Hybrid methods for elliptic PDEs

Enriching continuous Lagrange finite element approximation spaces

Victor Michel-Dansac*, joint work with
Hélène Barucq, Michel Duprez, Florian Faucher,
Emmanuel Franck, Frédérique Lecourtier, Vanessa Lleras, Nicolas Victorion

✨✨SPARCL✨✨ workshop, Paris, 08 April 2026

*MACARON project-team, Université de Strasbourg, CNRS, Inria, IRMA, France

1. Approximation 1
   1.1. Linear approximation spaces 2
   1.2. Nonlinear approximation spaces 7
2. Stationary PDEs 16
3. Enriching cG approximation spaces with neural networks (Barucq et al., M2AN, 2026) 21
   3.1. Enriched approximation spaces 22
   3.2. Numerical experiments 26
4. Conclusion and perspectives 36

1. Approximation

Numerical analysis and supervised learning

Function approximation is paramount in several fields of applied mathematics. Examples include **supervised learning** and **numerical analysis**.

- **Numerical analysis**: Given a differential operator $\mathcal{D}$, we seek a function u such that $\mathcal{D}[u] \approx 0$.
- **Supervised learning**: Given a set of data $(x_k, y_k)_{k=1}^K$, we seek a function u such that for all $k \in \{1, \dots, K\}$, $u(x_k) \approx y_k$.

In both cases, the sought solution u is in some infinite-dimensional Hilbert space $\mathcal{H}$.

Main idea: reduce this problem to a finite-dimensional one by approximating u in a **finite-dimensional function space** $V_N \subset \mathcal{H}$.

1.1. Linear approximation spaces

Linear approximation spaces: definition

Definition (Linear approximation space): Let $\Omega \in \mathbb{R}^d$ be a domain and $\mathcal{H} \subset L^2(\Omega)$ be a Hilbert space of functions defined on Ω. A linear approximation space V_N is defined by

$$V_N = \mathrm{Span}(\varphi_1, \dots, \varphi_n) = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\},$$

where $\varphi = (\varphi_j)_{j=1}^N \in \mathcal{H}^N$ are the basis functions of V_N and $\theta = (\theta_j)_{j=1}^N \in \Theta \subset \mathbb{R}^N$ are the parameters to be determined. The parameters θ are called **degrees of freedom** (dofs).

The choice of the basis functions φ completely defines the approximation space V_N.

Linear approximation spaces: definition

Definition (Linear approximation space): Let $\Omega \in \mathbb{R}^d$ be a domain and $\mathcal{H} \subset L^2(\Omega)$ be a Hilbert space of functions defined on Ω. A linear approximation space V_N is defined by

$$V_N = \mathrm{Span}(\varphi_1, \dots, \varphi_n) = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\},$$

where $\varphi = (\varphi_j)_{j=1}^N \in \mathcal{H}^N$ are the basis functions of V_N and $\theta = (\theta_j)_{j=1}^N \in \Theta \subset \mathbb{R}^N$ are the parameters to be determined. The parameters θ are called **degrees of freedom** (dofs).

The choice of the basis functions φ completely defines the approximation space V_N.

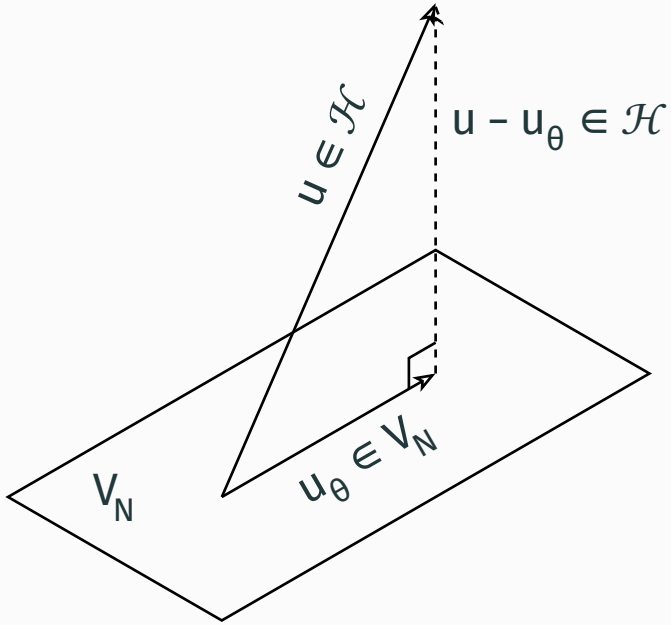
Goal: Let $u \in \mathcal{H}$. Given a finite-dimensional subspace $V_N \subset \mathcal{H}$, we seek to approximate u by a function $u_\theta \in V_N$.

⇝ How to find the dofs θ?

Linear approximation spaces: orthogonal projection

Finding u_θ amounts to determining the dofs θ such that the distance in $\mathcal{H}$ between u et u_θ is minimized:

$$\theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \mathcal{J}(\vartheta) = \operatorname*{argmin}_{\vartheta \in \Theta} \frac{1}{2}\|u - u_\vartheta\|_{\mathcal{H}}^2 = \operatorname*{argmin}_{\vartheta \in \Theta} \frac{1}{2}\int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx.$$



We thus seek $u_\theta \in V_N$ such that, for all $v \in V_N$, $\langle u - u_\theta, v\rangle_{\mathcal{H}} = 0$. This amounts to an orthogonal projection of $u \in \mathcal{H}$ onto V_N.

Therefore, we seek $\theta = (\theta_j)_{j=1}^N$ such that

$$\forall i \in \{1, \dots, N\}, \quad \sum_{j=1}^N \theta_j \langle \varphi_i, \varphi_j\rangle_{\mathcal{H}} = \langle u, \varphi_i\rangle_{\mathcal{H}}.$$

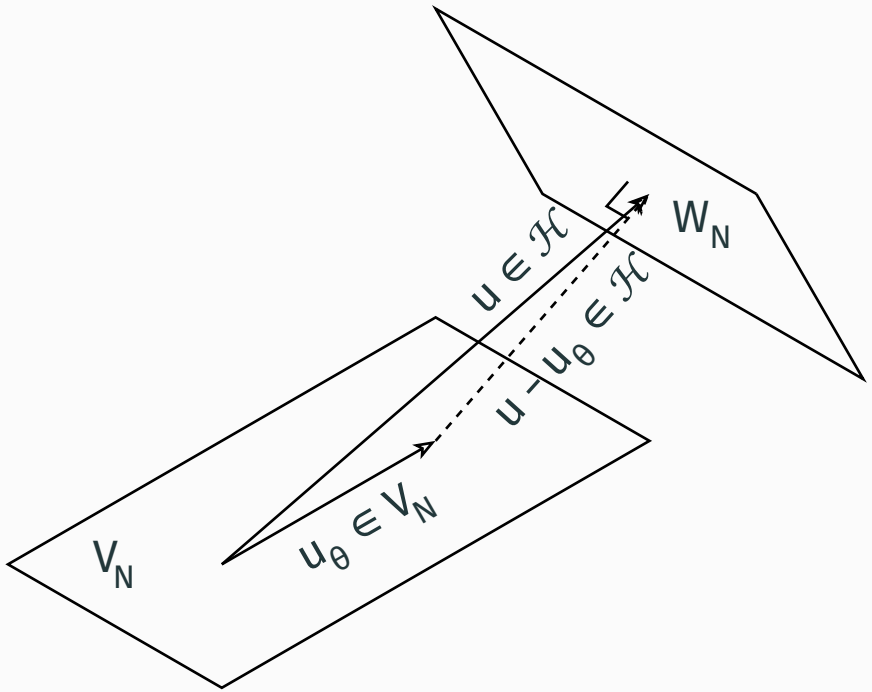
This is nothing but a **linear system** $M\theta = U$, where the matrix M and right-hand side U are given by

$$M_{ij} = \langle \varphi_i, \varphi_j\rangle_{\mathcal{H}}, \qquad U_i = \langle u, \varphi_i\rangle_{\mathcal{H}}.$$

Linear approximation spaces: oblique projection

To find $u_\theta \in V_N$, we compute the dofs θ such that the residual is orthogonal to a test space $W_N = \mathrm{Span}(\psi_1, \dots, \psi_n)$:

$$u_\theta \in V_N \quad \text{s.t.} \quad \langle u - u_\theta, w \rangle_{\mathcal{H}} = 0, \quad \forall w \in W_N.$$



We seek $u_\theta \in V_N$ such that the $u - u_\theta$ is orthogonal to W_N.
If $V_N = W_N$, we recover the **orthogonal projection**.
If $V_N \neq W_N$, this is an **oblique projection**.

Therefore, we seek $\theta = (\theta_j)_{j=1}^N$ such that

$$\forall i \in \{1, \dots, N\}, \quad \sum_{j=1}^{N} \theta_j \langle \varphi_j, \psi_i \rangle_{\mathcal{H}} = \langle u, \psi_i \rangle_{\mathcal{H}}.$$

This yields the linear system $K\theta = U$, where:

$$K_{ij} = \langle \varphi_j, \psi_i \rangle_{\mathcal{H}}, \quad U_i = \langle u, \psi_i \rangle_{\mathcal{H}}.$$

The matrix K is often called the **Petrov-Galerkin** or **trial-test** matrix.

Degrees of freedom for finite volume and RBF methods

- As an example, the **finite volume method** introduces a mesh made of cells Ω_j on the domain Ω. Then, the basis functions are indicator functions: $\varphi_j = \chi_{\Omega_j}$. In this case, the right-hand side and matrix are given by

$$U_i = \int_\Omega u(x)\,\varphi_i(x)\,dx = \int_{\Omega_i} u(x)\,dx \qquad \text{and} \qquad M_{ij} = \int_\Omega \varphi_i(x)\varphi_j(x)\,dx = \begin{cases} |\Omega_j| \text{ if } i = j\,, \\ 0 \text{ otherwise.} \end{cases}$$

Therefore, the degrees of freedom θ are given by

$$\theta_j = \frac{1}{|\Omega_j|}\int_{\Omega_j} u(x)\,dx \quad \Longrightarrow \quad u_\theta = \sum_{j=1}^{N} \frac{\chi_{\Omega_j}}{|\Omega_j|}\int_{\Omega_j} u(x)\,dx.$$

Degrees of freedom for finite volume and RBF methods

- As an example, the **finite volume method** introduces a mesh made of cells Ω_j on the domain Ω. Then, the basis functions are indicator functions: $\varphi_j = \chi_{\Omega_j}$. In this case, the right-hand side and matrix are given by

$$U_i = \int_\Omega u(x)\,\varphi_i(x)\,dx = \int_{\Omega_i} u(x)\,dx \qquad \text{and} \qquad M_{ij} = \int_\Omega \varphi_i(x)\varphi_j(x)\,dx = \begin{cases} |\Omega_j| \text{ if } i = j\,, \\ 0 \text{ otherwise.} \end{cases}$$

  Therefore, the degrees of freedom θ are given by

$$\theta_j = \frac{1}{|\Omega_j|}\int_{\Omega_j} u(x)\,dx \quad \Longrightarrow \quad u_\theta = \sum_{j=1}^{N} \frac{\chi_{\Omega_j}}{|\Omega_j|}\int_{\Omega_j} u(x)\,dx.$$

- Another example is the **Gaussian radial basis function methods**, where centers $c_j \in \Omega$ are defined, and the basis functions are Gaussian functions:

$$\varphi_j(x) = \exp\left(-\frac{|x - c_j|^2}{2\sigma^2}\right).$$

In this case, finding the degrees of freedom require solving a linear system with a full matrix.

Advantages and drawbacks of linear methods

Advantage (Convergence): Linear function approximation methods generally satisfy, for some $p > 0$,

$$\left\|u - u_\theta\right\|_{\mathcal{H}} = \mathcal{O}\left(N^{-\frac{p}{d}}\right) \quad \text{or} \quad \left\|u - u_\theta\right\|_{\mathcal{H}} = \mathcal{O}\left(e^{-cN^{\frac{1}{d}}}\right).$$

This is a great convergence result! As the number N of dofs increases, the error quantifiably decreases.

Advantages and drawbacks of linear methods

Advantage (Convergence): Linear function approximation methods generally satisfy, for some $p > 0$,

$$\left\|u - u_\theta\right\|_{\mathcal{H}} = \mathcal{O}\left(N^{-\frac{p}{d}}\right) \quad \text{or} \quad \left\|u - u_\theta\right\|_{\mathcal{H}} = \mathcal{O}\left(e^{-cN^{\frac{1}{d}}}\right).$$

This is a great convergence result! As the number N of dofs increases, the error quantifiably decreases.

Drawback (Curse of dimensionality): To reach a given error target ε, one requires roughly

$$N = \mathcal{O}\left(\varepsilon^{-\frac{d}{p}}\right) \quad \text{or} \quad N = \mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)^d\right)$$

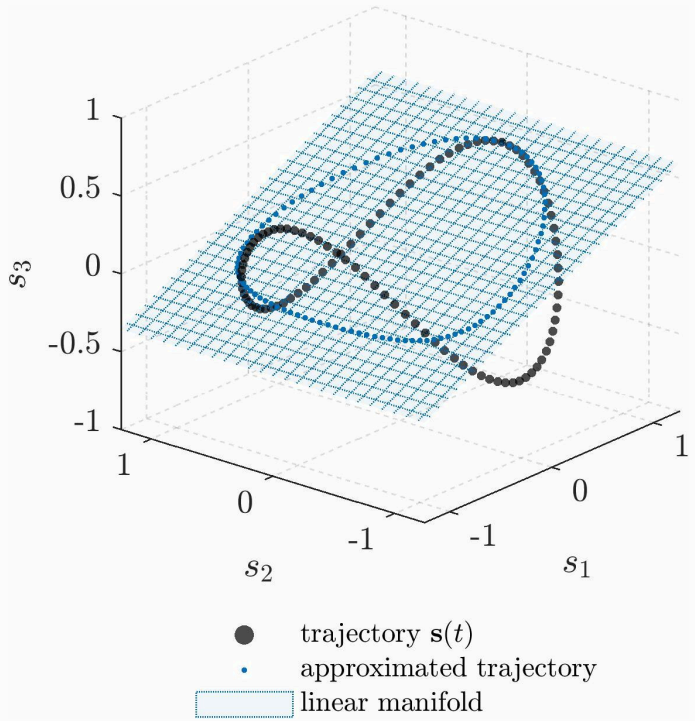
dofs for a d-dimensional domain Ω.

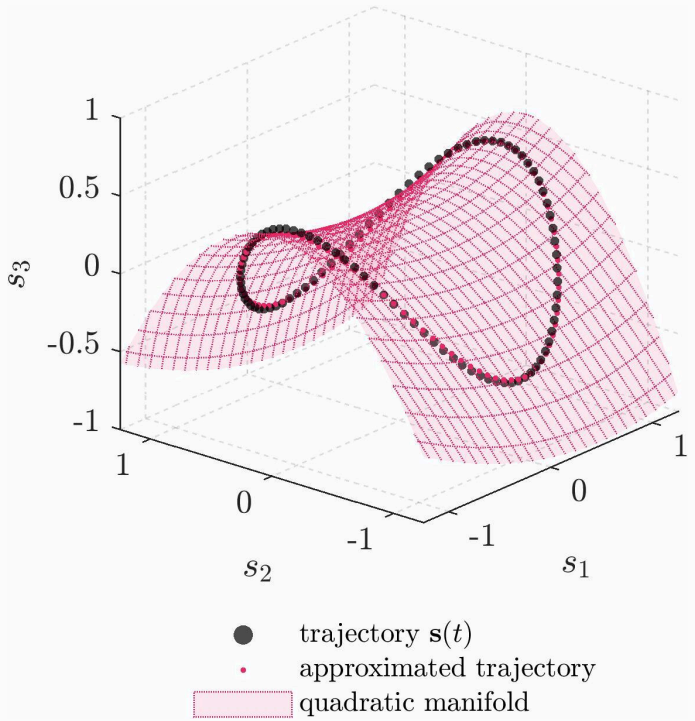
This is a disheartening result, as the number of dofs increases exponentially with the dimension of the domain.

1.2. Nonlinear approximation spaces

Linear vs nonlinear approximation spaces

Nonlinear numerical methods rely on replacing the linear subspace $V_N \in \mathcal{H}$ with a nonlinear subspace $\mathcal{V}_N \in \mathcal{H}$.





figures from Geelen et al, 2022

Example: linear vs nonlinear RBFs

For **linear Gaussian RBFs**, the centers $c_j \in \Omega$ and the variance σ of the Gaussian basis functions are fixed:

$$u_\theta(x) = \sum_{j=1}^{N} \theta_j \exp\left(-\frac{|x - c_j|^2}{2\sigma^2}\right).$$

Example: linear vs nonlinear RBFs

For **linear Gaussian RBFs**, the centers $c_j \in \Omega$ and the variance σ of the Gaussian basis functions are fixed:

$$u_\theta(x) = \sum_{j=1}^{N} \theta_j \exp\left(-\frac{|x - c_j|^2}{2\sigma^2}\right).$$

For **nonlinear Gaussian RBFs**, the centers and variances are also dofs:

$$u_{\alpha,c,\sigma}(x) = \sum_{j=1}^{N} \alpha_j \exp\left(-\frac{|x - c_j|^2}{2\sigma_j^2}\right).$$

We define a vector $\theta = (\alpha_j, c_j, \sigma_j)_{j=1}^N \in \mathbb{R}^{3N}$ and a nonlinear function $\mathcal{N}$ such that

$$u_\theta(x) = \mathcal{N}(x;\theta) = \sum_{j=1}^{N} \alpha_j \exp\left(-\frac{|x - c_j|^2}{2\sigma_j^2}\right).$$

Nonlinear approximation spaces

Definition (Nonlinear approximation space): Let $\mathcal{N} : \Omega \times \mathbb{R}^N \to \mathbb{R}$ be a nonlinear function. The nonlinear approximation space $\mathcal{V}_N \subset \mathcal{H}$, based on $\mathcal{N}$, is defined by

$$\mathcal{V}_N = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \mathcal{N}(x;\theta) \right\},$$

where $\theta = (\theta_j)_{j=1}^N \in \Theta \subset \mathbb{R}^N$ remain the dofs to be determined.

Nonlinear approximation spaces

Definition (Nonlinear approximation space): Let $\mathcal{N} : \Omega \times \mathbb{R}^N \to \mathbb{R}$ be a nonlinear function. The nonlinear approximation space $\mathcal{V}_N \subset \mathcal{H}$, based on $\mathcal{N}$, is defined by

$$\mathcal{V}_N = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \mathcal{N}(x;\theta) \right\},$$

where $\theta = (\theta_j)_{j=1}^N \in \Theta \subset \mathbb{R}^N$ remain the dofs to be determined.

Remark: Such nonlinear spaces may be rewritten as

$$\mathcal{V}_N = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \alpha \in \mathbb{R}^{N_1}, \exists \beta \in \mathbb{R}^{N_2}, \forall x \in \Omega,\ u_{\alpha,\beta}(x) = \sum_{j=1}^{N_1} \alpha_j \varphi_j(x,\beta) \right\},$$

and so we may think of them as linear spaces with nonlinear basis functions.

Determining the degrees of freedom

To approximate u in $\mathcal{V}_N$, we seek the dofs θ in a **Petrov-Galerkin** (oblique projection) way:

1. consider an approximation u_θ in $\mathcal{V}_N$ with dofs θ: $u_\theta(x) = \mathcal{N}(x;\theta)$;
2. define m *collocation points* $(x_1, \dots, x_m) \in \Omega^m$;
3. define the test space $W_n = \mathrm{Span}\,(\delta_{x_1}, \dots, \delta_{x_m})$;
4. **project the residual** $u - u_\theta$ onto W_n:

$$\text{find } \theta \text{ such that } \quad \forall i \in \{1,\dots,m\}, \quad \langle u - u_\theta, \delta_{x_i} \rangle_{\mathcal{H}} = 0 \qquad \Longleftrightarrow \qquad \theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \sum_{i=1}^{m} \left| u(x_i) - u_\vartheta(x_i) \right|^2.$$

Determining the degrees of freedom

To approximate u in $\mathcal{V}_N$, we seek the dofs θ in a **Petrov-Galerkin** (oblique projection) way:

1. consider an approximation u_θ in $\mathcal{V}_N$ with dofs θ: $u_\theta(x) = \mathcal{N}(x;\theta)$;
2. define m *collocation points* $(x_1, \dots, x_m) \in \Omega^m$;
3. define the test space $W_n = \mathrm{Span}\,(\delta_{x_1}, \dots, \delta_{x_m})$;
4. **project the residual** $u - u_\theta$ onto W_n:

$$\text{find } \theta \text{ such that } \quad \forall i \in \{1,\dots,m\}, \quad \langle u - u_\theta, \delta_{x_i} \rangle_{\mathcal{H}} = 0 \qquad \Longleftrightarrow \qquad \theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \sum_{i=1}^{m} \left| u(x_i) - u_\vartheta(x_i) \right|^2.$$

This leads to a **collocation method**, with the following advantages and drawbacks:

😄 **No mesh is required**; we need to "sample" the domain to obtain the collocation points.
😄 **High dimensions** become reachable, even with a low number of dofs.
😱 We obtain a **non-quadratic and non-convex minimization problem** in θ! This does not lead to a linear system.

Error sources

We introduce the **approximation error** $E_{app} = \|u - u_\theta\|_{\mathcal{H}}$. In a perfect world, we would minimize this error:

$$\theta = \operatorname*{argmin}_{\vartheta \in \Theta} \mathcal{E}(\vartheta) = \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx.$$

Error sources

We introduce the **approximation error** $E_{app} = \|u - u_\theta\|_{\mathcal{H}}$. In a perfect world, we would minimize this error:

$$\theta = \operatorname*{argmin}_{\vartheta \in \Theta} \mathcal{E}(\vartheta) = \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx.$$

We approximate the integral $\mathcal{E}$ by $\mathcal{E}_m$:

$$\int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx \approx \frac{|\Omega|}{m} \sum_{i=1}^{m} |u(x_i) - u_\vartheta(x_i)|^2.$$

This introduces an **integration error**

$$E_{int} = \sup_{\theta \in \Theta} |\mathcal{E}(\theta) - \mathcal{E}_m(\theta)|.$$

Error sources

We introduce the **approximation error** $E_{app} = \|u - u_\theta\|_{\mathcal{H}}$. In a perfect world, we would minimize this error:

$$\theta = \operatorname*{argmin}_{\vartheta \in \Theta} \mathcal{E}(\vartheta) = \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx.$$

We approximate the integral $\mathcal{E}$ by $\mathcal{E}_m$:

$$\int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx \approx \frac{|\Omega|}{m} \sum_{i=1}^{m} |u(x_i) - u_\vartheta(x_i)|^2.$$

This introduces an **integration error**

$$E_{int} = \sup_{\theta \in \Theta} |\mathcal{E}(\theta) - \mathcal{E}_m(\theta)|.$$

We approximate the solution of the optimization problem, which yields an approximate solution θ^*.

This introduces an **optimization error**

$$E_{opt} = \mathcal{E}_m(\theta^*) - \min_\theta \mathcal{E}_m(\theta).$$

Error sources

We introduce the **approximation error** $E_{app} = \|u - u_\theta\|_{\mathcal{H}}$. In a perfect world, we would minimize this error:

$$\theta = \operatorname*{argmin}_{\vartheta \in \Theta} \mathcal{E}(\vartheta) = \operatorname*{argmin}_{\vartheta \in \Theta} \int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx.$$

We approximate the integral $\mathcal{E}$ by $\mathcal{E}_m$:

$$\int_\Omega |u(x) - u_\vartheta(x)|^2 \, dx \approx \frac{|\Omega|}{m} \sum_{i=1}^{m} |u(x_i) - u_\vartheta(x_i)|^2.$$

This introduces an **integration error**

$$E_{int} = \sup_{\theta \in \Theta} |\mathcal{E}(\theta) - \mathcal{E}_m(\theta)|.$$

We approximate the solution of the optimization problem, which yields an approximate solution θ^*.

This introduces an **optimization error**

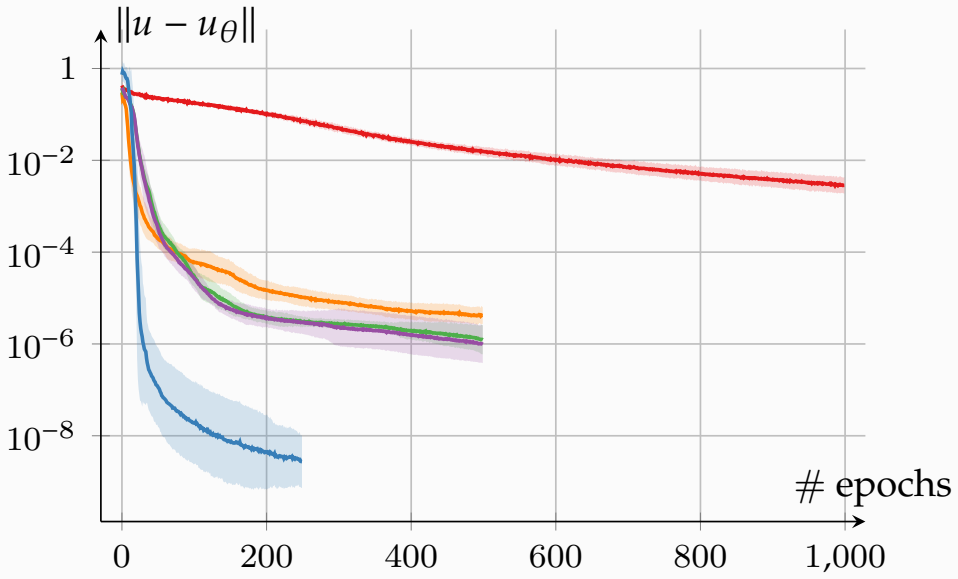
$$E_{opt} = \mathcal{E}_m(\theta^*) - \min_\theta \mathcal{E}_m(\theta).$$

-For **linear methods**, in general, $E_{app} > E_{int} > E_{opt} \approx 0$.

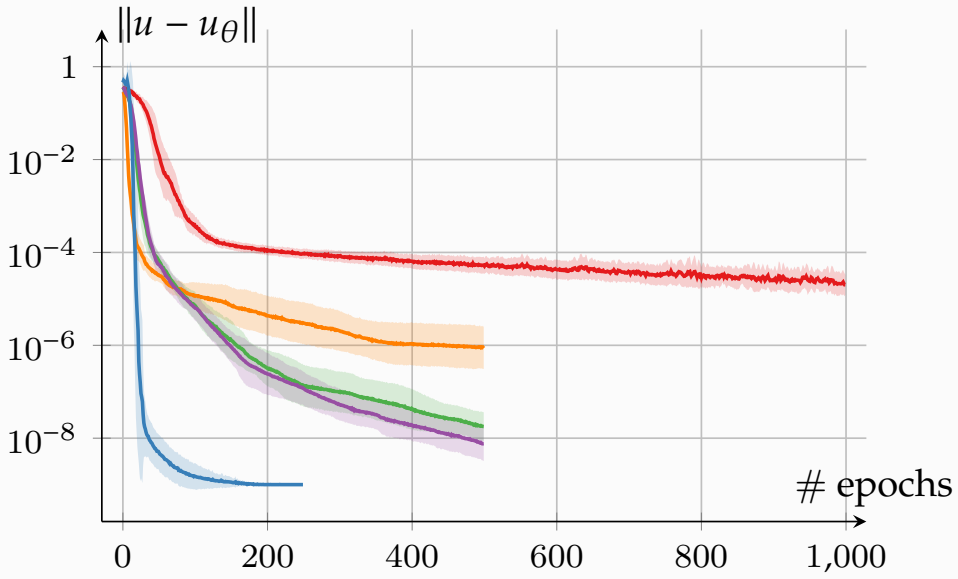
-For **nonlinear methods**, in general, $E_{opt} > E_{int} > E_{app}$: one needs a very high precision on the optimization!

Comparison of nonlinear optimizers for a small model

(a) approximation with 128 dofs

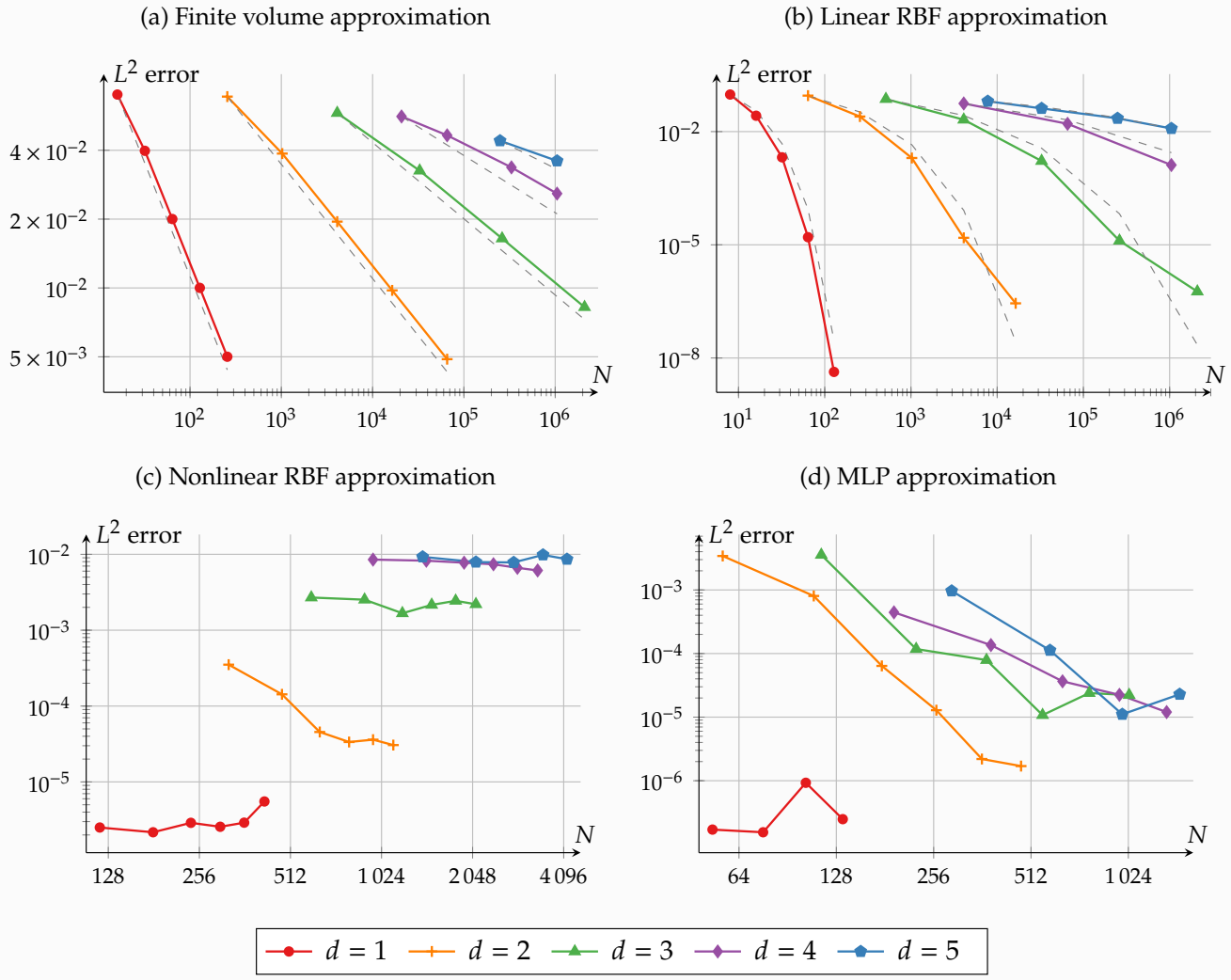


(b) approximation with 608 dofs



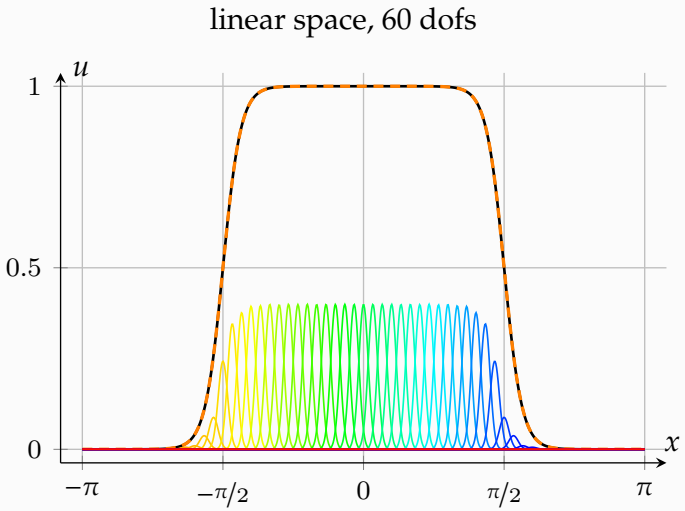
Conclusion: Even on a simple problem, optimization efficiency is crucial for obtaining good accuracy.

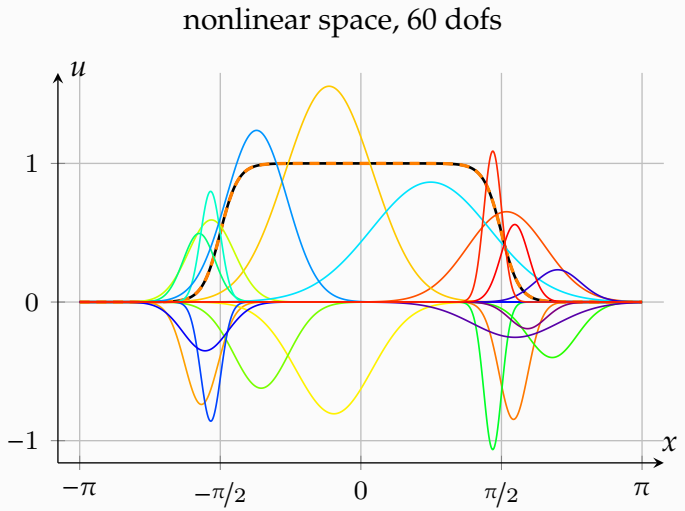
Numerical experiment: convergence with a smooth target



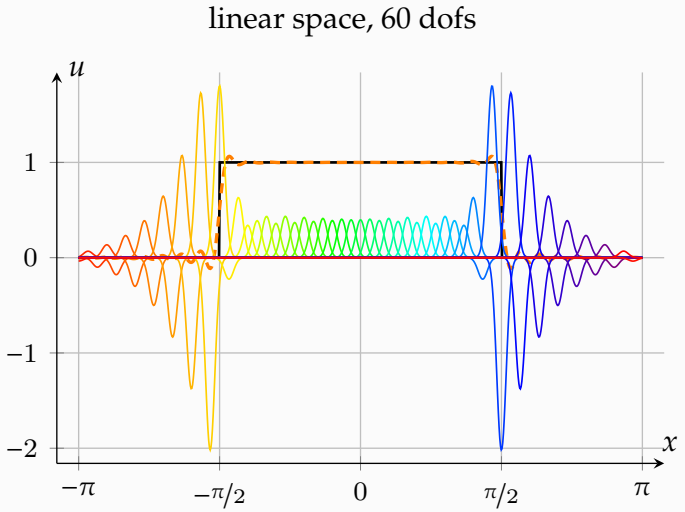
- The nonlinear RBFs have trouble converging but yield a better error than the linear ones.
- The bottom-right convergence plot is obtained when $\mathcal{N}$ is a multilayer perceptron (MLP, a fully-connected neural network).
- We observe a somewhat better convergence rate with the MLP compared to the RBFs.

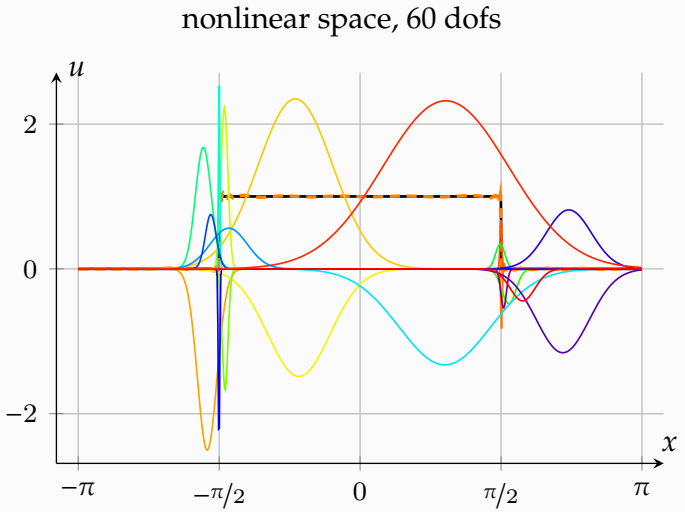
Numerical experiment: displaying the RBFs





The nonlinear RBFs are able to adapt their centers and variances to better approximate the *smooth* target function.





When the target function is *non-smooth*, some very spiky nonlinear RBFs are concentrated around the discontinuity.

Advantages and drawbacks

Advantages:
- **Universal approximation theorems**: for some neural network architectures or RBFs, $\mathcal{V}_N$ is **dense** in $\mathcal{H}$.
- The number of dofs is **less dependent on the dimension** of the domain Ω than for linear methods.
- The method **does not require meshing** Ω, which is a great advantage for complex geometries.

Drawbacks:
- Despite the density result, **convergence rates (as the dofs increase) are unavailable**.
- Theoretical results on solving the optimization problem are still lacking. In practice, **the optimization problem is very difficult to solve** (non-convex, non-quadratic, etc.).

In spite of these drawbacks, dimension-insensitivity and the lack of meshing are very appealing features of neural numerical methods, especially when approximating solutions of PDEs in high dimensions.

2. Stationary PDEs

Linear methods for elliptic problems

With $\mathcal{D}$ a differential operator, we consider the elliptic problem (disregarding boundary conditions):

$$\forall x \in \Omega, \quad \mathcal{D}[u](x) = f(x).$$

Similarly to the function approximation task, we define a **linear approximation space** V_N:

$$V_N = \left\{ f_\theta =: \mathcal{D}[u_\theta] \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

Linear methods for elliptic problems

With $\mathcal{D}$ a differential operator, we consider the elliptic problem (disregarding boundary conditions):

$$\forall x \in \Omega, \quad \mathcal{D}[u](x) = f(x).$$

Similarly to the function approximation task, we define a **linear approximation space** V_N:

$$V_N = \left\{ f_\theta =: \mathcal{D}[u_\theta] \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

We then solve the **oblique projection**:

$$\text{find } f_\theta \in V_N \text{ s.t. } \forall w \in W_N, \quad \langle f - f_\theta, w \rangle_{\mathcal{H}} = 0,$$

$$\text{or equivalently, find } \theta \in \Theta \text{ s.t. } \forall w \in W_N, \quad \langle f - \mathcal{D}[u_\theta], w \rangle_{\mathcal{H}} = 0.$$

Linear methods for elliptic problems

With $\mathcal{D}$ a differential operator, we consider the elliptic problem (disregarding boundary conditions):

$$\forall x \in \Omega, \quad \mathcal{D}[u](x) = f(x).$$

Similarly to the function approximation task, we define a **linear approximation space** V_N:

$$V_N = \left\{ f_\theta =: \mathcal{D}[u_\theta] \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

We then solve the **oblique projection**:

$$\text{find } f_\theta \in V_N \quad \text{s.t.} \quad \forall w \in W_N, \quad \langle f - f_\theta, w \rangle_{\mathcal{H}} = 0,$$

$$\text{or equivalently, find } \theta \in \Theta \quad \text{s.t.} \quad \forall w \in W_N, \quad \langle f - \mathcal{D}[u_\theta], w \rangle_{\mathcal{H}} = 0.$$

If $W_N = V_N$, we get the **Galerkin method**:

$$\forall i \in \{1, \dots, N\}, \quad \left\langle \mathcal{D}\left[\sum_{j=1}^{N} \theta_j \varphi_j\right], \varphi_i \right\rangle_{\mathcal{H}} = \langle f, \varphi_i \rangle_{\mathcal{H}}.$$

If $W_n = \text{Span}(\delta_{x_1}, \dots, \delta_{x_m})$, we get the **collocation method**:

$$\forall i \in \{1, \dots, N\}, \quad \mathcal{D}\left[\sum_{j=1}^{N} \theta_j \varphi_j\right](x_i) = f(x_i).$$

Nonlinear method for elliptic problems

Just like before, we define a nonlinear approximation space $\mathcal{V}_N$:

$$\mathcal{V}_N = \left\{ f_\theta =: \mathcal{D}[u_\theta] \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \mathcal{N}(x;\theta) \right\}.$$

To approximate f in $\mathcal{V}_N$, we seek the dofs θ in a **Petrov-Galerkin** (oblique projection) way:

1. consider an approximation f_θ in $\mathcal{V}_N$ with dofs θ: $f_\theta(x) = \mathcal{D}[\mathcal{N}(\cdot\,;\theta)](x)$;
2. define m *collocation points* $(x_1, \dots, x_m) \in \Omega^m$;
3. define the test space $W_n = \mathrm{Span}\,(\delta_{x_1}, \dots, \delta_{x_m})$;
4. **project the residual** $f - f_\theta = f - \mathcal{D}[u_\theta]$ onto W_n:

$$\theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \sum_{i=1}^{m} \left| \mathcal{D}[u_\vartheta](x_i) - f(x_i) \right|^2.$$

This formulation includes **physics-informed neural networks** (PINNs, [Raissy et al., 2019]).

Parameter-dependent PDEs

We can also consider parameter-dependent PDEs, where the solution depends on some parameters $\mu \in \mathbb{M} \subset \mathbb{R}^p$:

$$\forall x \in \Omega, \quad \mathcal{D}[u](x, \mu) = f(x, \mu).$$

We emphasize that u **is treated as a function of the joint variable** $(x, \mu) \in \Omega \times \mathbb{M}$. The current approach is **monolithic**: the parameters are seen as additional coordinates in the input space.

Parameter-dependent PDEs

We can also consider parameter-dependent PDEs, where the solution depends on some parameters $\mu \in \mathbb{M} \subset \mathbb{R}^p$:

$$\forall x \in \Omega, \quad \mathcal{D}[u](x, \mu) = f(x, \mu).$$

We emphasize that u **is treated as a function of the joint variable** $(x, \mu) \in \Omega \times \mathbb{M}$. The current approach is **monolithic**: the parameters are seen as additional coordinates in the input space.

We can then define a parametric nonlinear approximation space

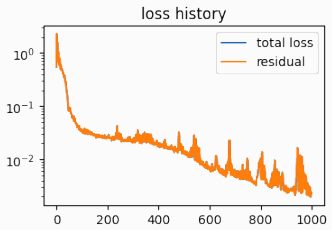
$$\mathcal{V}_N = \left\{ f_\theta =: \mathcal{D}[u_\theta] \in L^2(\Omega \times \mathbb{M}) \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega, \forall \mu \in \mathbb{M},\ u_\theta(x, \mu) = \mathcal{N}(x, \mu; \theta) \right\}.$$

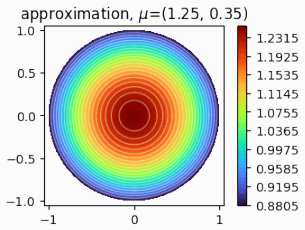
The projection is treated in the same way as before, but now the collocation points are defined in $\Omega \times \mathbb{M}$:

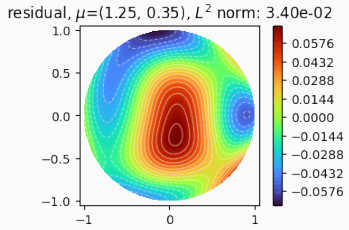
$$\theta \in \operatorname*{argmin}_{\vartheta \in \Theta} \sum_{i=1}^{m} \left| \mathcal{D}[u_\theta](x_i, \mu_i) - f(x_i, \mu_i) \right|^2.$$

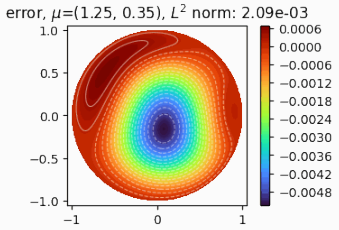
Result on a parametric Poisson problem in a disk

Adam

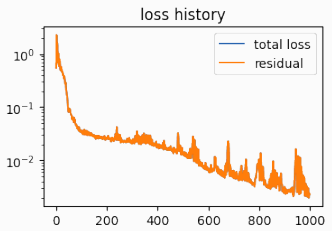


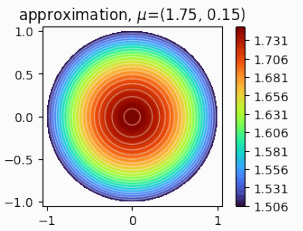


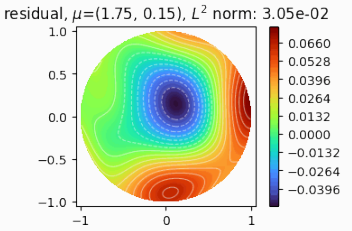


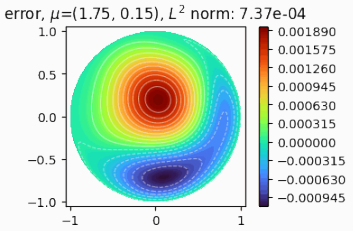


Adam

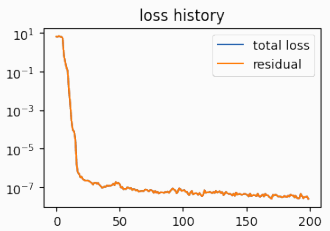


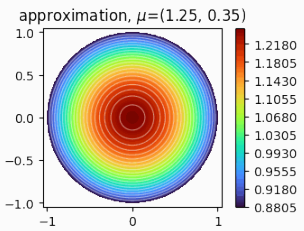


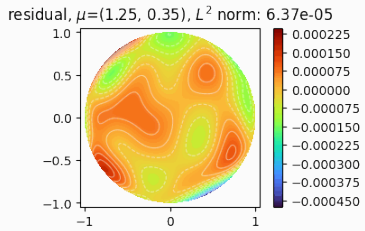


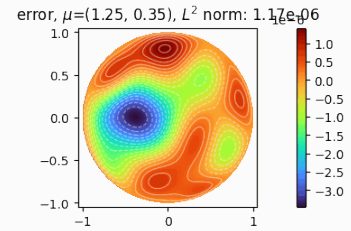


ENG

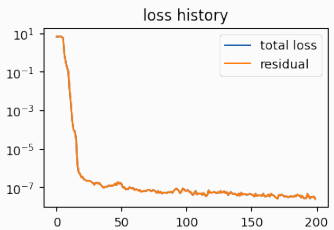


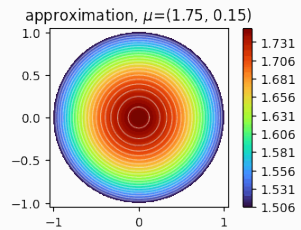


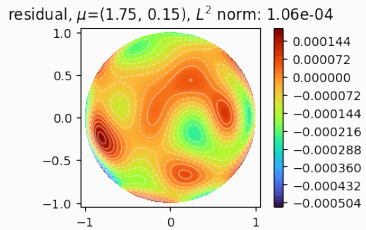


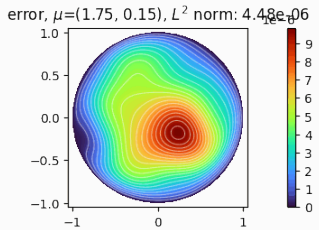


ENG









- parametric Poisson problem:

$$-\mu_1 \Delta u(x, y, \mu_1, \mu_2) = f(x, y, \mu_1, \mu_2)$$

2D (space) + 2D (parameters) = 4D problem, solved with only 960 dofs!

- At the end of the optimization, we obtain a vector $\theta \in \Theta$, and so a function $u_\theta : \Omega \times \mathbb{M} \to \mathbb{R}$.
We can evaluate this function for any $x \in \Omega$ and $\mu \in \mathbb{M}$.

- Once again, it is essential to have a good optimization strategy to obtain good accuracy.

Advantages and drawbacks

Advantages:
- Very few dofs are required to reach good accuracy, even in high dimensions.
- Complex domains may be easier to deal with (no meshing required).

Drawbacks:
- The cost per dof is generally much higher than with linear methods.
- No good convergence results are available. In practice, adding dofs does not necessarily reduce the error.

Advantages and drawbacks

Advantages:
- Very few dofs are required to reach good accuracy, even in high dimensions.
- Complex domains may be easier to deal with (no meshing required).

Drawbacks:
- The cost per dof is generally much higher than with linear methods.
- No good convergence results are available. In practice, adding dofs does not necessarily reduce the error.

The main objectives of this work are to improve:
- **the accuracy of parametric PINNs**, and
- **the error constant of linear methods**

by **hybridizing PINNs with linear numerical methods**.

3. Enriching cG approximation spaces with neural networks (Barucq et al., M2AN, 2026)

Motivation

- On the one hand, **nonlinear approximation spaces** can be very powerful, but they are **difficult to optimize** and **do not have convergence guarantees**.
- On the other hand, **linear approximation spaces** have **convergence guarantees**, but suffer from the **curse of dimensionality**.

⇝ Can we combine the best of both worlds, in the continuous Galerkin (cG) framework?

Motivation

- On the one hand, **nonlinear approximation spaces** can be very powerful, but they are **difficult to optimize** and **do not have convergence guarantees**.
- On the other hand, **linear approximation spaces** have **convergence guarantees**, but suffer from the **curse of dimensionality**.

⇝ Can we combine the best of both worlds, in the continuous Galerkin (cG) framework?

We propose a two-step hybrid method:

1. **Offline phase**: train a neural network to approximate a large family of solutions to the PDE;
2. **Online phase**: use the trained network to correct the cG approximation space, making it possible to run the cG simulations on a coarse grid.

3.1. Enriched approximation spaces

Classical cG (finite element) method

The classical cG method relies on a **approximation space V_N** with **piecewise polynomial basis functions**:

$$V_N = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

The number N of dofs increases with the number of mesh points, the polynomial degree, and the dimension.

Classical cG (finite element) method

The classical cG method relies on a **approximation space V_N** with **piecewise polynomial basis functions**:

$$V_N = \left\{ u_\theta \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_\theta(x) = \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

The number N of dofs increases with the number of mesh points, the polynomial degree, and the dimension.

1. Rewrite the PDE $\mathcal{D}[u] = f$ as a variational problem ($\mathcal{D}$ is supposed to be linear for simplicity):

$$\text{find } u \in \mathcal{H} \text{ such that } a(u, v) = \ell(v), \quad \forall v \in \mathcal{H},$$

where $\mathcal{H} = \mathcal{H}_0^m(\Omega)$ is a Hilbert space, $a(\cdot,\cdot)$ a bilinear form, and $\ell(\cdot)$ a linear form.

2. Discretize the domain Ω and use the **finite-dimensional nonlinear subspace V_N**, to get

$$\text{find } u_N \in V_N \text{ such that } a(u_N, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

3. Solve the above linear system to get the approximation u_N of u.

Enriching the approximation space

Assumption: **Some prior information** $u^\star \in V$ **is known**, giving an approximation of u for all $x \in \Omega$ and $\mu \in \mathbb{M}$.

For instance, it can be given by a PINN trained during the offline phase.

Enriching the approximation space

Assumption: **Some prior information** $u^\star \in V$ **is known**, giving an approximation of u for all $x \in \Omega$ and $\mu \in \mathbb{M}$.

For instance, it can be given by a PINN trained during the offline phase.

We propose to **modify the cG approximation space**, replacing V_N by V_N^+, defined by:

$$V_N^+ = u^\star + V_N = \left\{ u_N^+ = u^\star + \eta_N^+ \quad \text{such that} \quad \eta_N^+ \in V_N \right\}.$$

Hence, η_N^+ is seen as a piecewise polynomial correction to the prior information $u^\star$.

More precisely, the enriched approximation space is defined as

$$V_N^+ = \left\{ u_N^+ \in \mathcal{H} \quad \text{such that} \quad \exists \theta \in \Theta, \forall x \in \Omega,\ u_N^+(x) = u^\star(x) + \sum_{j=1}^{N} \theta_j \varphi_j(x) \right\}.$$

This enriched space is affine, but it remains a linear approximation space!

Determining the dofs in the enriched space

In the non-enriched method, the variational problem is defined on the space V_N:

$$\text{find } u_N \in V_N \text{ such that } a(u_N, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

Determining the dofs in the enriched space

In the non-enriched method, the variational problem is defined on the space V_N:

$$\text{find } u_N \in V_N \text{ such that } a(u_N, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

To determine the enriched dofs, we elect to use a Petrov-Galerkin method, where the test space is still V_N:

$$\text{find } u_N^+ \in V_N^+ \text{ such that } a(u_N^+, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

Determining the dofs in the enriched space

In the non-enriched method, the variational problem is defined on the space V_N:

$$\text{find } u_N \in V_N \text{ such that } a(u_N, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

To determine the enriched dofs, we elect to use a Petrov-Galerkin method, where the test space is still V_N:

$$\text{find } u_N^+ \in V_N^+ \text{ such that } a(u_N^+, v_N) = \ell(v_N), \quad \forall v_N \in V_N.$$

Using the definition of the enriched space, this is equivalent to:

$$\text{find } \eta_N^+ \in V_N \text{ such that } a(\eta_N^+, v_N) = \ell(v_N) - a(u^\star, v_N), \quad \forall v_N \in V_N.$$

The variational problem only has a **modified right-hand side** that incorporates the prior information $u^\star$.

Error analysis

Performing an error analysis of the enriched cG method is straightforward (it uses standard cG techniques). The following result holds.

Theorem: Let $u \in \mathcal{H}_0^m(\Omega)$ be the exact solution of the elliptic problem, $u^\star \in \mathcal{H}_0^m(\Omega)$ a prior, and $u_N^+ \in V_N^+$ the enriched cG solution (considering $\mathbb{P}_q$ polynomials, with $m \leq q$). Then, with h the mesh size, we have:

$$\|u - u_N^+\|_{H^m} \lesssim C_{\text{gain}}^+ \underbrace{h^{q+1-m} |u|_{H^{q+1}}}_{\text{classical cG error}}.$$

The constant $C_{\text{gain}}^+ = \dfrac{|u - u^\star|_{H^{q+1}}}{|u|_{H^{q+1}}}$ represents the potential gain compared the classical cG method.

Key remark: The prior $u^\star$ must be a good approximation of the $(q+1)^{\text{th}}$ derivative of u. This is why we use PINNs, rather than purely data-driven priors!

3.2. Numerical experiments

Poisson problem in a square: setup

We consider a Poisson problem in two space dimensions and with two parameters:

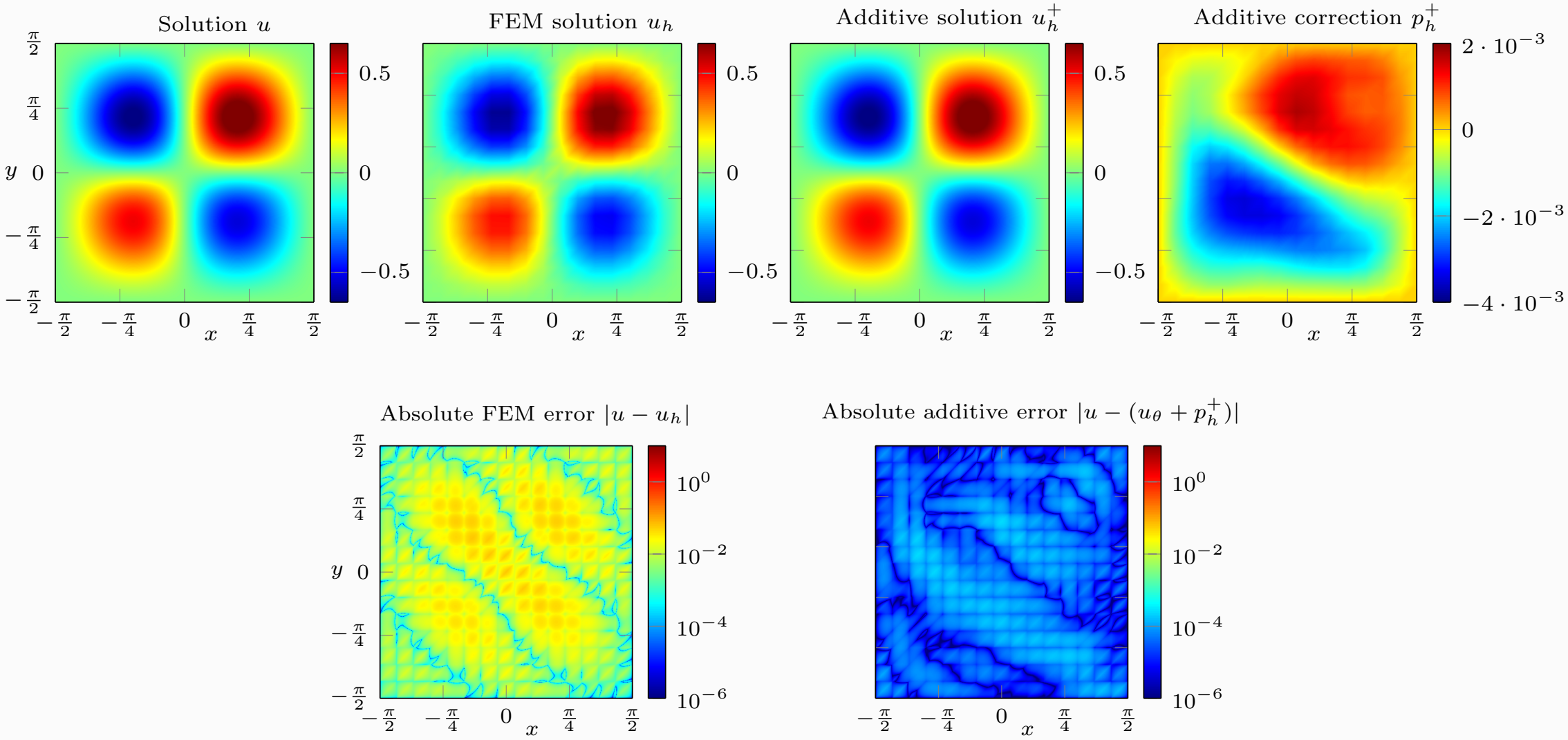
$$\begin{cases} -\Delta u(x, y, x_0, y_0) = f(x, y, x_0, y_0), & (x, y) \in \Omega = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^2, \quad (x_0, y_0) \in \mathbb{M} = \left(-\frac{1}{2}, \frac{1}{2}\right)^2 \\ u(x, y, x_0, y_0) = 0, & (x, y) \in \partial\Omega, \quad (x_0, y_0) \in \mathbb{M} = \left(-\frac{1}{2}, \frac{1}{2}\right)^2. \end{cases}$$

The right-hand side f is defined such that the solution is given by

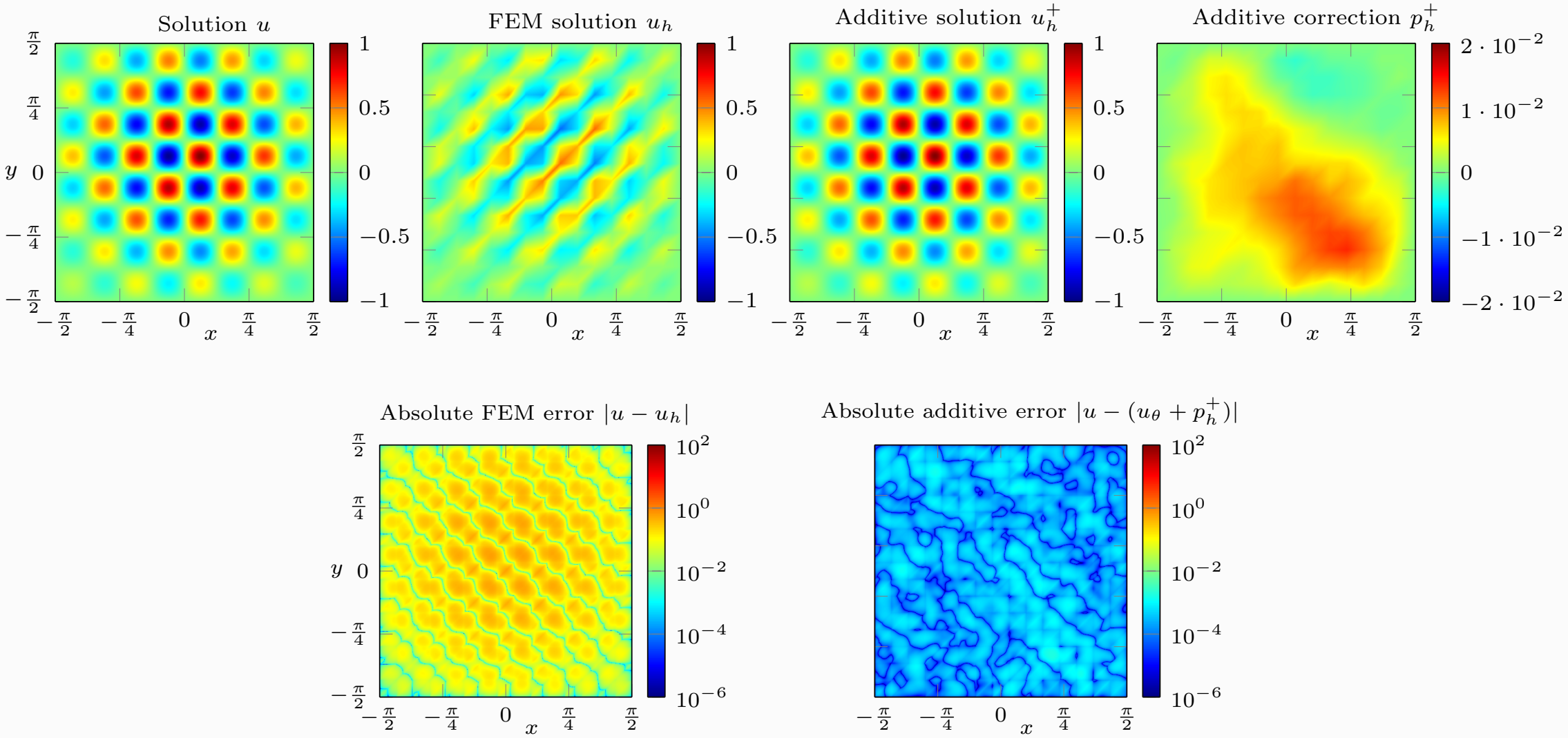
$$u(x, y, x_0, y_0) = \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2}\right) \sin(\kappa x) \sin(\kappa y).$$

The quantity κ controls the frequency of the solution, and so the difficulty of the approximation task.

Poisson problem in a square: low-frequency solution (κ=2)



Poisson problem in a square: high-frequency solution (κ=8)



Poisson problem in a square: convergence and errors for κ=8

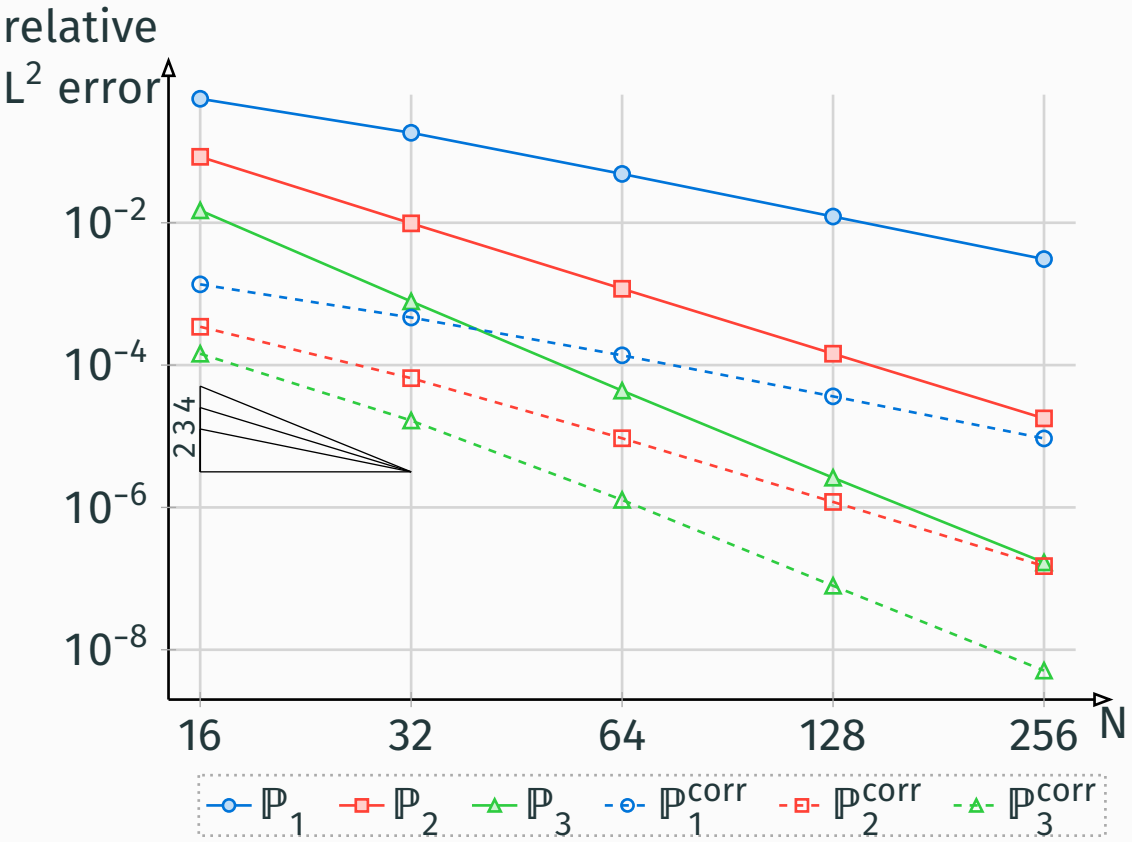


Figure: Relative L^2 error with respect to number of dofs per direction.

Poisson problem in a square: convergence and errors for κ=8

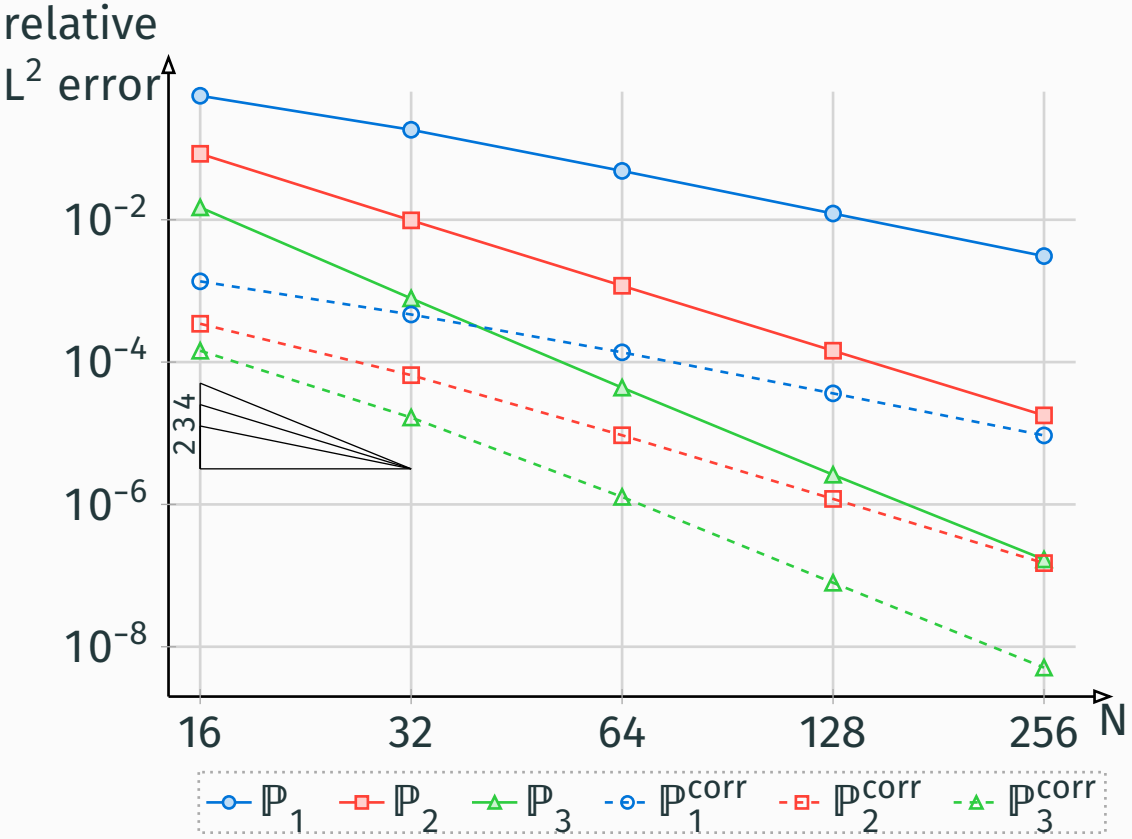


Figure: Relative L^2 error with respect to number of dofs per direction.

q	N	Mean gain wrt PINN	Mean gain wrt cG
1	20	1.98×10^1	3.50×10^2
	40	5.88×10^1	3.08×10^2
2	20	8.74×10^1	1.59×10^2
	40	5.17×10^2	1.10×10^2
3	20	2.53×10^2	6.30×10^1
	40	2.59×10^3	3.35×10^1

Table: Mean L^2 gains for 50 parameter instances, comparing enriched cG to PINNs and cG.

Poisson problem in a donut: setup

We consider a Poisson problem with mixed boundary conditions, in two dimensions and with one parameter:

$$\begin{cases} -\Delta u(x, y, \mu) = f(x, y, \mu), & (x, y) \in \Omega = \mathbb{D}_1 \setminus \mathbb{D}_{\frac{1}{4}}, \quad \mu \in \mathbb{M} = (2.4, 2.6), \\ u(x, y, \mu) = g_D(x, y, \mu), & (x, y) \in \partial\Omega, \quad \mu \in \mathbb{M} = (2.4, 2.6), \\ \dfrac{du}{dn}(x, y, \mu) + u(x, y, \mu) = g_R(x, y, \mu), & (x, y) \in \mathbb{S}_{\frac{1}{4}}, \quad \mu \in \mathbb{M} = (2.4, 2.6). \end{cases}$$

The right-hand side f is defined such that the solution is given by

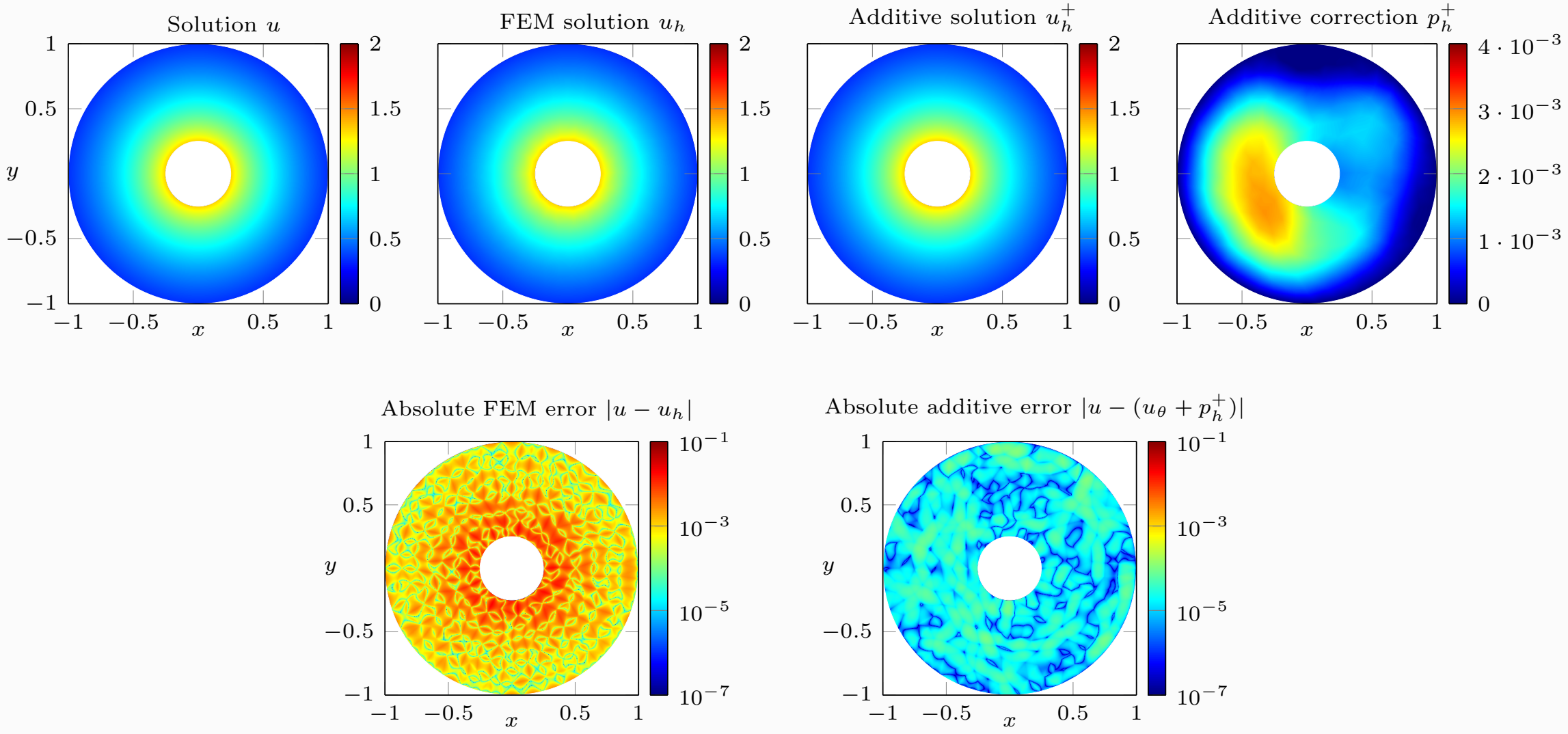
$$u(x, y, \mu) = 1 - \frac{\log\left(\mu\sqrt{x^2 + y^2}\right)}{\log 4},$$

and the boundary conditions are

$$g_D(x, y, \mu) = 1 - \frac{\log \mu}{\log 4} \quad \text{and} \quad g_R(x, y, \mu) = 2 + \frac{4 - \log \mu}{\log 4}.$$

We have **parameter-dependent mixed boundary conditions** on a more complex domain than a square.

Poisson problem in a donut: solution and approximation



Poisson problem in a donut: convergence and errors

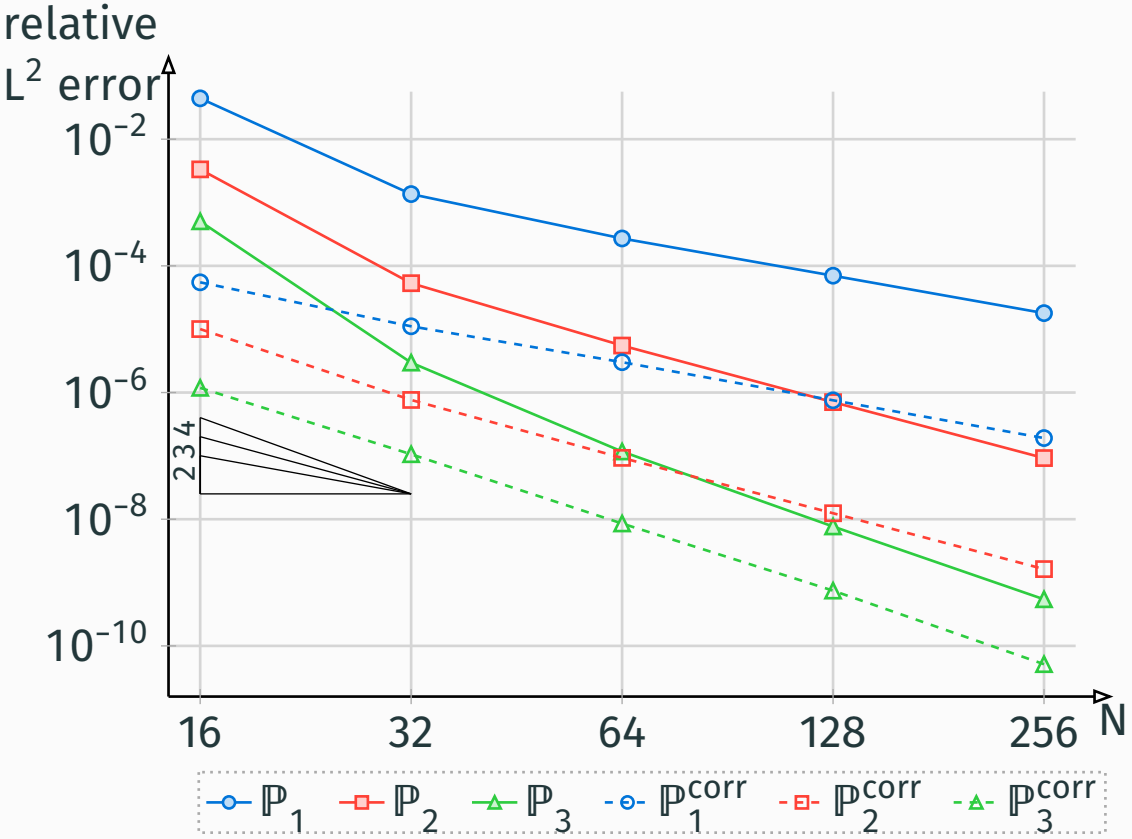


Figure: Relative L^2 error with respect to number of dofs per direction.

q	N	Mean gain wrt PINN	Mean gain wrt cG
1	1.33e-1	1.25×10^2	5.55×10^1
	6.90e-2	4.61×10^2	5.17×10^1
2	1.33e-1	3.40×10^3	5.84×10^1
	6.90e-2	2.16×10^4	4.72×10^1
3	1.33e-1	1.98×10^4	2.06×10^1
	6.90e-2	2.81×10^5	1.93×10^1

Table: Mean L^2 gains for 50 parameter instances, comparing enriched cG to PINNs and cG.

Poisson problem in a cube: setup

We consider a Poisson problem in three space dimensions and with three parameters:

$$\begin{cases} -\Delta u(x, y, z, x_0, y_0, z_0) = f(x, y, z, x_0, y_0, z_0), & (x, y, z) \in \Omega = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^3, \quad (x_0, y_0, z_0) \in \mathbb{M} = \left(-\frac{1}{2}, \frac{1}{2}\right)^3 \\ u(x, y, z, x_0, y_0, z_0) = 0, & (x, y, z) \in \partial\Omega, \quad (x_0, y_0, z_0) \in \mathbb{M} = \left(-\frac{1}{2}, \frac{1}{2}\right)^3. \end{cases}$$

The right-hand side f is defined such that the solution is given by

$$u(x, y, z, x_0, y_0, z_0) = \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}{2}\right) \sin(2x) \sin(2y) \sin(2z).$$

Poisson problem in a cube: convergence and errors

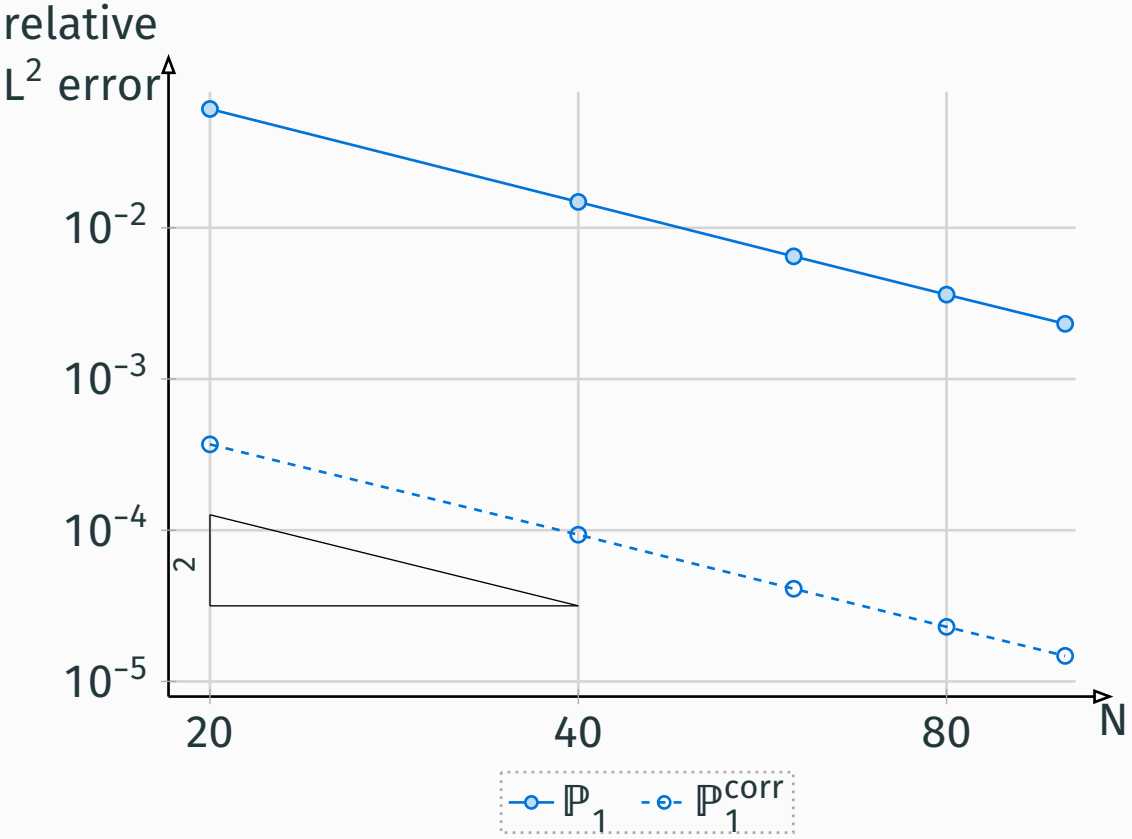


Figure: Relative L^2 error with respect to number of dofs per direction.

q	N	Mean gain wrt PINN	Mean gain wrt cG
1	20	1.41×10^1	1.37×10^2
	40	5.65×10^1	1.33×10^2

Table: Mean L^2 gains for 50 parameter instances, comparing enriched cG to PINNs and cG.

Poisson problem in a cube: how many simulations to recoup offline cost?

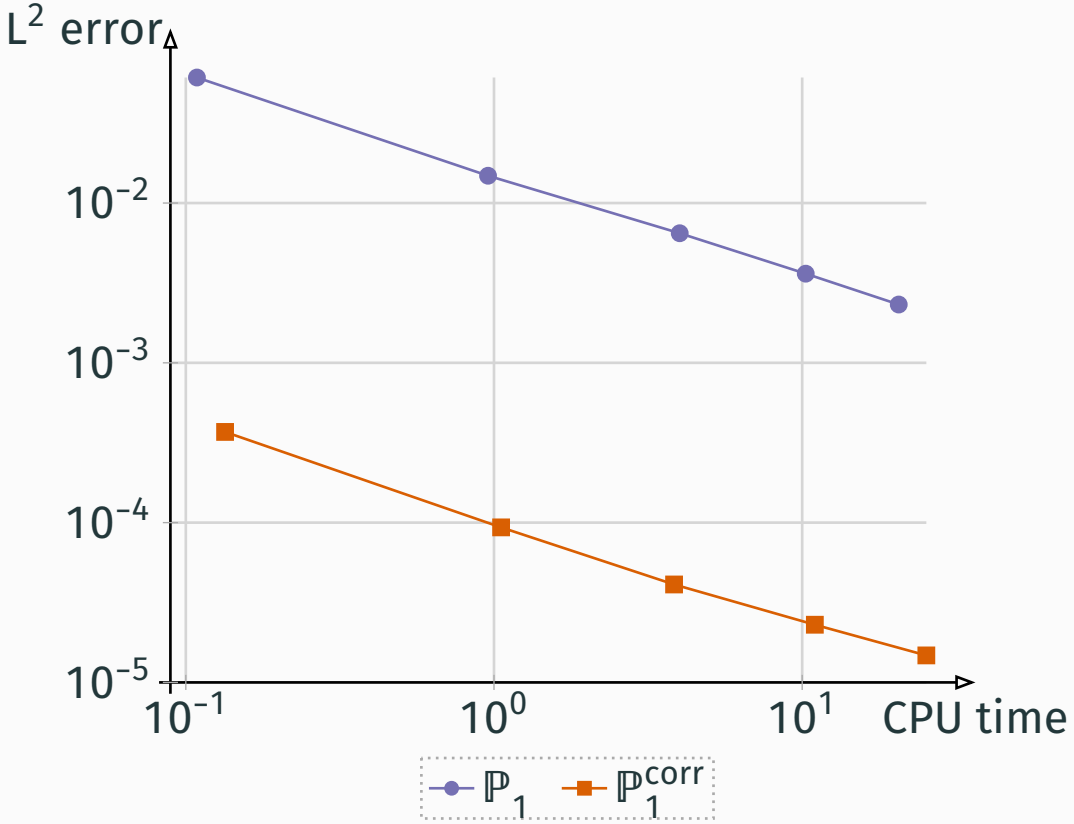


Figure: L^2 error with respect to computation time.

Poisson problem in a cube: how many simulations to recoup offline cost?

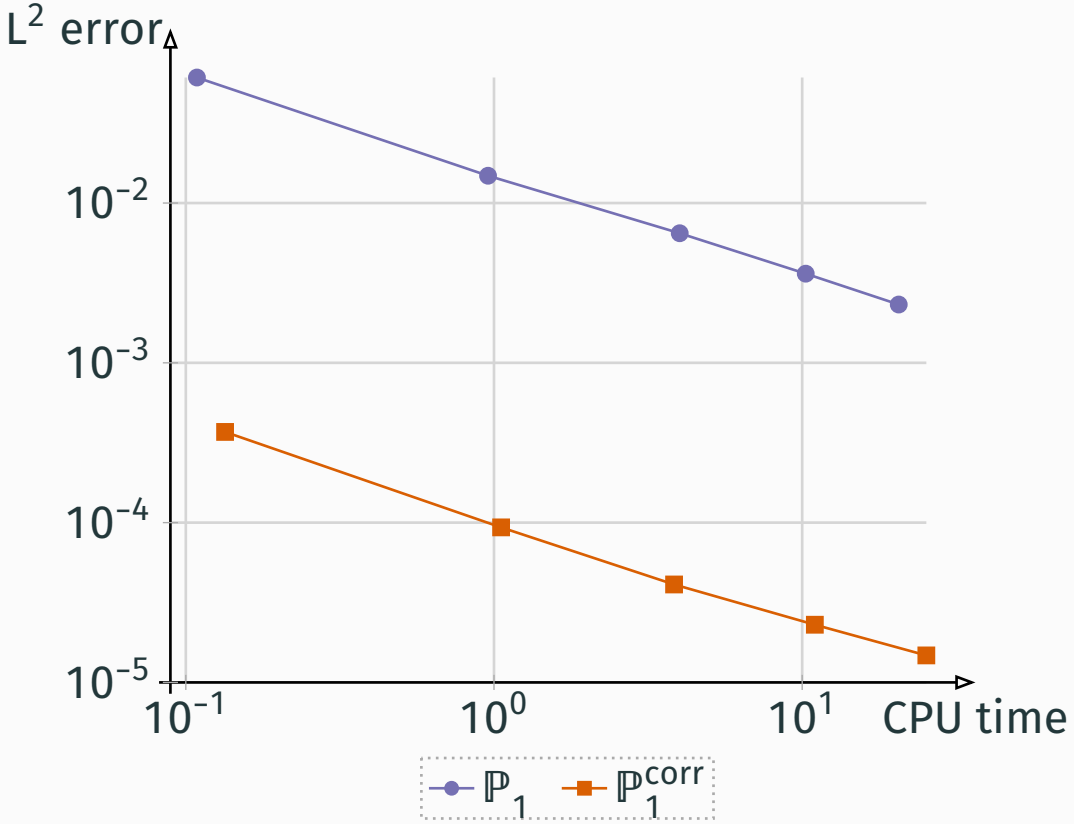


Figure: L^2 error with respect to computation time.

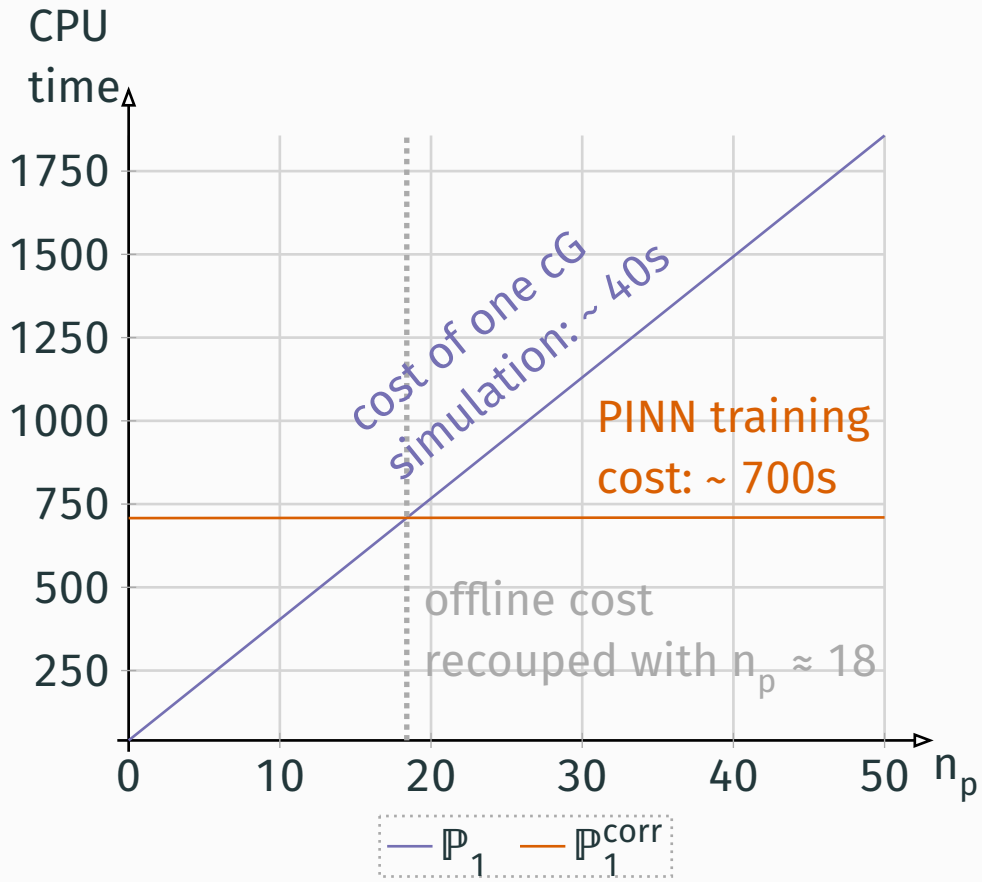


Figure: Computation time (in s) to reach an error of 10^{-3} for a given number of parameter instances.

4. Conclusion and perspectives

Summary

We introduced:

- a framework for **approximating functions** in linear or **nonlinear spaces**,
- an extension to **approximating solutions to PDEs**,
- and a **hybrid method** between cG and PINNs, applied to elliptic problems.

Perspectives include:

- learning bases in the **dG framework** and tackling **time-dependent** problems,
- applying the methodology to **nonlinear problems**, improving both initial guess and error constant.

Scimba is a `Python` + `pytorch` **library** with SciML methods for PDEs.

A `jax` version is also available; more features are coming soon!

More information is available at:

https://www.scimba.org

We are organizing a Workshop on **JAX for Scientific Computing** in **Strasbourg** in **June 2026**: https://majsc2026.pages.math.unistra.fr/

✨✨Thank you for your attention!✨✨