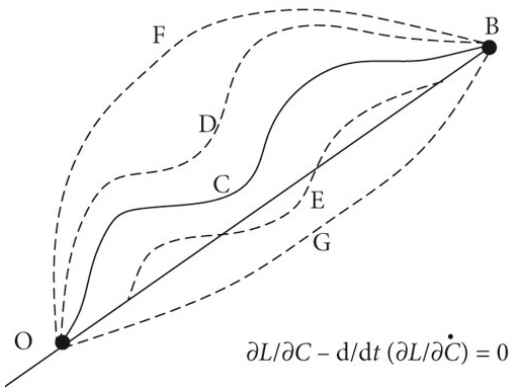
```
LO[fr_] := LinearOptimization[
  fr n1 + f0 - f1 + fr n2 + f0 - f2 + fr n3 + f0 - f3 + fr n4 + f0 - f4,
  {f1m ≤ f1 ≤ f1x,
   f2m ≤ f2 ≤ f2x,
   f3m ≤ f3 ≤ f3x,
   f4m ≤ f4 ≤ f4x,
   0 ≤ f0, f0 < fr,
   fr n1 + f0 - f1 == 0,
   fr n2 + f0 - f2 == 0,
   fr n3 + f0 - f3 == 0,
   fr n4 + f0 - f4 == 0,
   {n1, n2, n3, n4} ∈ Integers},
  {n1, n2, n3, n4, f0, f1, f2, f3, f4}]

For[fr = frm, fr ≤ frm + 1000 000, fr += 1,
 If[Mod[fr, 10 000] == 0, Print["### Status fr=", fr]];
 sol = Quiet@LO[fr];
 {a, b, c, d} =
  {fr n1 + f0 - f1, fr n2 + f0 - f2, fr n3 + f0 - f3, fr n4 + f0 - f4} /. sol;
 If[a == 0 && b == 0 && c == 0 && d == 0,
  Print["fr=", DecimalForm[fr, 20]];
  Print[DecimalForm[sol, 20]];
 ]
]
```

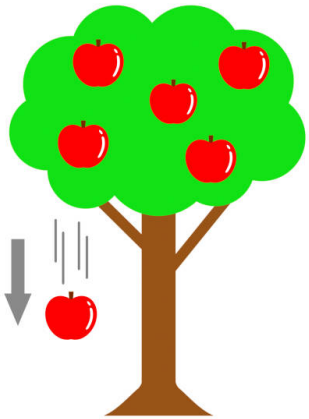
General & powerful idea

Reduction to optimization is one of the most fruitful ideas in history.

Formulation of the entire physics

$$\delta \int_{t_0}^{t_f} \mathcal{L} dt = 0$$

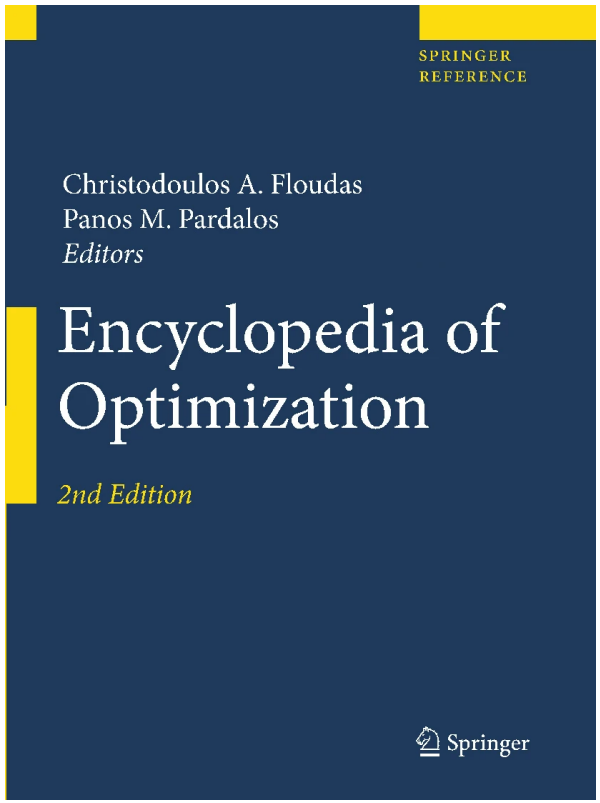




This is not our angle today...

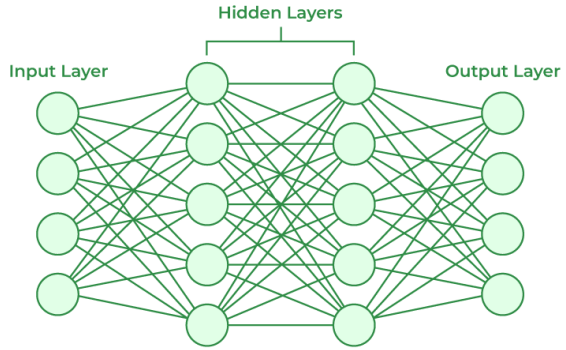
Many real problems reduced to opt.

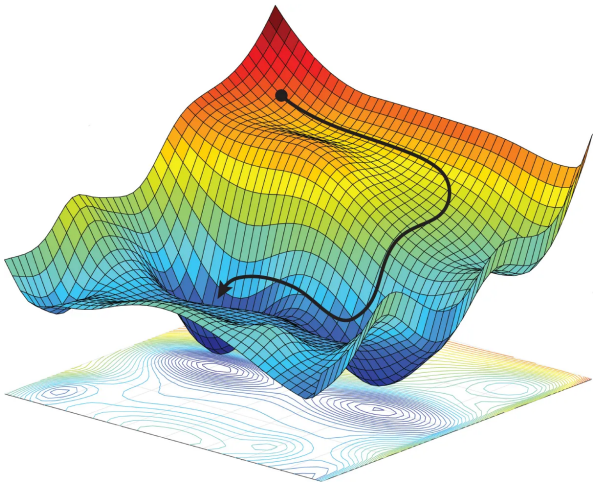
Springer, 2001, 4,800 pages, EUR 2,000

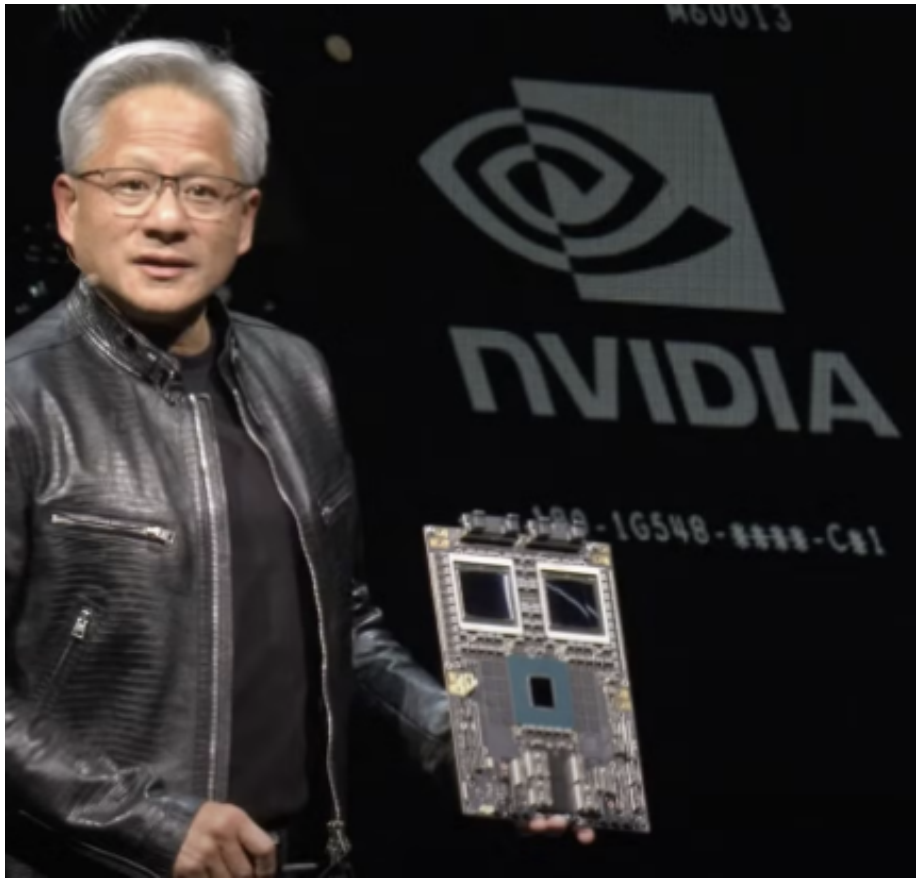


Exascale computing

Training NNs is optimization - this simple fact is our opportunity.







Physics-informed neural networks

The paper

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b], G.E. Karniadakis [a]

arXiv: 28 Nov 2017 Journal: 1 Feb 2019

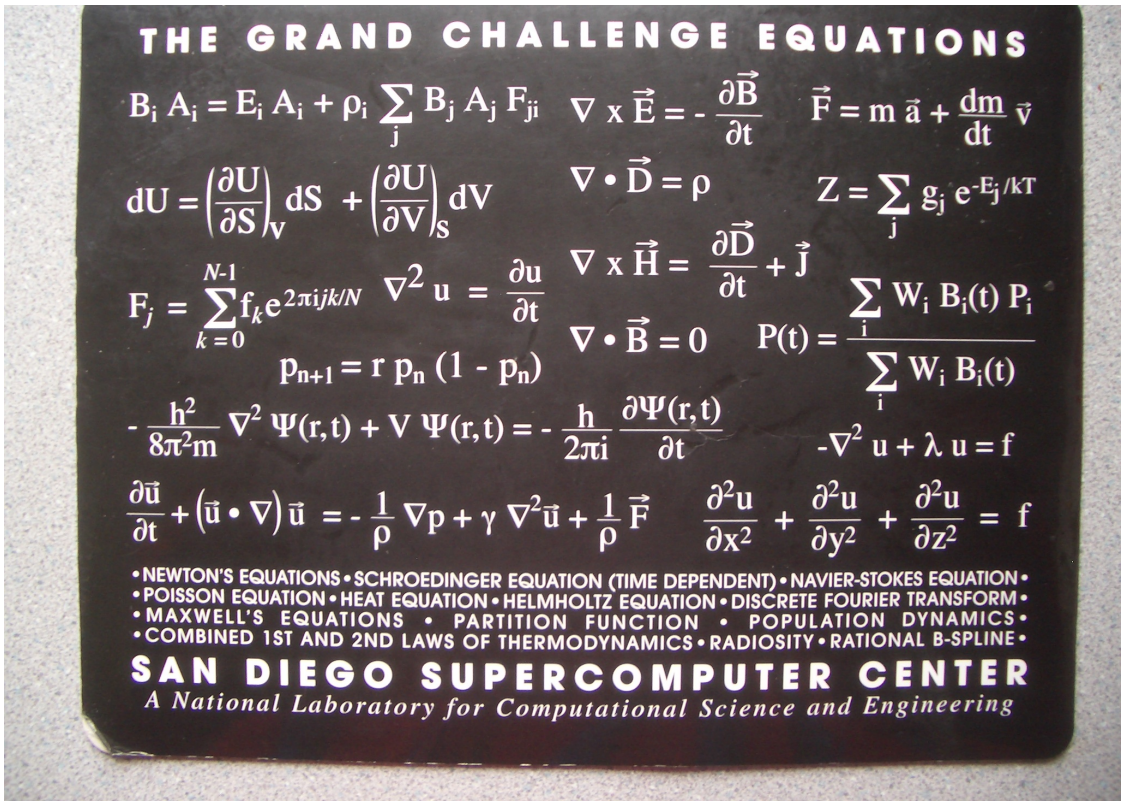
We introduce physics-informed neural networks – neural networks that are trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear partial differential equations.

The problem - Solve the governing equations

Physical laws can be, and usually are, described in terms of differential equations.

$$\mathbf{F} = m \cdot \mathbf{a} \quad \Rightarrow \quad \frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} = \mathbf{f} \quad \Rightarrow \quad \mathbf{x}_{tt} = \mathbf{f}$$

Entire physics can be "coded" that way!



The problem - optimization formulation

$$x_{tt} = f, \quad x(0) = x_0, \quad x_t(0) = v_0$$

$$x_{tt} - f = 0, \quad x(0) - x_0 = 0, \quad x_t(0) - v_0 = 0$$

For n time intervals:

$$\sum_{i=0}^{n} (x_{tt}^i - f^i)^2 + (x^0 - x_0)^2 + (x_t^0 - v_0)^2 = 0$$

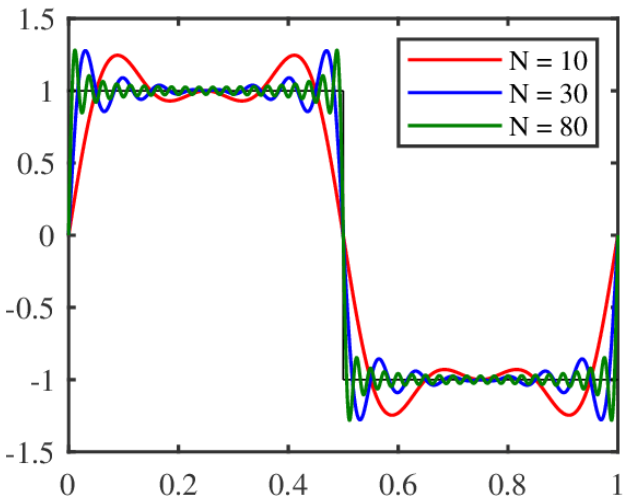
$$f(a_1 \dots a_k) = \sum_{i=0}^{n} (x_{tt}^i - f_i)^2 + (x(0) - x_0)^2 + (x_t(0) - v_0)^2$$

$$\underset{a_1 \dots a_k}{minimize}\, f(a_1 \dots a_k)$$

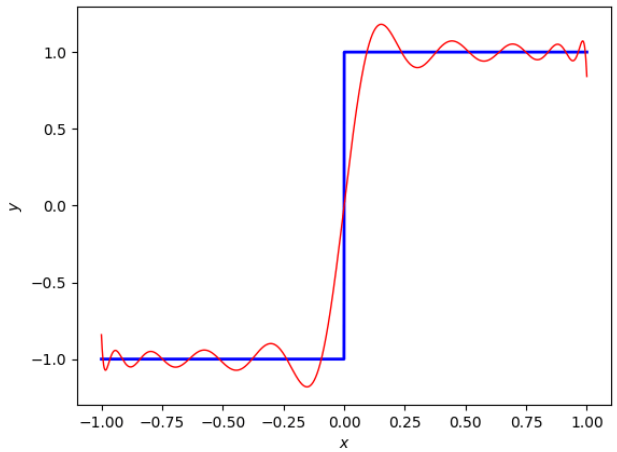
Solution representation - spectral methods

Representing the solution in terms of basis functions:

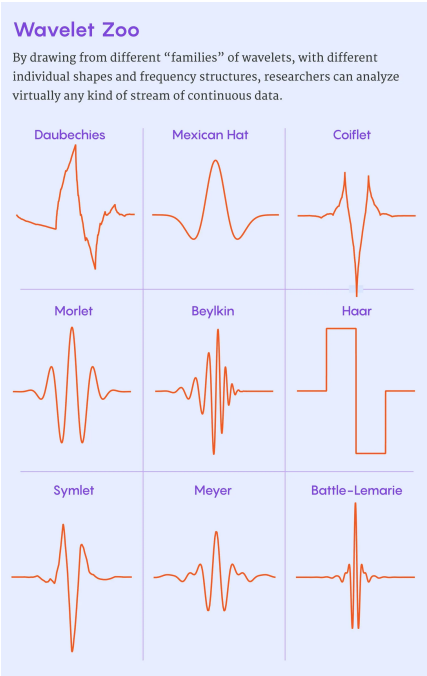
Fourier



Chebyshev



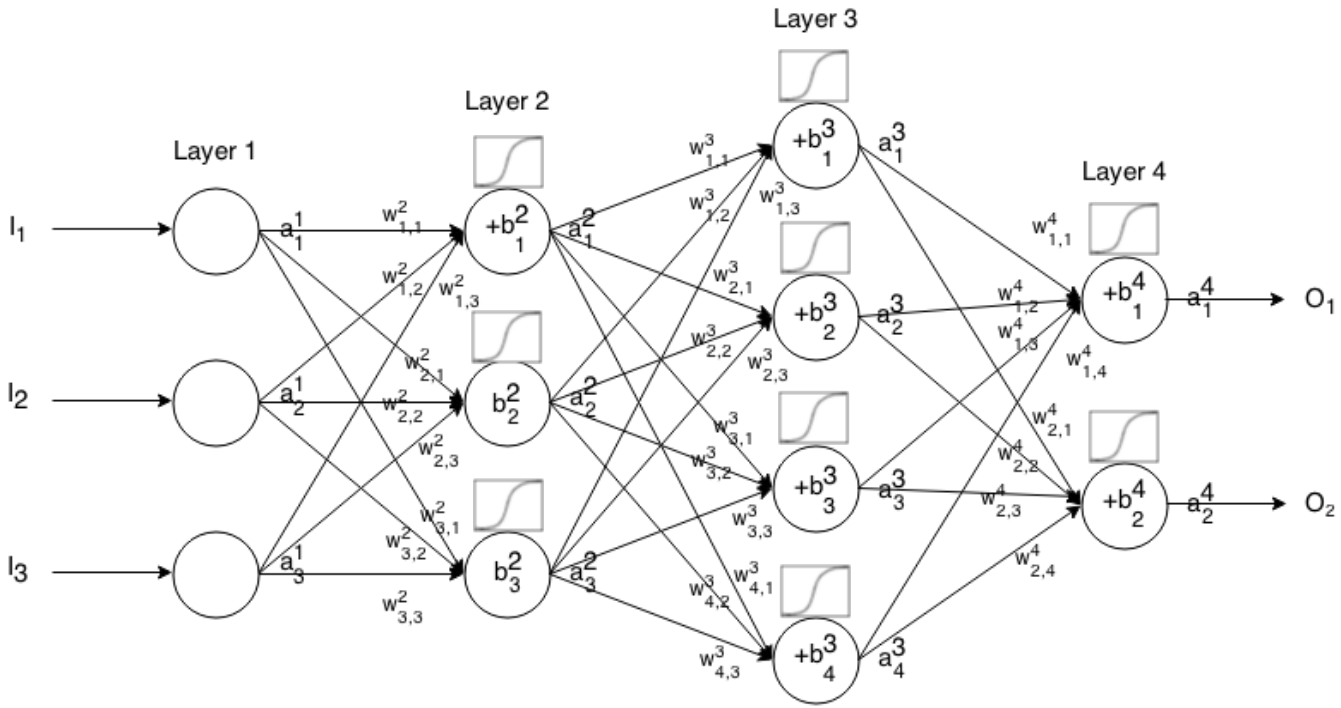
Wavelet



Orthonormal, complete, error bounds, approximation theorems...

Neural networks

Why mentioning spectral methods? Because this is in essence what we will be doing with neural network - represent our solution as composition & superposition of NN activation functions, e.g. $tanh(x)$.

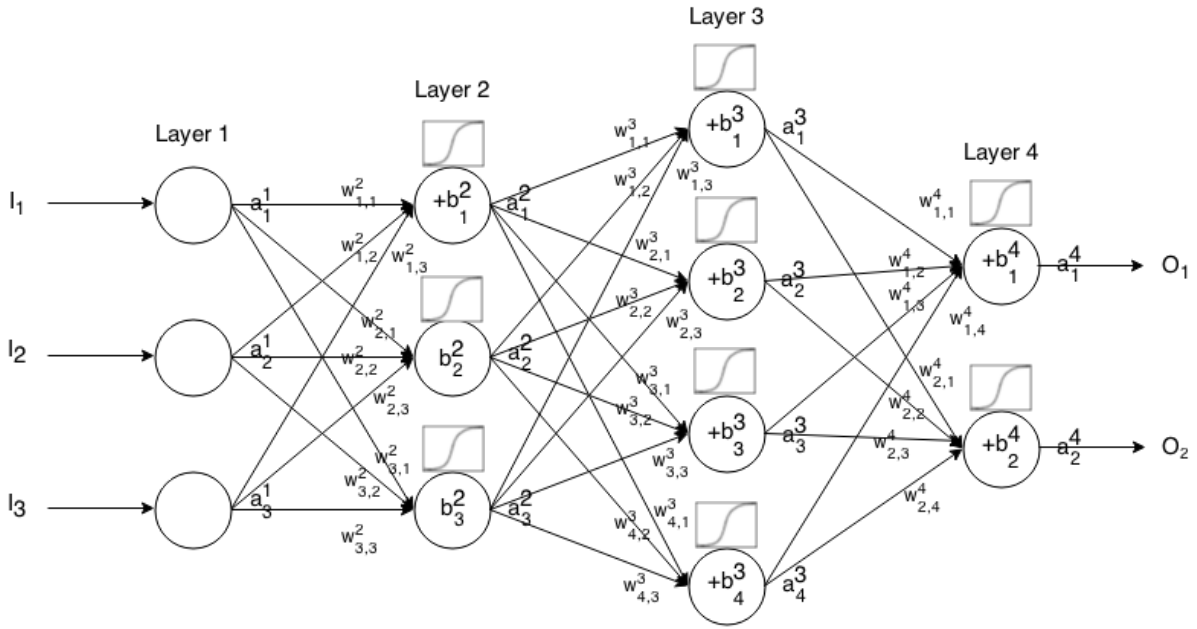


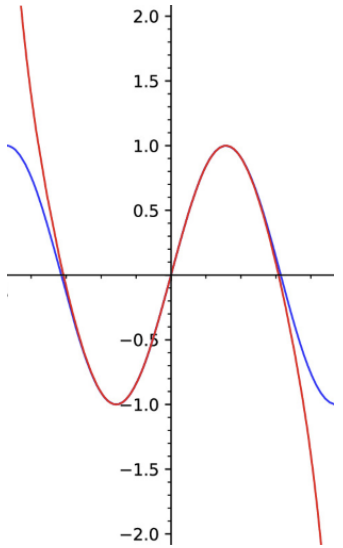
$$\textbf{output} = tanh(w_n \dots tanh(w_2 \cdot tanh(w_1 \cdot \textbf{input} + b_1) + b_2) \cdots + b_n)$$

Universal approximation theorem

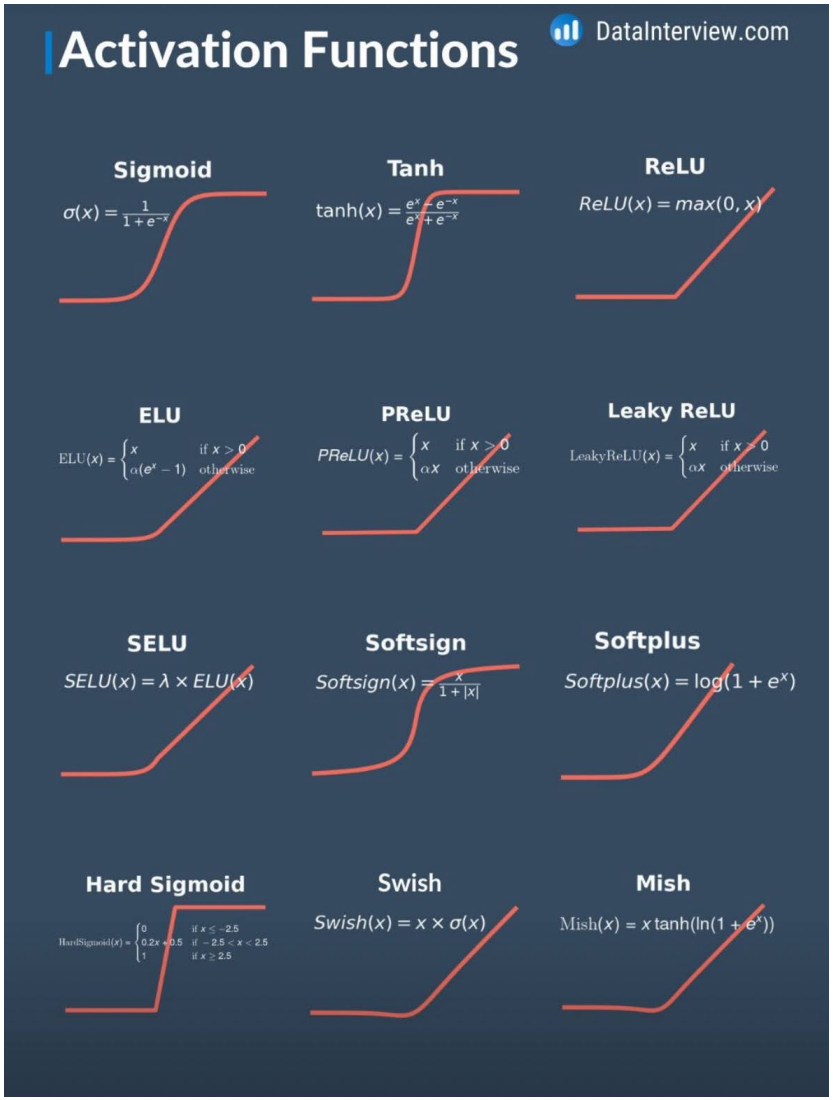
G. Cybenko, 1989, "Approximation by Superpositions of a Sigmoidal Function"

"In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables..."





Activation functions overview



Leaky ReLu - A variant of ReLU that has a small slope for negative values, preventing neurons from "dying".

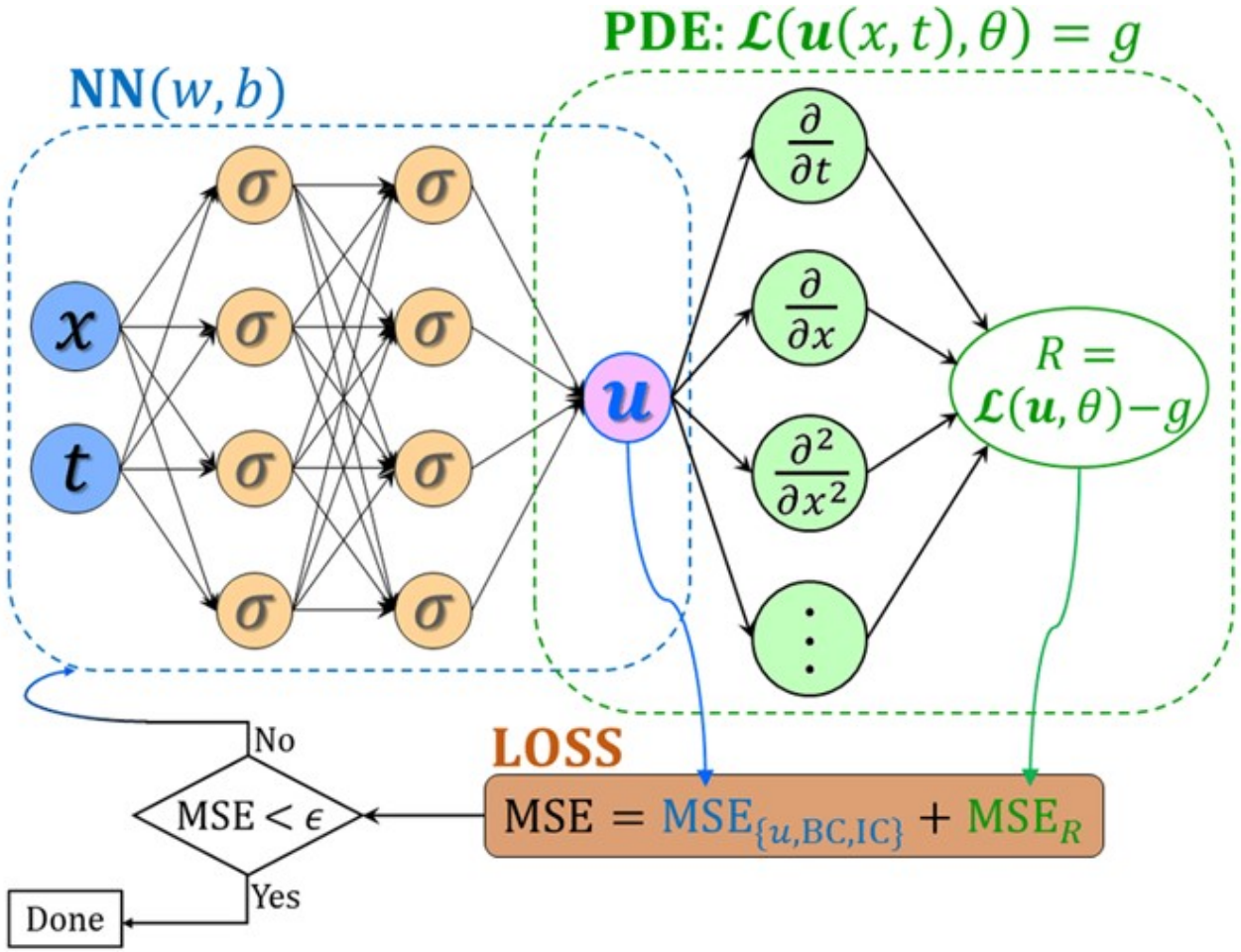
Swish - A self-gated activation function discovered by researchers at Google. It's computationally efficient and has been found to work better than ReLU in some cases.

Mish - A newer activation function that uses a combination of softplus and tanh functions. It has been shown to outperform many traditional activation functions in deep networks.

Physics-informed neural networks

Central idea: *Loss function contains the differential operator error!*

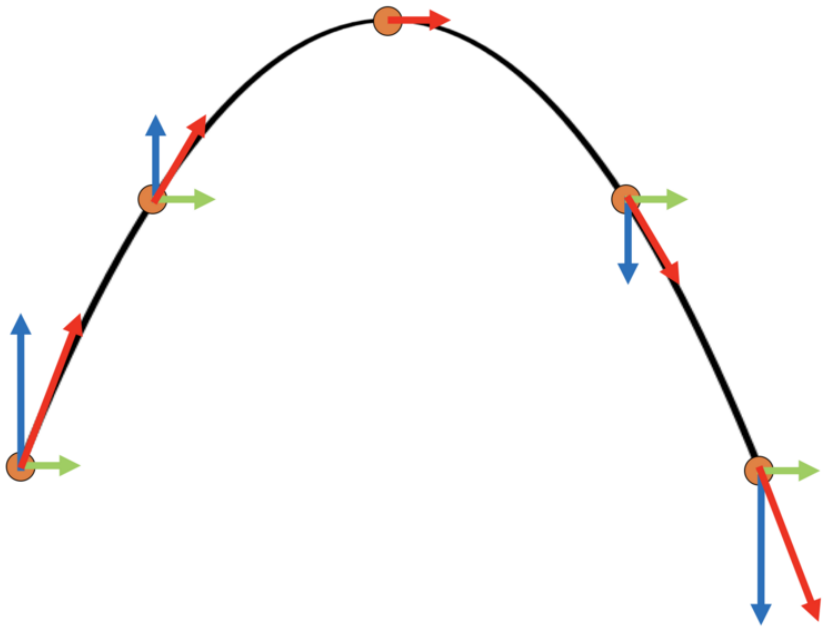
$$\mathbf{x} = [x_0, x_1 \dots x_n]$$
$$\mathbf{t} = [t_0, t_1 \dots t_n]$$



Physics-informed neural networks

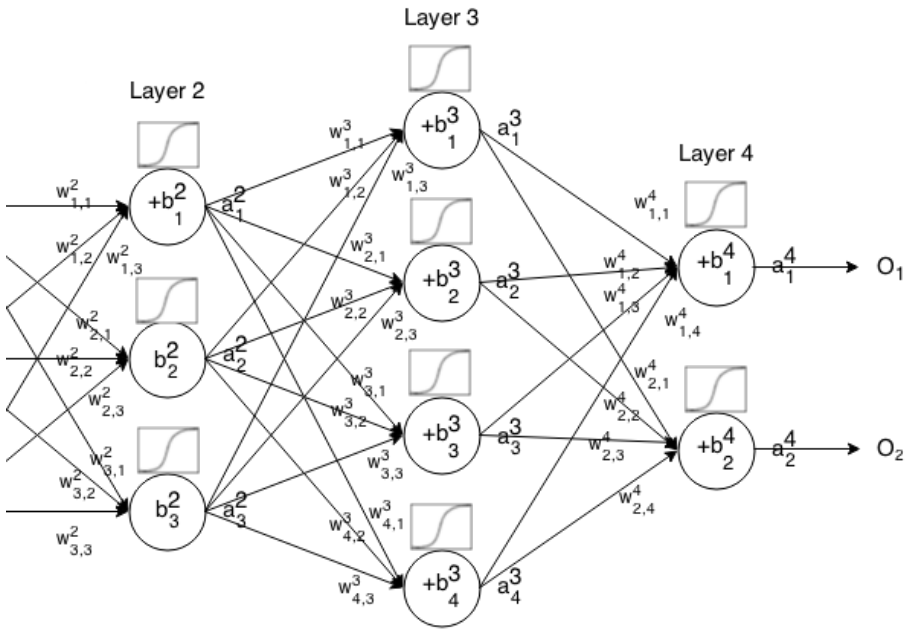
$Loss = Loss_{conditions} + Loss_{equations}$

Can you find the function respecting the boundary conditions with the curvature specified by the laws of physics?



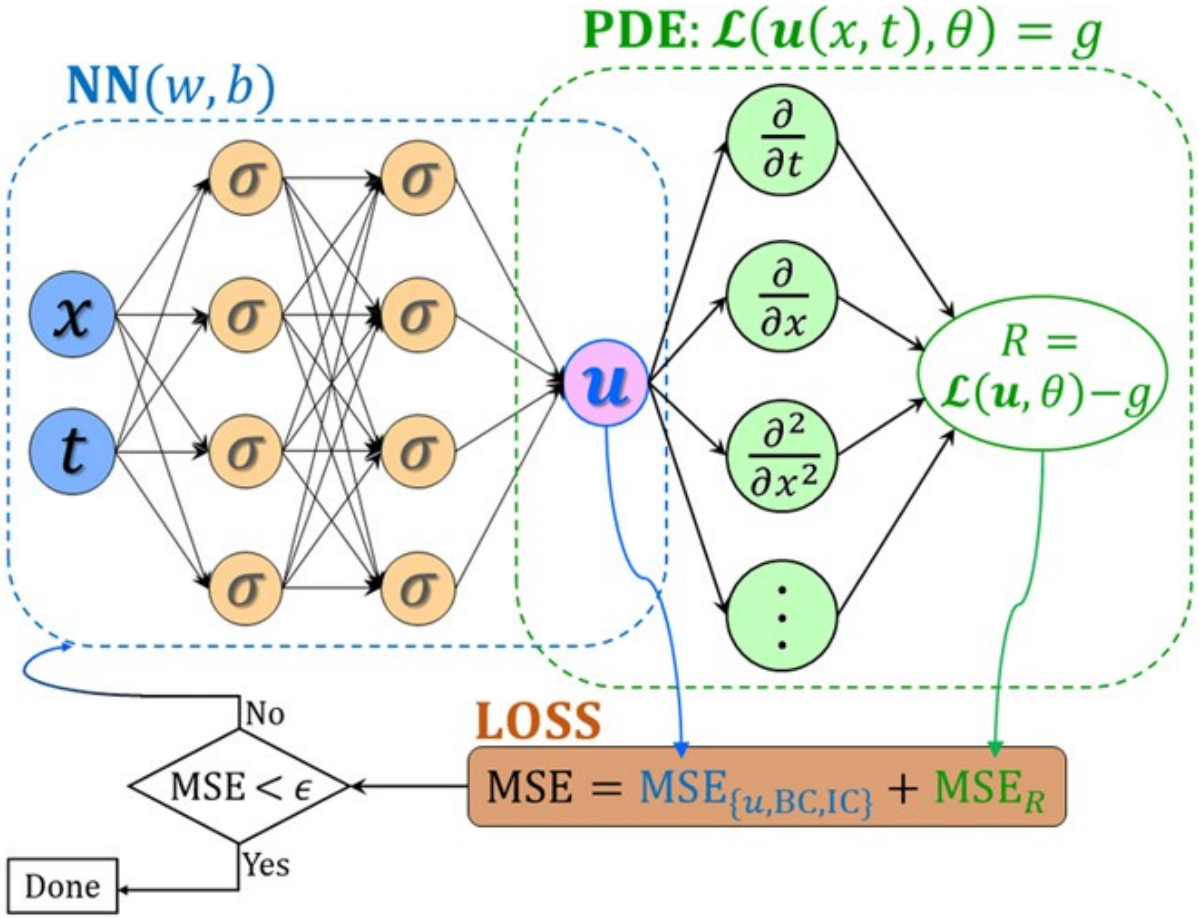
NN training

OK, I will try by finding the optimal combination of my activation functions!



Derivative

Derivative calculation is readily available in modern NN frameworks and can be directly used in the definition of the loss function - automatic differentiation!



TensorFlow

```
tf.gradients
```

PyTorch

```
requires_grad
autograd.grad
```

Google JAX

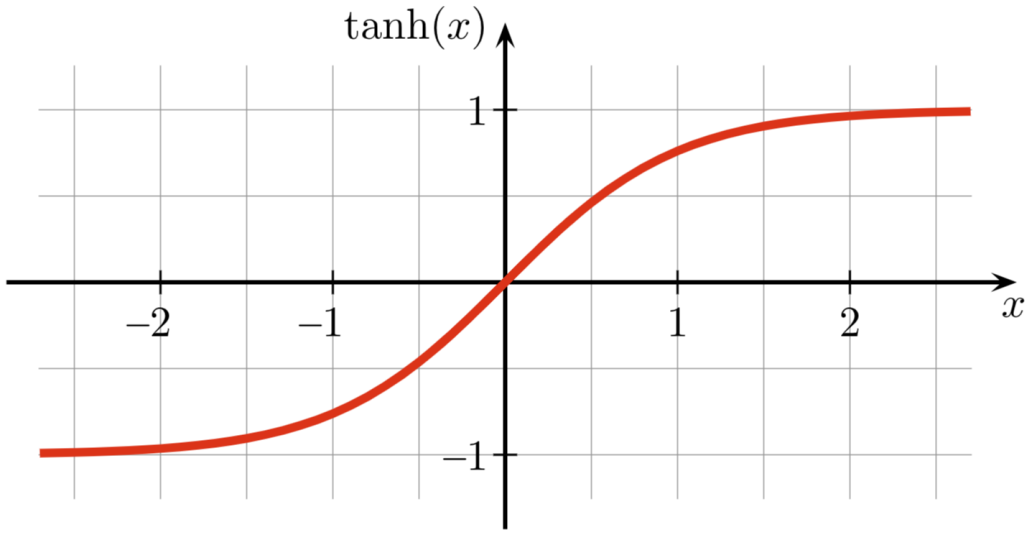
```
jax.grad
```

Activation function constraints

Universal approx. prop. - activation function "must" be nonpolynomial.

Multiple-derivable output - activation function has to be multi-derivable.

$\tanh x$



Representing the equations

Burger's equation from the original paper:

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

Correspondingly, the *physics informed neural network* $f(t, x)$ takes the form

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

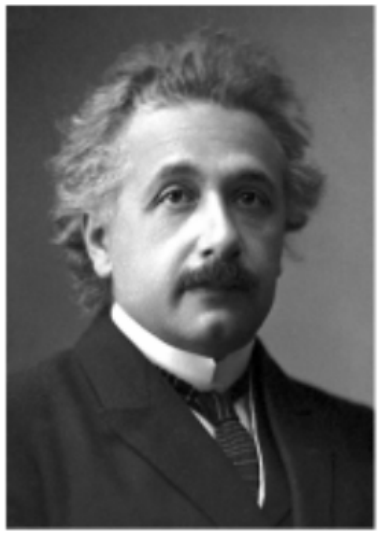
Existence & uniqueness of the solution

Extremely complex topic which we will reduce to "a dogma of physics":

Solution exists and it's unique. We did our best to make it so!

Isaac Newton

James Clerk Maxwell



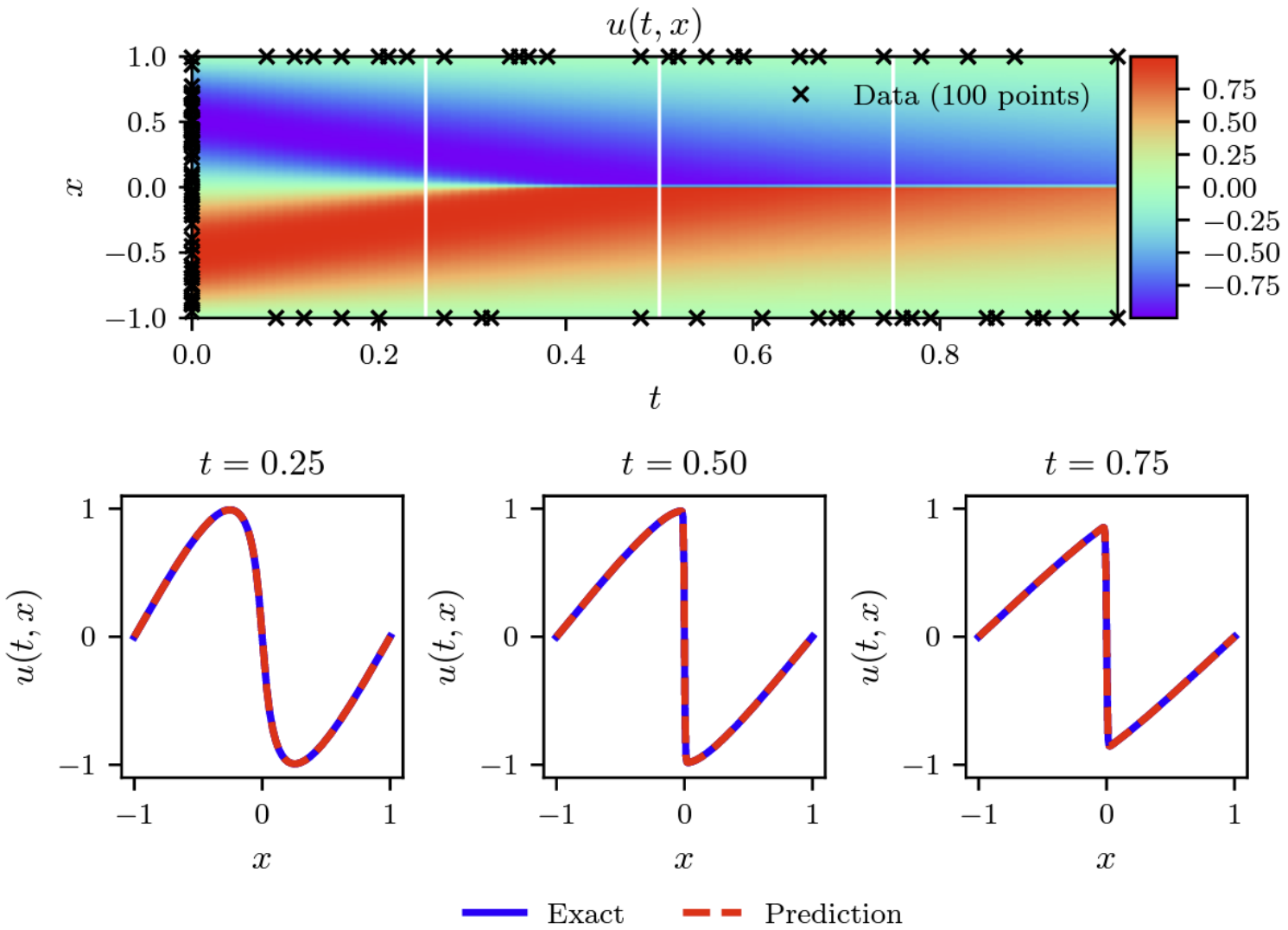
Albert Einstein

Paul A. M. Dirac

Clay Mathematics Institute $1 million Millennium Prize Problem:

Existence of solution of Naiver-Stokes equations.

The solution - Burgers' equation

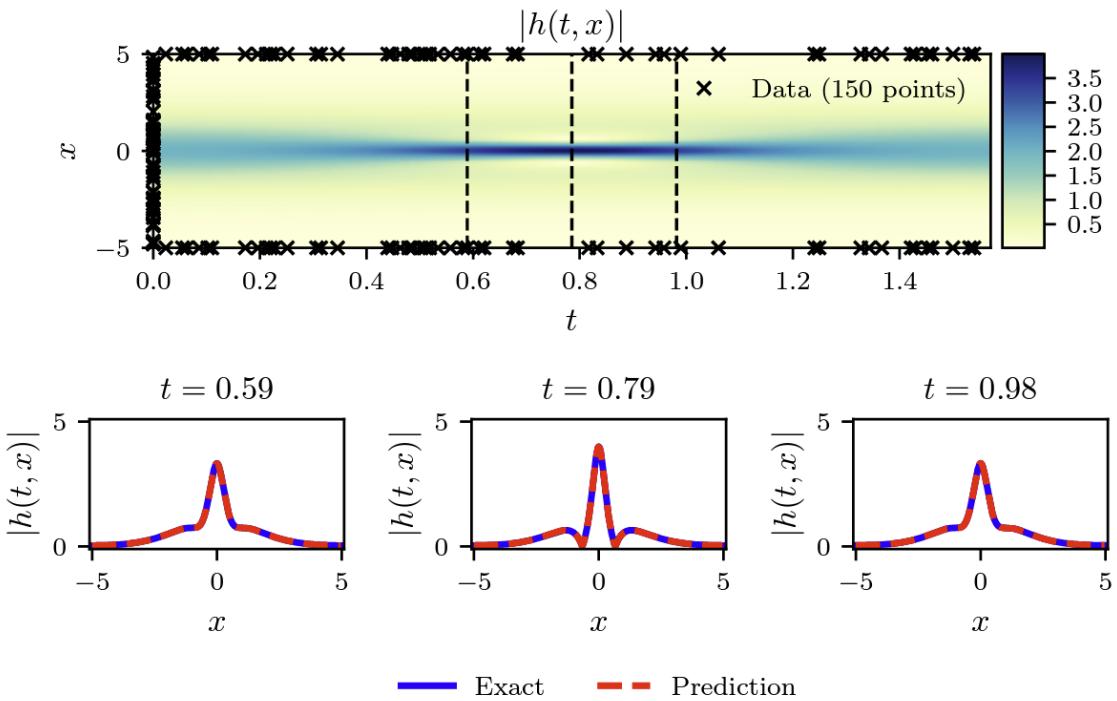


The solution - Schrödinger's equation

$$
\begin{aligned}
&ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2], \qquad (5)\\
&h(0, x) = 2\ \mathrm{sech}(x),\\
&h(t, -5) = h(t, 5),\\
&h_x(t, -5) = h_x(t, 5),
\end{aligned}
$$

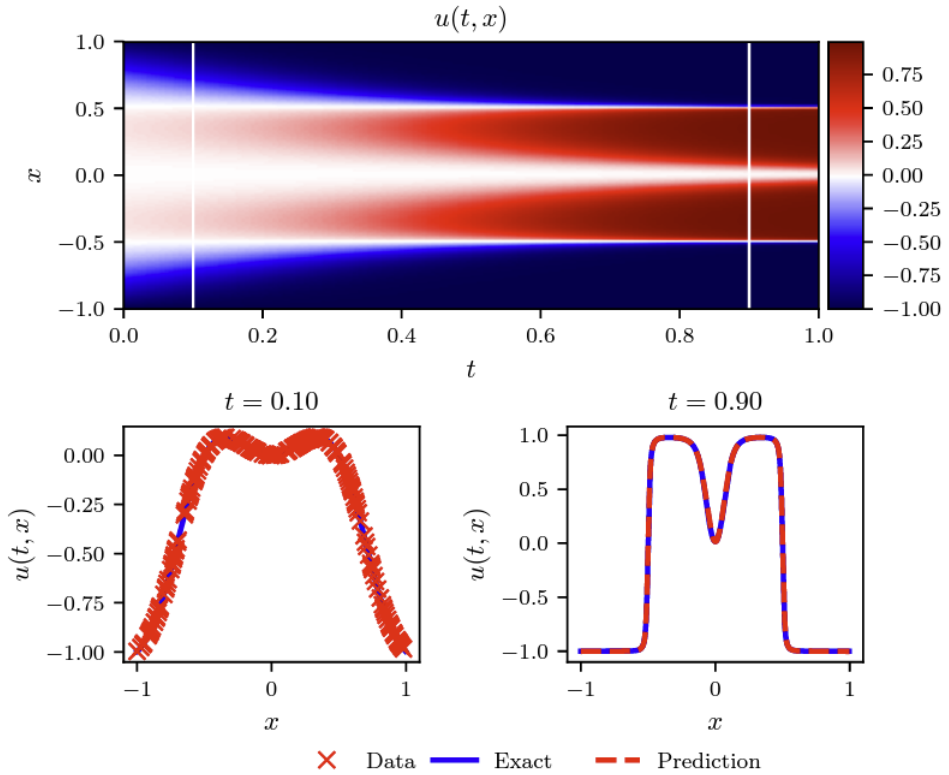
where $h(t, x)$ is the complex-valued solution. Let us define $f(t, x)$ to be given by

$$f := ih_t + 0.5h_{xx} + |h|^2 h,$$

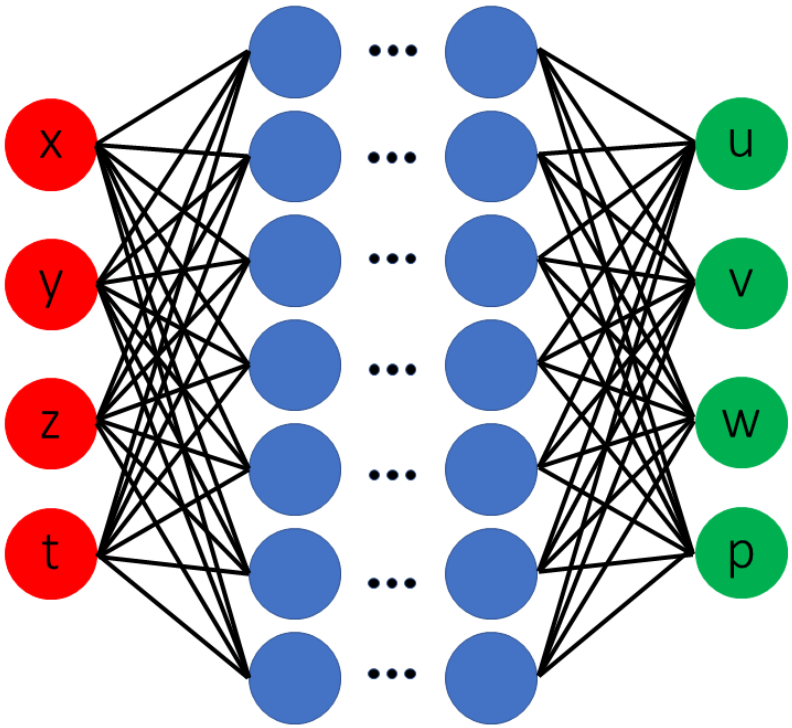


The solution - Allen-Cahn equation

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = x^2 \cos(\pi x),$$
$$u(t, -1) = u(t, 1),$$
$$u_x(t, -1) = u_x(t, 1).$$



PINNs architecture



Burgers' equation: 4 hidden layers

$$2 \times 20 \times 20 \times 20 \times 20 \times 2$$

Schrödinger's equation: 4 hidden layers

$$2 \times 100 \times 100 \times 100 \times 100 \times 2$$

Allen-Cahn equation: 4 hidden layers

$$1 \times 200 \times 200 \times 200 \times 200 \times 101$$

Navier-Stokes: 8 hidden layers

$$3 \times 20^8 \times 2$$

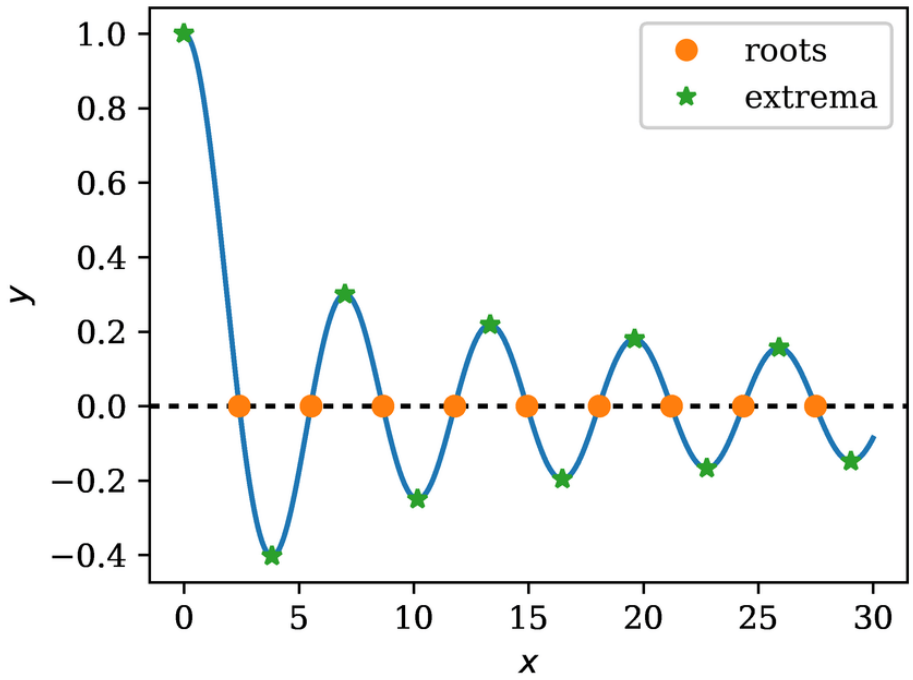
Discussion

What did we get

Surrogate model - a single instance of solution for fixed conditions (boundary & initial conditions, geometry, and material properties).

New training for every new geometry and boundary & initial conditions.

Solution approx. ready for differentiation, integration, root-finding, optimization, feature & spectral analysis - in the spirit of spectral methods.



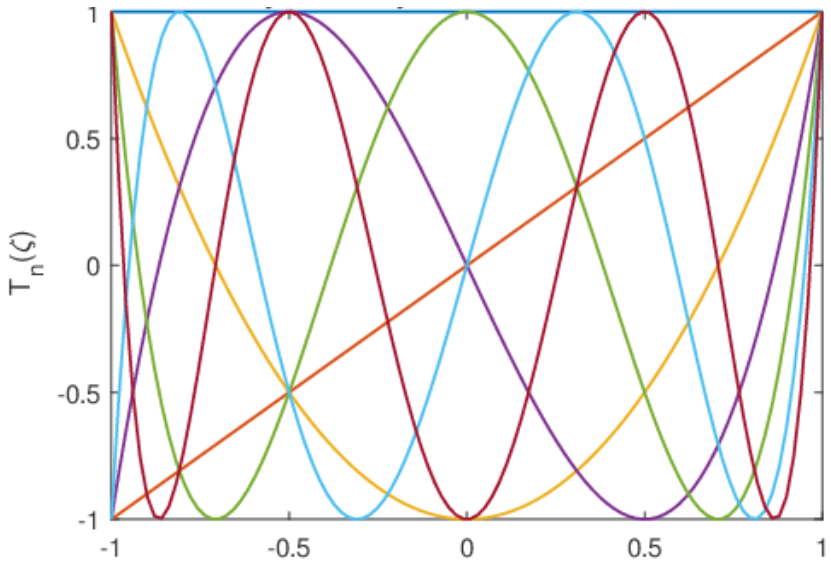
chebfun Matlab package - inspired by?

chebfun

Approximations, root finding, quadratures, optimizations...

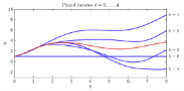
Spectral method package - functions represented as superposition of Chebyshev polynomials.

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x),\ T_k(\cos x) = \cos kx$$

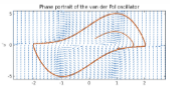


Two electrons orbiting symmetrically about a nucleus
Jeremy Fleury and Nick Trefethen, June 2016

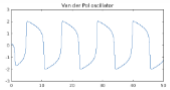
Picard iteration for ODE existence proof
Nick Trefethen, January 2016



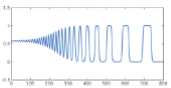
Phase portraits with chebop/quiver
Asgeir Birkisson, November 2015



Chebfun's new IVP capabilities
Asgeir Birkisson, February 2015



A nonlinear system of Guckenheimer and Holmes
Nick Trefethen, February 2015



Lyapunov exponent of the Lorenz system
Hrothgar, January 2015

Pythagorean planets
Behnam Hashemi and Nick Trefethen, December 2014

Data-driven PINNs

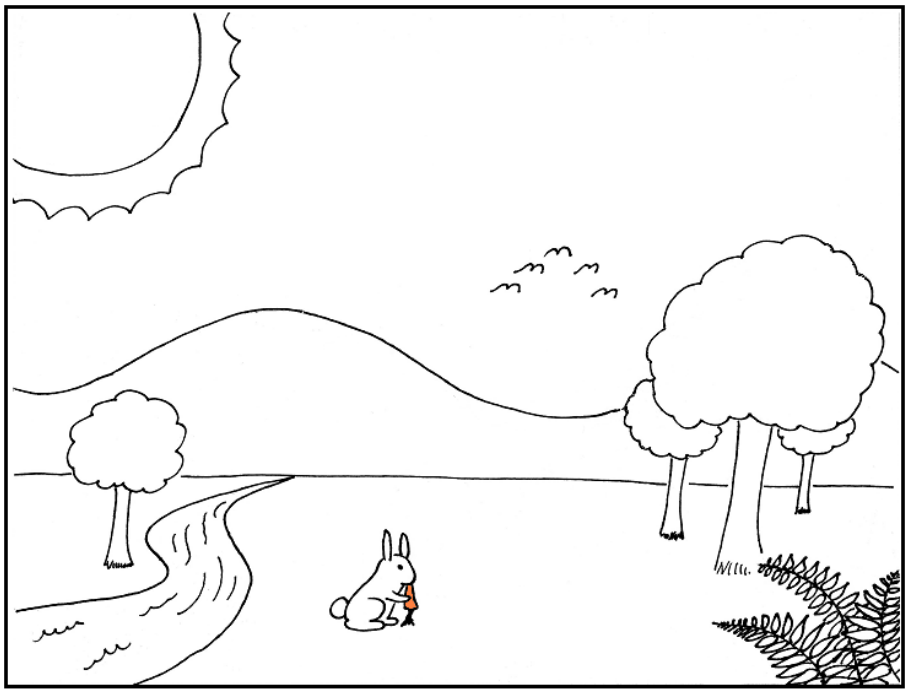
NN contains additional unknowns on geometry or material properties and solves for those during the optimization (training).

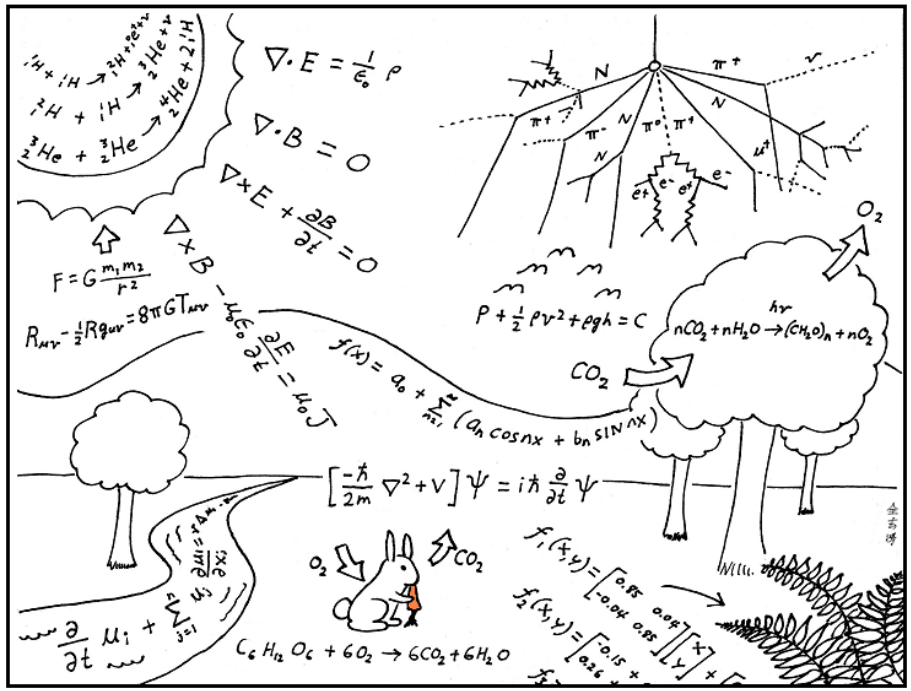
E.g. deduct the property of the liquid from video - ask the NN to find the most probable material properties considering the physics (Navier-Stokes equations):

Is the man using water, alcohol, or gasoline?!

Data-driven PINNs

How PINNs see the world:





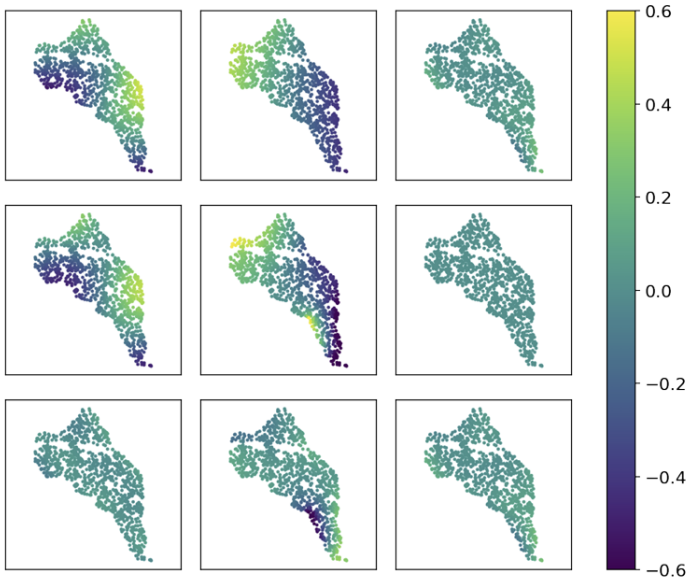
Physics-informed deduction of material & process parameters in the environment.

Data-driven PINNs

Physics-informed computer sensing - vision, sound, wind, vibration, thermal, radiation - deducting the model consistent with all the laws of physics.

Real-life application

"Highly-scalable, physics-informed GANs for learning solutions of stochastic PDEs", 2019, `https://arxiv.org/pdf/1910.13444`

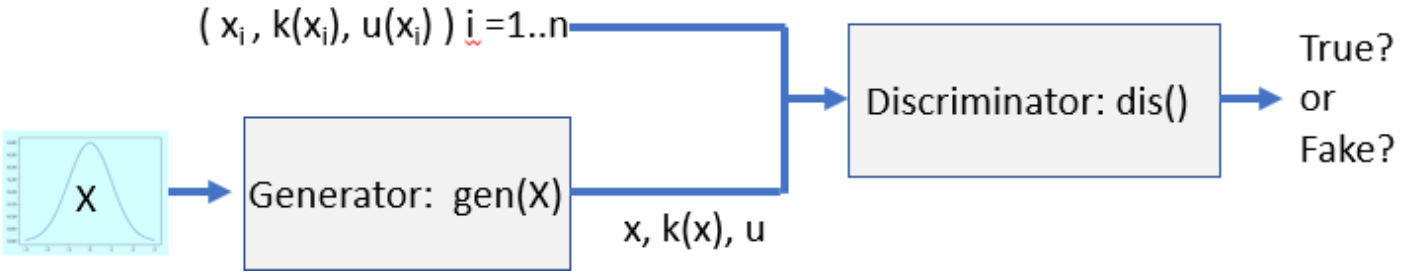


"... modeling subsurface flow at the Hanford Site by combining stochastic computational models with observational data using physics-informed GAN models. The geographic extent, spatial heterogeneity, and multiple correlation length scales of the Hanford Site require training a computationally intensive GAN model to thousands of dimensions."

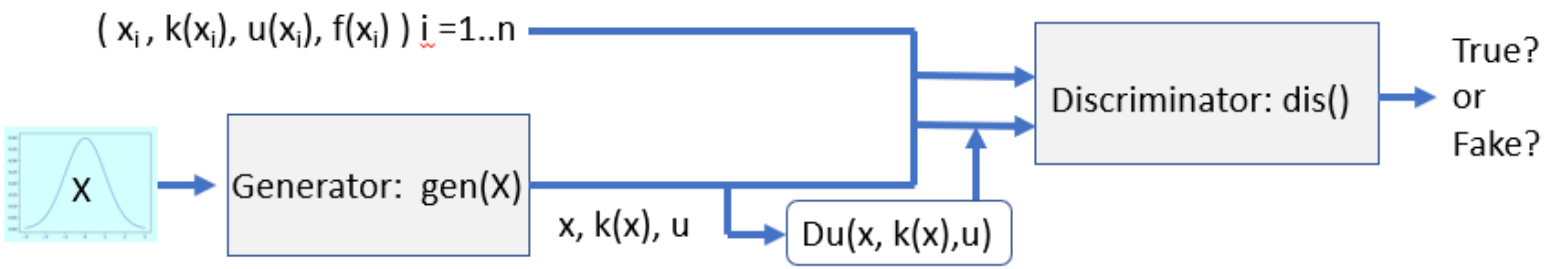
Discussion on using PINNs as the best available computational approach considering the requirements - number of unknowns, multiple correlation scales; only partially known physics resolved by "big data".

Physics-informed GANs

Basic GAN configuration:



Full GAN that also satisfies the differential equation:

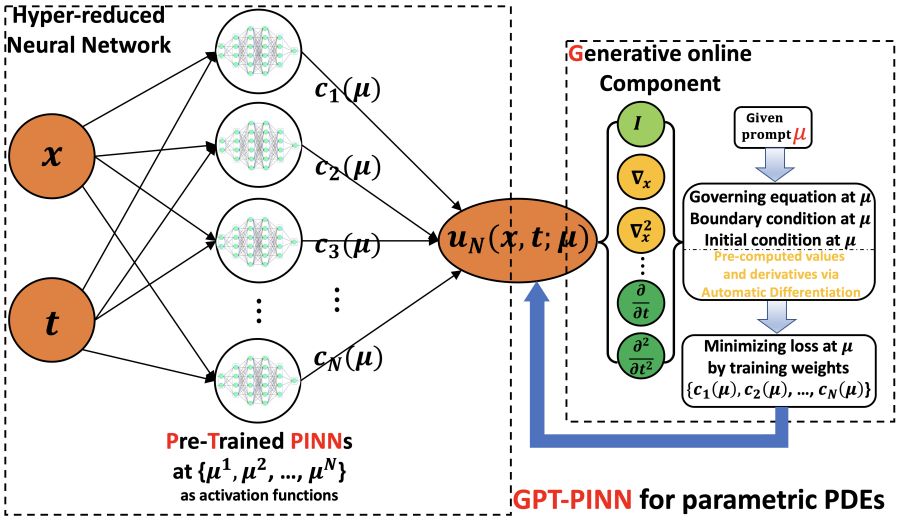


https://cloud4scieng.org/2020/06/10/notes-on-deep-learning-and-differential-equations/

Generative model generating PINNs

"GPT-PINN: Generative Pre-Trained Physics-Informed Neural Networks toward non-intrusive Meta-learning of parametric PDEs", Jan 2024, https://www.sciencedirect.com/science/article/pii/S0168874X23001403

"TGPT-PINN: Nonlinear model reduction with transformed GPT-PINNs", Mar 2024, https://arxiv.org/abs/2403.03459

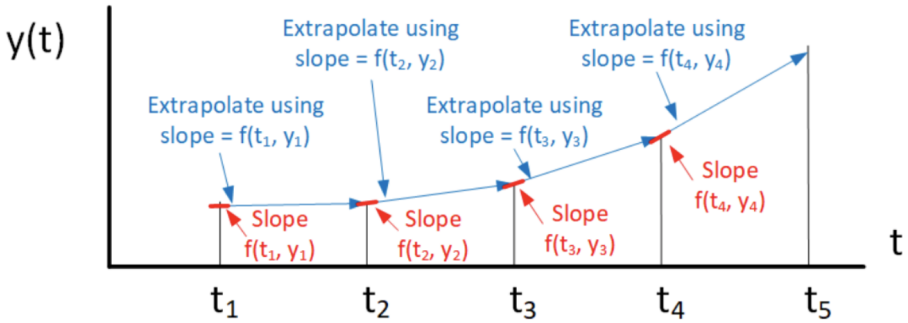


"Its infrastructure is a network of networks. The inner networks are the full PINNs. ... learns and grows hidden layer one neuron/network at a time. ... In the end is capable of generating surrogate solutions across the entire parameter domain accurately and efficiently, with a cost independent of the size of the full PINN."

Performance mindset

Classical approach: finite differences

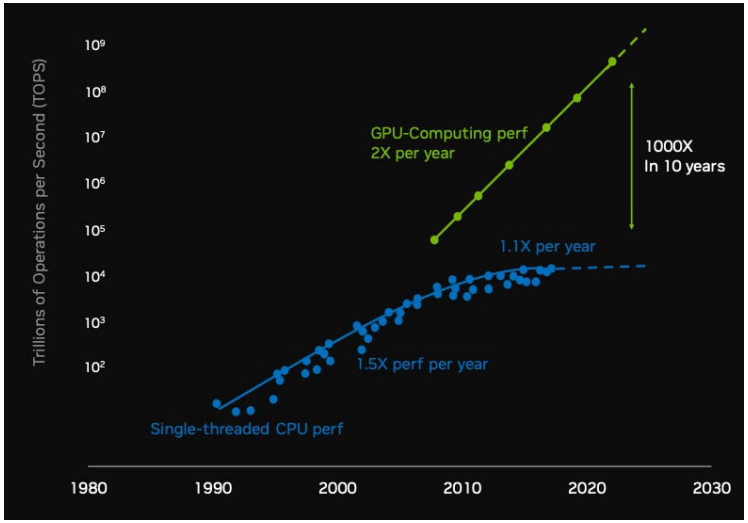
$$\frac{\partial u}{\partial x} = \frac{u_{n+1,j} - u_{n-1,j}}{2\Delta x} + \mathrm{O}(\Delta x^2)$$



Good: very efficient in number of ops

Bad: specialized schemes, not easy to really parallelize, decoupled pipeline

PINNs



Good: paralelized (NN training), universal, directly plugged into NN

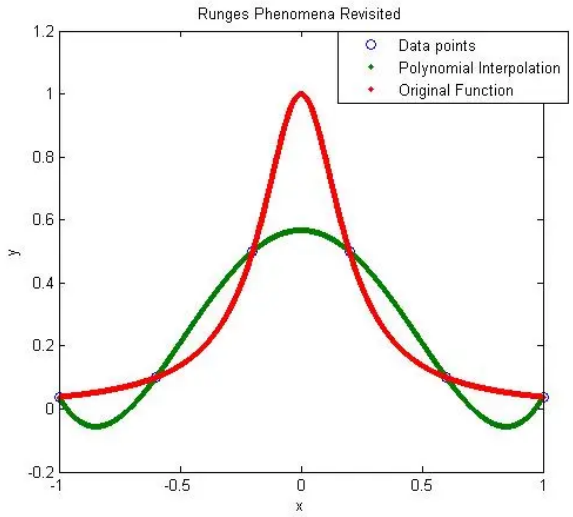
Bad: number of parameters, still dubious whether scaled with good prospects

Could be user will gain time saving with method running much more operations - new computational mindset considering exp. scaling of hardware capabilities.

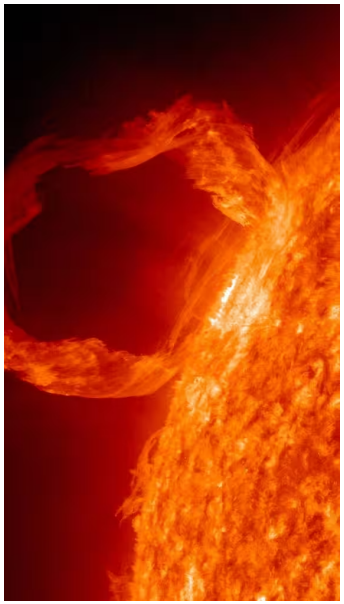
Challenges

Chaotic systems

Narrow resonances



Non-linear dynamics



Very active area of research...

The Theory of Functional Connections

"The Theory of Functional Connections", 2021, https://arxiv.org/abs/2105.08034

PINN loss function is sum of both governing equations and conditions loss:

$$f(a_1 \dots a_k) = \sum_{i=0}^{n} (x_{tt}^i - f_i)^2 + \alpha(x(0) - x_0)^2 + \beta(x_t(0) - v_0)^2$$

Could slow the optimization, e.g. ill-conditioned narrow canyon surface.

TFC packs initial & boundary conditions into the equations, so we are left with a reduced number of terms and more efficient optimization:

$$f^{TFC}(a_1 \dots a_k) = \sum_{i=0}^{n} (x_{tt}^i - f_i^{TFC})^2$$

Kolmogorov-Arnold Networks 1956-2024

Learnable activation functions (e.g. B-splines), not edge weights:

Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) MLP(x); $\mathbf{W}_3$, σ_2, $\mathbf{W}_2$, σ_1, $\mathbf{W}_1$; *nonlinear, fixed*; *linear, learnable*; x	(d) KAN(x); $\mathbf{\Phi}_3$, $\mathbf{\Phi}_2$, $\mathbf{\Phi}_1$; *nonlinear, learnable*; x

KAN: Kolmogorov-Arnold Networks, Apr 2024, `https://arxiv.org/abs/2404.19756`

For some other opportunity...

- Details of PINNs architecture - currently simple
- Utilizing transformer architectures or attention mechanisms?
- Recent developments on accuracy, scaling, physics multimodality...
- Implementation details, e.g. Meta PyTorch, Google JAX, Julia
- More of nice examples and technical overview from great resources
  - Physics-based Deep Learning Book
  - `https://physicsbaseddeeplearning.org/`

Thank You So Much!

Questions, comments...