Neural Operators Meet Conjugate Gradients: The FCG-NO Method for Efficient PDE Solving

Alexander Rudikov[1,2], Vladimir Fanaskov[2], Ekaterina Muravleva[2,4], Yuri M. Laevsky[3], Ivan Oseledets[1,2]

[1] Artificial Intelligence Research Institute, [2] Skolkovo Institute of Science and Technology, [3] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, [4] Sberbank PJSC

Motivation

Main problems: Deep learning methods like U-Net and Neural Operators (NOs) typically achieve a relative L_2-norm of 0.1% to 1% solving partial differential equations (PDEs), while classical numerical methods can reach arbitrary accuracy but are less time efficient.

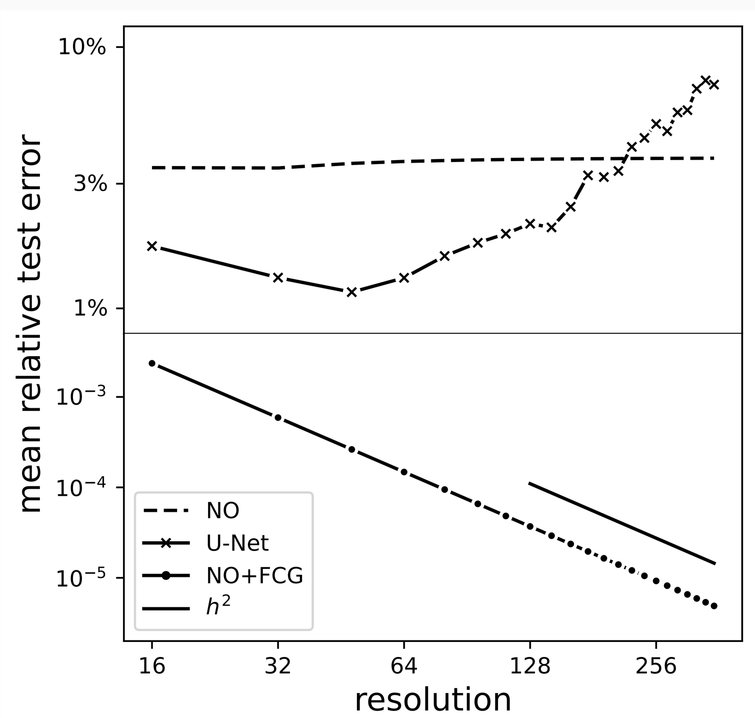


Fig. 1: Comparison of accuracy for three approaches: U-Net, NO, NO+FCG – hybrid approach advocated in the present article.

Contributions

- **Neural Operator Preconditioner**: Train a neural operator as a non-linear preconditioner for the flexible conjugate gradient (FCG) method, effective across resolutions.
- **Krylov Subspace Learning**: Use random vectors from the Krylov subspace $p_{\mathcal{K}_m}(r)$ for training; abandoning this subspace and using random right-hand sides $p_{\mathcal{K}_0}(r)$ impairs training.
- **Energy Norm Loss**: Introduce a novel loss function, which provides convergence guarantees and outperforms the traditional L_2 loss.

Flexible Conjugate Gradients

Algorithm 1

Input: $A,\ \mathcal{B},\ f,\ m_{\max} > 0$, iter
Ensure: $u_{\text{iter}}, r_{\text{iter}}$.
Initialize $u_0 \leftarrow \mathcal{N}(0,\ 1) \in \mathbb{R}^n$, $r_0 \leftarrow f - Au_0 \in \mathbb{R}^n$.
for $i = 0$ **to** iter $- 1$ **do**
  $w_i \leftarrow \mathcal{B}(r_i)$
  $m_i \leftarrow \min(i,\ \max(1,\ \bmod(i,\ m_{\max}+1)))$
  $p_i \leftarrow w_i - \sum_{k=i-m_i}^{i-1} \frac{(w_i, s_k)}{(p_k, s_k)}\, p_k$
  $s_i \leftarrow Ap_i$
  $u_{i+1} \leftarrow u_i + \frac{(p_i, r_i)}{(p_i, s_i)}\, p_i$
  $r_{i+1} \leftarrow r_i - \frac{(p_i, r_i)}{(p_i, s_i)}\, s_i$
end for

Notay's theorem

*Let $A,\ B \in \mathbb{R}^{n\times n}$ be symmetric positive definite matrices and $\mathcal{B}: \mathbb{R}^n \to \mathbb{R}^n$ be a nonlinear operator. Let $f,\ u_0$ be the vectors of $\mathbb{R}^n$, and let $\{r_i\}_{i=0,1,\ldots}$, $\{p_i\}_{i=0,1,\ldots}$, $\{u_i\}_{i=1,2,\ldots}$ be the sequences of vectors generated by applying **Algorithm 1** to $A,\ \mathcal{B},\ f$, and u_0 with some given sequences of non-negative integer parameters $\{m_i\}_{i=0,1,\ldots}$.*
If, for any i,

$$\frac{\left\|\mathcal{B}(r_i) - B^{-1}r_i\right\|_B}{\left\|B^{-1}r_i\right\|_B} \leqslant \varepsilon_i < 1,$$

then

$$\frac{\left\|u - u_{i+1}\right\|_A}{\left\|u - u_i\right\|_A} \leqslant \frac{\kappa(B^{-1}A)\cdot\gamma_i - 1}{\kappa(B^{-1}A)\cdot\gamma_i + 1},$$

where $\gamma_i = \frac{1+\varepsilon_i}{1-\varepsilon_i}\cdot\frac{(1+\varepsilon_i^2)^2}{(1-\varepsilon_i^2)}$, *and* $\|u\|_A = \sqrt{(u, Au)}$.

Notay loss

Consider boundary-value problem (BVP)

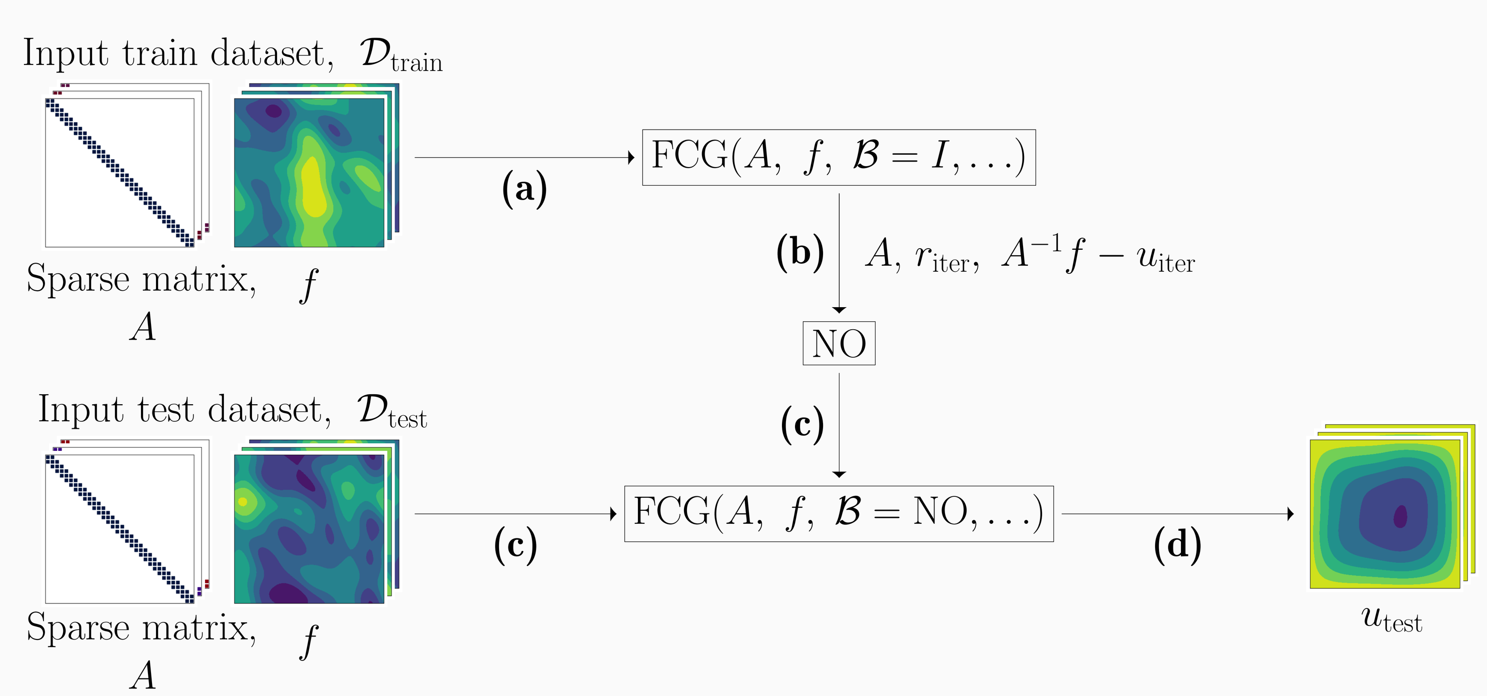
$$-\sum_{ij=1}^{2}\frac{\partial}{\partial x_i}\left(a(x)\frac{\partial u(x)}{\partial x_j}\right) = f(x)$$
$$x \in \Gamma \equiv (0,\ 1)^2,\ u(x)\big|_{x\in\partial\Gamma} = 0,$$

where $\partial\Gamma$ is a boundary of the unit hypercube Γ, and $a(x) \geq \epsilon > 0$.

$$L_{\text{Notay}}(\theta) = \mathbb{E}_{r,a,f}\frac{\left\|\mathcal{B}(r;\theta) - A^{-1}r\right\|_A}{\left\|A^{-1}r\right\|_A},$$

where A depends on a via discretization, r depends on f and the distribution of initial guess $u_0 \sim \mathcal{N}(0, I)$.

Proposed approach



The full scheme of the proposed approach: starts from the input train dataset, $\mathcal{D}_{\text{train}} = (A, f)$, where $f = Au_{\text{exact}}$. **(a)** Submit $\mathcal{D}_{\text{train}}$ to the CG (FCG with $\mathcal{B} = I$) with 100 iterations to generate residuals r_{iter} from the Krylov subspaces. **(b)** Train the NO on the FCG output r_{iter} with the use of A, u_{iter} for the calculation of L_{Notay}. **(c)** Apply the FCG with $\mathcal{B} = \text{NO}$ with the test dataset, $\mathcal{D}_{\text{test}}$. **(d)** Output u_{test}.

Experiments

FCG vs classical preconditioning techniques

The table contains the first iteration number i such that $\|r_i\|_2 / \|r_0\|_2 \leq 10^{-6}$ for different resolutions.

Dataset	grid	NO+FCG	CG	Jacobi(4)	GS(1)	GS(4)	ILU(1)	ILU(8)
Poisson	32	**9**	74	30	27	13	69	27
	64	**14**	130	58	47	23	110	77
	128	**20**	216	112	78	37	185	128
Diffusion	32	**9**	75	32	27	13	69	27
	64	**14**	132	61	46	22	110	74
	128	**19**	215	115	78	36	177	128

Time: FCG-NO vs CG

$(1 - t_{\text{FCG}}/t_{\text{CG}})\cdot 100\%$

		$\|r_i\|_2/\|r_0\|_2$		
Dataset	grid	10^{-3}	10^{-6}	10^{-12}
Poisson	32	43%	34%	9%
	64	58%	31%	14%
	128	74%	40%	33%
Diffusion	32	22%	21%	5%
	64	32%	32%	11%
	128	66%	44%	42%

Notay loss vs L_2-loss

		L_{Notay}			L_2		
		$\|r_i\|_2/\|r_0\|_2$			$\|r_i\|_2/\|r_0\|_2$		
Dataset	grid	10^{-3}	10^{-6}	10^{-12}	10^{-3}	10^{-6}	10^{-12}
Poisson	32	4	9	20	5	15	34
	64	5	14	31	7	22	53
	128	6	20	48	21	86	210
Diffusion	32	4	9	31	8	22	50
	64	5	14	36	11	36	80
	128	5	19	47	—	—	—

Differently obtained residuals

		$r \sim p_{\mathcal{K}_m}(r)$			$r \sim p_{\mathcal{K}_0}(r)$		
		$\|r_i\|_2/\|r_0\|_2$			$\|r_i\|_2/\|r_0\|_2$		
Dataset	grid	10^{-3}	10^{-6}	10^{-12}	10^{-3}	10^{-6}	10^{-12}
Poisson	32	4	9	20	4	10	21
	64	5	14	31	5	18	67
	128	6	20	48	7	49	153
Diffusion	32	4	9	31	5	23	56
	64	5	14	36	6	—	—
	128	5	19	47	10	—	—

Different grids

			$\|r_i\|_2/\|r_0\|_2$		
Dataset	Train grid	Test grid	10^{-3}	10^{-6}	10^{-12}
Poisson	32	64	9	18	39
	64	128	12	29	68
	32	128	24	61	141
	64	256	26	67	150
	128	256	16	45	103
Diffusion	32	64	8	18	42
	64	128	9	24	59
	32	128	17	57	148
	64	256	24	87	233
	128	256	16	46	112

Non-smooth diffusion coefficient $a(x)$

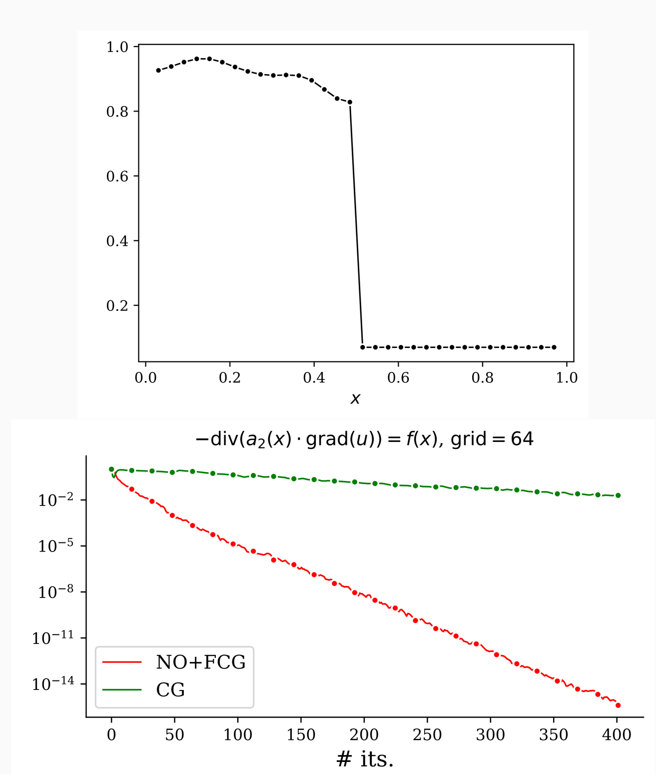


Fig. 2: Residuals' decline (bottom figure) for non-smooth diffusion coefficient (top figure).