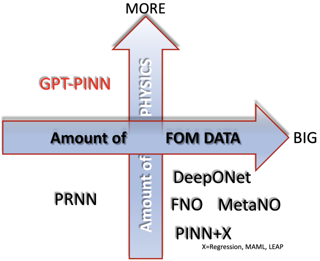
GPT-PINN: Generative Pre-Trained Physics-Informed Neural Networks toward non-intrusive Meta-learning of parametric PDEs

Yanlai Chen

UMass Dartmouth

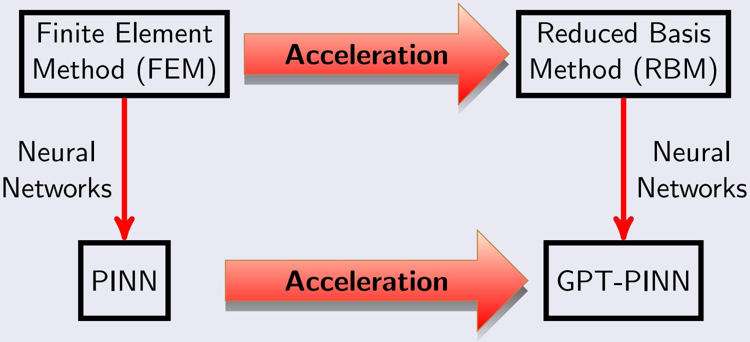
CBMS 2023, Deep Learning & Numerical PDEs



Joint work with Shawn Koohy

GPT-PINN to PINN is what RBM is to FEM

Structure-preserving accelerations

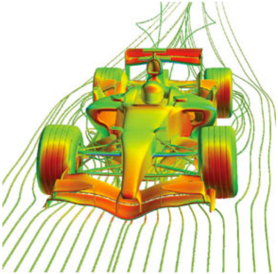


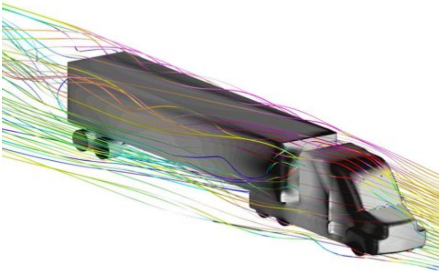
Outline

- Projection-based Model Order Reduction (PMOR): a primer
- Physics-Informed Neural Networks (PINNs)
- From PINN to GPT-PINN: Design and details
- GPT-PINN: Numerical results

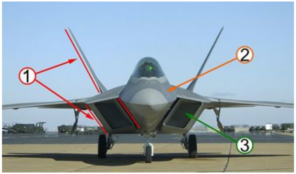
PMOR for the multi-query context

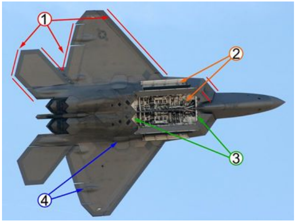
Aerodynamics





Stealth Technology

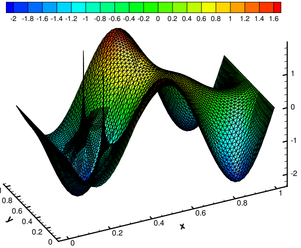




Optimization, Inverse problems, Sensitivity analysis, Uncertainty quantification . . .

PMOR: intuition and idea

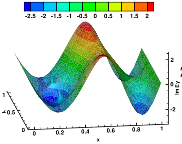
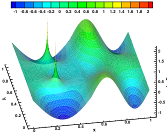
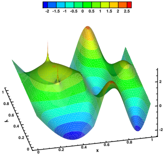
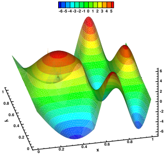
Traditional methods:



References: Nagy 1979; Noor, Peters 1980; Rozza, Huynh, Patera 2008; Benner, Gugercin, Willcox 2015; Haasdonk 2017; Quarteroni, Manzoni, Negri 2016; Hesthaven, Rozza, Stamm 2016; Binev, Cohen, Dahmen, DeVore, Petrova, Wojtaszczyk 2011; Buffa, Maday, Patera, Prudhomme, Turinici 2012; Maday, Patera, Turinici 2002; C., Gottlieb, Ji, Maday 2021; Berkooz, Holmes, Lumley 1993; Willcox, Peraire 2002;

PMOR relies on traditional methods but strives to accelerate parameter (denoted by $\boldsymbol{\mu}$, could be implicit) dependent simulations.

Ansatz for linear reduction: $u(\boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) u(\boldsymbol{\mu}^i)$


$\approx c_1(\boldsymbol{\mu})$

$+ c_2(\boldsymbol{\mu})$

$+ c_3(\boldsymbol{\mu})$


The Parameter to Code (P2C) Map

$$\mu \mapsto \boldsymbol{c}(\mu) := \begin{pmatrix} c_1(\boldsymbol{\mu}) \\ c_2(\boldsymbol{\mu}) \\ \vdots \\ c_N(\boldsymbol{\mu}) \end{pmatrix}$$

PMOR for $\mu \mapsto \boldsymbol{c}(\mu)$ embeds physics

$$-\nabla \cdot (\kappa \nabla u) + c\, u = f \quad \text{in } \Omega; u = 0 \quad \text{on } \partial\Omega. \quad \boldsymbol{\mu} := \{\kappa, c\} \in \mathcal{D}.$$

PMOR for $\mu \mapsto \boldsymbol{c}(\mu)$ embeds physics

$$-\nabla \cdot (\kappa \nabla u) + c\, u = f \quad \text{in } \Omega; u = 0 \quad \text{on } \partial\Omega. \quad \boldsymbol{\mu} := \{\kappa, c\} \in \mathcal{D}.$$

$$\Downarrow$$

Find $u(\boldsymbol{\mu}) \in H_0^1(\Omega)$ such that $a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v)\ \forall v \in H_0^1(\Omega)$ with

$$a(u, v; \boldsymbol{\mu}) = \int_\Omega (\kappa \nabla u \cdot \nabla v + cuv)\ dx \quad f(v) = \int_\Omega fv\, dx$$

PMOR for $\mu \mapsto \boldsymbol{c}(\mu)$ embeds physics

$$-\nabla \cdot (\kappa \nabla u) + c\, u = f \quad \text{in } \Omega; u = 0 \quad \text{on } \partial\Omega. \quad \boldsymbol{\mu} := \{\kappa, c\} \in \mathcal{D}.$$

$$\Downarrow$$

$$\text{Find } u(\boldsymbol{\mu}) \in H_0^1(\Omega) \text{ such that } a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v) \ \forall v \in H_0^1(\Omega) \text{ with}$$

$$a(u, v; \boldsymbol{\mu}) = \int_\Omega (\kappa \nabla u \cdot \nabla v + cuv)\, dx \quad f(v) = \int_\Omega fv\, dx$$

$$\Downarrow$$

$$\text{FEM : Find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in V_h \text{ such that } a_h(u^{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f_h(v) \ \forall v \in V_h$$

$$\dim(V_h) = \mathcal{N}$$

PMOR for $\mu \mapsto \boldsymbol{c}(\mu)$ embeds physics

$$-\nabla \cdot (\kappa \nabla u) + c\, u = f \quad \text{in } \Omega;\, u = 0 \quad \text{on } \partial\Omega. \quad \boldsymbol{\mu} := \{\kappa, c\} \in \mathcal{D}.$$

$$\Downarrow$$

$$\text{Find } u(\boldsymbol{\mu}) \in H_0^1(\Omega) \text{ such that } a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v)\ \forall v \in H_0^1(\Omega) \text{ with}$$

$$a(u, v; \boldsymbol{\mu}) = \int_\Omega (\kappa \nabla u \cdot \nabla v + cuv)\, dx \quad f(v) = \int_\Omega fv\, dx$$

$$\Downarrow$$

$$\text{FEM : Find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in V_h \text{ such that } a_h(u^{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f_h(v)\ \forall v \in V_h$$

$$\dim(V_h) = \mathcal{N}$$

$$\Downarrow \text{(Structure-preserving)}$$

$$\text{RBM : Find } u_N(\boldsymbol{\mu}) \in W_N \text{ such that } a_h(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f_h(v)\ \forall v \in W_N$$

$$\dim(W_{RB}) = N << \mathcal{N}, \;\; W_N \subset V_h$$

Theory: **Fast decay** of Kolmogorov N-width

$$d_N\left[u\left(\cdot;\mathcal{D}\right)\right] := \inf_{\substack{X_N \subset V_h \\ \dim X_N = N}} \mathrm{Dist}\left(u\left(\cdot;\mathcal{D}\right), X_N\right)$$

Theory: **Fast decay** of Kolmogorov N-width

$$\sup_{\boldsymbol{\mu}\in\mathcal{D}} \inf_{v\in X_N} \|u(\cdot,\boldsymbol{\mu}) - v\|_{V_h}$$

$$d_N\left[u\left(\cdot;\mathcal{D}\right)\right] := \inf_{\substack{X_N\subset V_h \\ \dim X_N = N}} \mathrm{Dist}\left(u\left(\cdot;\mathcal{D}\right), X_N\right)$$

How to construct that W_N

RBM-type of approaches

Greedy algorithm

A Posteriori error estimate

$\Downarrow$

N full order queries

How to construct that W_N

RBM-type of approaches

Greedy algorithm

A Posteriori error estimate

$\Downarrow$

N full order queries

POD-type of approaches

A Priori sampling of μ-domain

SVD

Truncation down to N

$\Downarrow$

$\gg N$ full order queries

How to construct that W_N

RBM-type of approaches

Greedy algorithm

A Posteriori error estimate

$\Downarrow$

N full order queries

POD-type of approaches

A Priori sampling of μ-domain

SVD

Truncation down to N

$\Downarrow$

$\gg N$ full order queries

Small **Amount of FOM data** Big

Neural Network (NN) approaches for the $\mu \mapsto \boldsymbol{c}(\mu)$ map

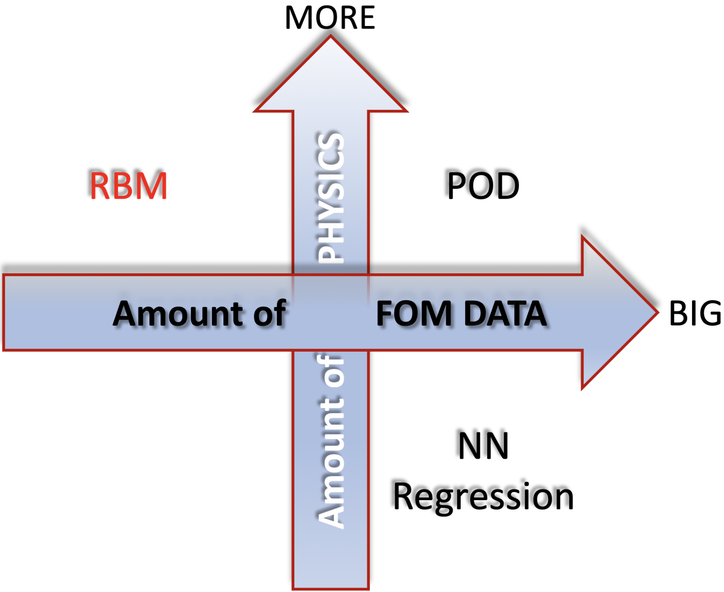
POD-NN (Hesthaven, Ubbiali, 2018 Wang, Hesthaven, Ray, 2019): Map recovered via direct evaluation of NN.

DL-ROM, POD-DL-ROM (Fresca, Dedè, Manzoni, 2021; Fresca, Manzoni, 2021): Avoids the projection stage, map recovered via direct evaluation of a NN.

Deep convolutional autoencoders (Lee, Carlberg, 2020): nonlinear reduction, map recovered via direct evaluation of a NN.

Analysis (Kutyniok, Petersen, Raslan, Schneider, 2022.): Theoretical upper bound on the complexity of DNN approximating parametric solution maps.

Physics-Data cataloging when solving the $\mu \mapsto \boldsymbol{c}(\mu)$ map



PINN

$$\Psi_{\text{NN}}^{\theta}(\boldsymbol{x}, t) = C_K \circ \sigma \circ C_{K-1} \ldots \circ C_1(\boldsymbol{x}, t), \quad C_k(Z) = W_k\, Z + b_k.$$

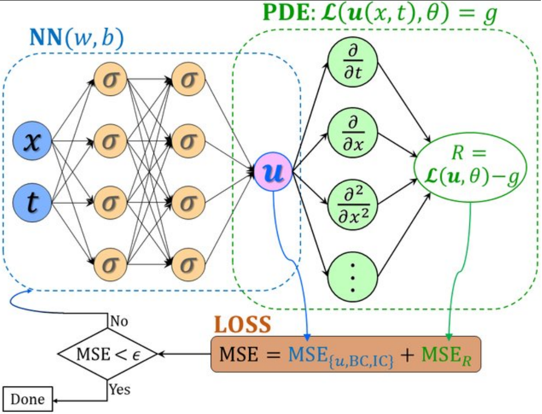
$$\theta := \{W_k, b_k\}_{k=1}^{K}$$

$$\theta^* = \arg\min_{\theta} \mathrm{MSE}(\theta)$$

(via training, automatic differentiation)

PINN approximation:

$$u(\boldsymbol{x}, t) \approx \Psi_{\text{NN}}^{\theta^*}(\boldsymbol{x}, t)$$



(Credit: Meng, Li, Zhang, Karniadakis, PPINN, CMAME 2020.)

PINN for parametric PDE / Meta-learning

Challenges of PINNs for parametric PDEs

$\theta^* = \theta^*(\boldsymbol{\mu})$: high-dimension (over parameterization), multi-query.
Lack of low-rank structure in θ^*.

PINN for parametric PDE / Meta-learning

Challenges of PINNs for parametric PDEs

$\theta^* = \theta^*(\boldsymbol{\mu})$: high-dimension (over parameterization), multi-query.
Lack of low-rank structure in θ^*.

Related existing attempts

Exploring $\boldsymbol{\mu}$-dependence of θ^*: regression/interpolation of $\theta^*(\boldsymbol{\mu})$ with labeled data, or adopting standard meta-learning techniques (MAML, LEAP). See Penwarden, Zhe, Narayan, & Kirby 2022, called **PINN+X** herein.

PINN for parametric PDE / Meta-learning

Challenges of PINNs for parametric PDEs

$\theta^* = \theta^*(\boldsymbol{\mu})$: high-dimension (over parameterization), multi-query.
Lack of low-rank structure in θ^*.

Related existing attempts

Exploring $\boldsymbol{\mu}$-dependence of θ^*: regression/interpolation of $\theta^*(\boldsymbol{\mu})$ with labeled data, or adopting standard meta-learning techniques (MAML, LEAP). See Penwarden, Zhe, Narayan, & Kirby 2022, called **PINN+X** herein.

DeepONet, FNO: POD-based thinking - pervasive sampling of input function space. See Lu, Jin, Pang, Zhang & Karniadakis, 2021. Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2021.

PINN for parametric PDE / Meta-learning

Challenges of PINNs for parametric PDEs

$\theta^* = \theta^*(\boldsymbol{\mu})$: high-dimension (over parameterization), multi-query.
Lack of low-rank structure in θ^*.

Related existing attempts

Exploring $\boldsymbol{\mu}$-dependence of θ^*: regression/interpolation of $\theta^*(\boldsymbol{\mu})$ with labeled data, or adopting standard meta-learning techniques (MAML, LEAP). See Penwarden, Zhe, Narayan, & Kirby 2022, called **PINN+X** herein.

DeepONet, FNO: POD-based thinking - pervasive sampling of input function space. See Lu, Jin, Pang, Zhang & Karniadakis, 2021. Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2021.

MetaNO: Transferring $\theta^*(\boldsymbol{\mu})$. Surrogate is fully data-driven. See Zhang, You, Gao, Yu, Lee, & Yu, 2023.

PINN for parametric PDE / Meta-learning

Challenges of PINNs for parametric PDEs

$\theta^* = \theta^*(\boldsymbol{\mu})$: high-dimension (over parameterization), multi-query. Lack of low-rank structure in θ^*.

Related existing attempts

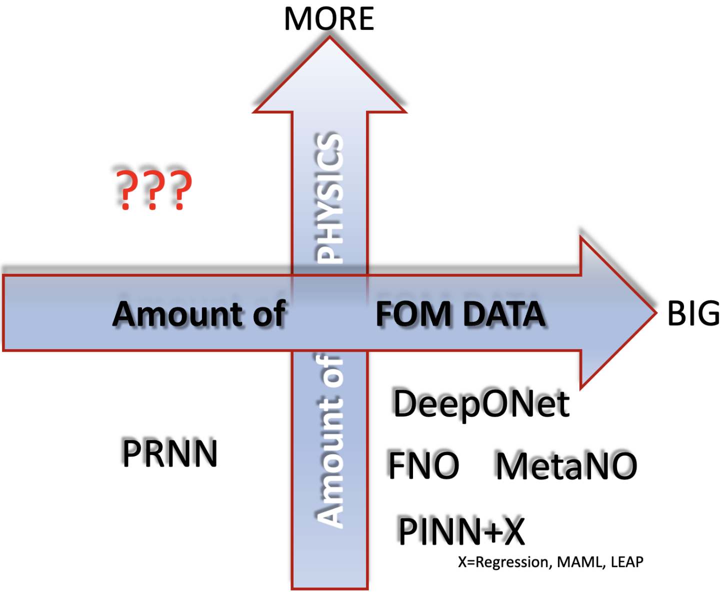
Exploring $\boldsymbol{\mu}$-dependence of θ^*: regression/interpolation of $\theta^*(\boldsymbol{\mu})$ with labeled data, or adopting standard meta-learning techniques (MAML, LEAP). See Penwarden, Zhe, Narayan, & Kirby 2022, called **PINN+X** herein.

DeepONet, FNO: POD-based thinking - pervasive sampling of input function space. See Lu, Jin, Pang, Zhang & Karniadakis, 2021. Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2021.

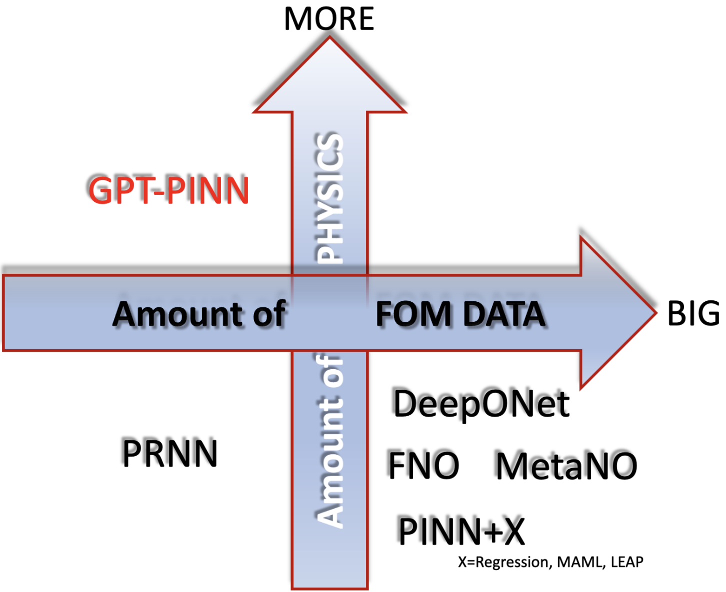
MetaNO: Transferring $\theta^*(\boldsymbol{\mu})$. Surrogate is fully data-driven. See Zhang, You, Gao, Yu, Lee, & Yu, 2023.

PRNN: Needs less data, but $(\boldsymbol{\mu}, t) \mapsto \mathbf{c}(\boldsymbol{\mu})$ is still regression based (i.e. no physics) although both residual and labelled data are used in training of the map. See Chen, Wang, Hesthaven, Zhang, 2021.

Physics-Data cataloging of the methods: RBM for PINN?



Physics-Data cataloging of the methods: RBM for PINN?



GPT-PINN: the inspiration

$$\boxed{\text{RBM Ansatz: } u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) u(\boldsymbol{\mu}^i)}$$

GPT-PINN: the inspiration

$$\boxed{\text{RBM Ansatz: } u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) \Psi_{\mathrm{NN}}^{\theta^i}(x, t)}$$

GPT-PINN: the inspiration

One hidden layer,0-bias
Hyper-reduced NN, $\mathrm{NN}^{\mathrm{r}}(2, N, 1)$

RBM Ansatz: $u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) \Psi_{\mathrm{NN}}^{\theta^i}(x, t)$

GPT-PINN: the inspiration

One hidden layer,0-bias
Hyper-reduced NN, $\mathrm{NN}^{\mathrm{r}}(2, N, 1)$

RBM Ansatz: $u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) \Psi_{\mathrm{NN}}^{\theta^i}(x, t)$

Width of the hidden layer
Growth of Hyper-reduced NN

GPT-PINN: the inspiration

One hidden layer,0-bias
Hyper-reduced NN, $\mathrm{NN}^{\mathrm{r}}(2, N, 1)$

Weight at the i^{th} node.

RBM Ansatz: $u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) \Psi_{\mathrm{NN}}^{\theta^i}(x, t)$

Width of the hidden layer
Growth of Hyper-reduced NN

GPT-PINN: the inspiration

One hidden layer,0-bias
Hyper-reduced NN, $\mathrm{NN}^{\mathrm{r}}(2, N, 1)$

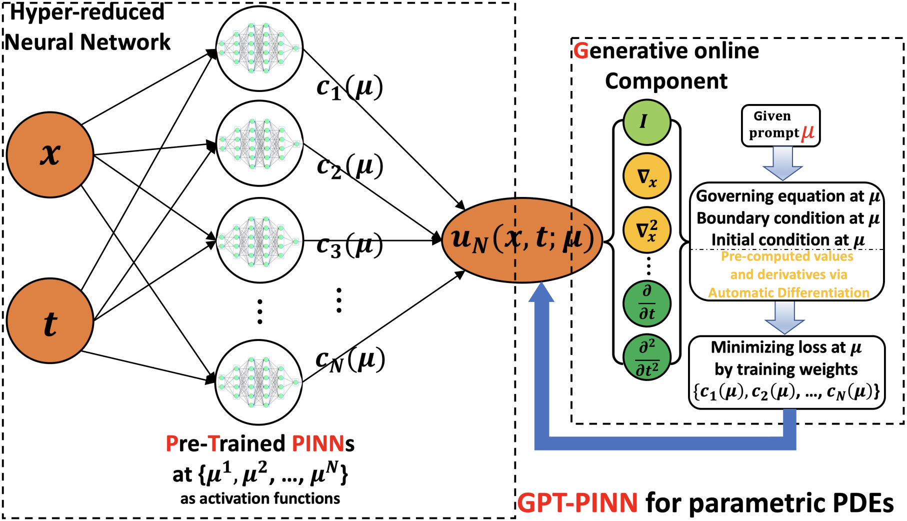
Weight at the i^{th} node.

RBM Ansatz: $u_N(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \sum_{i=1}^{N} c_i(\boldsymbol{\mu}) \Psi_{\mathrm{NN}}^{\theta^i}(x, t)$

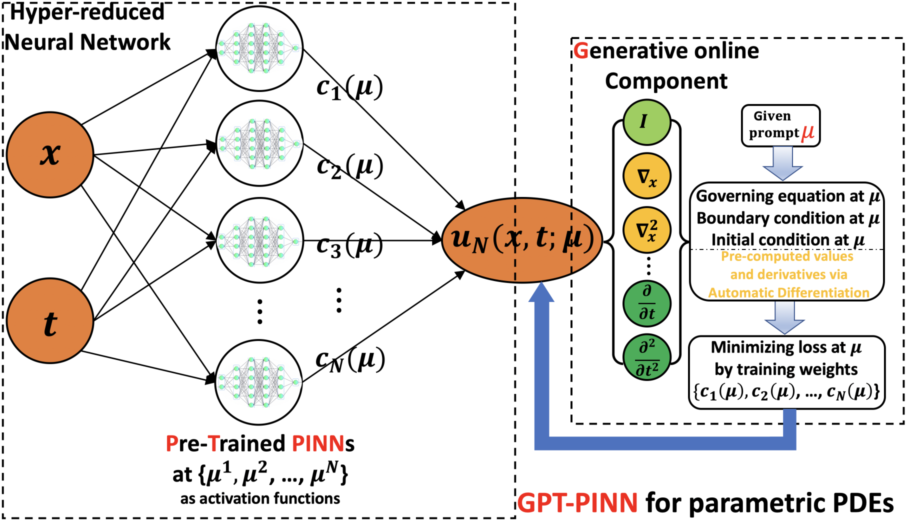
Width of the hidden layer
Growth of Hyper-reduced NN

Customized activation function
Well-chosen Pre-Trained PINN

GPT-PINN: the schematics

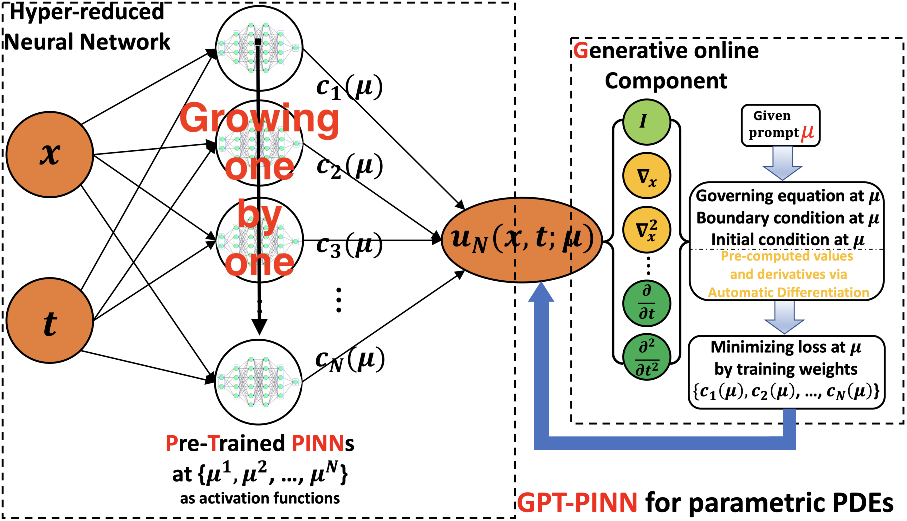


GPT-PINN: the schematics



$\text{MSE}(\mu) = \text{MSE}_{\{u_N,\text{BC},\text{IC}\}} + \text{MSE}_{\text{PDE Residual}}$, denoted by $\mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c}(\mu))$

GPT-PINN: the schematics



$$\text{MSE}(\mu) = \text{MSE}_{\{u_N,\text{BC},\text{IC}\}} + \text{MSE}_{\text{PDE Residual}}, \text{ denoted by } \mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c}(\mu))$$

GPT-PINN features

It embeds physics

$\mathbf{c} \leftarrow \mathbf{c} - \delta_r \nabla_{\mathbf{c}} \mathcal{L}^{\mathrm{GPT}}_{\mathrm{PINN}}(\mathbf{c})$, with $\mathcal{L}^{\mathrm{GPT}}_{\mathrm{PINN}}(\mathbf{c})$ including residual, BC, and IC.

GPT-PINN features

It embeds physics

$\mathbf{c} \leftarrow \mathbf{c} - \delta_r \nabla_{\mathbf{c}} \mathcal{L}^{\text{GPT}}_{\text{PINN}}(\mathbf{c})$, with $\mathcal{L}^{\text{GPT}}_{\text{PINN}}(\mathbf{c})$ including residual, BC, and IC.

It needs small FOM data

Like RBM, number of Full PINN queries is minimum (N) with no truncation.

GPT-PINN features

It embeds physics

$\mathbf{c} \leftarrow \mathbf{c} - \delta_r \nabla_{\mathbf{c}} \mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$, with $\mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$ including residual, BC, and IC.

It needs small FOM data

Like RBM, number of Full PINN queries is minimum (N) with no truncation.

It has Super/Meta neurons

Neurons are adaptively built and customized for the problem at hand.

GPT-PINN features

It embeds physics

$\mathbf{c} \leftarrow \mathbf{c} - \delta_r \nabla_{\mathbf{c}} \mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$, with $\mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$ including residual, BC, and IC.

It needs small FOM data

Like RBM, number of Full PINN queries is minimum (N) with no truncation.

It has Super/Meta neurons

Neurons are adaptively built and customized for the problem at hand.

It is non-intrusive

Can plug in any existing PINN. The meta network is independent of those pre-trained at the neurons.

GPT-PINN features

It embeds physics

$\mathbf{c} \leftarrow \mathbf{c} - \delta_r \nabla_{\mathbf{c}} \mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$, with $\mathcal{L}_{\text{PINN}}^{\text{GPT}}(\mathbf{c})$ including residual, BC, and IC.

It needs small FOM data

Like RBM, number of Full PINN queries is minimum (N) with no truncation.

It has Super/Meta neurons

Neurons are adaptively built and customized for the problem at hand.

It is non-intrusive

Can plug in any existing PINN. The meta network is independent of those pre-trained at the neurons.

It accelerates PINN

2 - 3 orders of magnitude speedup, see numerical results.

GPT-PINN delicacies: Precomputation for fast training

Insights

Linearity of the derivative operations $\oplus$ Collocation nature of the PINN/GPT-PINN loss function

GPT-PINN delicacies: Precomputation for fast training

Insights

Linearity of the derivative operations $\oplus$ Collocation nature of the PINN/GPT-PINN loss function

Precompute, precompute, precompute

$$\Psi_{\mathrm{NN}}^{\theta^i}(\mathcal{C}),\ \frac{\partial^k}{\partial t^k}\left(\Psi_{\mathrm{NN}}^{\theta^i}\right)(\mathcal{C})\,(k=1,2,\cdots),\ \nabla_{\boldsymbol{x}}^{\ell}\Psi_{\mathrm{NN}}^{\theta^i}(\mathcal{C})\,(\ell=1,2,\cdots).$$

Here, $\mathcal{C}$ denotes the collocation sets for interior residuals, boundary and initial conditions.

GPT-PINN delicacies: A natural error indicator toward a greedy algorithm

RBM *a posteriori* error estimators/indicators

- ✓ Guides the generation of the reduced solution space.
- ✓ Certifies the accuracy of the surrogate solution.
- ✓ Often residual-based.

GPT-PINN delicacies: A natural error indicator toward a greedy algorithm

RBM *a posteriori* error estimators/indicators

- ✓ Guides the generation of the reduced solution space.
- ✓ Certifies the accuracy of the surrogate solution.
- ✓ Often residual-based.

Accuracy of surrogate network $\mathrm{NN}^{\mathrm{r}}(2, n, 1)$

- ✓ PINN/GPT-PINN training loss is residual-based.

GPT-PINN delicacies: A natural error indicator toward a greedy algorithm

RBM *a posteriori* error estimators/indicators

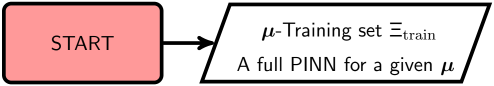
- ✓ Guides the generation of the reduced solution space.
- ✓ Certifies the accuracy of the surrogate solution.
- ✓ Often residual-based.

Accuracy of surrogate network $\mathrm{NN}^{\mathrm{r}}(2, n, 1)$

- ✓ PINN/GPT-PINN training loss is residual-based.

$$\Delta^r_{\mathrm{NN}}(\mathbf{c}(\boldsymbol{\mu})) \triangleq \mathcal{L}^{\mathrm{GPT}}_{\mathrm{PINN}}(\mathbf{c}(\boldsymbol{\mu})).$$

GPT-PINN: the greedy algorithm



GPT-PINN: the greedy algorithm

START

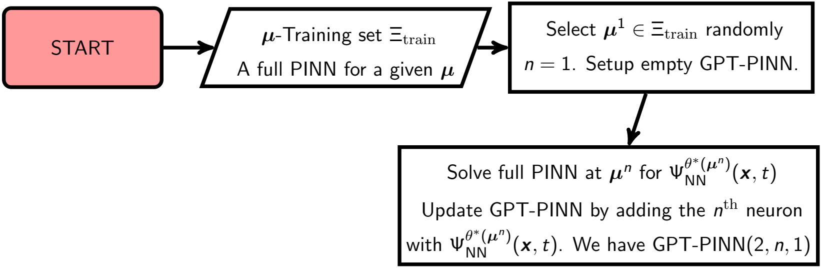
$\boldsymbol{\mu}$-Training set Ξ_{train}

A full PINN for a given $\boldsymbol{\mu}$

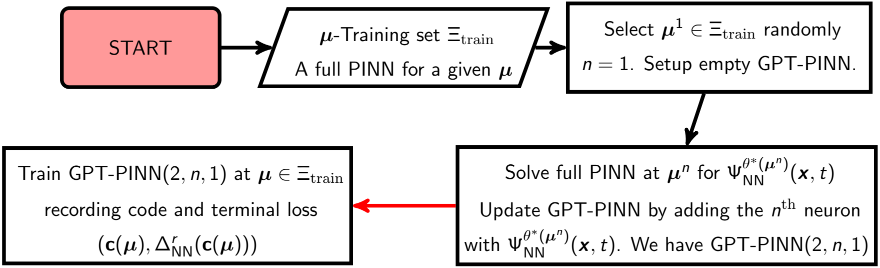
Select $\boldsymbol{\mu}^1 \in \Xi_{\mathrm{train}}$ randomly

$n = 1$. Setup empty GPT-PINN.

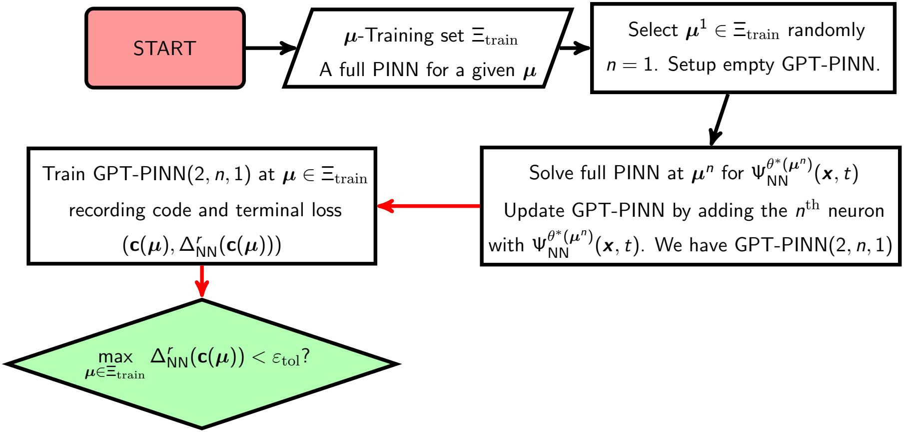
GPT-PINN: the greedy algorithm



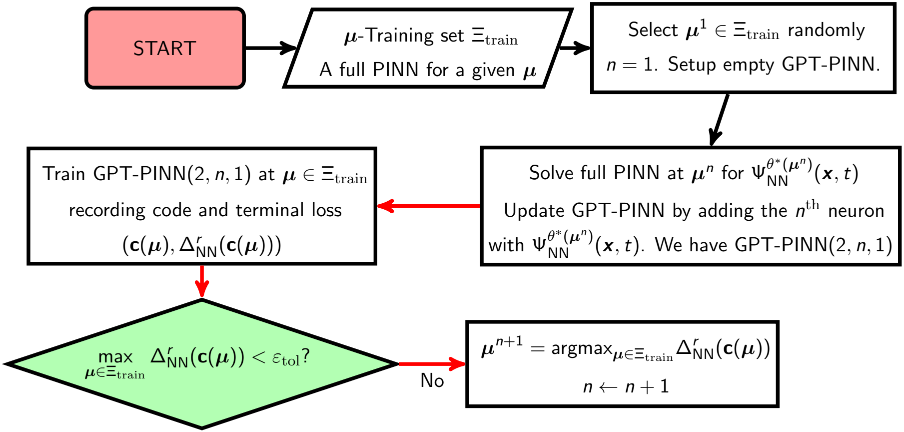
GPT-PINN: the greedy algorithm



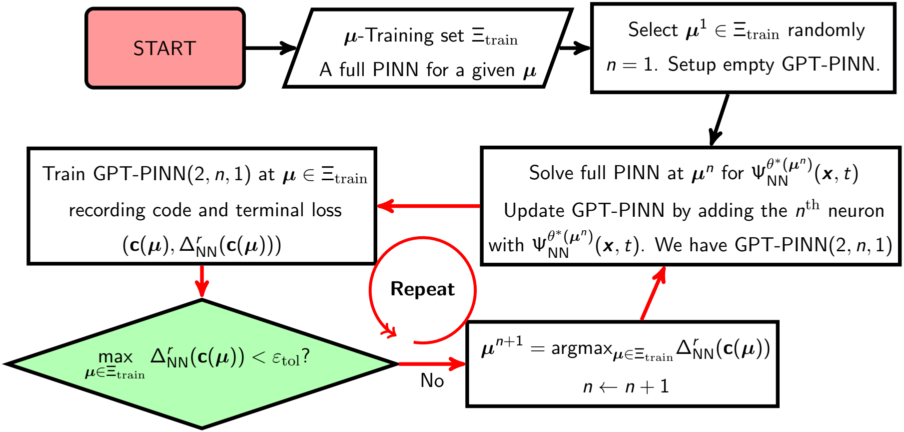
GPT-PINN: the greedy algorithm



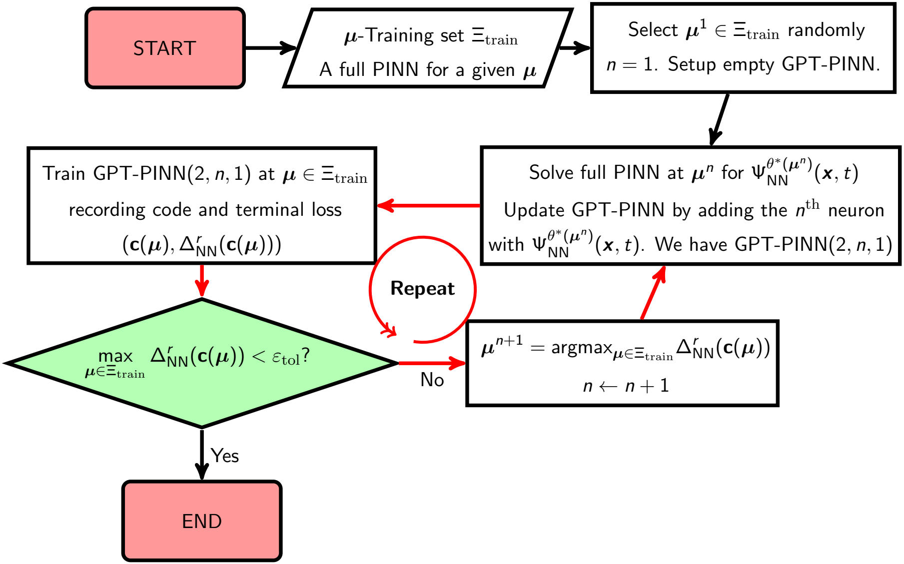
GPT-PINN: the greedy algorithm



GPT-PINN: the greedy algorithm



GPT-PINN: the greedy algorithm



GPT-PINN numerical results: test problems

KG: Klein-Gordon equation with $(\alpha, \beta, \gamma) \in [-2, -1] \times [0, 1] \times [0, 1]$

$$u_{tt} + \alpha u_{xx} + \beta u + \gamma u^2 + x \cos(t) - x^2 \cos^2(t) = 0$$

$$(x, t) \in [-1, 1] \times [0, 5], \text{ with } u(\pm 1, t), u(x, 0), u_t(x, 0) \text{ given.}$$

B: Burgers' equation with $\nu \in [0.005, 1]$

$$u_t + u u_x - \nu u_{xx} = 0, \quad (x, t) \in [-1, 1] \times [0, 1]$$

with $u(\pm 1, t), u(x, 0)$ given.

AC: Allen-Cahn equation with $(\lambda, \epsilon) \in [0.0001, 0.001] \times [1, 5]$

$$u_t - \lambda u_{xx} + \epsilon(u^3 - u) = 0, \quad (x, t) \in [-1, 1] \times [0, 1]$$

with $u(\pm 1, t), u(x, 0)$ given.

GPT-PINN numerical results: setup

Equation	KG	B	AC
Architecture	$[2, 40, 40, 1]$, fully connected	$[2, 20, 20, 20, 20, 1]$, fully connected	$[2, 128, 128, 128, 128, 1]$, fully connected
Activation	$\cos(z)$	$\tanh(z)$	$\tanh(z)$
Collocation set	Uniform: $(10,000, 512, 512)$ for interior, boundary, and initial	Uniform: $(10,000, 100, 100)$ for interior, boundary, and initial	Latin hypercube $(20,000, 100, 512)$ for interior, boundary, and initial
Optimizer	ADAM	ADAM	(ADAM, L-BFGS)
Learning rate	0.0005	0.005	(0.005, 0.8)
Max epochs	$75,000$	$60,000$	$(10,000, 10,000)$
Ξ_{train}	Tensorial $10 \times 10 \times 10$	size 129 uniform	size 121 uniform
GPT-PINN learning rate	0.025	0.02	0.0025
GPT-PINN epochs	2000	2000	2000

Setup: Additional details for the Burgers' equation

"Mini-batching" to accommodate near-discontinuities/shocks

$\left(\Psi_{\text{NN}}^{\theta^i}\right)_x$ and $\left(\Psi_{\text{NN}}^{\theta^i}\right)_{xx}$ are of little value in the training of GPT-PINN when x is close to these regions.

Setup: Additional details for the Burgers' equation

"Mini-batching" to accommodate near-discontinuities/shocks

$\left(\Psi_{\text{NN}}^{\theta^i}\right)_x$ and $\left(\Psi_{\text{NN}}^{\theta^i}\right)_{xx}$ are of little value in the training of GPT-PINN when x is close to these regions.

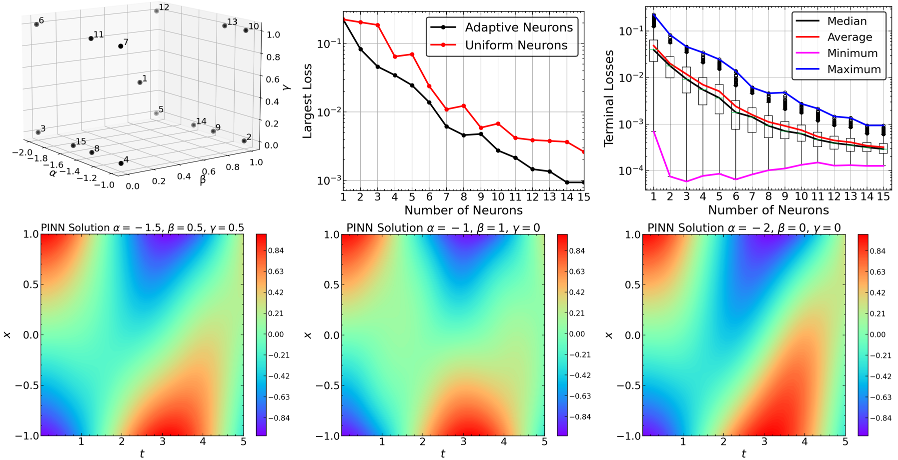
Strategy: Excluding the collocation points where $\left|\left(\Psi_{\text{NN}}^{\theta^i}\right)_{xx}\right|$ is within the top 20% of all such values:

$$\mathcal{C}_{pos}^r = \mathcal{C}_{pos} \setminus \left\{ x : \left|\left(\Psi_{\text{NN}}^{\theta^i}\right)_{xx}(x)\right| > 0.8 \max_x \left|\left(\Psi_{\text{NN}}^{\theta^i}\right)_{xx}(x)\right| \right\}.$$

$\mathcal{C}_{pos}$: Collocation sets for the full PINN, with *pos* indexing interior, boundary, and initial.

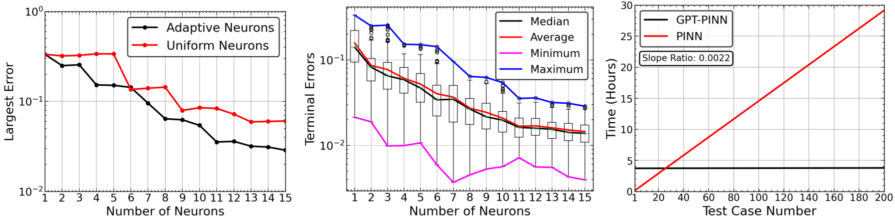
$\mathcal{C}_{pos}^r$: Collocation sets for the GPT-PINN, with *pos* indexing interior, boundary, and initial.

GPT-PINN numerical results: KG training



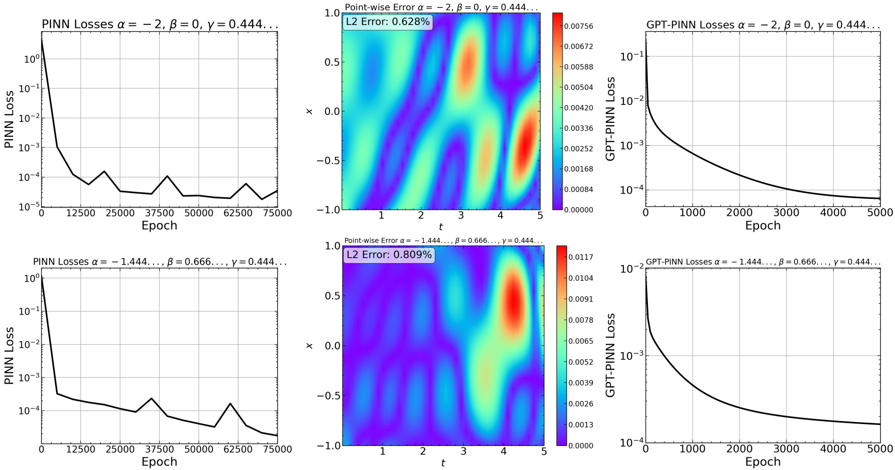
The adaptively chosen parameter values, GPT-PINN training losses, and the first three full PINN solutions GPT-PINN adopts as the activation functions.

GPT-PINN numerical results: KG testing



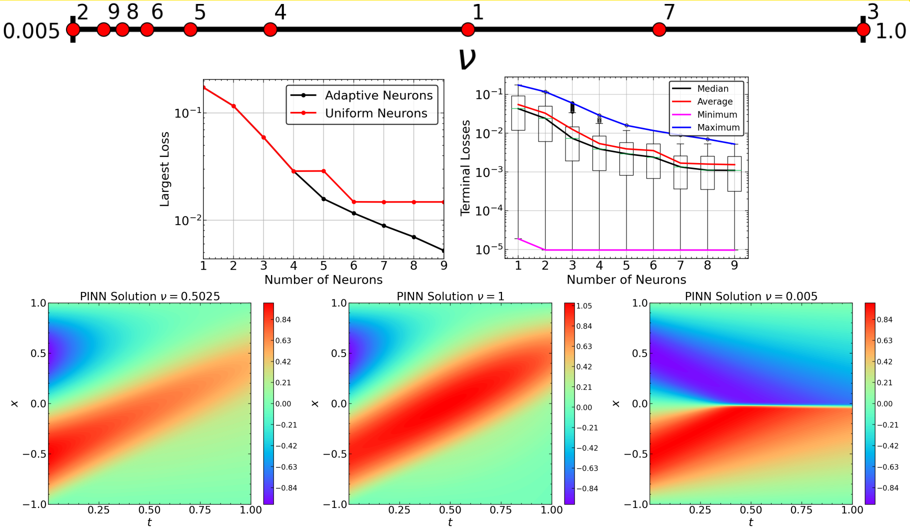
Test errors of the GPT-PINN of various sizes, and cumulative run time of the full PINN versus the GPT-PINN (slope ratio: 454).

GPT-PINN numerical results: KG losses & errors



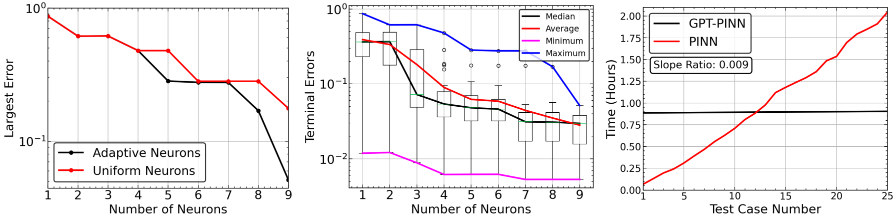
Full PINN training loss (Left) and GPT-PINN training loss (Right) as functions of epochs. Plotted in the middle are the point-wise errors of the corresponding GPT-PINN solution.

GPT-PINN numerical results: B training



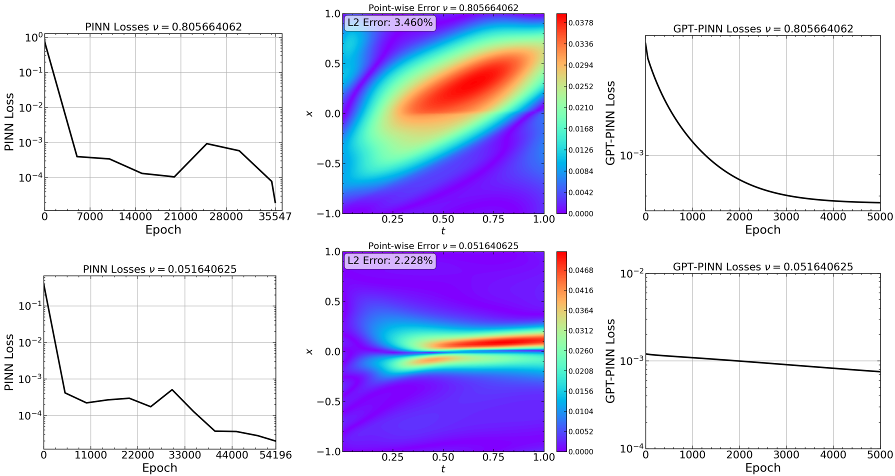
The adaptively chosen parameter values, GPT-PINN training losses, and the first three full PINN solutions GPT-PINN adopts as activation functions.

GPT-PINN numerical results: B testing



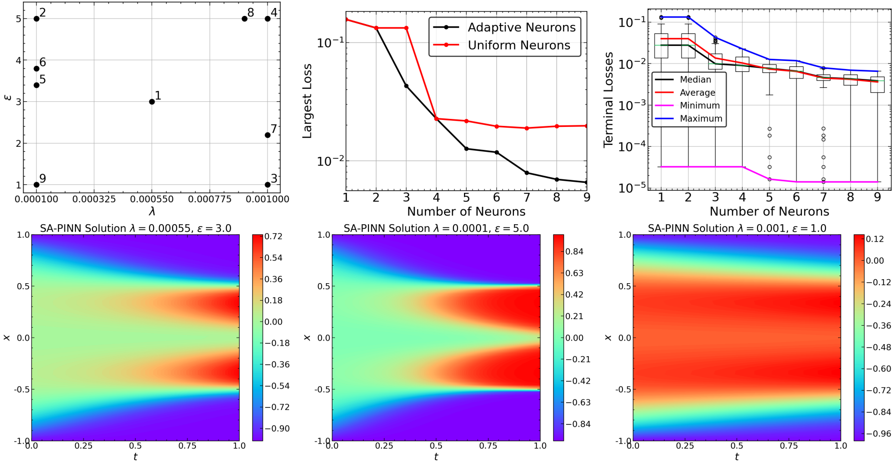
Test error of the GPT-PINN of various sizes and cumulative run time of the full PINN versus the GPT-PINN (slope ratio: 111).

GPT-PINN numerical results: B loses & errors



Full PINN training loss (Left) and GPT-PINN training loss (Right) as functions of epochs. Plotted in the middle are the point-wise errors of the corresponding GPT-PINN solution.

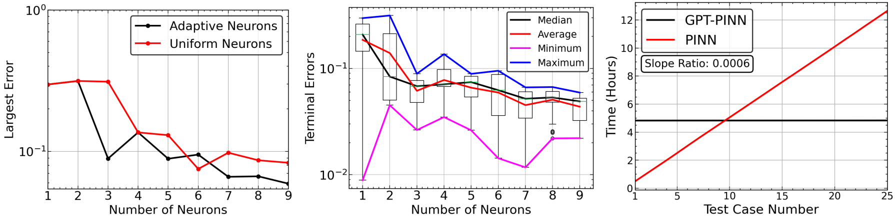
GPT-PINN numerical results: AC training



The chosen parameter values, GPT-PINN training losses, and the first three SA-PINN solutions GPT-PINN adopts as the activation functions.

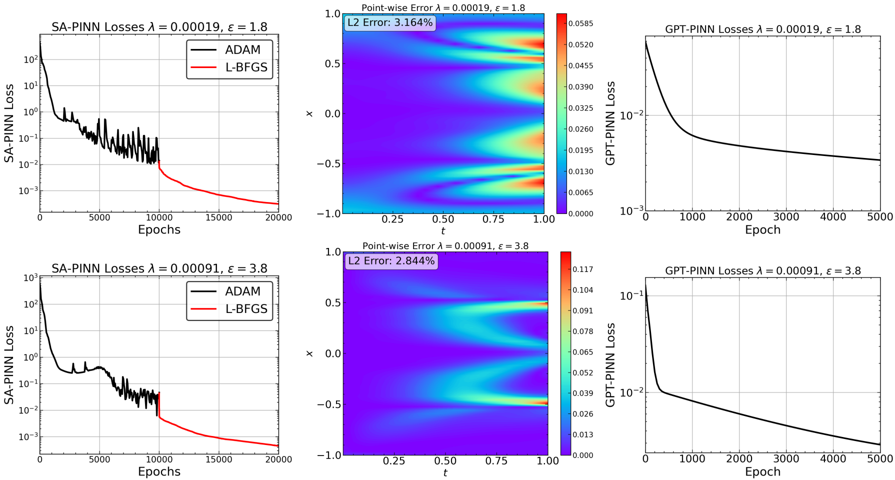
SA-PINN: McClenny, Braga-Neto, 2020.

GPT-PINN numerical results: AC testing



Test error of the GPT-PINN of various sizes, and cumulative run time of the full PINN versus the GPT-PINN (slope ratio: 1,667)

GPT-PINN numerical results: AC losses & errors



SA-PINN training loss (Left) and GPT-PINN training loss (Right) as functions of epochs. Plotted in the middle are the point-wise errors of the corresponding GPT-PINN solution.

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

✓ Preserves the PINN structure and physics-conforming property,

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

- ✓ Preserves the PINN structure and physics-conforming property,
- ✓ Relies on minimum amount of FOM data,

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

- ✓ Preserves the PINN structure and physics-conforming property,
- ✓ Relies on minimum amount of FOM data,
- ✓ Non-instrusive $\rightarrow$ portable,

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

- ✓ Preserves the PINN structure and physics-conforming property,
- ✓ Relies on minimum amount of FOM data,
- ✓ Non-instrusive $\rightarrow$ portable,
- ✓ Adopts a full PINN as a single neuron, and loss for guiding greedy construction,

Conclusion

The first PINN accelerator, the RBM framework for PINNs, that

- ✓ Preserves the PINN structure and physics-conforming property,
- ✓ Relies on minimum amount of FOM data,
- ✓ Non-instrusive $\rightarrow$ portable,
- ✓ Adopts a full PINN as a single neuron, and loss for guiding greedy construction,
- ✓ Naturally features an offline-online decomposition leading to practical speedups.

Conclusion

PAPER

CODE

Conclusion

PAPER

CODE

Yanlai.Chen @ umassd.edu