Hybrid methods: coupling numerical solvers with learned components

Scientific Machine Learning

Alena Kopaničáková

ENSEEIHT / Toulouse INP

Spring 2026

Outline

Why Hybrid?

Why Hybrid?

Classical numerical analysis

- Discretise: $A\,u = b$, with A encoding the PDE
- Solve: direct factorisation/iterative methods
- Guarantees: consistency, stability, convergence, a-priori rates
- Weakness: per-instance; ignores structure of a *family* of problems

Scientific machine learning

- Train a neural operator $\widehat{\mathcal{G}} \approx \mathcal{G}^\dagger$ on $\{(\mu_i, u_i)\}$
- Fast at inference, amortises across the parametric family
- Weakness: no a-priori accuracy guarantee

The hybrid premise

Each method's weakness is the other's strength $\implies$ *Combine* them so the classical machinery retains the guarantees and the learned component injects parametric prior knowledge

What does “hybrid” actually mean?

Hybrid algorithms differ in *where* the learned component is inserted into the classical numerical pipeline

Today’s example:

1. **Hybrid linear solvers (algorithm-in-the-loop)**

Linear Iterative Solvers: A Refresher

The linear system

Discretisation of a PDE yields

$$A\,u \;=\; b, \qquad A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^{n}$$

- Symmetric positive definite for elliptic problems (Poisson, Darcy)
- Sparsity inherited from the stencil
- Condition number $\kappa(A) \sim h^{-2}$ for 2nd-order elliptic

Two routes:

- *Direct*: factorise $A = LU$ - cost $\mathcal{O}(n^{\alpha})$, $\alpha \in [1.5, 3]$ - prohibitive in 3D
- *Iterative*: build a sequence of iterates $\{u^k\}$ converging to u^*

Splittings and stationary iterations

Pick a splitting $A = M - N$ with M *cheap to invert*
The associated iteration:

$$u^{k+1} \;=\; M^{-1}(Nu^k + b) \;=\; u^k + M^{-1}\underbrace{(b - Au^k)}_{r^k}$$

Apply a cheap approximate inverse to the current residual; add as correction

Canonical choices ($A = D - L - U$)

- **Richardson**: $M = \omega^{-1} I$
- **Jacobi**: $M = D$
- **Gauss–Seidel**: $M = D - L$

Richardson iteration with fixed step size

The simplest member of the splitting family: $M = \alpha^{-1} I$, i.e. apply the *identity* as preconditioner with a single relaxation parameter $\alpha > 0$

$$u^{k+1} \;=\; u^k \;+\; \alpha\, r^k, \qquad r^k \;=\; b - A u^k$$

Two viewpoints

- *Preconditioned residual:* cheapest possible $M^{-1} = \alpha I$ — no diagonal scaling, no factorisation
- *Gradient descent* on the quadratic energy $J(u) = \frac{1}{2} u^\top A u - b^\top u$ (A SPD) with step size α, since $\nabla J(u) = Au - b = -r$

Error propagation for Richardson

Let $e^k = u^k - u^\star$. Subtracting the fixed point $u^\star = u^\star + \alpha(b - Au^\star)$:

$$\boxed{e^{k+1} \;=\; G_\alpha\, e^k, \qquad G_\alpha \;=\; I - \alpha A}$$

The iteration is *linear in the error*, i.e., $e^k = G_\alpha^k\, e^0$

Spectrum inherited from A

G_α and A share eigenvectors; their eigenvalues are

$$\mu_i(G_\alpha) \;=\; 1 - \alpha\,\lambda_i(A),$$

where $\lambda_i(A)$ is eigenvalue of A. Convergence control: For which α does $\max_i |1 - \alpha\lambda_i| < 1$?

Convergence (general G)

For any splitting $A = M - N$ with $G = I - M^{-1}A$:

$$u^k \to u^\star \;\forall u^0 \iff \rho(G) < 1,$$

$$\left\|e^k\right\| \;\le\; \rho(G)^k \,\left\|e^0\right\|$$

$\rho(G_\alpha)$ **controls everything:** whether the method works, and how fast

Error propagation, component-wise

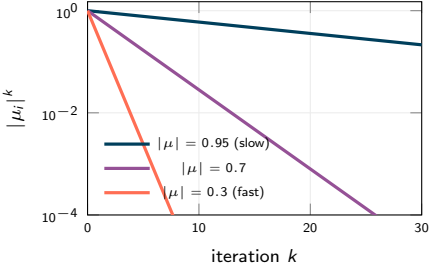
Let $\{(\lambda_i, v_i)\}_{i=1}^{n}$ be the eigenpairs of A (assume A diagonalisable). Expand the error in this basis:

$$e^0 = \sum_{i=1}^{n} c_i\, v_i, \qquad e^k = G_\alpha^k e^0 = \sum_{i=1}^{n} c_i\,(1-\alpha\lambda_i)^k\, v_i.$$

Each eigenmode of A evolves independently: Richardson is n decoupled geometric sequences with rates $\mu_i = 1 - \alpha\lambda_i$, i.e.,

- $|\mu_i| < 1 \Rightarrow$ mode v_i decays as $|\mu_i|^k$
- $|\mu_i| = 1 \Rightarrow$ mode persists
- $|\mu_i| > 1 \Rightarrow$ mode diverges

Worst mode $\rho(G_\alpha) = \max_i |\mu_i|$ dominates asymptotically



Mode-wise decay: the slowest mode sets asymptotic rate

Example (Saad, Ex. 4.1): Richardson iteration

Stationary Richardson: $M = \alpha^{-1} I$, $\alpha > 0$, so

$$u^{k+1} = u^k + \alpha(b - Au^k), \qquad G_\alpha = I - \alpha A, \qquad \mu_i(G_\alpha) = 1 - \alpha\lambda_i.$$

If $\lambda_{\min} \le \lambda_i \le \lambda_{\max}$: $\rho(G_\alpha) = \max\{|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\}$.

Convergence (SPD: $\lambda_{\min} > 0$)

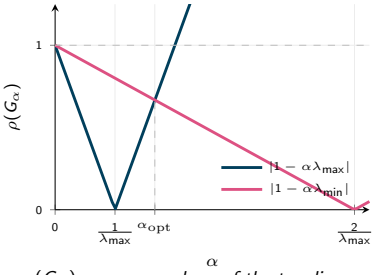
$$0 < \alpha < \frac{2}{\lambda_{\max}}.$$

Indefinite A ($\lambda_{\min} < 0 < \lambda_{\max}$): $\rho(G_\alpha) > 1$ for *every* α

Optimal α (curve crossing)

$$\alpha_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}}, \qquad \rho_{\text{opt}} = \frac{\kappa(A) - 1}{\kappa(A) + 1}$$

Ill-conditioned $\Rightarrow \rho_{\text{opt}} \to 1$



$\rho(G_\alpha)$: *upper envelope of the two lines.*

Jacobi iteration

Richardson treats all coordinates with the same step size α
Jacobi adapts to each row by preconditioning with the diagonal: $M = D = \mathrm{diag}(A)$

$$u^{k+1} \;=\; u^k \;+\; D^{-1}\, r^k \;=\; u^k \;+\; \omega\, D^{-1}(b - Au^k)$$

The damping $\omega \in (0, 1]$ is the same role as α in Richardson — a free relaxation parameter

Error propagation

$$G_\omega \;=\; I - \omega\, D^{-1}A,$$

eigenvalues $\mu_i = 1 - \omega\, \lambda_i(D^{-1}A)$.
Same one-dimensional spectral picture, on the spectrum of $D^{-1}A$ instead of A.

Convergence

- A strictly diagonally dominant $\Rightarrow \rho(G_1) < 1$ (undamped Jacobi converges)
- A SPD: damped Jacobi converges for $\omega \in (0,\, 2/\rho(D^{-1}A))$ (Ostrowski–Reich-type)
- Diagonal rescaling typically gives a much smaller $\kappa(D^{-1}A)$ than $\kappa(A)$

Jacobi = Richardson on the diagonally rescaled system $D^{-1}A\,u = D^{-1}b$

Damped Jacobi on the 1D Poisson model

A concrete spectrum lets us cash in the component-wise analysis
Take $A = h^{-2}\,\mathrm{tridiag}(-1, 2, -1) \in \mathbb{R}^{n\times n}$, $h = 1/(n+1)$

Eigenpairs of A (closed form)

$$\lambda_j(A) = \tfrac{4}{h^2}\sin^2\!\big(\tfrac{j\pi h}{2}\big), \quad \phi_j^{(i)} = \sqrt{2h}\,\sin(ij\pi h), \quad j = 1, \dots, n$$

$\lambda_1 \approx \pi^2$, $\lambda_n \approx 4h^{-2}$, $\kappa(A) \sim (2/\pi)^2 h^{-2}$ — *ill-conditioned as* $h \to 0$

Damped Jacobi: $D = (2/h^2)I$, so $D^{-1}A$ shares eigenvectors with A and has eigenvalues $\lambda_j(D^{-1}A) = 2\sin^2(j\pi h/2) \in (0, 2)$. By the previous slide, $G_\omega = I - \omega D^{-1}A$ acts diagonally:

$$\boxed{G_\omega\,\phi_j \;=\; \mu_j(\omega)\,\phi_j, \qquad \mu_j(\omega) \;=\; 1 - 2\omega\sin^2\!\big(\tfrac{j\pi h}{2}\big)}$$

The eigenmodes ϕ_j are discrete sine waves: j small = *smooth (low-frequency)*, j large = *oscillatory (high-frequency)*. Plugging into previous example with $\lambda_{\min,\max}(D^{-1}A)$:

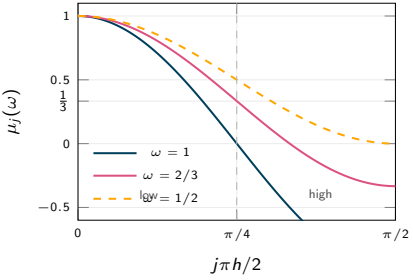
$$\omega_{\mathrm{opt}} = \tfrac{2}{\lambda_{\min}+\lambda_{\max}} \approx 1, \qquad \rho_{\mathrm{opt}}(G_\omega) = \tfrac{\kappa(D^{-1}A)-1}{\kappa(D^{-1}A)+1} \xrightarrow{h\to 0} 1$$

Smoothing factor vs. asymptotic factor

Split the spectrum at the Nyquist boundary $j = n/2$: $V_{\text{low}} = \text{span}\{\phi_1, \ldots, \phi_{n/2}\}$, $V_{\text{high}} = \text{span}\{\phi_{n/2+1}, \ldots, \phi_n\}$

Per-mode contraction $|\mu_j(\omega)|$

- *Low* j ($j\pi h \ll 1$): $\mu_j \approx 1 - \frac{\omega}{2}(j\pi h)^2 \to 1$ — smooth modes barely decay
- *High* j ($j\pi h/2 \to \pi/2$): $\mu_j \approx 1 - 2\omega$ — mesh-indep. decay



Two descriptors

$$\rho(G_\omega) = \max_{1 \le j \le n} |\mu_j| \quad \textit{(asymptotic)}$$

$$\mu_s(G_\omega) = \max_{n/2 \le j \le n} |\mu_j| \quad \textit{(smoothing)}$$

Optimal damping for the smoother

$\omega^\star = \frac{2}{3}$, $\mu_s(G_{\omega^\star}) = \frac{1}{3}$ *independent of* h

The classical scaling problem

Damped Jacobi at $\omega^\star = 2/3$ as a *stand-alone* solver: the asymptotic factor ρ is set by the *slowest* mode ϕ_1, not by the smoothing factor μ_s:

$$\rho(G_{\omega^\star}) \;=\; |\mu_1(\omega^\star)| \;=\; 1 - 2\omega^\star \sin^2\!\left(\tfrac{\pi h}{2}\right) \;=\; 1 - \tfrac{\omega^\star \pi^2 h^2}{2} + \mathcal{O}(h^4) \;\xrightarrow{h\to 0}\; 1$$

Plugging $\rho \to 1$ into the bound $\left\|e^k\right\| \le C\rho^k \left\|e^0\right\|$:

$$k\big(\|e^k\| \le \varepsilon \|e^0\|\big) \;\sim\; \frac{\log \varepsilon^{-1}}{1-\rho(G_{\omega^\star})} \;\sim\; \frac{\log \varepsilon^{-1}}{h^2} \;\sim\; \log \varepsilon^{-1} \cdot n$$

Iteration count grows like the mesh size — on a 3D grid with $n \sim h^{-3}$ unknowns, this is intractable

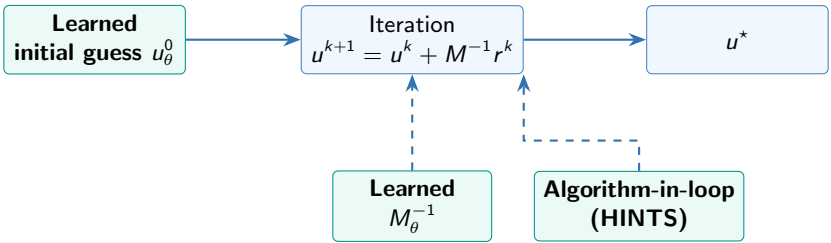
Smoothing $\neq$ solving

The two factors disagree: $\mu_s(G_{\omega^\star}) = 1/3$ (mesh-indep.), but $\rho(G_{\omega^\star}) \to 1$.
Damped Jacobi is a great *smoother* and a *useless solver:*

- Learned fix: **HINTS** — pair the smoother with a *neural operator* that does the same job on V_{low}

Where Can Machine Learning Help?

A map of ML inside an iterative solver



- Blue boxes are classical
- Green boxes are learned

1. Learned initial guess

- Replace $u^0 = 0$ by $u_\theta^0(\mu) \approx u^\star(\mu)$ from a surrogate (DeepONet, FNO, ...) trained on the parametric family

Effect on the error bound

$$\left\|e^k\right\| \;\leq\; C\,\rho^k\,\left\|e_\theta^0\right\|$$

The surrogate attacks the *multiplicative constant* $\left\|e^0\right\|$; the convergence rate ρ is unchanged

- Pays off when surrogate is much better than "zero start" but not accurate enough to skip the solver
- Complements every other hybrid approach, we will investigate below

2. Learned step size

Previous example gives the optimal Richardson step $\alpha_{\text{opt}}(A) = 2/(\lambda_{\min} + \lambda_{\max})$ have to be know. Moreover, they depend on the problem parameter μ, and computing them online is exactly the cost we wanted to avoid

Idea: Learn the step size from data:

$$\boxed{u^{k+1} \;=\; u^k \;+\; \alpha_\theta(\mu)\, r^k,}$$

with $\alpha_\theta : \mathcal{M} \to \mathbb{R}_{>0}$ a tiny network mapping problem parameters to a scalar

Three variants

- *Constant per problem:* $\alpha_\theta(\mu)$
- *Step-dependent schedule:* $\alpha_\theta^k(\mu)$ (T scalars, like a learned ω-sequence in SOR)
- *Residual-adaptive:* $\alpha_\theta(\mu, r^k)$ — closes the loop on the current error

Why it can beat α_{opt}

- α_{opt} minimises the *worst-case* spectral radius; learned α_θ minimises the *actual* T-step error on the parameter distribution

3. Algorithm-in-the-loop: Learned step

Replace the cheap inverse M^{-1} by a learned operator $\mathcal{N}_\theta : \mathbb{R}^n \to \mathbb{R}^n$ that approximates $A(\mu)^{-1}$ on the residual:

$$u^{k+1} \;=\; u^k \;+\; \mathcal{N}_\theta(\mu,\, r^k)$$

- Apply once per outer iteration
- Encodes parametric structure: $\mathcal{N}_\theta$ depends on μ
- Convergence depends on $\rho\big(I - \mathcal{N}_\theta A\big)$ — a quantity that can be monitored offline

Possible network architectures - operator learning is natural choice

DeepONet, FNO, ...

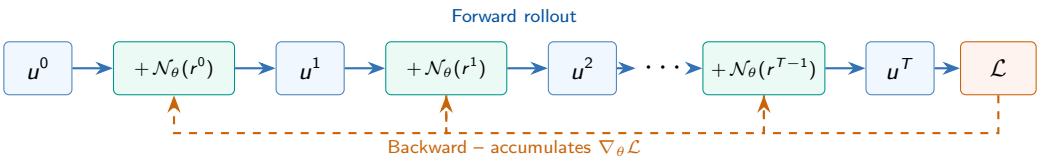
Training pipeline: Unrolling the hybrid solver

Treat the algorithm-in-the-loop iteration as a *differentiable program* of depth T:

$$u^{k+1} \;=\; u^{k} \;+\; \mathcal{N}_\theta(r^k), \qquad r^k = b - Au^k, \qquad k = 0, \dots, T-1$$

Train θ end-to-end against the exact solution $u^\star = A^{-1}b$:

$$\theta^\star \;=\; \arg\min_{\theta} \; \mathbb{E}_{(A,b)\sim\mu} \left\| \, u_T(\theta; A, b) \;-\; u^\star \, \right\|^2$$



- Forward: T residual evaluations $r^k = b - Au^k$ and T applications of the learned preconditioner $\mathcal{N}_\theta$
- Backward: reverse-mode AD

Diff.-physics example: Learned preconditioner for Poisson

$-u''(x) = f(x)$ on $(0,1)$, $u(0) = u(1) = 0$

Algorithm-in-the-loop:
$u^{k+1} = u^k + \mathcal{N}_\theta(r^k)$, $r^k = b - Au^k$

Train through T steps against $u^\star = A^{-1}b$

```
class HybridPoisson(nn.Module):
    def __init__(self, A):
        super().__init__()
        self.A = A
        n = A.shape[0]
        self.N_theta = nn.Sequential(
            nn.Linear(n,128), nn.GELU(),
            nn.Linear(128,n))
    def forward(self, u, b, T):
        for _ in range(T):
            r = b - self.A @ u
            u = u + self.N_theta(r)
        return u

loss = ((model(u0, b, T) - u_star)**2).mean()
loss.backward()
```

Reading the snippet

- `N_theta` *replaces* the cheap inverse M^{-1}; ideally approximates A^{-1} on the residual
- One residual evaluation $r = b - Au$ + one neural apply per step
- `forward(u, b, T)` unrolls T steps

4. Intertwining iterative methods with DNNs: HINTS (Zhang et al, '24)

Instead of replacing the smoother, *alternate* it with a neural correction:

$$\underbrace{\widetilde{u} \leftarrow u^k + \omega D^{-1}(b - Au^k) \ (n_s \text{ times})}_{\text{smoother kills high freq.}} \quad \rightarrow \quad \underbrace{u^{k+1} \leftarrow \widetilde{u} + \mathcal{N}_\theta(\mu, b - A\widetilde{u})}_{\text{network kills low freq.}}$$

Why this is the natural pairing

- Jacobi/SOR smooth high frequencies, stall on low ones
- Neural operators learn low frequencies first — the well-known *spectral bias*
- The two are spectrally complementary $\implies$ combine them and the whole spectrum is covered

The HINTS iteration

One outer HINTS sweep (relaxation budget n_s)

Smoothing: $\widetilde{u} \leftarrow u^k$; for $s = 1, \ldots, n_s$: $\widetilde{u} \leftarrow \widetilde{u} + \omega D^{-1}(b - A\widetilde{u})$

Neural correction: $u^{k+1} \leftarrow \widetilde{u} + \mathcal{N}_\theta(\mu,\ b - A\widetilde{u})$

Error propagation: Set $e^k = u^\star - u^k$ and assume $\mathcal{N}_\theta$ is linear in its argument
Smoothing gives $\widetilde{e} = S^{n_s} e^k$ with $S = I - \omega D^{-1} A$ (one damped-Jacobi step)
The neural correction adds $\mathcal{N}_\theta(b - A\widetilde{u}) = -\mathcal{N}_\theta A\,\widetilde{e}$:

$$e^{k+1} \;=\; T_{\text{HINTS}}\, e^k, \qquad T_{\text{HINTS}} \;=\; (I - \mathcal{N}_\theta A)\,(I - \omega D^{-1} A)^{n_s}$$

Convergence is controlled by $\rho(T_{\text{HINTS}})$ — the product of the smoother and network propagators

HINTS as an algorithm

Algorithm 1 HINTS

Input: A, b, $\mathcal{N}_\theta$, ω, n_s, tolerance ε, initial u^0
$k \leftarrow 0$; $r \leftarrow b - Au^0$
while $\|r\| > \varepsilon \, \|b\|$ **do**
    $\widetilde{u} \leftarrow u^k$
    for $s = 1, \ldots, n_s$ **do** ▷ Smoothing phase
        $\widetilde{u} \leftarrow \widetilde{u} + \omega \, D^{-1}(b - A\widetilde{u})$
    $r \leftarrow b - A\widetilde{u}$
    $u^{k+1} \leftarrow \widetilde{u} + \mathcal{N}_\theta(\mu, r)$ ▷ Neural correction phase
    $r \leftarrow b - Au^{k+1}$
    $k \leftarrow k + 1$
return u^k

- n_s is typically small in practice, i.e., $n_s \in \{1, 2, 3\}$
- $\mathcal{N}_\theta$ called only *once* per outer sweep

Dataset construction: the distribution-shift trap

Common pitfall

Neural operator surrogate $f \mapsto u$ (trained on *forcing functions*) performs well at iter 0 inside HINTS and degrades badly later

Three remedies

1. **Synthetic random residuals**: sample $u^{(i)} \sim \mathcal{D}_u$ broadband, set $r^{(i)} = Au^{(i)}$
2. **On-policy harvesting**: log residuals of the deployed HINTS run; retrain periodically
3. **Smoothed curriculum**: $r^{(i,j)} = G_\omega^j\, b^{(i)}$, $j = 0, \ldots, J$ — spectrum shifts low $\rightarrow$ high with j

Normalisation and loss

Residual magnitude drops orders of magnitude across iterations
Train and infer on unit-norm residuals:

$$\mathcal{N}_\theta(r) \;=\; \|r\|\,\widetilde{\mathcal{N}}_\theta\big(r/\,\|r\|\big)$$

Ensures positive homogeneity; removes scale variance

Loss possibilities (in increasing alignment with the deployed objective):

- *Direct supervision*: $\mathbb{E}\,\|\mathcal{N}_\theta r - A^{-1}r\|^2$ (needs reference solves)
- *Residual reduction*: $\mathbb{E}\,\|A\mathcal{N}_\theta r - r\|^2$ (no reference solve)
- *Subspace-targeted*: weight by $P_{V_{\text{low}}}$ — force accuracy where needed
- *Unrolled HINTS*: $\mathbb{E}\,\|u_K(\theta) - A^{-1}b\|^2$ — trains through the iteration via the differentiable solver; best alignment; checkpointing for memory

Choice of operator network architectures: examples

DeepONet

$$\mathcal{N}_\theta(\mu, r) = \sum_i b_i(\mu)\ \langle \xi_i,\ r \rangle\ \zeta_i$$

- Linear in r by construction
- Low-rank parametric preconditioner $M_\theta(\mu)$
- Safe inside CG (strict linearity)
- Mesh-fixed; tight per-call budget

FNO

$$\mathcal{N}_\theta = \mathcal{Q} \circ (\sigma \circ \mathcal{K}_\ell)_{\ell=1}^{L} \circ \mathcal{P}$$

- Mode truncation $\Rightarrow$ hard low-pass
- Discretisation invariant — transfers across meshes
- Not strictly linear in r
- Natural fit for the HINTS role

Spectral analysis: the idealised case

Recall the sine-mode basis $\{\phi_j\}$ of A and the split $V_{\text{low}} = \text{span}\{\phi_1, \ldots, \phi_{n/2}\}$, $V_{\text{high}} = \text{span}\{\phi_{n/2+1}, \ldots, \phi_n\}$. Let $P_{V_{\text{low}}}$, $P_{V_{\text{high}}}$ denote the orthogonal projectors onto these subspaces; note $P_{V_{\text{low}}} + P_{V_{\text{high}}} = I$

Idealised assumption

$\mathcal{N}_\theta$ inverts A exactly on V_{low} and zeroes the rest:

$$\mathcal{N}_\theta(\phi_j) = \begin{cases} \lambda_j^{-1}\phi_j, & j \le n/2 \\ 0, & j > n/2 \end{cases} \quad \Longleftrightarrow \quad \mathcal{N}_\theta A = P_{V_{\text{low}}}$$

Then $I - \mathcal{N}_\theta A = I - P_{V_{\text{low}}} = P_{V_{\text{high}}}$: the neural step *zeros out* V_{low} exactly and acts as the identity on V_{high}

Theorem (HINTS contraction, idealised)

With damped Jacobi at $\omega = 2/3$ as smoother: $\rho(T_{\text{HINTS}}) \le (1/3)^{n_s}$ — h-independent and exponential in the relaxation budget

Spectral analysis: realistic bound (setup)

The idealised theorem assumed $\mathcal{N}_\theta A = P_{V_{\text{low}}}$ exactly.
A trained network only approximates this projector — quantify the two error modes by:

$$\delta_{\text{low}} = \max_{v \in V_{\text{low}}} \frac{\|(I - \mathcal{N}_\theta A)\, v\|_2}{\|v\|_2} \qquad \textit{shortfall} \text{ on } V_{\text{low}} \text{ (should be 0)}$$

$$\delta_{\text{high}} = \max_{v \in V_{\text{high}}} \frac{\|(\mathcal{N}_\theta A)\, v\|_2}{\|v\|_2} \qquad \textit{leakage} \text{ onto } V_{\text{high}} \text{ (should be 0)}$$

Proposition (HINTS contraction, realistic case)

$$\|T_{\text{HINTS}}\|_2 \le \sqrt{2}\, \max\Big(\underbrace{\delta_{\text{low}}}_{\text{low-freq. residual}} , \underbrace{(1+\delta_{\text{high}})\, \mu_s^{n_s}}_{\text{high-freq. residual, inflated by leakage}} \Big)$$

Why each δ matters:
$\delta_{\text{low}} = 0$ means $\mathcal{N}_\theta$ *is* A^{-1} on the low subspace
$\delta_{\text{high}} = 0$ means it never moves a vector in the high subspace
The bound says HINTS converges if and only if both hold approximately

Spectral analysis: Where does the bound come from?

Decompose the error orthogonally: $e = e_L + e_H$ with $e_L \in V_{\text{low}}$, $e_H \in V_{\text{high}}$, so $\|e\|_2^2 = \|e_L\|_2^2 + \|e_H\|_2^2$

Step 1 – Smoothing: $V_{\text{low}}, V_{\text{high}}$ are S-invariant (they share eigenvectors with A):

$$\|S^{n_s} e_L\|_2 \le \|e_L\|_2\,, \qquad \|S^{n_s} e_H\|_2 \le \mu_s^{n_s}\ \|e_H\|_2$$

Step 2 – Neural correction: Apply $(I - \mathcal{N}_\theta A)$ to each piece, using the definitions of δ_{high} and δ_{low}:

$$\|(I - \mathcal{N}_\theta A)\, S^{n_s} e_L\|_2 \le \delta_{\text{low}}\ \|e_L\|_2\,, \quad \|(I - \mathcal{N}_\theta A)\, S^{n_s} e_H\|_2 \le (1 + \delta_{\text{high}})\, \mu_s^{n_s}\ \|e_H\|_2$$

(Triangle inequality on the high part: $\|v - \mathcal{N}_\theta A v\| \le (1 + \delta_{\text{high}})\, \|v\|$)

Combine: With $M = \max\big(\delta_{\text{low}}, (1 + \delta_{\text{high}})\mu_s^{n_s}\big)$ and Cauchy–Schwarz:
$\|T_{\text{HINTS}} e\|_2 \le M(\|e_L\| + \|e_H\|) \le \sqrt{2}\, M\, \|e\|_2$

Spectral analysis: the a priori certificate

Why we care: The bound predicts HINTS convergence from $(\delta_{\text{low}}, \delta_{\text{high}})$
Both are properties of $(\mathcal{N}_\theta, A)$ alone

How to measure each δ

δ_{low} is the operator norm of $(I - \mathcal{N}_\theta A)\big|_{V_{\text{low}}}$:

1. Pick a basis $\{\phi_1, \ldots, \phi_{n/2}\}$ of V_{low}
2. Build $M_L = [(I - \mathcal{N}_\theta A)\phi_j]_{j \le n/2}$ (one network call per column)
3. $\delta_{\text{low}} = \sigma_{\max}(M_L)$ via a few Lanczos / power iterations

Same recipe for δ_{high} with $\mathcal{N}_\theta A$ on $\{\phi_{n/2+1}, \ldots, \phi_n\}$

Deployment workflow

1. Train $\mathcal{N}_\theta$
2. Measure $\delta_{\text{low}}, \delta_{\text{high}}$ once
3. Check $\sqrt{2}\max(\delta_{\text{low}}, (1 + \delta_{\text{high}})\mu_s^{n_s}) < 1$
4. If yes: deploy HINTS with guarantees
5. If no: retrain with adjusted dataset, loss, stopping, ...