Hybrid finite element and neural network methods for parametric PDEs

Frédérique Lecourtier[1]

Supervised by: Michel Duprez[1], Emmanuel Franck[2] and Vanessa Lleras[3]

Joint work with: Hélène Barucq[4], Victor Michel-Dansac[2], Florian Faucher[4] and Nicolas Victorion[4]

[1]Project-Team MIMESIS, Inria, Strasbourg
[2]Project-Team MACARON, Inria, Strasbourg
[3]IMAG, University of Montpellier, Montpellier
[4]Project-Team MAKUTU, Inria, Pau

Context and motivation

Many-query contexts: real-time simulation, digital twins, optimization, uncertainty quantification

$\Rightarrow$ **Goal:** solve a parametric PDE accurately and quickly, for many parameters $\boldsymbol{\mu}$

Context and motivation

Many-query contexts: real-time simulation, digital twins, optimization, uncertainty quantification

$\Rightarrow$ **Goal:** solve a parametric PDE accurately and quickly, for many parameters $\boldsymbol{\mu}$

Classical methods (FEM, FV, FD, spectral)

guarantees	+ error analysis
single solve	+ efficient
geometry	— mesh generation
many queries	— one solve per $\boldsymbol{\mu}$
expressivity	— fixed spaces

Context and motivation

Many-query contexts: real-time simulation, digital twins, optimization, uncertainty quantification

$\Rightarrow$ **Goal:** solve a parametric PDE accurately and quickly, for many parameters $\boldsymbol{\mu}$

Classical methods (FEM, FV, FD, spectral)

guarantees	+ error analysis
single solve	+ efficient
geometry	— mesh generation
many queries	— one solve per $\boldsymbol{\mu}$
expressivity	— fixed spaces

Neural networks (PINN, neural operator)

guarantees	— none
single solve	— costly training
geometry	+ mesh-free
many queries	+ parametric
expressivity	+ flexible

Context and motivation

Many-query contexts: real-time simulation, digital twins, optimization, uncertainty quantification

$\Rightarrow$ **Goal:** solve a parametric PDE accurately and quickly, for many parameters $\boldsymbol{\mu}$

Classical methods (FEM, FV, FD, spectral)

guarantees	+ error analysis
single solve	+ efficient
geometry	— mesh generation
many queries	— one solve per $\boldsymbol{\mu}$
expressivity	— fixed spaces

Neural networks (PINN, neural operator)

guarantees	— none
single solve	— costly training
geometry	+ mesh-free
many queries	+ parametric
expressivity	+ flexible

Goal: classical guarantees + neural expressivity

Enriching the approximation space

Hand-crafted bases: GFEM [Strouboulis et al., 2001], XFEM [Moës et al., 1999], Trefftz [Moiola and Perugia, 2018]

Enriching the approximation space

Hand-crafted bases: GFEM [Strouboulis et al., 2001], XFEM [Moës et al., 1999], Trefftz [Moiola and Perugia, 2018]

$\Rightarrow$ problem-dependent

Enriching the approximation space

Hand-crafted bases: GFEM [Strouboulis et al., 2001], XFEM [Moës et al., 1999], Trefftz [Moiola and Perugia, 2018]

$\Rightarrow$ problem-dependent

Geometric prior: φ-FEM [Duprez and Lozinski, 2020], $u = \varphi\, w$

Enriching the approximation space

Hand-crafted bases: GFEM [Strouboulis et al., 2001], XFEM [Moës et al., 1999], Trefftz [Moiola and Perugia, 2018]

$\Rightarrow$ problem-dependent

Geometric prior: φ-FEM [Duprez and Lozinski, 2020], $u = \varphi\, w$

Neural prior: learned level-set [Wang et al., 2025], learned solution [Franck et al., 2024]

Enriching the approximation space

Hand-crafted bases: GFEM [Strouboulis et al., 2001], XFEM [Moës et al., 1999], Trefftz [Moiola and Perugia, 2018]

$\Rightarrow$ problem-dependent

Geometric prior: φ-FEM [Duprez and Lozinski, 2020], $u = \varphi\, w$

Neural prior: learned level-set [Wang et al., 2025], learned solution [Franck et al., 2024]

Ours: classical method + neural prior, two hybridizations

Two hybridizations

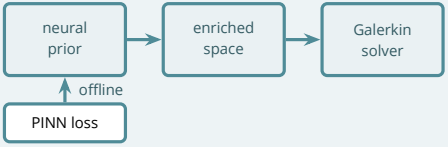
How does the neural network enter the approximation space of a classical method?

Two hybridizations

How does the neural network enter the approximation space of a classical method?

1st hybridization

Enriched FEM using PINNs



Network: pretrained, frozen

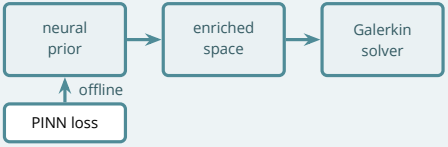
Guarantees: a priori error estimates

Two hybridizations

How does the neural network enter the approximation space of a classical method?

1st hybridization

Enriched FEM using PINNs

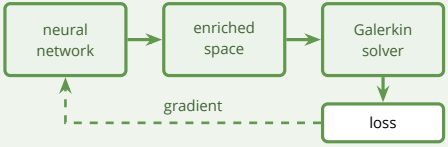


Network: pretrained, frozen

Guarantees: a priori error estimates

2nd hybridization

Learnable Galerkin methods



Network: learned through the solver

Guarantees: none (experimental)

Parametric elliptic PDEs

Let $\Omega \subset \mathbb{R}^d$ be the spatial domain ($d = 1, 2, 3$) and $\mathcal{M} \subset \mathbb{R}^p$ the parameter space.
For one or several $\boldsymbol{\mu} \in \mathcal{M}$, find $u(\,\cdot\,;\boldsymbol{\mu}) : \Omega \to \mathbb{R}$ such that

$$\begin{cases} \mathcal{L}\big(u\,;\boldsymbol{x},\boldsymbol{\mu}\big) = f(\boldsymbol{x},\boldsymbol{\mu}), & (\boldsymbol{x},\boldsymbol{\mu}) \in \Omega \times \mathcal{M}, \\ \text{Dirichlet, Neumann or Robin BC}, & (\boldsymbol{x},\boldsymbol{\mu}) \in \Gamma \times \mathcal{M}, \end{cases} \tag{$\mathcal{P}$}$$

with $\Gamma := \partial\Omega$ the boundary of Ω, f the source term and $\mathcal{L}$ the linear differential operator defined by

$$\mathcal{L}(\cdot\,;\boldsymbol{x},\boldsymbol{\mu}) : u \mapsto R(\boldsymbol{x},\boldsymbol{\mu})\,u + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\mu}) \cdot \nabla u - \frac{1}{\mathrm{Pe}} \nabla \cdot \big(D(\boldsymbol{x},\boldsymbol{\mu}) \nabla u\big).$$

where R is the reaction, $\boldsymbol{C}$ the convection, D the (symmetric positive definite) diffusion matrix and Pe the Péclet number.

Numerical methods

Numerical methods for parametric PDEs*

* presented with homogeneous Dirichlet boundary conditions

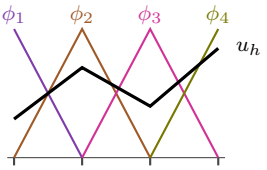
Continuous Galerkin (CG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω.

Let V_h be a continuous space of piecewise polynomials of degree k on $\mathcal{T}_h$. Its basis functions $(\phi_i)_i$, with ϕ_i supported on the cells sharing the dof i, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{i=1}^{N_{\text{dofs}}} \beta_i(\boldsymbol{\mu})\, \phi_i(\boldsymbol{x}).$$

$N_{\text{cells}} = 3$, $N_{\text{dofs}} = 4$ $(k = 1)$



[1] [Ern and Guermond, 2004]

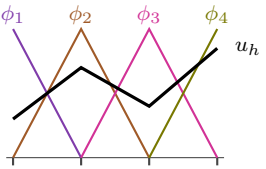
Continuous Galerkin (CG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω.

Let V_h be a continuous space of piecewise polynomials of degree k on $\mathcal{T}_h$. Its basis functions $(\phi_i)_i$, with ϕ_i supported on the cells sharing the dof i, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{i=1}^{N_{\text{dofs}}} \beta_i(\boldsymbol{\mu})\, \phi_i(\boldsymbol{x}).$$

$N_{\text{cells}} = 3$, $N_{\text{dofs}} = 4$ $(k = 1)$



Variational problem: Find $u_h \in V_h^0$ such that, $\forall v_h \in V_h^0$, $a(u_h, v_h; \boldsymbol{\mu}) = l(v_h; \boldsymbol{\mu}) = \int_\Omega f\, v_h$,

$$a(u_h, v_h; \boldsymbol{\mu}) = \int_\Omega \left(R\, u_h v_h + \boldsymbol{C} \cdot \nabla u_h\, v_h + \frac{1}{\text{Pe}} D \nabla u_h \cdot \nabla v_h \right)$$

with $V_h^0 \subset V_h$ vanishing on Γ (homogeneous Dirichlet condition).

[1] [Ern and Guermond, 2004]

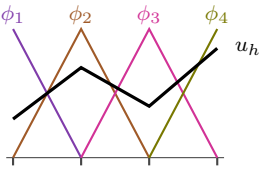
Continuous Galerkin (CG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω.

Let V_h be a continuous space of piecewise polynomials of degree k on $\mathcal{T}_h$. Its basis functions $(\phi_i)_i$, with ϕ_i supported on the cells sharing the dof i, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{i=1}^{N_{\text{dofs}}} \beta_i(\boldsymbol{\mu})\, \phi_i(\boldsymbol{x}).$$

$N_{\text{cells}} = 3$, $N_{\text{dofs}} = 4$ $(k = 1)$



Variational problem: Find $u_h \in V_h^0$ such that, $\forall v_h \in V_h^0$, $a(u_h, v_h; \boldsymbol{\mu}) = l(v_h; \boldsymbol{\mu}) = \int_\Omega f\, v_h$,

$$a(u_h, v_h; \boldsymbol{\mu}) = \int_\Omega \left(R\, u_h v_h + \boldsymbol{C} \cdot \nabla u_h\, v_h + \frac{1}{\text{Pe}} D \nabla u_h \cdot \nabla v_h \right)$$

with $V_h^0 \subset V_h$ vanishing on Γ (homogeneous Dirichlet condition).

Linear system:

$$\underset{\text{matrix}}{A_h(\boldsymbol{\mu})}\ \underset{\text{dofs}}{\boldsymbol{\beta}_h(\boldsymbol{\mu})} = \underset{\text{right-hand side}}{\boldsymbol{b}_h(\boldsymbol{\mu})}, \qquad A_{ij} = a(\phi_j, \phi_i; \boldsymbol{\mu}), \quad b_i = l(\phi_i; \boldsymbol{\mu})$$

[1] [Ern and Guermond, 2004]

Physics-informed neural networks (PINNs)[1]

Let $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ be a neural network with weights θ and inputs $(\boldsymbol{x}, \boldsymbol{\mu}) \in \Omega \times \mathcal{M}$, approximating the solution u.

Training: Find the weights $\theta^\star$ such that

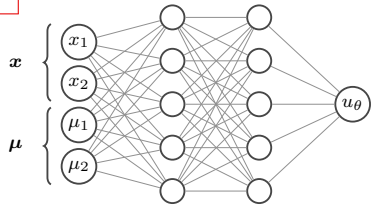
$$\theta^\star = \arg\min_\theta \big(\omega_r \, \mathcal{J}_r(\theta) + \omega_b \, \mathcal{J}_b(\theta)\big), \qquad (\mathcal{P}_\theta)$$

with ω_r and ω_b the weights of the two loss terms, and

residual loss
$$\mathcal{J}_r(\theta) = \int_{\mathcal{M}} \int_{\Omega} \big|\mathcal{L}\big(u_\theta(\boldsymbol{x}, \boldsymbol{\mu}); \boldsymbol{x}, \boldsymbol{\mu}\big) - f(\boldsymbol{x}, \boldsymbol{\mu})\big|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{\mu},$$

boundary loss
$$\mathcal{J}_b(\theta) = \int_{\mathcal{M}} \int_{\Gamma} \big|u_\theta(\boldsymbol{x}, \boldsymbol{\mu})\big|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{\mu}.$$

Monte-Carlo method:
Discretize the cost functions by random process.



[1] [Raissi, Perdikaris, and Karniadakis, 2019]

Physics-informed neural networks (PINNs)[1]

Let $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ be a neural network with weights θ and inputs $(\boldsymbol{x}, \boldsymbol{\mu}) \in \Omega \times \mathcal{M}$, approximating the solution u.

Training: Find the weights $\theta^\star$ such that

$$\theta^\star = \arg\min_\theta \big(\omega_r \, \mathcal{J}_r(\theta) + \omega_b \, \mathcal{J}_b(\theta)\big), \qquad (\mathcal{P}_\theta)$$

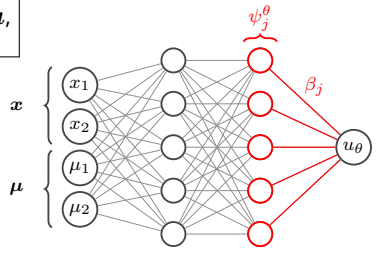
with ω_r and ω_b the weights of the two loss terms, and

residual loss $$\mathcal{J}_r(\theta) = \int_{\mathcal{M}} \int_{\Omega} \big|\mathcal{L}\big(u_\theta(\boldsymbol{x}, \boldsymbol{\mu}); \boldsymbol{x}, \boldsymbol{\mu}\big) - f(\boldsymbol{x}, \boldsymbol{\mu})\big|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{\mu},$$

boundary loss $$\mathcal{J}_b(\theta) = \int_{\mathcal{M}} \int_{\Gamma} \big|u_\theta(\boldsymbol{x}, \boldsymbol{\mu})\big|^2 \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{\mu}.$$

Monte-Carlo method:
Discretize the cost functions by random process.

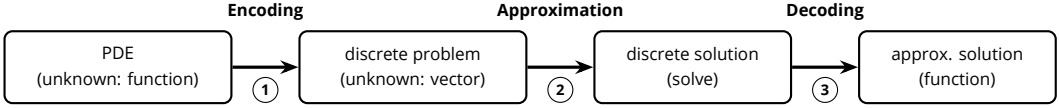
$$u_\theta(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_j \beta_j \, \psi_j^\theta$$



[1] [Raissi, Perdikaris, and Karniadakis, 2019]

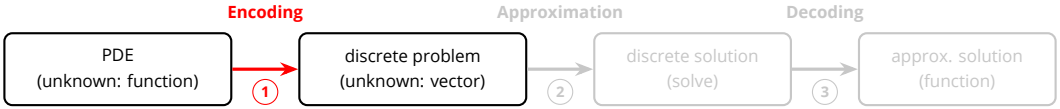
Link between Galerkin and PINNs

Numerical method (three steps):



Link between Galerkin and PINNs

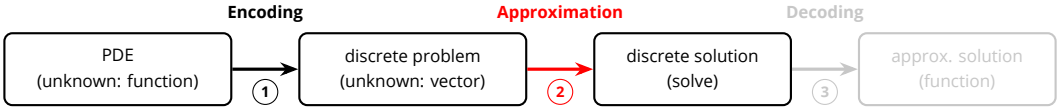
Numerical method (three steps):



		Galerkin (CG)	PINN
	Space	vector space V_h	manifold of the u_θ
①	Unknowns	dofs $\boldsymbol{\beta}_h(\boldsymbol{\mu})$	weights θ

Link between Galerkin and PINNs

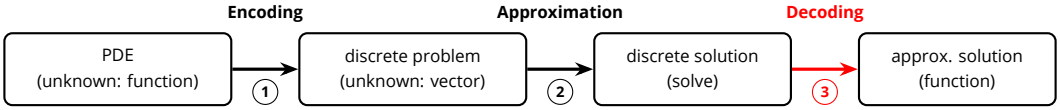
Numerical method (three steps):



		Galerkin (CG)	PINN
①	Space	vector space V_h	manifold of the u_θ
	Unknowns	dofs $\boldsymbol{\beta}_h(\boldsymbol{\mu})$	weights θ
②	Parameters	fixed $\boldsymbol{\mu}$, one solve per $\boldsymbol{\mu}$	$\boldsymbol{\mu} \in \mathcal{M}$ as input, one training
	Resolution	linear system $A_h(\boldsymbol{\mu})\boldsymbol{\beta}_h(\boldsymbol{\mu}) = \boldsymbol{b}_h(\boldsymbol{\mu})$	optimization on θ

Link between Galerkin and PINNs

Numerical method (three steps):

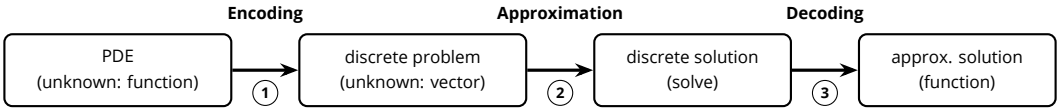


		Galerkin (CG)	PINN
①	Space	vector space V_h	manifold of the u_θ
	Unknowns	dofs $\boldsymbol{\beta}_h(\boldsymbol{\mu})$	weights θ
②	Parameters	fixed $\boldsymbol{\mu}$, one solve per $\boldsymbol{\mu}$	$\boldsymbol{\mu} \in \mathcal{M}$ as input, one training
	Resolution	linear system $A_h(\boldsymbol{\mu})\boldsymbol{\beta}_h(\boldsymbol{\mu}) = \boldsymbol{b}_h(\boldsymbol{\mu})$	optimization on θ
③	Solution	$u_h(\boldsymbol{x}, \boldsymbol{\mu})$, linear in $\boldsymbol{\beta}_h$	$u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$, non-linear in θ

Link between Galerkin and PINNs

Numerical method (three steps):

Same philosophy, different approximation space



		Galerkin (CG)	PINN
①	Space	vector space V_h	manifold of the u_θ
	Unknowns	dofs $\boldsymbol{\beta}_h(\boldsymbol{\mu})$	weights θ
②	Parameters	fixed $\boldsymbol{\mu}$, one solve per $\boldsymbol{\mu}$	$\boldsymbol{\mu} \in \mathcal{M}$ as input, one training
	Resolution	linear system $A_h(\boldsymbol{\mu})\boldsymbol{\beta}_h(\boldsymbol{\mu}) = \boldsymbol{b}_h(\boldsymbol{\mu})$	optimization on θ
③	Solution	$u_h(\boldsymbol{x}, \boldsymbol{\mu})$, linear in $\boldsymbol{\beta}_h$	$u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$, non-linear in θ

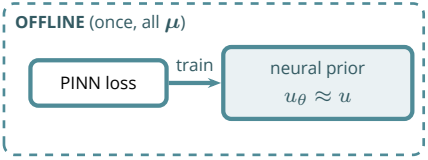
1st hybridization

Enriched finite element method using PINNs

Presentation*

* presented with homogeneous Dirichlet boundary conditions

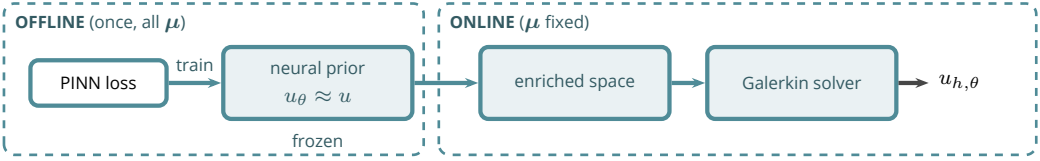
Pipeline of the hybrid method[1]



Offline: train a parametric PINN $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ once, for all $\boldsymbol{\mu} \in \mathcal{M}$.

[1] [Lecourtier et al., 2026]

Pipeline of the hybrid method[1]

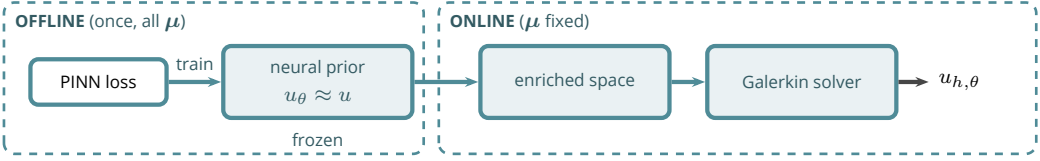


Offline: train a parametric PINN $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ once, for all $\boldsymbol{\mu} \in \mathcal{M}$.

Online (μ fixed): evaluate the prior and its derivatives, solve the enriched FE problem.

[1] [Lecourtier et al., 2026]

Pipeline of the hybrid method[1]



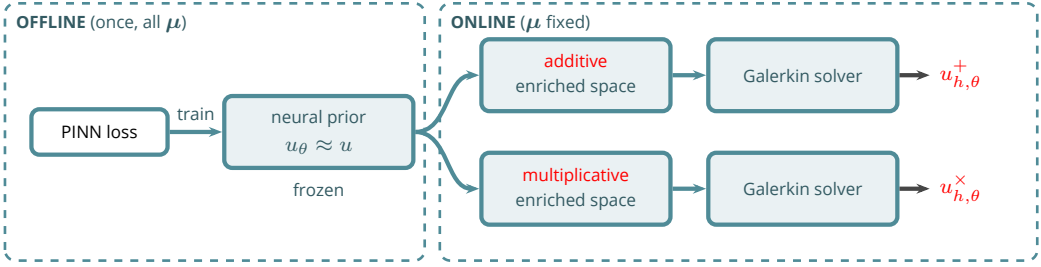
Offline: train a parametric PINN $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ once, for all $\boldsymbol{\mu} \in \mathcal{M}$.

Online (μ fixed): evaluate the prior and its derivatives, solve the enriched FE problem.

Goal: reach the accuracy of a fine mesh with a coarser mesh.

[1] [Lecourtier et al., 2026]

Pipeline of the hybrid method[1]



Offline: train a parametric PINN $u_\theta(\boldsymbol{x}, \boldsymbol{\mu})$ once, for all $\boldsymbol{\mu} \in \mathcal{M}$.

Online (μ fixed): evaluate the prior and its derivatives, solve the enriched FE problem.

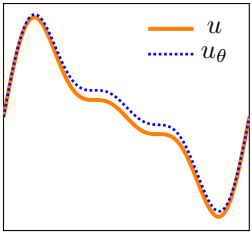
Goal: reach the accuracy of a fine mesh with a coarser mesh.

Two versions of $u_{h,\theta}$: additive $u^+_{h,\theta}$ and multiplicative $u^\times_{h,\theta}$.

[1] [Lecourtier et al., 2026]

Additive approach[1]

Prior: Let $u_\theta \in H^{k+1}(\Omega)$ be the PINN prediction ($\boldsymbol{\mu}$ fixed).



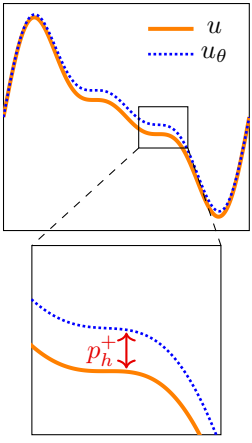
[1] [Lecourtier et al., 2026]

Additive approach[1]

Prior: Let $u_\theta \in H^{k+1}(\Omega)$ be the PINN prediction ($\boldsymbol{\mu}$ fixed).

Enriched trial space:

$$V_h^+ = \left\{ u_{h,\theta}^+ = u_\theta + p_h^+, \quad p_h^+ \in V_h^0 \right\}$$



[1] [Lecourtier et al., 2026]

Additive approach[1]

Prior: Let $u_\theta \in H^{k+1}(\Omega)$ be the PINN prediction ($\boldsymbol{\mu}$ fixed).

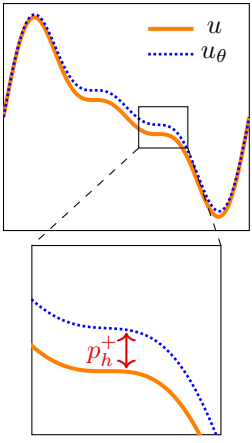
Enriched trial space:

$$V_h^+ = \left\{ u_{h,\theta}^+ = u_\theta + p_h^+, \quad p_h^+ \in V_h^0 \right\}$$

Variational problem: Find $p_h^+ \in V_h^0$ such that, $\forall v_h \in V_h^0$,

$$a(p_h^+, v_h) = l(v_h) - a(u_\theta, v_h) \qquad (\mathcal{P}_h^+)$$

with modified Dirichlet BC.



[1] [Lecourtier et al., 2026]

Multiplicative approach[1]

Enriched trial space:

$$V_h^{\times} = \left\{ u_{h,\theta}^{\times} = u_\theta \quad p_h^{\times} \qquad , \qquad \right\}$$

[1] [Lecourtier et al., 2026]

Multiplicative approach[1]

Enriched trial space:

$$V_h^\times = \left\{ u_{h,\theta}^\times = u_\theta \quad p_h^\times \qquad , \qquad \right\}$$

Lifted problem: Need a prior u_θ that never vanishes.

Let $M \in \mathbb{R}_+$ be a constant such that $u_{\theta,M} = u_\theta + M > 0$. Then $u_M = u + M$ solves

$$\begin{cases} \mathcal{L}(u_M) = f & \text{in } \Omega, \\ u_M = M & \text{on } \Gamma, \end{cases} \qquad \text{so that } u = u_M - M.$$

[1] [Lecourtier et al., 2026]

Multiplicative approach[1]

Enriched trial space:

$$V_h^\times = \left\{ u_{h,\theta}^\times = u_{\theta,M}\, p_h^\times - M, \quad p_h^\times \in 1 + V_h^0 \right\}$$

Lifted problem: Need a prior u_θ that never vanishes.

Let $M \in \mathbb{R}_+$ be a constant such that $u_{\theta,M} = u_\theta + M > 0$. Then $u_M = u + M$ solves

$$\begin{cases} \mathcal{L}(u_M) = f & \text{in } \Omega, \\ u_M = M & \text{on } \Gamma, \end{cases} \qquad \text{so that } u = u_M - M.$$

[1] [Lecourtier et al., 2026]

Multiplicative approach[1]

Enriched trial space:

$$V_h^\times = \left\{ u_{h,\theta}^\times = u_{\theta,M}\, p_h^\times - M, \quad p_h^\times \in 1 + V_h^0 \right\}$$

Lifted problem: Need a prior u_θ that never vanishes.

Let $M \in \mathbb{R}_+$ be a constant such that $u_{\theta,M} = u_\theta + M > 0$. Then $u_M = u + M$ solves

$$\begin{cases} \mathcal{L}(u_M) = f & \text{in } \Omega, \\ u_M = M & \text{on } \Gamma, \end{cases} \qquad \text{so that } u = u_M - M.$$

Variational problem: Find $p_h^\times \in 1 + V_h^0$ such that, $\forall v_h \in V_h^0$,

$$a\big(u_{\theta,M}\, p_h^\times, u_{\theta,M}\, v_h\big) = l\big(u_{\theta,M}\, v_h\big) \qquad (\mathcal{P}_h^\times)$$

with modified Dirichlet BC.

[1] [Lecourtier et al., 2026]

Convergence analysis[1]

Theorem 1: Standard FEM [Ern and Guermond, 2004]

Let $u_h \in V_h^0$ be the solution of the standard FEM. Then,

$$\|u - u_h\|_{L^2} \leqslant C_{L^2}\, h^{k+1} |u|_{H^{k+1}}.$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Convergence analysis[1]

Theorem 1: Standard FEM [Ern and Guermond, 2004]

Let $u_h \in V_h^0$ be the solution of the standard FEM. Then,

$$\|u - u_h\|_{L^2} \leqslant C_{L^2}\, h^{k+1} |u|_{H^{k+1}}.$$

Theorem 2: Additive approach [Lecourtier et al., 2026]

Let $u_{h,\theta}^+ \in V_h^+$ be the solution of $(\mathcal{P}_h^+)$ with V_h^+ the enriched trial space. Then,

$$\|u - u_{h,\theta}^+\|_{L^2} \leqslant \boxed{C_{\text{gain}}^+} \left(C_{L^2}\, h^{k+1} |u|_{H^{k+1}}\right).$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Convergence analysis[1]

Theorem 1: Standard FEM [Ern and Guermond, 2004]

Let $u_h \in V_h^0$ be the solution of the standard FEM. Then,

$$\|u - u_h\|_{L^2} \leqslant C_{L^2}\, h^{k+1} |u|_{H^{k+1}}.$$

Theorem 2: Additive approach [Lecourtier et al., 2026]

Let $u_{h,\theta}^+ \in V_h^+$ be the solution of $(\mathcal{P}_h^+)$ with V_h^+ the enriched trial space. Then,

$$\|u - u_{h,\theta}^+\|_{L^2} \leqslant \boxed{C_{\text{gain}}^+} \left(C_{L^2}\, h^{k+1} |u|_{H^{k+1}}\right).$$

$$\boxed{C_{\text{gain}}^+ = \frac{|u - u_\theta|_{H^{k+1}}}{|u|_{H^{k+1}}}} < 1 \;\Rightarrow\; \textbf{gain over the standard FEM}$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Convergence analysis[1]

Theorem 1: Standard FEM [Ern and Guermond, 2004]

Let $u_h \in V_h^0$ be the solution of the standard FEM. Then,

$$\|u - u_h\|_{L^2} \leqslant C_{L^2}\, h^{k+1} |u|_{H^{k+1}}.$$

Theorem 3: Multiplicative approach [Lecourtier et al., 2026]

Let $u_{h,\theta}^{\times} \in V_h^{\times}$ be the solution of $(\mathcal{P}_h^{\times})$ with $V_h^{\times}$ the enriched trial space. Then,

$$\|u - u_{h,\theta}^{\times}\|_{L^2} \leqslant \boxed{C_{\text{gain}}^{\times,M}} \left(C_{L^2}\, h^{k+1} |u|_{H^{k+1}}\right).$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Convergence analysis[1]

Theorem 1: Standard FEM [Ern and Guermond, 2004]

Let $u_h \in V_h^0$ be the solution of the standard FEM. Then,

$$\|u - u_h\|_{L^2} \leqslant C_{L^2}\, h^{k+1} |u|_{H^{k+1}}.$$

Theorem 3: Multiplicative approach [Lecourtier et al., 2026]

Let $u_{h,\theta}^{\times} \in V_h^{\times}$ be the solution of $(\mathcal{P}_h^{\times})$ with $V_h^{\times}$ the enriched trial space. Then,

$$\|u - u_{h,\theta}^{\times}\|_{L^2} \leqslant \boxed{C_{\text{gain}}^{\times,M}} \left(C_{L^2}\, h^{k+1} |u|_{H^{k+1}}\right).$$

$$\boxed{C_{\text{gain}}^{\times,M} = C_{\theta,M} \left| \frac{u_M}{u_{\theta,M}} \right|_{H^{k+1}} \frac{\|u_{\theta,M}\|_{W^{1,\infty}}^2}{|u|_{H^{k+1}}}} < 1 \;\Rightarrow\; \textbf{gain over the standard FEM}$$

with $C_{\theta,M} = \|u_{\theta,M}^{-1}\|_{L^\infty} + 2|u_{\theta,M}^{-1}|_{W^{1,\infty}} + |u_{\theta,M}^{-1}|_{W^{2,\infty}}$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Multiplicative versus additive[1]

Theorem 4: Comparison of the gains [Lecourtier et al., 2026]

The multiplicative gain converges to the additive gain when $M \to \infty$:

$$C_{\text{gain}}^{\times,M} \underset{M\to\infty}{\longrightarrow} C_{\text{gain}}^{+}$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Multiplicative versus additive[1]

$M \to \infty$: **multiplicative approach $\to$ additive approach**

Theorem 4: Comparison of the gains [Lecourtier et al., 2026]

The multiplicative gain converges to the additive gain when $M \to \infty$:

$$C_{\text{gain}}^{\times,M} \underset{M\to\infty}{\longrightarrow} C_{\text{gain}}^{+}$$

[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Multiplicative versus additive[1]

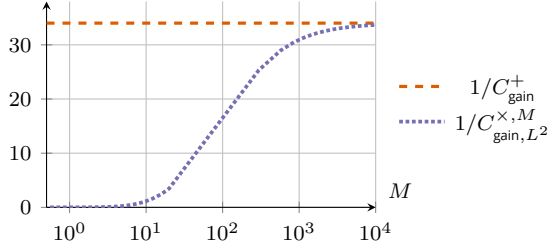
$M \to \infty$: **multiplicative approach $\to$ additive approach**

Theorem 4: Comparison of the gains [Lecourtier et al., 2026]

The multiplicative gain converges to the additive gain when $M \to \infty$:

$$C^{\times,M}_{\text{gain}} \underset{M\to\infty}{\longrightarrow} C^{+}_{\text{gain}}$$

Gain constants as a function of M :
1D Poisson, $\boldsymbol{\mu} = (0.3, 0.2, 0.1)$, $\mathbb{P}_1$



[1] Stated in L^2 norm; similar result in H^1 semi-norm.

Numerical results

1D Poisson
2D Poisson
3D Poisson

Numerical results

1D Poisson
2D Poisson
3D Poisson

Problem considered

Problem statement: Consider the Poisson problem in 1D with Dirichlet BC:

$$\begin{cases} -\partial_{xx} u = f, & \text{in } \Omega \times \mathcal{M}, \\ u = 0, & \text{on } \partial\Omega \times \mathcal{M}, \end{cases}$$

with $\Omega = (0,1)$ and $\mathcal{M} = [0,1]^3$ ($p = 3$ parameters).

Analytical solution: $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3) \in \mathcal{M}$

$$u(x, \boldsymbol{\mu}) = \mu_1 \sin(2\pi x) + \mu_2 \sin(4\pi x) + \mu_3 \sin(6\pi x).$$

Parametric PINN training:
MLP 6 layers (sine - 20, 80, 80, 80, 20, 10); Adam optimizer (10000 epochs - 5000 col. pts).
Imposing the Dirichlet BC exactly in the PINN with the levelset φ defined by

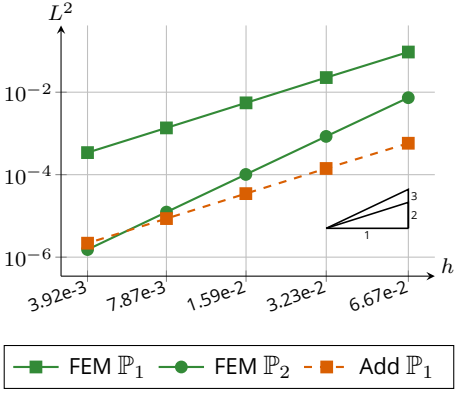
$$\varphi(x) = x(x-1).$$

Enriched FEM: $\mathbb{P}_1$, additive and multiplicative ($M = 3$ and $M = 100$).

Online phase - 1 parameters instance

$$\boldsymbol{\mu} = (0.8, 0.5, 0.8)$$

Error estimates in L^2 norm:



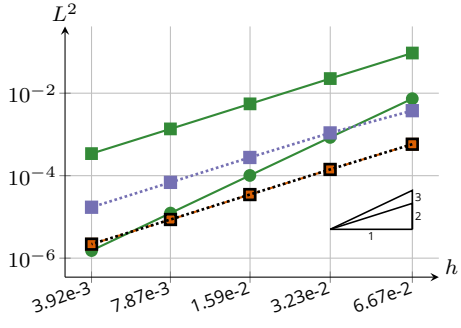
Add $\mathbb{P}_1$: order 2

error ÷ 160 vs. FEM $\mathbb{P}_1$

Online phase - 1 parameters instance

$$\boldsymbol{\mu} = (0.8, 0.5, 0.8)$$

Error estimates in L^2 norm:



Add $\mathbb{P}_1$: order 2

error ÷ 160 vs. FEM $\mathbb{P}_1$

Mult $\mathbb{P}_1$: order 2

$M = 3$: worse than Add

$M = 100 \approx$ Add

Numerical results

1D Poisson
2D Poisson
3D Poisson

Problem considered

Problem statement: Consider the Poisson problem in 2D with Dirichlet BC:

$$\begin{cases} -\Delta u = f, & \text{in } \Omega \times \mathcal{M}, \\ u = 0, & \text{on } \partial\Omega \times \mathcal{M}, \end{cases}$$

with $\Omega = (-0.5\pi, 0.5\pi)^2$ and $\mathcal{M} = [-0.5, 0.5]^2$ ($p = 2$ parameters).

Analytical solution: $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{M}$

$$u(\boldsymbol{x}, \boldsymbol{\mu}) = \exp\left(-\frac{(x-\mu_1)^2 + (y-\mu_2)^2}{2}\right) \sin(2x)\sin(2y).$$

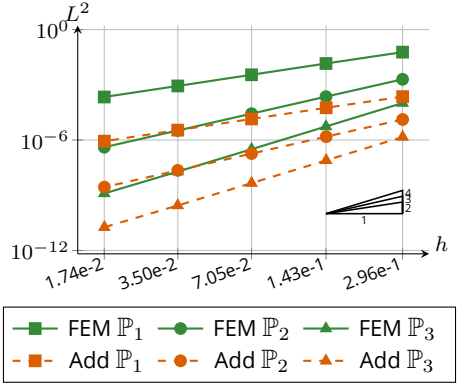
Parametric PINN training:
MLP 5 layers (sine - 40, 60, 60, 60, 40); Adam then LBFGS optimizer (5000 epochs - 6000 col. pts).
Imposing the Dirichlet BC exactly in the PINN with the levelset φ defined by

$$\varphi(\boldsymbol{x}) = (x + 0.5\pi)(x - 0.5\pi)(y + 0.5\pi)(y - 0.5\pi).$$

Online phase - 1 parameters instance

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22)$$

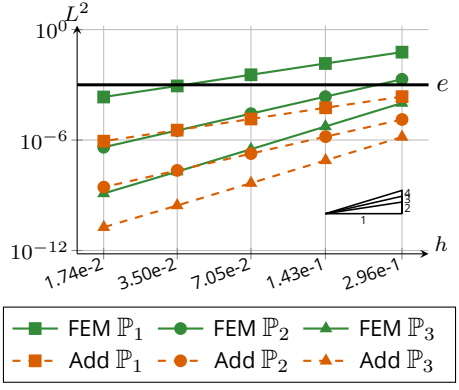
Error estimates in L^2 norm:



Online phase - 1 parameters instance

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22)$$

Error estimates in L^2 norm:



N_{dofs} required to reach the same error e:

k	e	N_{dofs} FEM	N_{dofs} Add
1	$1 \cdot 10^{-3}$	14 161	64
	$1 \cdot 10^{-4}$	143 641	576
2	$1 \cdot 10^{-4}$	6 889	225
	$1 \cdot 10^{-5}$	31 329	1 089
3	$1 \cdot 10^{-5}$	6 724	784
	$1 \cdot 10^{-6}$	20 164	2 704

$$h \approx N_{\text{dofs}}^{-1/2}$$

Numerical results

1D Poisson
2D Poisson
3D Poisson

Problem considered

Problem statement: Consider the Poisson problem in 3D with Dirichlet BC:

$$\begin{cases} -\Delta u = f, & \text{in } \Omega \times \mathcal{M}, \\ u = 0, & \text{on } \partial\Omega \times \mathcal{M}, \end{cases}$$

with $\Omega = (-0.5\pi, 0.5\pi)^3$ and $\mathcal{M} = [-0.5, 0.5]^3$ ($p = 3$ parameters).

Analytical solution: $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3) \in \mathcal{M}$

$$u(\boldsymbol{x}, \boldsymbol{\mu}) = \exp\left(-\frac{(x-\mu_1)^2 + (y-\mu_2)^2 + (z-\mu_3)^2}{2}\right) \sin(2x)\sin(2y)\sin(2z).$$

Parametric PINN training:

MLP 5 layers (tanh - 40, 60, 60, 60, 40); Adam then LBFGS optimizer (5000 epochs - 40000 col. pts).

Imposing the Dirichlet BC exactly in the PINN with the levelset φ defined by

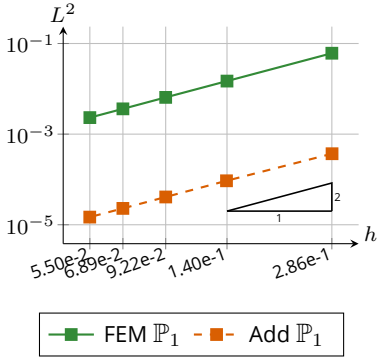
$$\varphi(\boldsymbol{x}) = (x + 0.5\pi)(x - 0.5\pi)(y + 0.5\pi)(y - 0.5\pi)(z + 0.5\pi)(z - 0.5\pi).$$

Training time: 12 minutes on NVIDIA RTX 2000 (8GB VRAM).

Online phase - 1 parameters instance I

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22, 0.1)$$

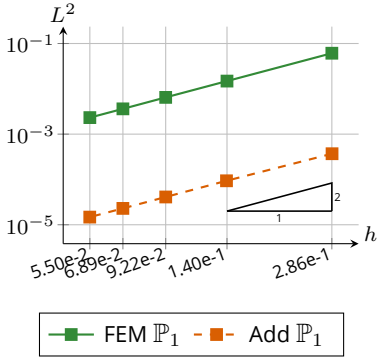
Error estimates in L^2 norm:



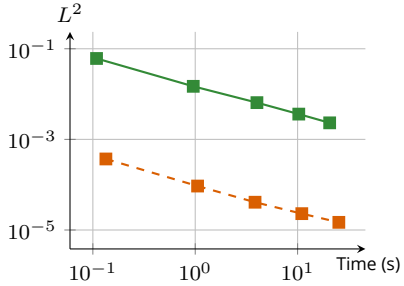
Online phase - 1 parameters instance I

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22, 0.1)$$

Error estimates in L^2 norm:



L^2 error w.r.t. online time*:

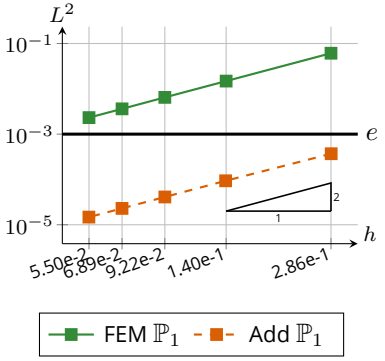


* mesh + assembly + solve
+ prior derivatives evaluation
~~+ training time~~

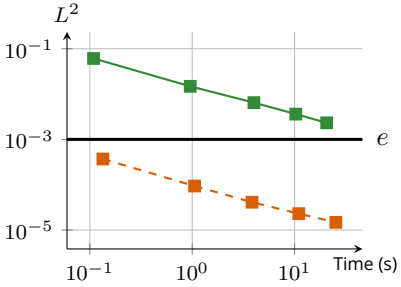
Online phase - 1 parameters instance I

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22, 0.1)$$

Error estimates in L^2 norm:



L^2 error w.r.t. online time*:



* mesh + assembly + solve
+ prior derivatives evaluation
~~+ training time~~

Online phase - 1 parameters instance II

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22, 0.1)$$

N_{dofs} and online time required to reach the same error e:

	N		N_{dofs}		Online time*	
e	**FEM**	**Add**	**FEM**	**Add**	**FEM**	**Add**
$1 \cdot 10^{-3}$	152	12	$3.51 \cdot 10^{6}$	$1.73 \cdot 10^{3}$	$(7.65 \cdot 10^{1})$	$5.2 \cdot 10^{-2}$
$1 \cdot 10^{-4}$	484	39	$1.13 \cdot 10^{8}$	$5.93 \cdot 10^{4}$	$(2.8 \cdot 10^{3})$	$9.26 \cdot 10^{-1}$
$1 \cdot 10^{-5}$	1 539	122	$3.65 \cdot 10^{9}$	$1.82 \cdot 10^{6}$	$(1.02 \cdot 10^{5})$	$(5.26 \cdot 10^{1})$

$\div 1471$

Cartesian mesh: $N_{\text{dofs}} = N^3$ nodes ($\mathbb{P}_1$ elements).

* The results in brackets are extrapolations.

Online phase - 1 parameters instance II

$$\boldsymbol{\mu}^{(1)} = (0.05, 0.22, 0.1)$$

N_{dofs} and online time required to reach the same error e:

	N		N_{dofs}		Online time*	
e	**FEM**	**Add**	**FEM**	**Add**	**FEM**	**Add**
$1 \cdot 10^{-3}$	152	12	$3.51 \cdot 10^{6}$	$1.73 \cdot 10^{3}$	$(7.65 \cdot 10^{1})$	$5.2 \cdot 10^{-2}$
$1 \cdot 10^{-4}$	484	39	$1.13 \cdot 10^{8}$	$5.93 \cdot 10^{4}$	$(2.8 \cdot 10^{3})$	$9.26 \cdot 10^{-1}$
$1 \cdot 10^{-5}$	1 539	122	$3.65 \cdot 10^{9}$	$1.82 \cdot 10^{6}$	$(1.02 \cdot 10^{5})$	$(5.26 \cdot 10^{1})$

$\div 1471$

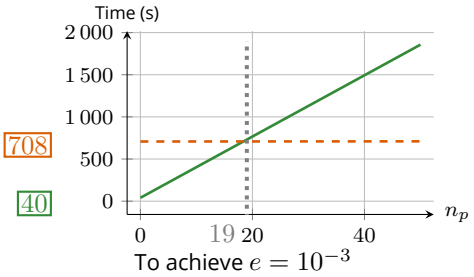
Cartesian mesh: $N_{\text{dofs}} = N^3$ nodes ($\mathbb{P}_1$ elements).

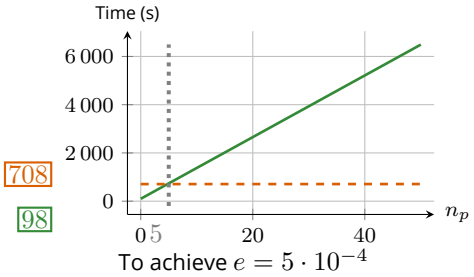
* The results in brackets are extrapolations.

⚠ Training time !

Parametric framework

How many parameters instance n_p are needed to balance training?





FEM $\mathbb{P}_1$ Add $\mathbb{P}_1$ Offline cost

Online: assembly + solve + prior derivatives evaluation

Offline: mesh + training time

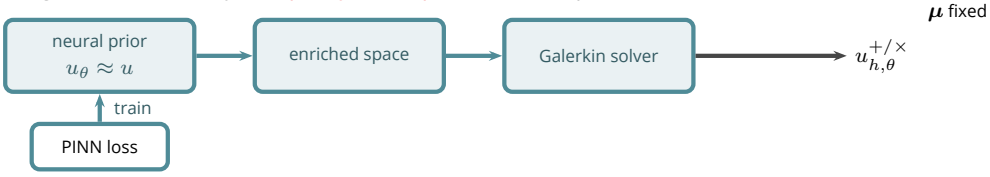
2nd hybridization

Learnable Galerkin methods

Introduction

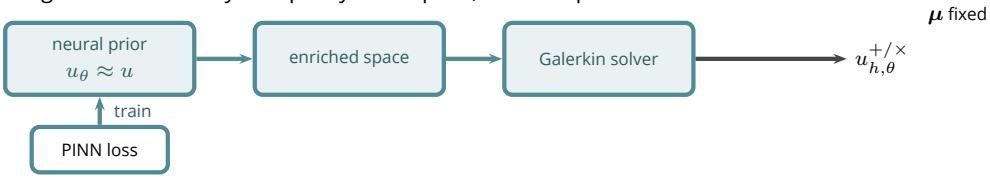
From frozen prior to learned basis

1st hybridization: neural prior trained offline, then frozen: a priori error estimates.
Limit: the gain is bounded by the quality of the prior, trained apart from the scheme.

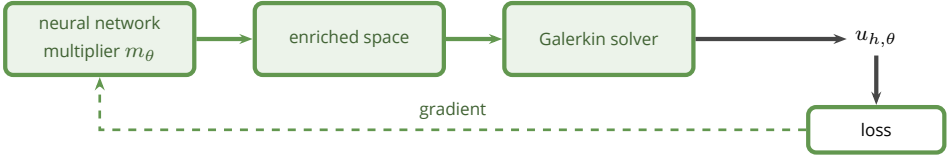


From frozen prior to learned basis

1st hybridization: neural prior trained offline, then frozen: a priori error estimates.
Limit: the gain is bounded by the quality of the prior, trained apart from the scheme.



2nd hybridization: learn the basis through the solver, adapted to the solution of the scheme.
Advantage: basis adapted to the scheme, not bounded by a frozen prior.
Limit: no a priori guarantees, training cost.



Non-parametric enrichment (PINN loss)*

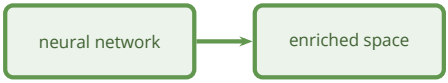
* $\boldsymbol{\mu}$ is fixed and omitted from the notation: basis, dofs and solution are valid for this $\boldsymbol{\mu}$ only

Neural-enriched Galerkin scheme

neural network

Neural network: Construct a single neural network $v_\theta(\boldsymbol{x})$ on Ω (with N_θ weights), shared by all cells.

Neural-enriched Galerkin scheme



Neural network: Construct a single neural network $v_\theta(\boldsymbol{x})$ on Ω (with N_θ weights), shared by all cells.

Enriched basis: Construct a multiplier $m_\theta(\boldsymbol{x})$ from $v_\theta(\boldsymbol{x})$ and define the enriched basis functions as:

$$\phi_i^{\boldsymbol{\theta}}(\boldsymbol{x}) = m_\theta(\boldsymbol{x})\,\phi_i(\boldsymbol{x}).$$

Neural-enriched Galerkin scheme



Neural network: Construct a single neural network $v_\theta(\boldsymbol{x})$ on Ω (with N_θ weights), shared by all cells.

Enriched basis: Construct a multiplier $m_\theta(\boldsymbol{x})$ from $v_\theta(\boldsymbol{x})$ and define the enriched basis functions as:

$$\phi_i^\theta(\boldsymbol{x}) = m_\theta(\boldsymbol{x})\,\phi_i(\boldsymbol{x}).$$

Galerkin scheme (μ fixed): Solve the linear system:

$$\boldsymbol{R}_h\big(\boldsymbol{\beta}_h(\theta);\theta\big) = A_h(\theta)\,\boldsymbol{\beta}_h(\theta) - \boldsymbol{b}_h(\theta) = \boldsymbol{0},$$

with $\quad A_{i,j}(\theta) = a(\phi_j^\theta, \phi_i^\theta), \qquad b_i(\theta) = l(\phi_i^\theta).$

Neural-enriched Galerkin scheme



Neural network: Construct a single neural network $v_\theta(\boldsymbol{x})$ on Ω (with N_θ weights), shared by all cells.

Enriched basis: Construct a multiplier $m_\theta(\boldsymbol{x})$ from $v_\theta(\boldsymbol{x})$ and define the enriched basis functions as:

$$\phi_i^\theta(\boldsymbol{x}) = m_\theta(\boldsymbol{x})\,\phi_i(\boldsymbol{x}).$$

Galerkin scheme ($\boldsymbol{\mu}$ fixed): Solve the linear system:

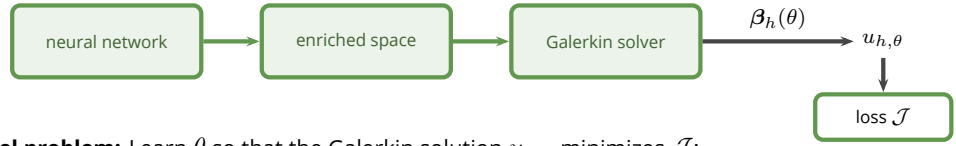
$$\boldsymbol{R}_h\big(\boldsymbol{\beta}_h(\theta);\theta\big) = A_h(\theta)\,\boldsymbol{\beta}_h(\theta) - \boldsymbol{b}_h(\theta) = \boldsymbol{0},$$

with $\quad A_{i,j}(\theta) = a(\phi_j^\theta, \phi_i^\theta), \qquad b_i(\theta) = l(\phi_i^\theta).$

Solution: Reconstruct the solution in the enriched space as:

$$u_{h,\theta}(\boldsymbol{x}) = \sum_i \beta_i(\theta)\,\phi_i^\theta(\boldsymbol{x}).$$

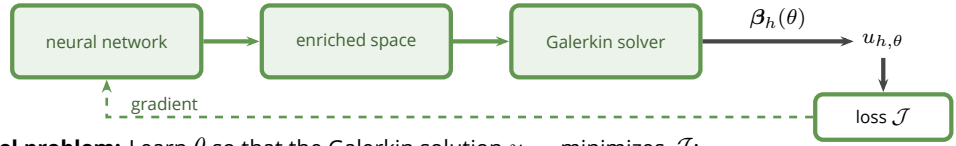
Learning θ: differentiating through the solver



Bi-level problem: Learn θ so that the Galerkin solution $u_{h,\theta}$ minimizes $\mathcal{J}$:

$$\theta^\star = \arg\min_\theta \ \mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) \quad \text{s.t.} \quad \boldsymbol{R}_h(\boldsymbol{\beta}_h(\theta);\theta) = \mathbf{0}$$

Learning θ: differentiating through the solver



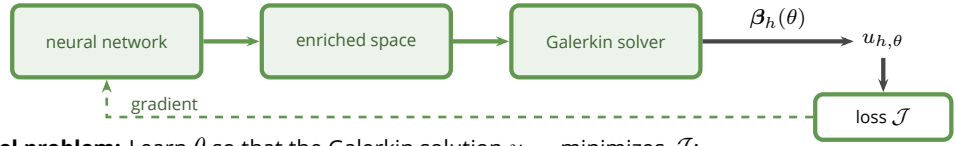
Bi-level problem: Learn θ so that the Galerkin solution $u_{h,\theta}$ minimizes $\mathcal{J}$:

$$\theta^\star = \arg\min_\theta \ \mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) \quad \text{s.t.} \quad \boldsymbol{R}_h(\boldsymbol{\beta}_h(\theta);\theta) = \mathbf{0}$$

Gradient: Differentiate through the solver (chain rule):

$$\frac{d\mathcal{J}}{d\theta} = \underbrace{\left.\frac{\partial \mathcal{J}}{\partial \theta}\right|_{\boldsymbol{\beta}_h}}_{\text{direct}} + \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}_h} \cdot \frac{d\boldsymbol{\beta}_h}{d\theta}}_{\text{through the dofs}}$$

Learning θ: differentiating through the solver



Bi-level problem: Learn θ so that the Galerkin solution $u_{h,\theta}$ minimizes $\mathcal{J}$:

$$\theta^\star = \arg\min_\theta \ \mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) \quad \text{s.t.} \quad \boldsymbol{R}_h(\boldsymbol{\beta}_h(\theta);\theta) = \mathbf{0}$$

Gradient: Differentiate through the solver (chain rule):

$$\frac{d\mathcal{J}}{d\theta} = \underbrace{\left.\frac{\partial \mathcal{J}}{\partial \theta}\right|_{\boldsymbol{\beta}_h}}_{\text{direct}} + \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}_h} \cdot \frac{d\boldsymbol{\beta}_h}{d\theta}}_{\text{through the dofs}}$$

Adjoint: Replace $d\boldsymbol{\beta}_h/d\theta$ by an adjoint state (only one linear solve, whatever N_θ):

$$A_h(\theta)^T \boldsymbol{\lambda}_h = \left(\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}_h}\right)^T \ \Rightarrow \ \frac{d\mathcal{J}}{d\theta} = \left.\frac{\partial \mathcal{J}}{\partial \theta}\right|_{\boldsymbol{\beta}_h} - \boldsymbol{\lambda}_h^T \frac{\partial \boldsymbol{R}_h}{\partial \theta}(\boldsymbol{\beta}_h;\theta)$$

Automatic differentiation and ScimBa

Need: the remaining derivatives of the adjoint formula, without deriving them by hand.

Automatic differentiation and ScimBa

Need: the remaining derivatives of the adjoint formula, without deriving them by hand.

Three ways to compute the derivatives:

	exact	use	cost per gradient
symbolic	✓	closed form only	huge expressions
finite differences	×	any program	N_θ solves
automatic diff.[1]	✓	any program	independent of N_θ

[1] [Baydin et al., 2018]

Automatic differentiation and ScimBa

Need: the remaining derivatives of the adjoint formula, without deriving them by hand.

Three ways to compute the derivatives:

	exact	use	cost per gradient
symbolic	✓	closed form only	huge expressions
finite differences	×	any program	N_θ solves
automatic diff.[1]	✓	any program	independent of N_θ

ScimBa: the whole pipeline is a JAX program[2]:

$$\text{network} \to \text{basis} \to \text{assembly} \to \text{solve} \to \text{loss } \mathcal{J}$$

One backward pass gives $\partial \mathcal{J} / \partial \theta$ through all the steps (the solve via its adjoint).

See ScimBa

[1] [Baydin et al., 2018]
[2] [Bradbury et al., 2018]

1D test case and PINN loss

Test case: 1D advection-diffusion with a boundary layer, $\Omega = (0,1)$, $\mathrm{Pe} = 20$, $f = 1$:

$$\begin{cases} -\dfrac{1}{\mathrm{Pe}}\,\partial_{xx}u + \boldsymbol{C}\,\partial_x u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(x,\boldsymbol{\mu}) = \mu_1 + \mu_2 x$$

with $\boldsymbol{\mu} = (\mu_1, \mu_2) = (0.8, 0.8)$ fixed.

1D test case and PINN loss

Test case: 1D advection-diffusion with a boundary layer, $\Omega = (0,1)$, $\mathsf{Pe} = 20$, $f = 1$:

$$\begin{cases} -\dfrac{1}{\mathsf{Pe}}\,\partial_{xx}u + \boldsymbol{C}\,\partial_x u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(x,\boldsymbol{\mu}) = \mu_1 + \mu_2 x$$

with $\boldsymbol{\mu} = (\mu_1, \mu_2) = (0.8, 0.8)$ fixed.

PINN loss: Minimize residual and boundary terms ($\omega_r = 1$, $\omega_b = 10$):

$$\mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) = \omega_r \underbrace{\int_\Omega \Big| -\frac{1}{\mathsf{Pe}}\,\partial_{xx}u_{h,\theta} + (\mu_1+\mu_2 x)\,\partial_x u_{h,\theta} - f\Big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_r} + \omega_b \underbrace{\int_\Gamma \big|u_{h,\theta}\big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_b}$$

1D test case and PINN loss

Test case: 1D advection-diffusion with a boundary layer, $\Omega = (0,1)$, $\mathsf{Pe} = 20$, $f = 1$:

$$\begin{cases} -\dfrac{1}{\mathsf{Pe}}\,\partial_{xx}u + \boldsymbol{C}\,\partial_x u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(x,\boldsymbol{\mu}) = \mu_1 + \mu_2 x$$

with $\boldsymbol{\mu} = (\mu_1, \mu_2) = (0.8, 0.8)$ fixed.

PINN loss: Minimize residual and boundary terms ($\omega_r = 1$, $\omega_b = 10$):

$$\mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) = \omega_r \underbrace{\int_\Omega \Big| -\frac{1}{\mathsf{Pe}}\,\partial_{xx}u_{h,\theta} + (\mu_1+\mu_2 x)\,\partial_x u_{h,\theta} - f\Big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_r} + \omega_b \underbrace{\int_\Gamma \big|u_{h,\theta}\big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_b}$$

Validity: Learned for this configuration only:

new $\boldsymbol{\mu}$, k or N_{cells} $\Rightarrow$ new training.

1D test case and PINN loss

Test case: 1D advection-diffusion with a boundary layer, $\Omega = (0,1)$, $\mathsf{Pe} = 20$, $f = 1$:

$$\begin{cases} -\dfrac{1}{\mathsf{Pe}}\,\partial_{xx}u + \boldsymbol{C}\,\partial_x u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(x,\boldsymbol{\mu}) = \mu_1 + \mu_2 x$$

with $\boldsymbol{\mu} = (\mu_1, \mu_2) = (0.8, 0.8)$ fixed.

PINN loss: Minimize residual and boundary terms ($\omega_r = 1$, $\omega_b = 10$):

$$\mathcal{J}(\boldsymbol{\beta}_h(\theta);\theta) = \omega_r \underbrace{\int_\Omega \Big| -\frac{1}{\mathsf{Pe}}\,\partial_{xx}u_{h,\theta} + (\mu_1+\mu_2 x)\,\partial_x u_{h,\theta} - f\Big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_r} + \omega_b \underbrace{\int_\Gamma \big|u_{h,\theta}\big|^2 \mathrm{d}\boldsymbol{x}}_{\mathcal{J}_b}$$

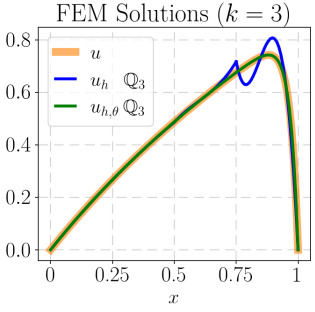
Validity: Learned for this configuration only:

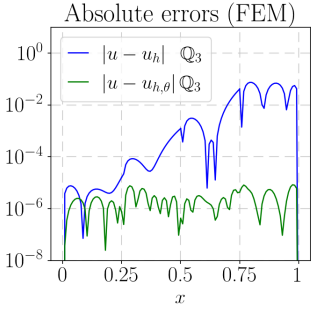
$$\text{new } \boldsymbol{\mu},\ k \text{ or } N_{\text{cells}} \ \Rightarrow\ \text{new training.}$$

Evaluation: Relative L^2 error w.r.t. a reference FEM solution (128 cells, $\mathbb{Q}_4$), never used in the training.

Learned basis: affine multiplier

Setting: CG, $N_{\text{cells}} = 4$ cells, $\mathbb{Q}_3$ ($N_{\text{dofs}} = 13$), affine multiplier $m_\theta = 2 + 0.1\, v_\theta$.
Training: MLP v_θ (tanh, 2×16, $N_\theta = 321$), ENG optimizer[1], 300 epochs, 6000 collocation points.



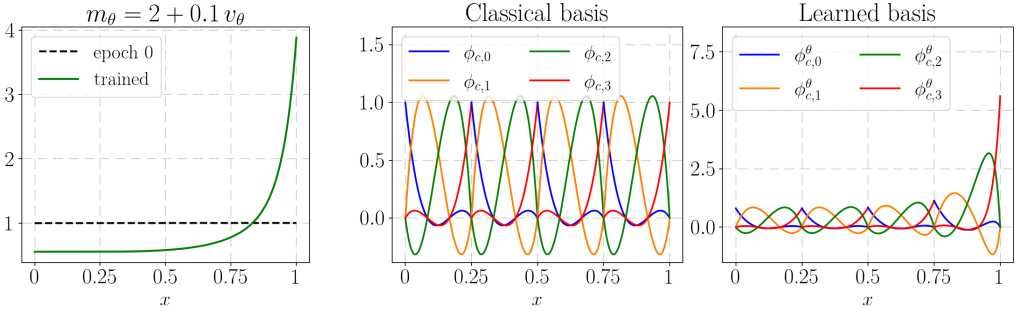


Solutions: Classical u_h oscillates in the boundary layer, $u_{h,\theta}$ matches the reference ($\sim 10^{-5}$ against $7 \cdot 10^{-2}$).

[1] [Müller and Zeinhofer, 2023]

Learned basis: affine multiplier

Setting: CG, $N_{\text{cells}} = 4$ cells, $\mathbb{Q}_3$ ($N_{\text{dofs}} = 13$), affine multiplier $m_\theta = 2 + 0.1\, v_\theta$.
Training: MLP v_θ (tanh, 2×16, $N_\theta = 321$), ENG optimizer[1], 300 epochs, 6000 collocation points.



Solutions: Classical u_h oscillates in the boundary layer, $u_{h,\theta}$ matches the reference ($\sim 10^{-5}$ against $7 \cdot 10^{-2}$).
Basis: The multiplier grows in the boundary layer: the last basis functions are amplified.

[1] [Müller and Zeinhofer, 2023]

Choice of the multiplier

Fix $N_{\text{dofs}} = 25$ and $N_\theta \in \{134, \dots, 151\}$, vary the levers: (N_{cells}, k) and the MLP shape (width, depth).

20 configurations $\times$ 20 seeds $=$ 400 trainings per multiplier (100 epochs, 5000 collocation points).

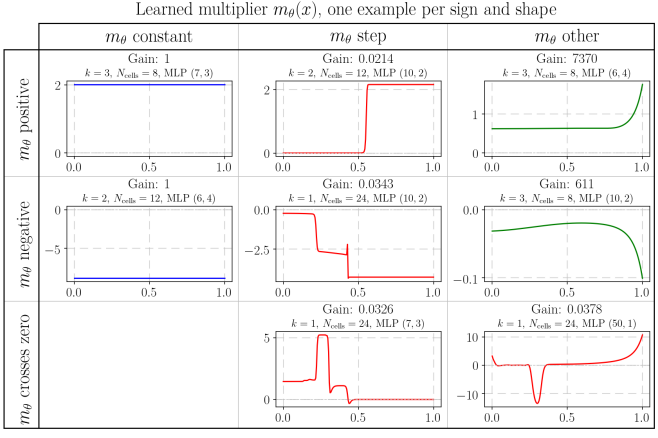
Choice of the multiplier

Fix $N_{\text{dofs}} = 25$ and $N_\theta \in \{134, \dots, 151\}$, vary the levers: (N_{cells}, k) and the MLP shape (width, depth).

20 configurations $\times$ 20 seeds $=$ 400 trainings per multiplier (100 epochs, 5000 collocation points).

Affine multiplier:

$$m_\theta = 2 + 0.1\, v_\theta$$



Choice of the multiplier

Fix $N_{\text{dofs}} = 25$ and $N_\theta \in \{134, \dots, 151\}$, vary the levers: (N_{cells}, k) and the MLP shape (width, depth).

20 configurations $\times$ 20 seeds $=$ 400 trainings per multiplier (100 epochs, 5000 collocation points).

Affine multiplier:

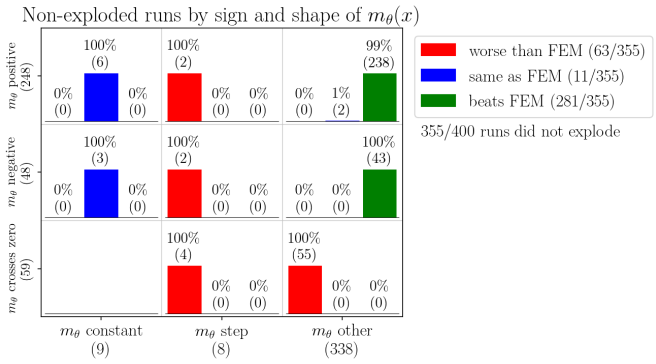
$$m_\theta = 2 + 0.1\, v_\theta$$

Diagnosis:

Some trainings explode (loss $=$ NaN).

Constant m_θ: same space, same error as FEM.

Steps and sign changes never beat FEM.



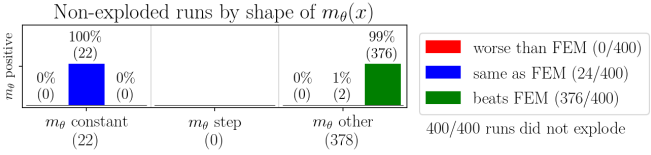
Choice of the multiplier

Fix $N_{\text{dofs}} = 25$ and $N_\theta \in \{134, \dots, 151\}$, vary the levers: (N_{cells}, k) and the MLP shape (width, depth).

20 configurations $\times$ 20 seeds $=$ 400 trainings per multiplier (100 epochs, 5000 collocation points).

Affine multiplier:

$$m_\theta = 2 + 0.1\, v_\theta$$



Diagnosis:

Some trainings explode (loss $=$ NaN).

Constant m_θ: same space, same error as FEM.

Steps and sign changes never beat FEM.

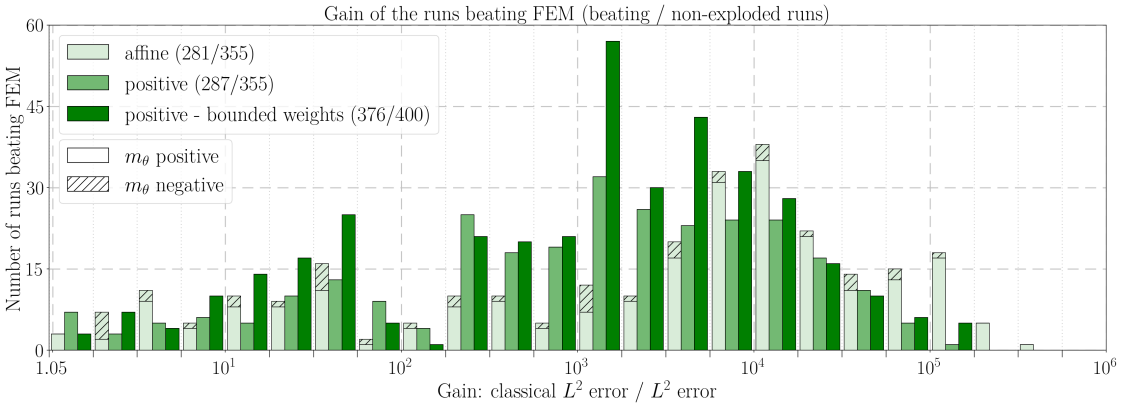
Fix: Positive multiplier

$$m_\theta = \exp(2 \tanh v_\theta)$$

with bounded weights:

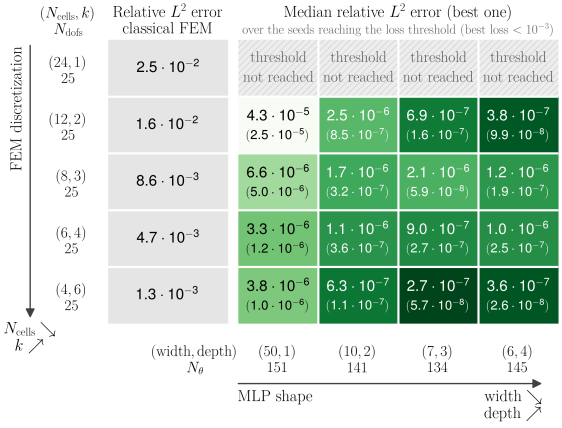
No sign change, no step.

Results at fixed budget



Gain: $376/400$ beat the FEM, none explodes; the affine multiplier keeps the highest gains.

Results at fixed budget



Gain: $376/400$ beat the FEM, none explodes; the affine multiplier keeps the highest gains.

Levers: degree and depth matter: $k = 1$ only a modest gain over FEM, $(50, 1)$ is the worst network.

3D test case

Test case: 3D advection-diffusion with a boundary layer, $\Omega = (0,1)^3$, $\mathrm{Pe} = 20$:

$$\begin{cases} -\dfrac{1}{\mathrm{Pe}}\,\Delta u + \boldsymbol{C}\cdot\nabla u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\mu}) = (\mu_1 + \mu_2\,\boldsymbol{x}\cdot\boldsymbol{e}_{\mathrm{diag}})\,\boldsymbol{e}_{\mathrm{diag}},$$

with $\boldsymbol{e}_{\mathrm{diag}} = \frac{1}{\sqrt{3}}(1,1,1)^\top$ the diagonal direction and $\boldsymbol{\mu} = (0.8, 0.8)$ fixed.

3D test case

Test case: 3D advection-diffusion with a boundary layer, $\Omega = (0,1)^3$, $\mathsf{Pe} = 20$:

$$\begin{cases} -\dfrac{1}{\mathsf{Pe}}\,\Delta u + \boldsymbol{C}\cdot\nabla u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\mu}) = (\mu_1 + \mu_2\,\boldsymbol{x}\cdot\boldsymbol{e}_{\mathsf{diag}})\,\boldsymbol{e}_{\mathsf{diag}},$$

with $\boldsymbol{e}_{\mathsf{diag}} = \frac{1}{\sqrt{3}}(1,1,1)^\top$ the diagonal direction and $\boldsymbol{\mu} = (0.8, 0.8)$ fixed.

Exact solution: manufactured, vanishing on the 6 faces (f deduced from it):

$$u(\boldsymbol{x}) = \prod_{i=1}^{3} x_i \Big(1 - \exp\Big(-\frac{\mathsf{Pe}\,(\mu_1+\mu_2)}{\sqrt{3}}\,(1-x_i)\Big)\Big)$$

3D test case

Test case: 3D advection-diffusion with a boundary layer, $\Omega = (0,1)^3$, $\mathsf{Pe} = 20$:

$$\begin{cases} -\dfrac{1}{\mathsf{Pe}}\,\Delta u + \boldsymbol{C}\cdot\nabla u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \qquad \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\mu}) = (\mu_1 + \mu_2\,\boldsymbol{x}\cdot\boldsymbol{e}_{\mathsf{diag}})\,\boldsymbol{e}_{\mathsf{diag}},$$

with $\boldsymbol{e}_{\mathsf{diag}} = \frac{1}{\sqrt{3}}(1,1,1)^\top$ the diagonal direction and $\boldsymbol{\mu} = (0.8, 0.8)$ fixed.

Exact solution: manufactured, vanishing on the 6 faces (f deduced from it):

$$u(\boldsymbol{x}) = \prod_{i=1}^{3} x_i \Big(1 - \exp\Big(-\frac{\mathsf{Pe}\,(\mu_1+\mu_2)}{\sqrt{3}}\,(1-x_i)\Big)\Big)$$

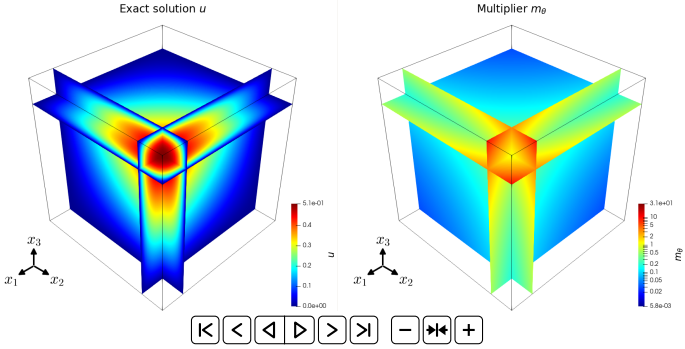
Evaluation: Relative L^2 error w.r.t. the manufactured solution u, never used in the training.

3D: learned multiplier

Setting: CG, $N_{\text{cells}} = 4 \times 4 \times 4$ cells, $\mathbb{Q}_2$ ($N_{\text{dofs}} = 729$), positive multiplier $m_\theta = \exp\left(6 \tanh v_\theta\right)$.
Training: MLP v_θ (tanh, 3×10, $N_\theta = 271$), ENG optimizer, 70 epochs, 6000 collocation points, 3 seeds.

3D: learned multiplier

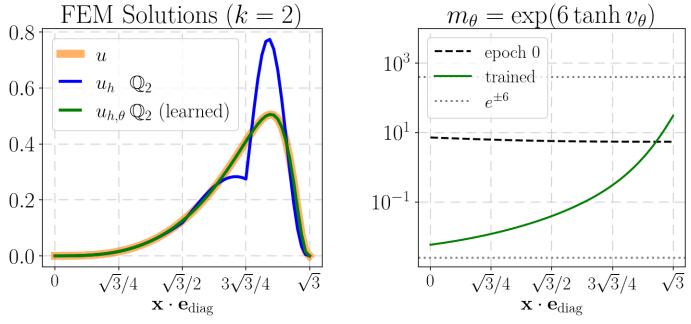
Setting: CG, $N_{\text{cells}} = 4 \times 4 \times 4$ cells, $\mathbb{Q}_2$ ($N_{\text{dofs}} = 729$), positive multiplier $m_\theta = \exp\left(6 \tanh v_\theta\right)$.
Training: MLP v_θ (tanh, 3×10, $N_\theta = 271$), ENG optimizer, 70 epochs, 6000 collocation points, 3 seeds.



Multiplier: grows in the boundary layer, from $6 \cdot 10^{-3}$ to 30 (seed 1).

3D: learned multiplier

Setting: CG, $N_{\text{cells}} = 4 \times 4 \times 4$ cells, $\mathbb{Q}_2$ ($N_{\text{dofs}} = 729$), positive multiplier $m_\theta = \exp\left(6 \tanh v_\theta\right)$.
Training: MLP v_θ (tanh, 3×10, $N_\theta = 271$), ENG optimizer, 70 epochs, 6000 collocation points, 3 seeds.



Multiplier: grows in the boundary layer, from $6 \cdot 10^{-3}$ to 30 (seed 1).
Solution: classical u_h overshoots, $u_{h,\theta}$ matches u.

3D: gain and cost

	time (s)									
	w/ comp.		w/o comp.		N_{dofs}			error		
seed	learn.	class.	learn.	class.	learn.	class.	ratio	learn.	class.	ratio
1	59	51	10.1	49.5	729	$5.31 \cdot 10^5$	729	$5.68 \cdot 10^{-5}$	$3.53 \cdot 10^{-4}$	6.2
2	41	37	6.8	26.6	729	$3.89 \cdot 10^5$	534	$8.13 \cdot 10^{-5}$	$4.82 \cdot 10^{-4}$	5.9
3	41	37	6.9	26.6	729	$3.89 \cdot 10^5$	534	$4.77 \cdot 10^{-4}$	$4.82 \cdot 10^{-4}$	1

Equal time: error $\times 6$ lower for seeds 1, 2, equal for seed 3.

3D: gain and cost

	time (s)									
	w/ comp.		w/o comp.		N_{dofs}			error		
seed	learn.	class.	learn.	class.	learn.	class.	ratio	learn.	class.	ratio
1	59	51	10.1	49.5	729	$5.31 \cdot 10^5$	729	$5.68 \cdot 10^{-5}$	$3.53 \cdot 10^{-4}$	6.2
2	41	37	6.8	26.6	729	$3.89 \cdot 10^5$	534	$8.13 \cdot 10^{-5}$	$4.82 \cdot 10^{-4}$	5.9
3	41	37	6.9	26.6	729	$3.89 \cdot 10^5$	534	$4.77 \cdot 10^{-4}$	$4.82 \cdot 10^{-4}$	1

Equal time: error $\times 6$ lower for seeds 1, 2, equal for seed 3.

Dofs: 534 to 729 times fewer for the learned basis (without counting the $N_\theta = 271$ weights).

3D: gain and cost

	time (s)									
	w/ comp.		w/o comp.		N_{dofs}			error		
seed	learn.	class.	learn.	class.	learn.	class.	ratio	learn.	class.	ratio
1	59	51	10.1	49.5	729	$5.31 \cdot 10^5$	729	$5.68 \cdot 10^{-5}$	$3.53 \cdot 10^{-4}$	6.2
2	41	37	6.8	26.6	729	$3.89 \cdot 10^5$	534	$8.13 \cdot 10^{-5}$	$4.82 \cdot 10^{-4}$	5.9
3	41	37	6.9	26.6	729	$3.89 \cdot 10^5$	534	$4.77 \cdot 10^{-4}$	$4.82 \cdot 10^{-4}$	1

Equal time: error $\times 6$ lower for seeds $1, 2$, equal for seed 3.

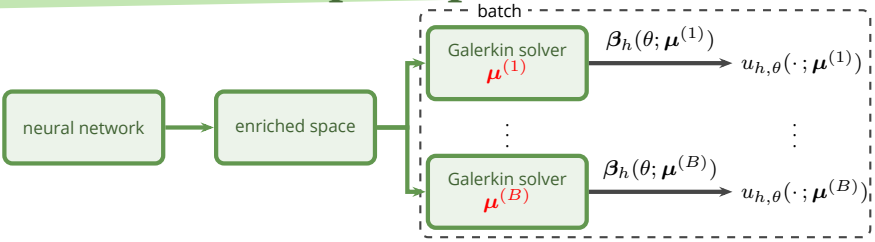
Dofs: 534 to 729 times fewer for the learned basis (without counting the $N_\theta = 271$ weights).

Cost: learned time dominated by compilation: 7 to 10 s without it, against 27 to 50 s for the FEM.

Parametric enrichment*

* $\boldsymbol{\mu}$ is no longer fixed: one network trained once, for all $\boldsymbol{\mu} \in \mathcal{M}$

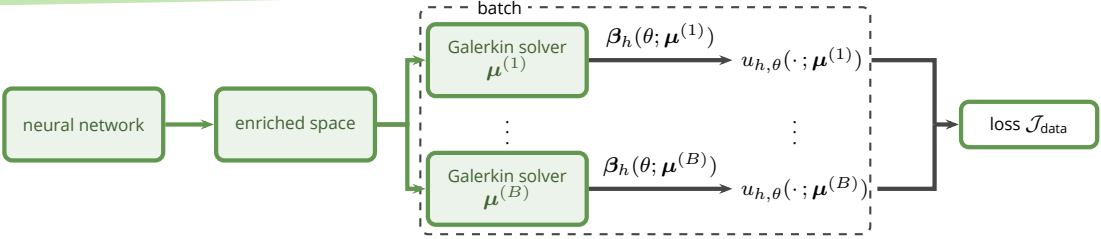
Parametric case: principle



Parameter sampling: $\mathcal{D}_{\text{train}} \subset \mathcal{M}$ chosen once, a random batch of $\boldsymbol{\mu}$ drawn at each epoch.

Parametric scheme: One network, one Galerkin solve per $\boldsymbol{\mu}$ of the batch: $A_h(\theta;\boldsymbol{\mu})\,\boldsymbol{\beta}_h(\theta;\boldsymbol{\mu}) = \boldsymbol{b}_h(\theta;\boldsymbol{\mu})$.

Parametric case: principle



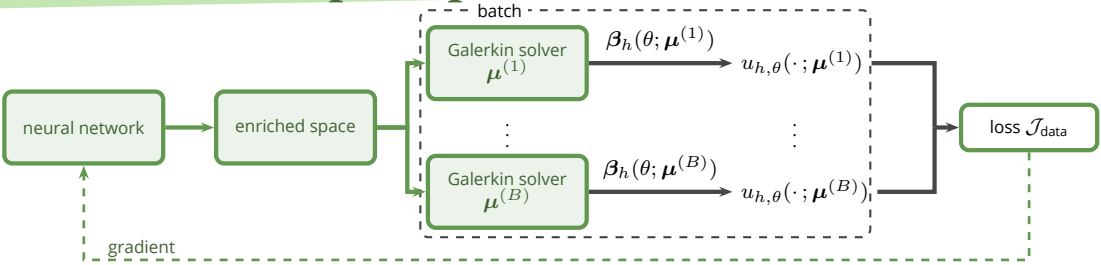
Parameter sampling: $\mathcal{D}_{\text{train}} \subset \mathcal{M}$ chosen once, a random batch of $\boldsymbol{\mu}$ drawn at each epoch.

Parametric scheme: One network, one Galerkin solve per $\boldsymbol{\mu}$ of the batch: $A_h(\theta;\boldsymbol{\mu})\,\boldsymbol{\beta}_h(\theta;\boldsymbol{\mu}) = \boldsymbol{b}_h(\theta;\boldsymbol{\mu})$.

Supervised loss: Fit reference solutions on the batch, at N_x points per $\boldsymbol{\mu}$:

$$\mathcal{J}_{\text{data}}(\theta) = \frac{1}{B\,N_x}\sum_{i=1}^{B}\sum_{j=1}^{N_x}\left|u_{h,\theta}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big) - u_{\text{ref}}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big)\right|^2.$$

Parametric case: principle



Parameter sampling: $\mathcal{D}_{\text{train}} \subset \mathcal{M}$ chosen once, a random batch of $\boldsymbol{\mu}$ drawn at each epoch.

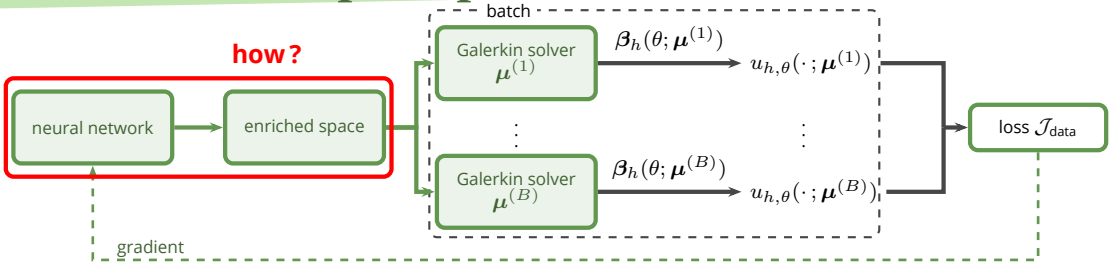
Parametric scheme: One network, one Galerkin solve per $\boldsymbol{\mu}$ of the batch: $A_h(\theta;\boldsymbol{\mu})\,\boldsymbol{\beta}_h(\theta;\boldsymbol{\mu}) = \boldsymbol{b}_h(\theta;\boldsymbol{\mu})$.

Supervised loss: Fit reference solutions on the batch, at N_x points per $\boldsymbol{\mu}$:

$$\mathcal{J}_{\text{data}}(\theta) = \frac{1}{B\,N_x}\sum_{i=1}^{B}\sum_{j=1}^{N_x}\Big|u_{h,\theta}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big) - u_{\text{ref}}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big)\Big|^2.$$

Gradient: one adjoint solve per $\boldsymbol{\mu}$ of the batch.

Parametric case: principle



Parameter sampling: $\mathcal{D}_{\text{train}} \subset \mathcal{M}$ chosen once, a random batch of $\boldsymbol{\mu}$ drawn at each epoch.

Parametric scheme: One network, one Galerkin solve per $\boldsymbol{\mu}$ of the batch: $A_h(\theta;\boldsymbol{\mu})\,\boldsymbol{\beta}_h(\theta;\boldsymbol{\mu}) = \boldsymbol{b}_h(\theta;\boldsymbol{\mu})$.

Supervised loss: Fit reference solutions on the batch, at N_x points per $\boldsymbol{\mu}$:

$$\mathcal{J}_{\text{data}}(\theta) = \frac{1}{B\,N_x}\sum_{i=1}^{B}\sum_{j=1}^{N_x}\Big|u_{h,\theta}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big) - u_{\text{ref}}\big(\boldsymbol{x}^{(j)};\boldsymbol{\mu}^{(i)}\big)\Big|^2.$$

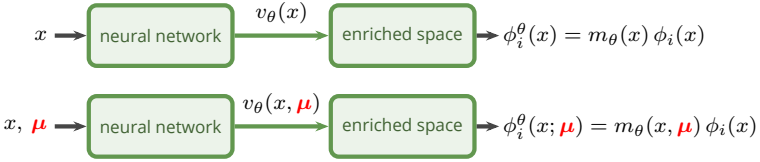
Gradient: one adjoint solve per $\boldsymbol{\mu}$ of the batch.

Constructing the enriched space (three ways)



Shared basis: same enriched space for all $\boldsymbol{\mu}$.

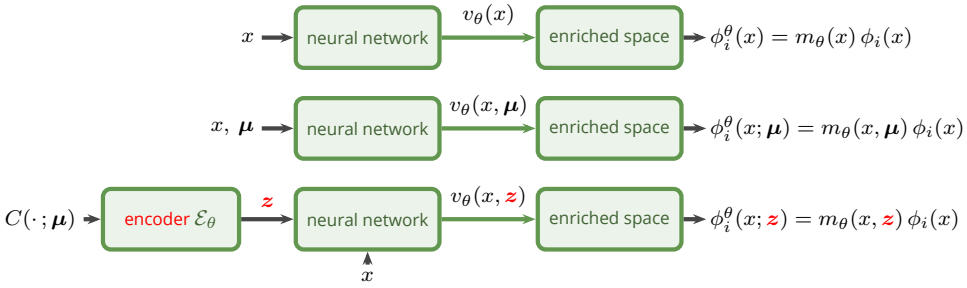
Constructing the enriched space (three ways)



Shared basis: same enriched space for all $\boldsymbol{\mu}$.

$\boldsymbol{\mu}$-dependent basis: one enriched space per $\boldsymbol{\mu}$.

Constructing the enriched space (three ways)



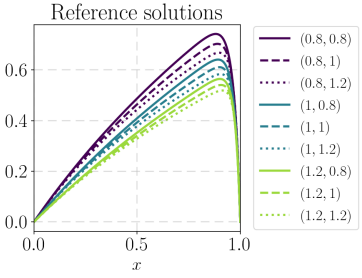
Shared basis: same enriched space for all $\boldsymbol{\mu}$.

$\boldsymbol{\mu}$-dependent basis: one enriched space per $\boldsymbol{\mu}$.

Field-dependent basis: an encoder summarizes the transport field in a latent state $\boldsymbol{z}$.

1D linear transport

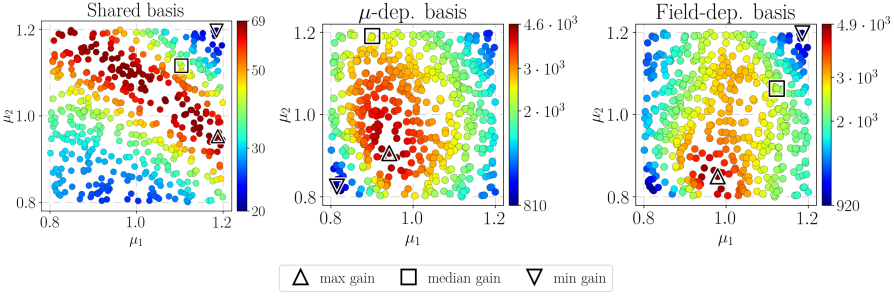
Test case: Same 1D case as the PINN, with $\boldsymbol{C}(x, \boldsymbol{\mu}) = \mu_1 + \mu_2 x$, $\boldsymbol{\mu} \in [0.8, 1.2]^2$: the layer stays at $x = 1$.



1D linear transport

Test case: Same 1D case as the PINN, with $\boldsymbol{C}(x, \boldsymbol{\mu}) = \mu_1 + \mu_2 x$, $\boldsymbol{\mu} \in [0.8, 1.2]^2$: the layer stays at $x = 1$.

Setup: DG, $N_{\text{cells}} = 8$, $k = 2$, 32 training $\boldsymbol{\mu}$, batch of 16, $N_x = 101$ points per $\boldsymbol{\mu}$.

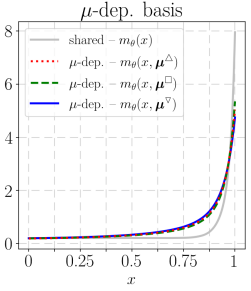


Gain (on 500 unseen $\boldsymbol{\mu}$):
shared 20–69 ($N_\theta = 321$), $\boldsymbol{\mu}$-dep. 811–$4.6 \cdot 10^3$ ($N_\theta = 353$), field-dep. 917–$4.9 \cdot 10^3$ ($N_\theta = 1381$).

1D linear transport

Test case: Same 1D case as the PINN, with $\boldsymbol{C}(x, \boldsymbol{\mu}) = \mu_1 + \mu_2 x$, $\boldsymbol{\mu} \in [0.8, 1.2]^2$: the layer stays at $x = 1$.

Setup: DG, $N_{\text{cells}} = 8$, $k = 2$, 32 training $\boldsymbol{\mu}$, batch of 16, $N_x = 101$ points per $\boldsymbol{\mu}$.

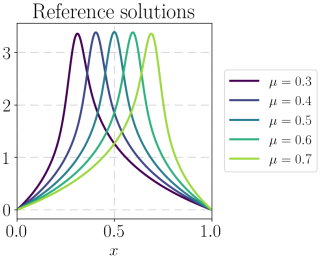


Gain (on 500 unseen $\boldsymbol{\mu}$):
shared 20–69 ($N_\theta = 321$), $\boldsymbol{\mu}$-dep. $811–4.6 \cdot 10^3$ ($N_\theta = 353$), field-dep. $917–4.9 \cdot 10^3$ ($N_\theta = 1381$).

Multipliers: shared, rises only in the last cells; $\boldsymbol{\mu}$-dep., rises over more cells.

1D turning point

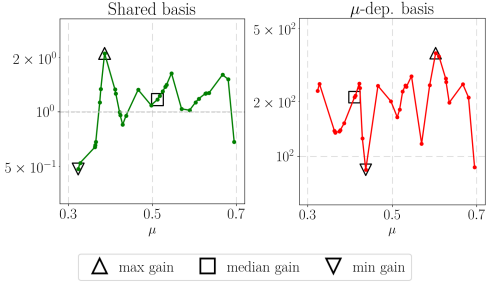
Test case: Same 1D equation as the PINN, $\boldsymbol{C}(x, \mu) = \mu - x$, $\mu \in [0.3, 0.7]$, Pe $= 1000$: peak moves with μ.



1D turning point

Test case: Same 1D equation as the PINN, $C(x, \mu) = \mu - x$, $\mu \in [0.3, 0.7]$, Pe $= 1000$: peak moves with μ.

Setup: Same as the linear transport (DG, $N_{\text{cells}} = 8$, $k = 2$), with a scalar μ.

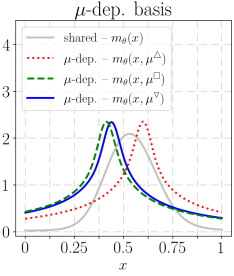


Gain (on 32 unseen μ):
shared 0.5–2.1 $(N_\theta = 321)$, μ-dep. 85–366 $(N_\theta = 337)$.

1D turning point

Test case: Same 1D equation as the PINN, $\boldsymbol{C}(x, \mu) = \mu - x$, $\mu \in [0.3, 0.7]$, Pe $= 1000$: peak moves with μ.

Setup: Same as the linear transport (DG, $N_{\text{cells}} = 8$, $k = 2$), with a scalar μ.



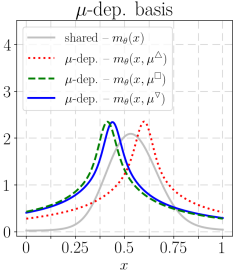
Gain (on 32 unseen μ):
shared 0.5–2.1 $(N_\theta = 321)$, μ-dep. 85–366 $(N_\theta = 337)$.

Multipliers: shared, one central bump; μ-dep., a peak at the turning point.

1D turning point

Test case: Same 1D equation as the PINN, $\boldsymbol{C}(x, \mu) = \mu - x$, $\mu \in [0.3, 0.7]$, Pe $= 1000$: peak moves with μ.

Setup: Same as the linear transport (DG, $N_{\text{cells}} = 8$, $k = 2$), with a scalar μ.



Gain (on 32 unseen μ):
shared 0.5–2.1 $(N_\theta = 321)$, μ-dep. 85–366 $(N_\theta = 337)$.

Multipliers: shared, one central bump; μ-dep., a peak at the turning point.

Conclusion

Conclusion

1st hybridization: enriched FEM with a frozen PINN prior

- Convergence orders preserved, gain set by the quality of the prior (its derivatives)
- Additive, $\mathbb{P}_1$: meshes 5 to 28 times coarser for the same accuracy
- 3D: cheaper as soon as 19 parameter sets are solved (training included, $e = 10^{-3}$)
- Multiplicative $\rightarrow$ additive as $M \rightarrow \infty$; better only in special cases

Conclusion

1st hybridization: enriched FEM with a frozen PINN prior

- Convergence orders preserved, gain set by the quality of the prior (its derivatives)
- Additive, $\mathbb{P}_1$: meshes 5 to 28 times coarser for the same accuracy
- 3D: cheaper as soon as 19 parameter sets are solved (training included, $e = 10^{-3}$)
- Multiplicative $\rightarrow$ additive as $M \rightarrow \infty$; better only in special cases

2nd hybridization: learnable Galerkin bases

- Differentiable solver: train the basis through the solver
- 1D and 3D: far fewer dofs for the same accuracy (1D: error 10^2 to 10^5 times lower at fixed N_{dofs}, best seed)
- Parametric (1D): unseen parameters, median gains from ≈ 9 to ≈ 7000

Limits and perspectives

1st hybridization:

- Gain bounded by the prior accuracy; scalar linear elliptic problems only
- Next: better priors, time-dependent and non-linear problems, complex geometries (φ-FEM)

Limits and perspectives

1st hybridization:

- Gain bounded by the prior accuracy; scalar linear elliptic problems only
- Next: better priors, time-dependent and non-linear problems, complex geometries (φ-FEM)

2nd hybridization: exploratory study

- Few test cases, parametric case in 1D only, uniform meshes only, no error estimate
- Training sensitive to the initialization (fails for $k = 1$ in 1D); training cost not accounted for in the parametric case
- Next: comparison at equal cost (training included), adapted meshes, parametric 2D

Limits and perspectives

1st hybridization:

- Gain bounded by the prior accuracy; scalar linear elliptic problems only
- Next: better priors, time-dependent and non-linear problems, complex geometries (φ-FEM)

2nd hybridization: exploratory study

- Few test cases, parametric case in 1D only, uniform meshes only, no error estimate
- Training sensitive to the initialization (fails for $k = 1$ in 1D); training cost not accounted for in the parametric case
- Next: comparison at equal cost (training included), adapted meshes, parametric 2D

Link between the two:
freeze the learned multiplier $m_\theta \Rightarrow$ multiplicative approach of the 1st hybridization, a priori estimate.

References

Arnold, Douglas N. et al. [2002]. “Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems”. In: *SIAM Journal on Numerical Analysis*.

Baydin, Atilim Gunes et al. [2018]. “Automatic Differentiation in Machine Learning: a Survey”. In: *Journal of Machine Learning Research*.

Bradbury, James et al. [2018]. *JAX: composable transformations of Python+NumPy programs*.

Duprez, Michel and Alexei Lozinski [2020]. “ϕ-FEM: A Finite Element Method on Domains Defined by Level-Sets”. In: *SIAM Journal on Numerical Analysis*.

Ern, A. and J.-L. Guermond [2004]. *Theory and Practice of Finite Elements*.

Franck, E., V. Michel-Dansac, and L. Navoret [2024]. “Approximately well-balanced Discontinuous Galerkin methods using bases enriched with Physics-Informed Neural Networks”. In: *Journal of Computational Physics*.

Lecourtier, Frédérique et al. [2026]. “Enriching continuous Lagrange finite element approximation spaces using neural networks”. In: *ESAIM: Mathematical Modelling and Numerical Analysis*.

Moës, Nicolas, John Dolbow, and Ted Belytschko [1999]. “A finite element method for crack growth without remeshing”. In: *International Journal for Numerical Methods in Engineering*.

Moiola, A. and I. Perugia [2018]. “A space–time Trefftz discontinuous Galerkin method for the acoustic wave equation in first-order formulation”. In: *Numerische Mathematik*.

Müller, Johannes and Marius Zeinhofer [2023]. “Achieving High Accuracy with PINNs via Energy Natural Gradient Descent”. In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Raissi, M., P. Perdikaris, and G. E. Karniadakis [2019]. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics*.

Strouboulis, T., K. Copps, and I. Babuška [2001]. “The generalized finite element method”. In: *Computer Methods in Applied Mechanics and Engineering*.

Sukumar, N. and A. Srivastava [2022]. “Exact imposition of boundary conditions with distance functions in physics-informed deep neural

Appendix – 1st hybridization

Enriched finite element method using PINNs

Additional details

General Dirichlet boundary conditions

Problem: $u = g$ on Γ, with u_θ the PINN prior.

Additive approach:

$$p_h^+ = g - u_\theta \quad \text{on } \Gamma.$$

Homogeneous as soon as u_θ is exact on Γ (exact BC in the PINN).

Multiplicative approach ($u_{h,\theta}^\times = u_{\theta,M}\, p_h^\times - M$, $u_{\theta,M} = u_\theta + M$):

$$\text{standard PINN:} \quad p_h^\times = \frac{g + M}{u_{\theta,M}} \quad \text{on } \Gamma,$$

$$\text{PINN with exact BC:} \quad p_h^\times = 1 \quad \text{on } \Gamma.$$

Homogeneous case: $M = 0$ possible, $p_h^\times$ left free (more freedom for the derivatives at the boundary).

Curved boundaries (disk, $k > 2$): the prior is not exact on $\partial\Omega_h$; in the tests, $g = u$ on $\partial\Omega_h$.

Extra results

Anisotropic 2D
Annulus 2D

Extra results

Anisotropic 2D

Annulus 2D

Anisotropic 2D: problem considered

Test case: $-\mathrm{div}(D\nabla u) = f$ on $\Omega = (0,1)^2$, $u = 0$ on $\partial\Omega$,
$\boldsymbol{\mu} = (\mu_1, \mu_2, \epsilon, \sigma) \in [0.4, 0.6]^2 \times [0.01, 1] \times [0.1, 0.8]$.

$$f = \exp\left(-\frac{(x_1-\mu_1)^2 + (x_2-\mu_2)^2}{0.025\sigma^2}\right), \qquad D = \begin{pmatrix} \epsilon x_1^2 + x_2^2 & (\epsilon-1)x_1x_2 \\ (\epsilon-1)x_1x_2 & x_1^2 + \epsilon x_2^2 \end{pmatrix}$$

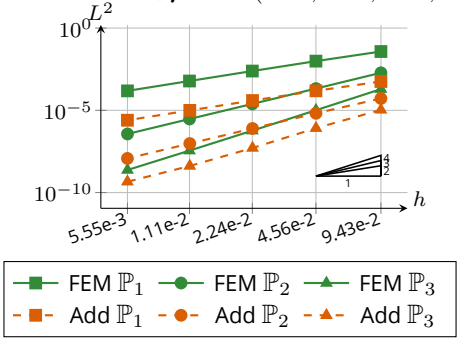
Anisotropy: eigenvalues $x_1^2 + x_2^2$ and $\epsilon\,(x_1^2 + x_2^2)$, stronger as ϵ decreases.

Parametric PINN: MLP $(40, 60, 60, 60, 40)$, tanh; Adam, 15000 epochs, 8000 col. pts; exact Dirichlet BC (level-set).

Reference: no analytical solution, FEM $\mathbb{P}_3$ on $N_{\text{ref}} = 1000$.

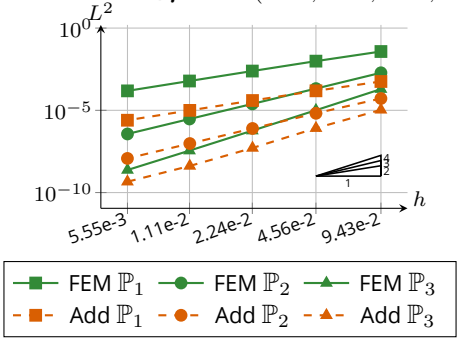
Anisotropic 2D: online phase

L^2 **error,** $\boldsymbol{\mu}^{(1)} = (0.51, 0.54, 0.52, 0.55)$



Anisotropic 2D: online phase

L^2 **error,** $\boldsymbol{\mu}^{(1)} = (0.51, 0.54, 0.52, 0.55)$



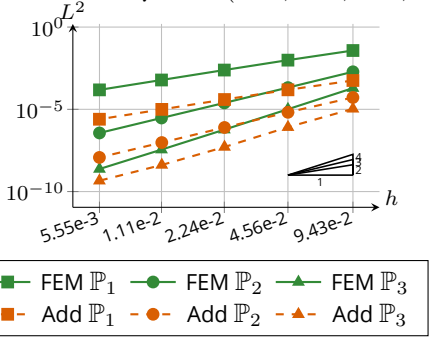
Gains over FEM: $\|u - u_h\|_{L^2} / \|u - u^+_{h,\theta}\|_{L^2}$

$N_\mu = 50$ parameters, $N = 20$.

	Gains in L^2 rel error of our method w.r.t. FEM		
k	**min**	**max**	**mean**
1	7.12	82.57	35.67
2	3.54	35.88	18.32
3	1.33	26.51	8.32

Anisotropic 2D: online phase

L^2 **error,** $\boldsymbol{\mu}^{(1)} = (0.51, 0.54, 0.52, 0.55)$



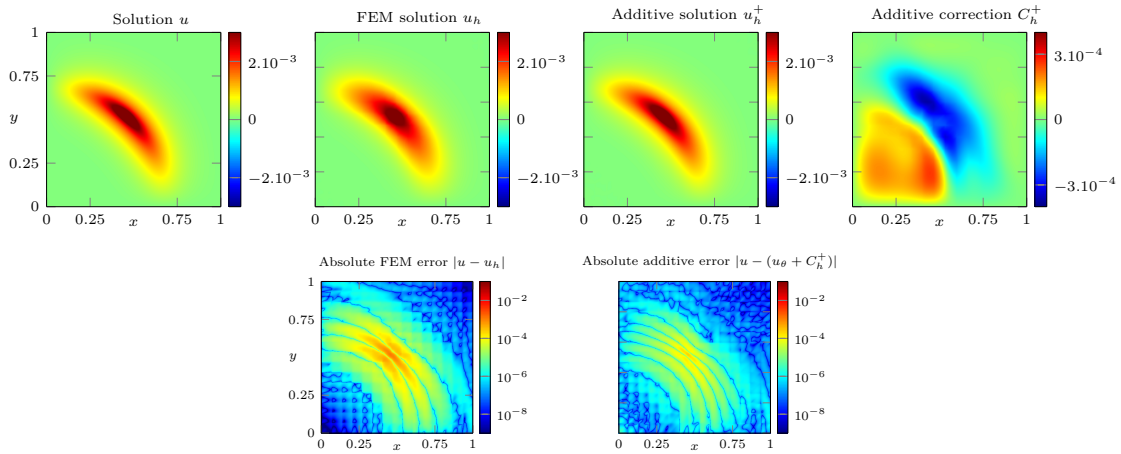
Gains over FEM: $\|u - u_h\|_{L^2} / \|u - u^+_{h,\theta}\|_{L^2}$

$N_\mu = 50$ parameters, $N = 20$.

	Gains in L^2 rel error of our method w.r.t. FEM		
k	**min**	**max**	**mean**
1	7.12	82.57	35.67
2	3.54	35.88	18.32
3	1.33	26.51	8.32

Smaller gains: the prior is less accurate on highly anisotropic problems.

Anisotropic 2D: solutions and errors



$$\boldsymbol{\mu}^{(3)} = (0.46, 0.52, 0.05, 0.12) \quad (k = 2, N = 16)$$

Extra results

Anisotropic 2D

Annulus 2D

Poisson on an annulus: problem considered[1]

Test case: $-\Delta u = f = 0$ on $\Omega = \{0.25 < \|\boldsymbol{x}\| < 1\}$, $\boldsymbol{\mu} = \mu_1 \in [2.4, 2.6]$.

$$\begin{cases} u = g, & \text{on } \Gamma_E \text{ (outer circle)}, \\ \dfrac{\partial u}{\partial n} + u = g_R, & \text{on } \Gamma_I \text{ (hole)}, \end{cases}$$

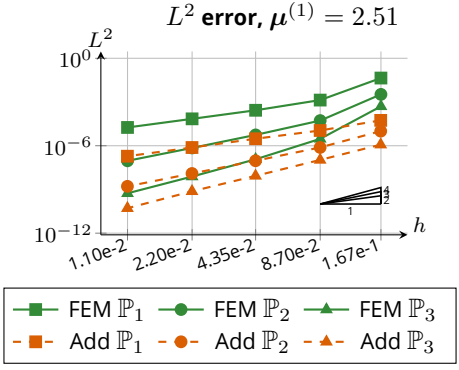
$$u(\boldsymbol{x}, \boldsymbol{\mu}) = 1 - \frac{\ln\left(\mu_1 \|\boldsymbol{x}\|\right)}{\ln(4)}, \quad g = 1 - \frac{\ln(\mu_1)}{\ln(4)}, \quad g_R = 2 + \frac{4 - \ln(\mu_1)}{\ln(4)}$$

Parametric PINN: MLP 5×40, tanh; 4000 epochs, 6000 col. pts; mixed BC exact + Sobolev training.

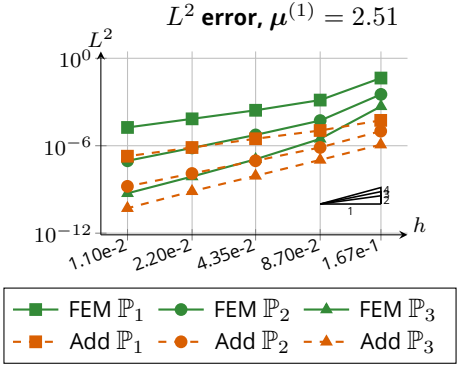
Boundary conditions depend on $\boldsymbol{\mu}$: harder than the previous cases.

[1] [Sukumar and Srivastava, 2022]

Poisson on an annulus: online phase



Poisson on an annulus: online phase

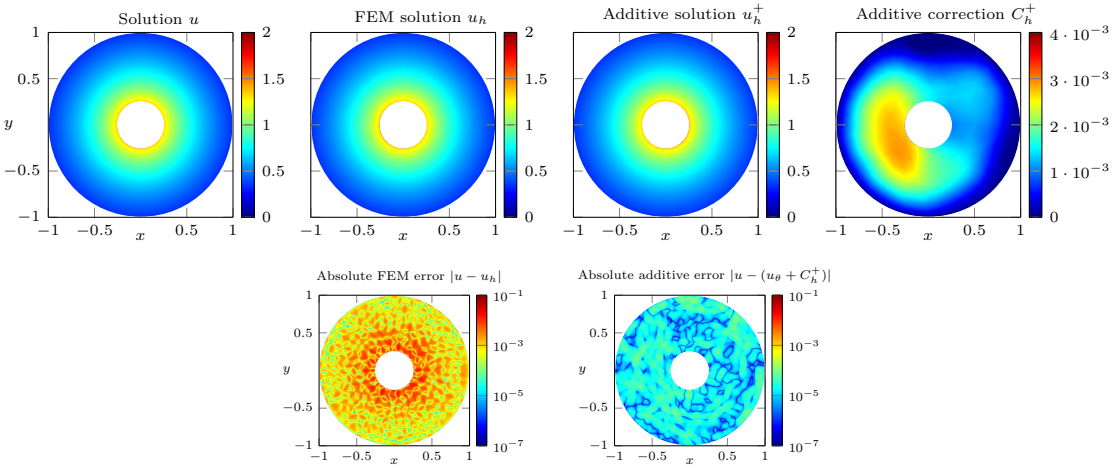


Gains over FEM: $\|u - u_h\|_{L^2} / \|u - u^+_{h,\theta}\|_{L^2}$

$N_\mu = 50$ parameters, $h = 1.33 \cdot 10^{-1}$.

	Gains in L^2 rel error of our method w.r.t. FEM		
k	**min**	**max**	**mean**
1	15.12	137.72	55.5
2	31	77.46	58.41
3	18.72	21.49	20.6

Poisson on an annulus: solutions and errors



$$\boldsymbol{\mu}^{(1)} = 2.51 \quad (k = 1, h = 1.67 \cdot 10^{-1})$$

Appendix – 2nd hybridization

Learnable Galerkin methods

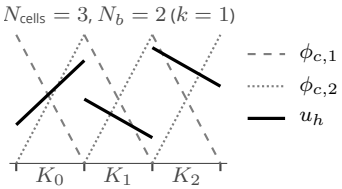
Additional details

Discontinuous Galerkin (DG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω of cells $(K_c)_{0 \leq c < N_{\text{cells}}}$.

Let V_h^{disc} be a discontinuous space of piecewise polynomials of degree k on the mesh $\mathcal{T}_h$. Its basis functions $(\phi_{c,i})_{c,i}$, with $\phi_{c,i}$ supported on a single cell K_c, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{c,i} \beta_{c,i}(\boldsymbol{\mu})\, \phi_{c,i}(\boldsymbol{x}).$$



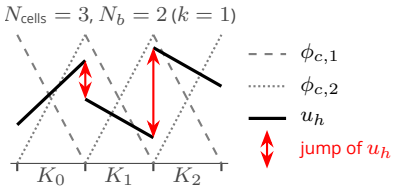
[1] [Arnold et al., 2002]

Discontinuous Galerkin (DG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω of cells $(K_c)_{0 \leq c < N_{\text{cells}}}$.

Let V_h^{disc} be a discontinuous space of piecewise polynomials of degree k on the mesh $\mathcal{T}_h$. Its basis functions $(\phi_{c,i})_{c,i}$, with $\phi_{c,i}$ supported on a single cell K_c, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{c,i} \beta_{c,i}(\boldsymbol{\mu})\, \phi_{c,i}(\boldsymbol{x}).$$



Coupling the cells: ambiguous traces on the interfaces $\Rightarrow$ numerical flux (it defines the method)

Diffusion-dominated regimes: flux for the diffusion term only, here SIPG (Symmetric Interior Penalty Galerkin)

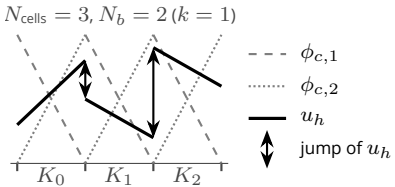
[1] [Arnold et al., 2002]

Discontinuous Galerkin (DG) method[1]

For a fixed $\boldsymbol{\mu} \in \mathcal{M}$, let $\mathcal{T}_h$ be a mesh of Ω of cells $(K_c)_{0 \leq c < N_{\text{cells}}}$.

Let V_h^{disc} be a discontinuous space of piecewise polynomials of degree k on the mesh $\mathcal{T}_h$. Its basis functions $(\phi_{c,i})_{c,i}$, with $\phi_{c,i}$ supported on a single cell K_c, decompose the discrete solution:

$$u_h(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{c,i} \beta_{c,i}(\boldsymbol{\mu})\, \phi_{c,i}(\boldsymbol{x}).$$



Variational problem (SIPG):

Find $u_h \in V_h^{\text{disc}}$ such that, $\forall v_h \in V_h^{\text{disc}}$, $a_h^{\text{disc}}(u_h, v_h; \boldsymbol{\mu}) = l(v_h; \boldsymbol{\mu}) = \int_\Omega f\, v_h$, with

$$a_h^{\text{disc}}(u_h, v_h; \boldsymbol{\mu}) = \underbrace{\int_\Omega \left(R\, u_h v_h + \boldsymbol{C} \cdot \nabla_h u_h\, v_h + \frac{1}{\text{Pe}} D \nabla_h u_h \cdot \nabla_h v_h \right)}_{\text{cell terms}} \underbrace{- \frac{1}{\text{Pe}} \int_{\mathcal{F}_h} \left([\![u_h]\!] \cdot \{D \nabla_h v_h\} + \{D \nabla_h u_h\} \cdot [\![v_h]\!] \right)}_{\text{consistency}} \underbrace{+ \frac{1}{\text{Pe}} \int_{\mathcal{F}_h} \kappa\, [\![u_h]\!] \cdot [\![v_h]\!]}_{\text{penalty}}$$

where $\mathcal{F}_h$ is the set of faces, ∇_h the broken gradient, $[\![\cdot]\!]$ the jump, $\{\cdot\}$ the average, κ the penalty parameter.

[1] [Arnold et al., 2002]