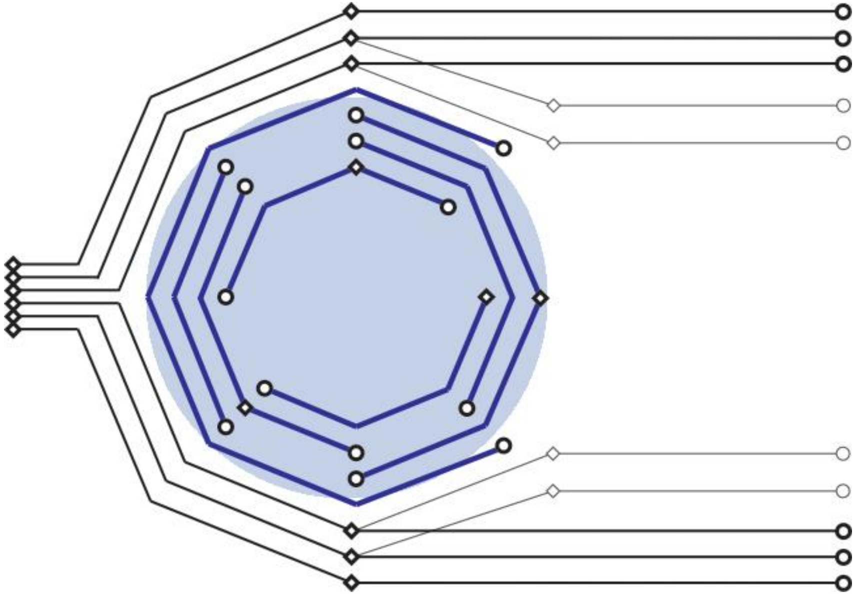


PINN Theories

Adhika Satyadharma

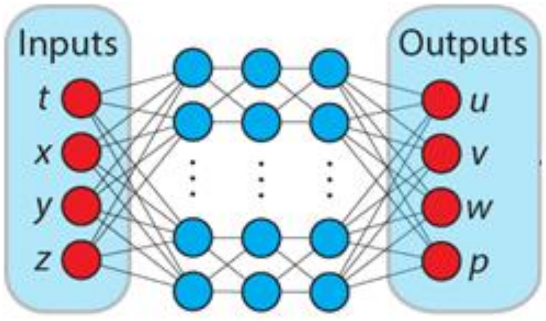
Computational Fluid Dynamics Laboratory
National Taiwan University of Science and Technology

Outline

- Basic PINN concepts
- Noise reduction /elimination
- Error estimation
- FAQ

Basic PINN concepts

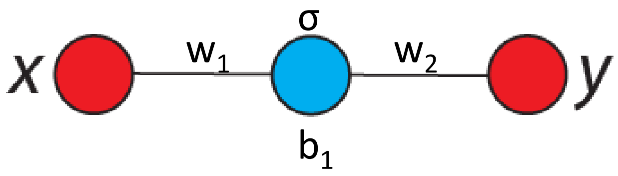
Network Structure



A **neural network (NN)** is inherently a mathematical function ***y** = f(**x**)*, which could be **trained** to learn any function *f*.

There are 3 components in NN:
Input, Output, Network

Example: 1 Layer, 1 Neuron



$$y = w_2\,\sigma(w_1 x + b_1)$$

- A network consists of:
  weight (*w*), bias (*b*) and activation function (σ)
- **Weight & bias** are coefficients/constants
- **Activation function** (e.g. sigmoid) adds non-linearity to the NN, so that it could model non-linear things.

Network Size

Polynomial Functions

$y = ax + b$

$y = ax^2 + bx + c$

$y = ax^3 + bx^2 + cx + d$

$y = ax^4 + bx^3 + cx^2 + dx + e$

- More complex model (more unknown parameters) lets a polynomial model more complex equations.

Neural network

- Bigger models (More parameters = weights & biases) allows to model more complex relationship
- Bigger model takes longer to calculate

ChatGPT

(v3) = 175×10^9 parameters

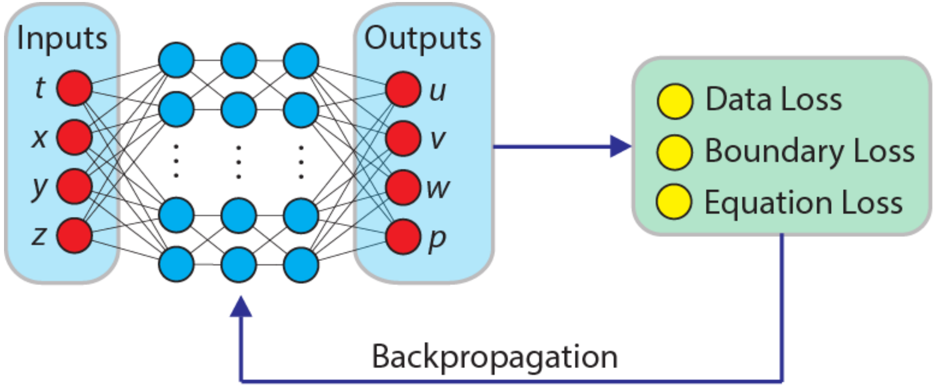
(v4) = 1.7×10^{12} parameters

deepseek

(v2) = 236×10^9 parameters

(v3) = 672×10^9 parameters

Losses



What a network learns depends on the loss.

•A **standard neural network** is usually data-driven (follows data), so it uses **data loss**

•For **PINN**, it uses both **boundary loss and governing equation (GE) / partial differential equation (PDE) loss**

To define what neural network learns, we use loss function:

- Learn to follow data:

Output = Data

$$L = \left(u_{PINN} - u_{Ref}\right)^2$$

- Learn to follow governing equation:

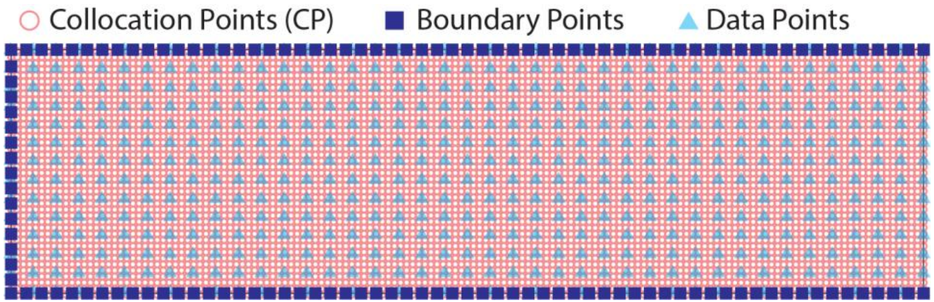
Residual = 0

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \Rightarrow L = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^2$$

*Boundary loss is similar to data loss,
but instead follows the boundary values

Points

The training of PINN is done by **evaluating each loss** at many points in the domain.



- Data Loss (optional) @ data points
- Boundary Loss @ boundary points
- Governing Equation Loss @ collocation points

- PINN check if the losses are satisfied in the respective points.
- If you only have 1 point, the PINN would **ONLY** satisfy the loss at that **single** point
- The points must be spread out in the domain, so that the physics are satisfied in the whole domain

- **Points ~ Mesh**
  More points usually means better result, but longer to train

Physical based loss = Governing equation & boundary

Weights

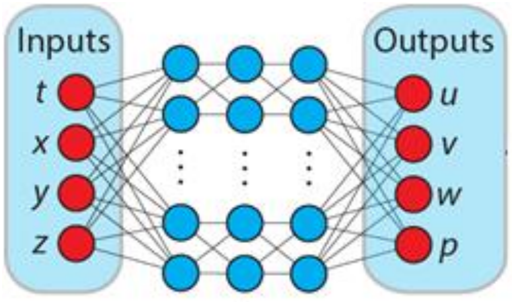
- A neural network only minimize a total loss

$$\textbf{Total Loss} = \boldsymbol{W_D L_D + W_B L_B + W_F L_F}$$

*D = Data
*B = Boundary
*F = Governing equation

- Every Loss has a weight (***W***).
- If you give a higher weight to a loss, it would try to satisfy that loss more.
- **Example:**
  $\boldsymbol{W_D}$ = 100, → (mostly) Follow data,
  $\boldsymbol{W_B}$ = 0.001 → (almost) ignore boundary

Auto differentiation

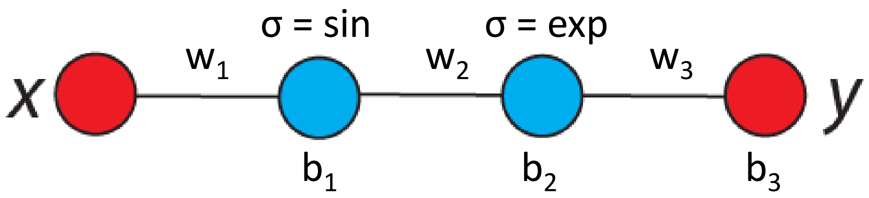


For fluid analysis purpose, we usually want to model:

$u\,(x, y, z, t),\ \ v\,(x, y, z, t)$
$w\,(x, y, z, t),\ p\,(x, y, z, t)$ $\Rightarrow$ **Input:** x, y, z, t
Output: u, v, w, p

*The result is a continues function

As a NN is a math function, we can mathematically calculate derivative using chain rules.



$$y = w_3\ exp(w_2\ \underline{sin(w_1 x + b_1)} + b_2) + b_3$$

$$y = w_3\ \underline{exp(w_2\ L_1 + b_2)} + b_3$$

$$y = w_3\ L_2 + b_3$$

$$dy/dx = dy/dL_2 \times dL_2/dL_1 \times dL_1/dx$$

$$dy/dx = w_3 w_2\ exp(w_2\ L_1 + b_2) \times w_1 \cos(w_1 x + b_1)$$

Noise Reduction

Main Reference: *https://pubs.aip.org/aip/pof/article/36/10/103619/3317204/Assessing-physics-informed-neural-network*

Basic concepts

Issues

- Assume that we have a data that is not completely physically accurate (inaccurate data)
- If we follow the data, our result is also not physically accurate

Solution

- Follow the physically accurate part of the data
- Resolve the non-physically accurate part by using physics

Basic concepts

Concepts

- If we just want to follow data, we could use data loss
- If we just want to follow physics, we could use the boundary and governing equation loss
- If we want to follow the data to a certain level then switch to physics we have to use both loss.

How

- Follow the data until the data loss reach a certain threshold
- After it reach this threshold, start ignoring the data and focus on physics
  (It doesnt completely ignore the data, just focus on physics more)

Technical Details

Noise Reduction Loss

$$\mathcal{L}_{\text{Total, NR}} = \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{D}1} + \mathcal{L}_{\mathcal{D}2} + \mathcal{L}_{\mathcal{F}}$$

$$\mathcal{L}_{\mathcal{D}1} = \omega_{\mathcal{D}1}\mathcal{L}_{\mathcal{D}}$$

Ignores the data as the data loss gets lower

$$\omega_{\mathcal{D}1} = \frac{2}{1+\exp\left(-\alpha_1 \log_{10}(\mathcal{L}_{\mathcal{D}}/\beta_1^2) + 0.5\alpha_1\right)}$$

$$\mathcal{L}_{\mathcal{D}2} = \omega_{\mathcal{D}2}\mathcal{L}_{\mathcal{D}}$$

Forces the physics to work harder
(optional, it is by default used for the noise reduction,
but not used for the error estimation)

$$\omega_{\mathcal{D}2} = \frac{1}{1+\exp(-\alpha_2 \log_{10}(\mathcal{L}_{\mathcal{F}}) + \alpha_2\beta_2)}$$

$\alpha_1 = \alpha_2 = 4.2$
$\beta_2 = 3$

Technical Details

Noise Reduction Loss

$$\mathcal{L}_{\text{Total, NR}} = \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{D}1} + \mathcal{L}_{\mathcal{D}2} + \mathcal{L}_{\mathcal{F}}$$

$$\mathcal{L}_{\mathcal{D}1} = \omega_{\mathcal{D}1}\mathcal{L}_{\mathcal{D}}$$

$$\omega_{\mathcal{D}1} = \frac{2}{1+\exp\left(-\alpha_1 \log_{10}(\mathcal{L}_{\mathcal{D}}/\beta_1^2) + 0.5\alpha_1\right)}$$

$$\mathcal{L}_{\mathcal{D}2} = \omega_{\mathcal{D}2}\mathcal{L}_{\mathcal{D}}$$

$$\omega_{\mathcal{D}2} = \frac{1}{1+\exp(-\alpha_2 \log_{10}(\mathcal{L}_{\mathcal{F}}) + \alpha_2\beta_2)}$$

$\alpha_1 = \alpha_2 = 4.2$

$\beta_2 = 3$

Threshold

- Threshold is set by β_1
- Higher β_1 → higher threshold
  → Ignores data more
  → Quickly focus on physics
  → tolerate more inaccuracies
- If β_1 is to low → Follow data too much
  → Follow the inaccuracies in the data
- If β_1 is to high → Relies on physics too much
  → May not be resolved properly

Error Estimation

Main Reference: (Paper is still under review)

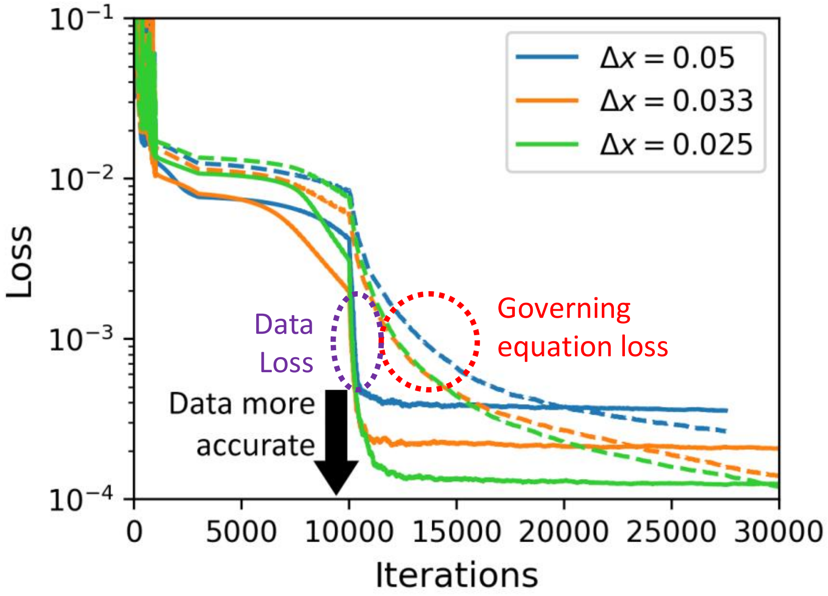
Idea & Goal

Idea

- If PINN follows not completely physically accurate data, the loss would stagnate (cannot go down)
- Better data → lower loss
  Worse data → higher loss

Goal

- Convert loss to indicate the data quality



The loss curve of PINN that follows a cavity flow simulation data at various cell resolution.
(Smaller cell → More accurate simulation data)

Methods

*Sorry but this is pure math

Idea

- Assume we have a fully physically accurate (*A*) PINN. The loss of this PINN is:
  $L^A_{Total} = 0$ (Total Loss)
  $L^A_B = 0$ (Boundary Loss)
  $L^A_F = 0$ (Governing equation Loss)

- Since our data is not physically accurate, the data loss is not zero.
- Since the PINN is physically accurate, it has the same solution as an actual / analytical solution ($u_{PINN} = u_{Actual}$)

- The data loss on a physically accurate PINN is the MSE (mean square error) of the data.
  $L^A_D = (u_{PINN} - u_{Ref})^2$
  $L^A_D = (u_{Actual} - u_{Ref})^2$

- If we can obtain a physically accurate PINN, we can easily evaluate the quality of the data.
  (MSE of the data = Data loss of PINN)

Methods

*Sorry but this is pure math

Issues

- It is very difficult to obtain a physically accurate PINN, so we cannot easily calculate the data MSE.

Solution

- We do not calculate the exact loss value, but estimate it (Min Guess < L^A_D < Max Guess)

Min Estimate

- A PINN always prefer the lowest total loss such as:

$$L_{Total} \leq L^A_{Total}$$

- Deriving this relationship:

$$L_D + L_B + L_F \leq L^A_D + L^A_B + L^A_F$$

$$L_D + L_B + L_F \leq L^A_D$$

$$L_D \leq L^A_D$$

(The data loss from PINN is always lower than the MSE of the data)

- **Min guess** = L_D

Methods

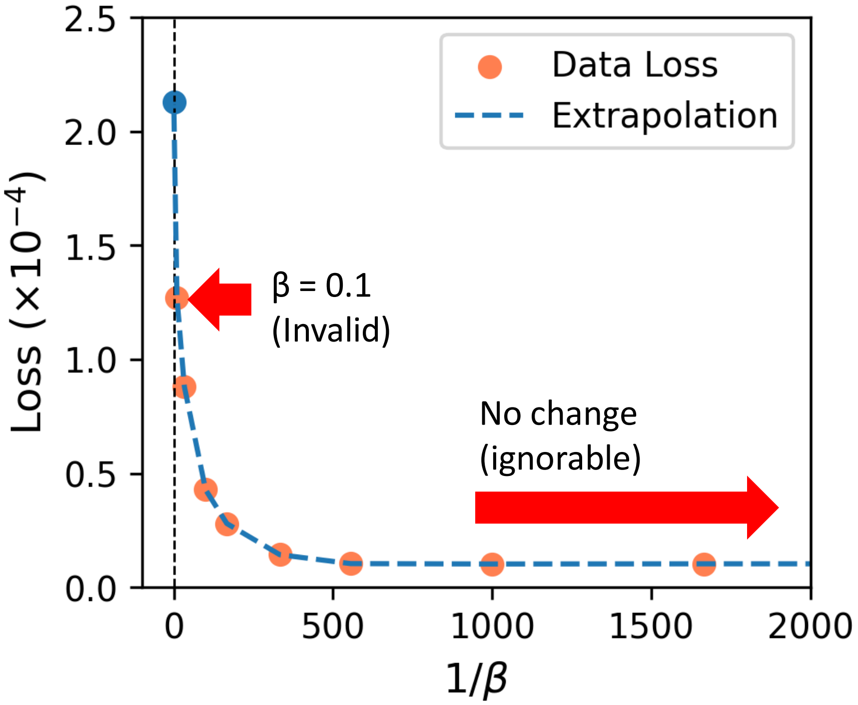
*Sorry but this is pure math

Max Estimate

- Based on NR-PINN (the one used in the noise reduction), when we use a higher $\beta = \beta_1$, it can ignore the inaccuracies in the data, and make the result more accurate.
  *Ignore the second optional term

- Higher β → More accurate, until the physics is incapable of resolving it correctly.

- To get the Max Guess, we simulate with several β, then extrapolate to get the data loss at β = infinity



*See https://doi.org/10.3390/computation9020010 for the extrapolation method

Methods

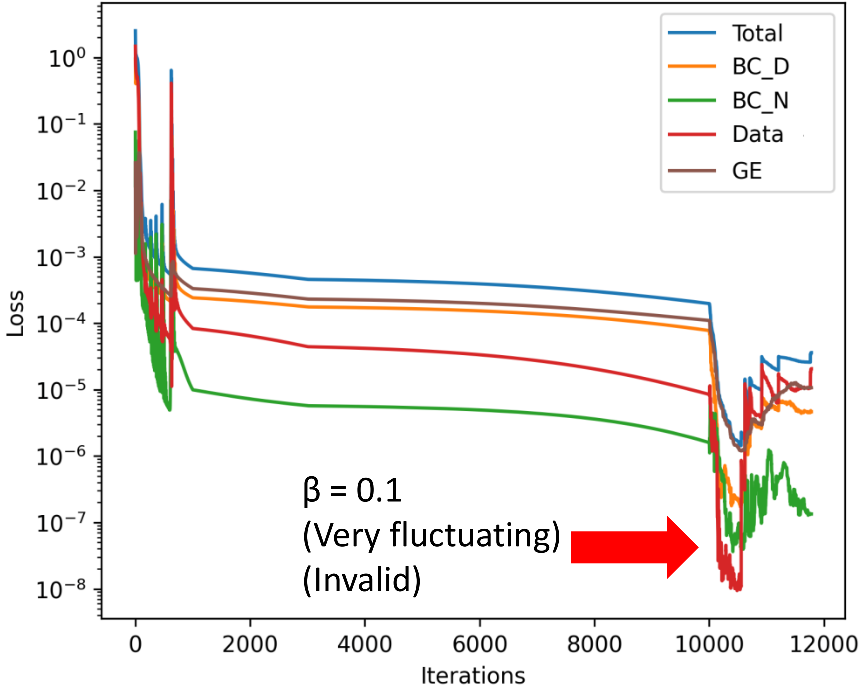
*Sorry but this is pure math

Min Estimate (Correction)

- It is better to get the minimum guess from the maximum β, before it is unstable.
- This would lead to the most accurate PINN, which leads to the better **Min guess** = L_D

Which estimation is invalid?

- If it highly fluctuates
- If the data loss is 20x lower than the physics
- If the change of β doesn't make any change in the data loss and physics loss.



FAQ

FAQ

- **Why the loss quickly dropped after 10000 iterations?**
  The code (by default) uses (in order):
  - ADAM optimizer for iterations 0 - 1000 with learning rate = 10^{-2}
  - ADAM optimizer for iterations 1000 - 3000 with learning rate = 10^{-3}
  - ADAM optimizer for iterations 3000 - 10000 with learning rate = 10^{-4}
  - LBFGS optimizer for iterations 10000 and above, until it is CONVERGED

- **What is the difference between Adam and LBFGS?**
  Adam is better, faster to converge when starting from a random solution. LBFGS is better to refine a solution.

FAQ

- By default, the network is initialized with random parameters (the weights and biases are random)
- The default activation function in this code is the hyperbolic tangent function

- **If your result looks odd, try:**
  - Try increasing the points
  - Try to use a larger network
  - Modify the weights

  * You can do (many) other things, but those techniques are too advance

  (*Im not expecting a 100% accurate result, if it is accurate enough it is fine. Getting a totally accurate result with PINN is hard (not its strong point).)

- **Tips for setting β :**
  - Try setting a wide range:

    1e-3, 1e-2, 1e-1, 1., 10., 100., 1000.

  - See where the most likely / optimal range is (based on the result or loss)
  - Refine β in that range