Direct optimization of BPX preconditioners

Vladimir Fanaskov,* Ivan Oseledets†

January 11, 2024

Abstract

We consider an automatic construction of locally optimal preconditioners for positive definite linear systems. To achieve this goal, we introduce a differentiable loss function that does not explicitly include the estimation of minimal eigenvalue. Nevertheless, the resulting optimization problem is equivalent to a direct minimization of the condition number. To demonstrate our approach, we construct a parametric family of modified BPX preconditioners. Namely, we define a set of empirical basis functions for coarse finite element spaces and tune them to achieve better condition number. For considered model equations (that includes Poisson, Helmholtz, Convection-diffusion, Biharmonic, and others), we achieve from two to twenty times smaller condition numbers for symmetric positive definite linear systems.

1 Introduction

In the present contribution, we consider two optimization problems. The first one is the optimization of a parametric family of preconditioners for a modified Richardson method applied to the matrix $A : A^T + A > 0$, that is,

$$\omega_{\text{opt}}, \ \theta_{\text{opt}} = \arg\min_{\omega,\theta} \rho\left(I - \theta B(A,\omega)\right), \tag{1}$$

where $B(A,\omega) = B(\omega)A$ (or $B(A,\omega) = B(\omega)AB(\omega)$) is a family of linear systems preconditioned from the left (or in a symmetric fashion), ρ is a spectral radius, and ω is a set of real numbers. Problem (1) corresponds to a direct optimization of asymptotic convergence speed of an iterative linear solver [25, Section 2.2.5].

The second related problem is the optimization of the condition number

$$\omega_{\text{opt}} = \arg\min_{\omega} \lambda_{\max}\left(B(A,\omega)\right) \big/ \lambda_{\min}\left(B(A,\omega)\right), \tag{2}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the smallest and the largest eigenvalues, and A is symmetric positive definite.

In both problems we follow the approach adopted in [29] and further generalized in [21], [33]. That is, we introduce a stochastic loss function that approximates an objective function – spectral radius or a condition number – and perform a direct gradient-based optimization. The details can be found in Section 2 and Section 3.

For $B(\omega)$ we use a modified BPX [7] preconditioner. General multilevel preconditioner operates on a chain of linear spaces $V_1 \subset V_2 \subset \cdots \subset V_L$, where $V_l,\ 1 \le l \le L$ is formed as a linear combination of the set of functions $\phi_k^l(x),\ k = 1,\ldots,N_l$. In the context of a finite element method, $\phi_k^l(x)$ is a tent function located at vertex k of a grid with the diameter of a cell $\simeq \text{const } 2^{-l}$ (grid corresponding to V_{l+1} is constructed from l-th grid by, for example, subdivision of coarse triangulation, see i.e. [50, Section 2]). BPX preconditioners were developed for an elliptic problem

$$-\sum_{i,j=1}^{D} \frac{\partial}{\partial x_i} a_{ij}(x) \frac{\partial}{\partial x_j} u(x) = f(x), \tag{3}$$

---

*v.fanaskov@skoltech.ru
†i.oseledets@skoltech.ru

with homogeneous Dirichlet boundary conditions and uniformly symmetric positive definite $a_{ij}(x)$. For equation (3) and a nested set of finite element spaces span $\left\{\phi_k^l:\ k = 1, \ldots, N_l\right\}$, original BPX and preconditioner reads

$$B_{\mathrm{BPX}}(\omega)v = \sum_{l=1}^{L} \sum_{k=1}^{N_l} \left(v, \phi_k^l\right) \phi_k^l, \tag{4}$$

where $(\psi, \chi) = \int \psi(x)\chi(x)dx$ is a L_2 scalar product. To improve BPX preconditioner we replace tent function with empirical basis functions $\widetilde{\phi}_k^l$, $l = 1, \ldots, L-1$ and introduce scalars $\widetilde{\alpha}_l$, $l = 1, \ldots, L-1$ that weight contributions from individual spaces V_l, that is

$$B_{\mathrm{BPX}}(\omega)v = \sum_{l=1}^{L} \widetilde{\alpha}_l \left(\sum_{k} \left(v, \widetilde{\phi}_k^l\right) \widetilde{\phi}_k^l \right). \tag{5}$$

The details of the parametrisation and more convenient form of preconditioners (5) are given in Section 4.

Together $\widetilde{\phi}_k^l$ and $\widetilde{\alpha}_l$ form a set of parameters ω in problems (1), (2). The results of the optimization can be found in Section 5. In short, our framework allows for up to two times smaller spectral radius of modified Richardson scheme and up to twenty times smaller condition number for selected problems.

2 Direct optimization of the spectral radius

Problem (1) can be viewed in the context of a general search for better linear iterative methods. As explained in [25, Section 2.2.2], an arbitrary consistent iterative method can be written in a form

$$x^{n+1} = M(\omega, A)x^n + N(\omega, A)b,\ I - M(\omega, A) = N(\omega, A)A. \tag{6}$$

The efficiency of the method can be characterised by spectral radius $\rho\left(M(\omega, A)\right)$, because it quantifies an asymptotic convergence rate in a following sense. Let e^n be an error vector on step n, $\|\cdot\|$ is arbitrary norm and $\rho_{m+k,m} = \left(\|e^{m+k}\| / \|e^m\|\right)^{1/k}$ is a geometric mean of a one-step error reduction factor $\rho_{m+1,m}$. It is known that $\lim_{k\to\infty} \max_{x_0} \left\{\rho_{m+k,m}(x_0)\right\} = \rho(M(\omega, A))$ (see [25, Remark 2.22]). That is, $\rho\left(M(\omega, A)\right)$ characterises a geometric mean of an error reduction per iteration in the worst case. Because of that it is a custom to use $\rho\left(M(\omega, A)\right)$ as an objective function. For example, classical schemes like SOR and instationary Richardson iteration were optimized analytically [26], [25, chapters 4, 8] and numerically [35], [38], to achieve better $\rho\left(M(\omega, A)\right)$. More modern attempts include optimization of multigrid with local Fourier analysis [9] and directly [42], [33], [21], [29].

To apply gradient-based optimization to (1) we need a differentiable approximation to the spectral radius. We consider three options.

The first one is an approximation of $\rho(A)$ by Gelfand formula [31] $\rho(A) = \lim_{k\to\infty} \left\|A^k\right\|^{1/k}$ combined with a stochastic trace approximation [2]:

$$\rho(A) \simeq \rho_1(A, k, N_{\mathrm{batch}}) \equiv \left(\frac{1}{N_{\mathrm{batch}}} \sum_{j=1}^{N_{\mathrm{batch}}} \left\|A^k z_j\right\|_2^2 \right)^{1/2k}, \tag{7}$$

$$\forall j : \mathbb{P}\left((z_j)_i = \pm 1\right) = 1/2,\ \forall i, j:\ z_i, z_j \text{ are independent}.$$

More details about this approach can be found in [29].

The second option is based on $\rho(A) = \lim_{k\to\infty} \left(\|e^{m+k}\| / \|e^m\|\right)^{1/k}$, $e^{m+l} = A^l e^m$, see [25, Remark 2.22 (b)] for details. This gives us another approximation

$$\rho(A) \simeq \rho_2\left(A, k\right) \equiv \left(\left\|A^k z\right\|_2 / \|z\|_2\right)^{1/k},\ (z_i)_j \sim \mathcal{N}(0, 1). \tag{8}$$

Algorithm 1 Minimization of L_1 (10).

Input: matrix $A > 0$, parametric family of preconditioners $B(A, \omega) : B(A, \omega) > 0$, stochastic gradient-based optimizer $\omega \leftarrow O(\omega, \partial_\omega (\text{loss function}))$ (f.e., ADAM, [30]), batch size N_{batch}, number of matrix-vector products k, number of epochs N_{epochs}, number of iterations for inner loop N_{inner}, estimator of the spectral radius $m \in \{1, 2, 3\}$.

---

for $i = 1 : N_{\text{epochs}}$ **do**
    for $j = 1 : N_{\text{inner}}$ **do**
        $\rho_m, \partial_\theta \rho_m \leftarrow \text{AD } \rho_m (I - \theta B(A, \omega), k, N_{\text{batch}})$ // AD – automatic differentiation
        $\theta \leftarrow O(\theta, \partial_\theta \rho_m)$
    end for
    $L_1, \partial_\omega L_1 \leftarrow \text{AD } \rho_m (I - \theta B(A, \omega), k, N_{\text{batch}})$
    $\omega \leftarrow O(\omega, \partial_\omega L_1)$
end for

Approximation (8) does not contain averaging, but we can introduce N_{batch} the same way as in (7). That gives us the following the last approximation

$$\rho(A) \simeq \rho_3 (A, k, N_{\text{batch}}) \equiv \frac{1}{N_{\text{batch}}} \sum_{j=1}^{N_{\text{batch}}} \left(\left\| A^k z_j \right\|_2 \big/ \left\| z_j \right\|_2 \right)^{1/k}, \tag{9}$$

$$\forall j : (z_j)_i \sim \mathcal{N}(0, 1), \ \forall i, j : \ z_i, z_j \text{ are independent.}$$

The resulting loss will measure how well matrix A damps nonzero initial vectors on average. We observed that introduction of $N_{\text{batch}} > 1$ in (9) leads to better convergence.

With approximations $\rho_i (A, k, N_{\text{batch}})$, $i = 1, 2, 3$ we can use forward mode automatic differentiation [39] and standard optimizers [20, Section 8.3] to solve problem (1). The resulting algorithm coincides with Algorithm 1 with $N_{\text{inner}} = 1$.

3 Direct optimization of the condition number

Unlike problem (1) the optimization of the condition number is not straightforward. The main problem is the presence of $\lambda_{\min}$ which is not readily available. The standard way to resolve this issue is to substitute spectral radius with more amenable loss. For example, objective functions $\|R - A\|$ and $\|I - R^{-1}A\|$ (here R is an easy invertible approximation to A) were used to construct optimal circulant [10], [48], [44] and sparse approximate inverse [23], [13] preconditioners. It is known that for nonsymmetric matrices optimization of $\|I - R^{-1}A\|$ can fail to deliver good preconditioner [12]. The same is true for symmetric positive definite matrices as illustrated on Figure 1.

For symmetric positive definite matrices, one can construct a loss function that leads to a direct minimization of the spectral condition number. It is well known that for arbitrary positive definite matrix C, optimal spectral radius of $I - \theta C$ is $(\lambda_{\max}(C) - \lambda_{\min}(C)) / (\lambda_{\max}(C) + \lambda_{\min}(C))$. Using this fact, we can consider the following loss function

$$L_1(\omega) = \rho(I - \theta_{\text{opt}}(\omega)B(A, \omega)), \ \theta_{\text{opt}}(\omega) = \arg\min_\theta \rho(I - \theta B(A, \omega)). \tag{10}$$

Evidently, the minimization of (10) is equivalent to the minimization of $(\kappa(B(A, \omega)) - 1) / (\kappa(B(A, \omega)) + 1)$, where κ is the spectral condition number. That means we constructed an optimization problem equivalent to (2) but without $\lambda_{\min}$. A procedure for minimization of loss (10) is summarised in Algorithm 1. The inner loop finds θ_{opt} for each ω and the outer loop optimizes ω. If an inner loop is reduced to a single iteration as it is done in many other situations (for example, generalized policy iteration [45, Section 4.6], and full
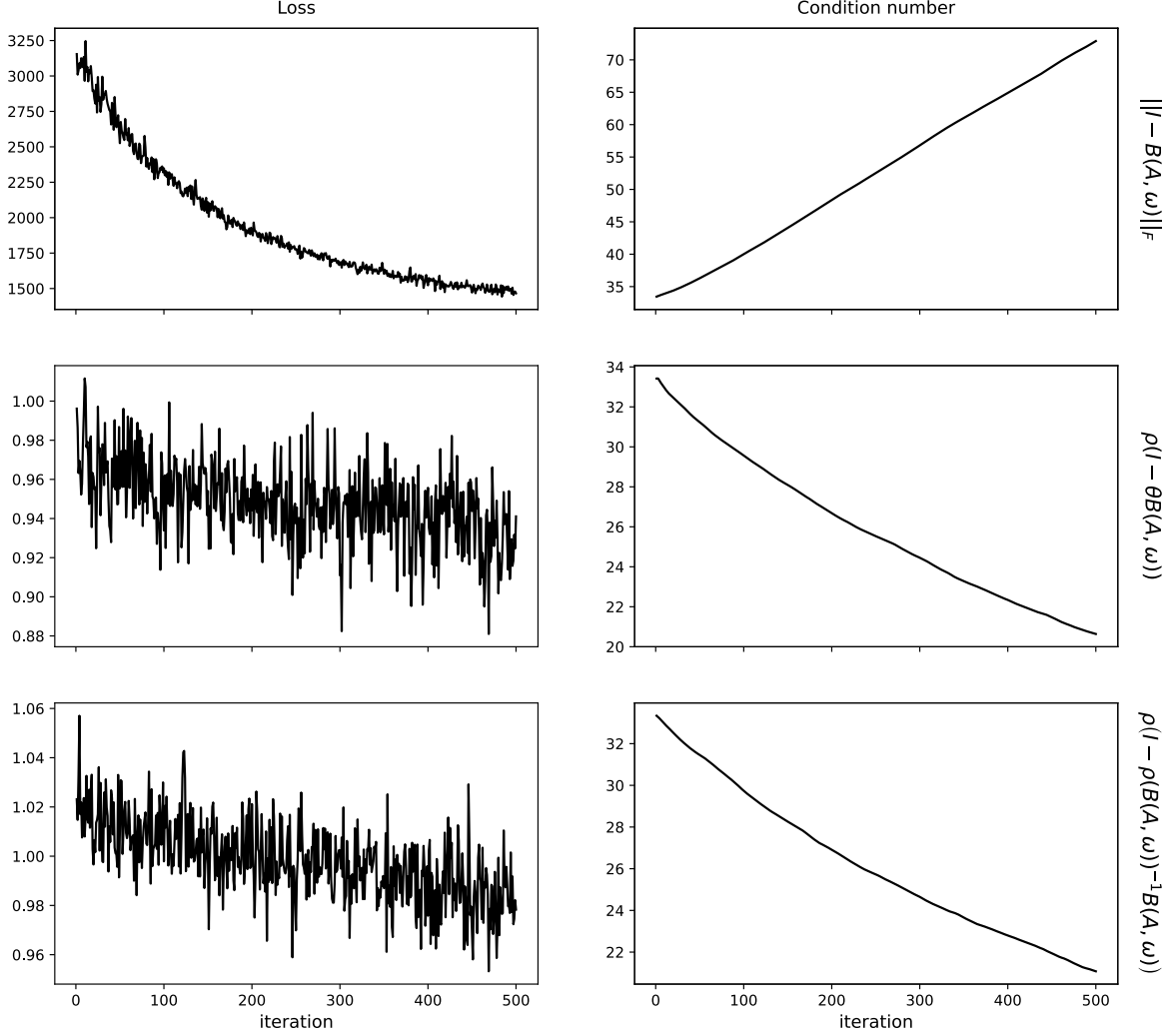
Figure 1: Comparison of three loss functions. The first column shows how the value of the loss function changes in the course of iterations, graphs in the second column demonstrate an evolution of condition number. The first row corresponds to the Frobenius norm $\|I - B(\omega)A\|$ used as a loss function, the second row shows minimization of L_1 by Algorithm 1 ($N_{\text{inner}} = 1$), the last row shows minimization of L_2 by Algorithm 2. For the last two cases, we used (7) to approximate spectral radius. It is clear that the decrease of both losses L_1 and L_2 lead to a smaller spectral condition number, whereas smaller Frobenius norm does not lead to a better spectral condition number. In all cases we use modified BPX preconditioner (15) as $B(\omega)$ and FEM discretization (see Section 4) of Poisson equation (16) in $D = 1$.

Algorithm 2 Minimization of L_2 (11).

Input: symmetric positive definite matrix $A > 0$, parametric family of preconditioners $B(A, \omega) : B(A, \omega) > 0$, stochastic gradient-based optimizer $\omega \leftarrow O(\omega, \partial_\omega (\text{loss function}))$ (f.e., ADAM, [30]), batch size N_{batch}, number of matrix-vector products k, number of epochs N_{epochs}, estimator of the spectral radius $m \in \{1, 2, 3\}$.

---

for $i = 1 : N_{\text{epochs}}$ **do**
  $\theta \leftarrow 1 / \rho_1 (B(A, \omega), k, N_{\text{batch}})$
  $\rho_m, \partial_\omega \rho_m \leftarrow \text{AD } \rho_m (B(A, \omega), k, N_{\text{batch}})$ // AD – automatic differentiation
  $L_2, \partial_\omega L_2, \partial_\theta L_2 \leftarrow \text{AD } \rho_m (I - \theta B(A, \omega), k, N_{\text{batch}})$
  $\omega \leftarrow O(\omega, \partial_\omega \rho_m - \theta^2 \partial_\theta L_2 \partial_\omega L_2)$
end for

---

approximation scheme [47, Section 5.3.1] follow the same pattern), we obtain an algorithm that minimizes spectral radius for modified Richardson scheme.

Another equivalent loss function is

$$L_2(\omega) = \rho \left(I - \frac{1}{\rho(B(A, \omega))} B(A, \omega) \right). \tag{11}$$

Indeed, $\rho(I - B(A, \omega)/\rho(B(A, \omega))) = 1 - \lambda_{\min}(B(A, \omega))/\lambda_{\max}(B(A, \omega))$, which means that a minimization of (11) is equivalent to minimization of $1 - 1/\kappa(B(A, \omega))$. Gradient-based optimization can be applied to (11) directly, but we can exploit a special structure of the problem to shorten the computation graph. Using a chain rule we get

$$
\begin{aligned}
\frac{\partial}{\partial \omega_i} \rho \left(I - \frac{1}{\rho(B(A, \omega))} B(A, \omega) \right) &= \left(\frac{\partial}{\partial \omega_i} \rho (I - \theta B(A, \omega)) \right) \Bigg|_{\theta = \rho(B(A, \omega))^{-1}} \\
&- \left(\theta^2 \frac{\partial}{\partial \theta} \rho (I - \theta B(A, \omega)) \right) \Bigg|_{\theta = \rho(B(A, \omega))^{-1}} \frac{\partial}{\partial \omega_i} \rho(B(A, \omega)).
\end{aligned} \tag{12}
$$

This leads to Algorithm 2. The performance of these two loss function is illustrated on Figure 1. In our experiments, we find little difference between Algorithm 1 and Algorithm 2. Because of that, we mainly use Algorithm 1, which requires a single computation of a gradient with respect to ω. However, unlike L_1 loss function L_2 is defined in terms of ρ in closed form, i.e., without an additional optimization problem, so it can be more advantageous in situations when a family of preconditioners is learned for a set of related linear equations, as it is done in [21] for the multigrid solver.

We summarize the results of this section in the following statement.

Proposition 1. *Let $A > 0$ and $B(\omega) > 0$ for all ω. For left $B(A, \omega) = B(\omega)A$, symmetric $B(A, \omega) = B(\omega)AB(\omega)$ and right $B(A, \omega) = AB(\omega)$ preconditioners the following three optimization problems are equivalent:*

- $\min_\omega \rho(I - \theta_{opt}(\omega)B(A, \omega))$, *where* $\theta_{opt}(\omega) = \arg\min_\theta \rho(I - \theta B(A, \omega))$ *– loss function* (10)

- $\min_\omega \rho(I - B(A, \omega)/\rho(B(A, \omega)))$ *– loss function* (11)

- $\min_\omega (\lambda_{\max}(B(A, \omega))/\lambda_{\min}(B(A, \omega)))$

4 Modified BPX preconditioner

We already specified algorithms that can be used to optimize condition number (optimization problem (2)). In this section, we describe a parametric family of positive definite preconditioners that we use in optimization.

(a) Bilinear FEM

	BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N
3	0.621	4.277	5	0.314	1.915	2
4	0.701	5.678	7	0.386	2.259	3
5	0.746	6.867	8	0.432	2.523	3
6	0.774	7.866	10	0.46	2.706	3

(b) Mehrstellen

	BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N
3	0.62	4.269	5	0.427	2.488	3
4	0.7	5.678	7	0.448	2.621	3
5	0.746	6.867	8	0.454	2.666	3
6	0.774	7.867	10	0.472	2.791	4

Figure 2: Results of optimization for 2D Poisson equation (16). Here $\rho = \lambda_{\max}\left(I - \theta_{\mathrm{opt}} BAB\right)$ – a spectral radius of optimal Richardson iteration for a given preconditioner, $\kappa = \lambda_{\max}(BAB)/\lambda_{\min}(BAB)$ – spectral condition number, and N – the number of iteration needed to drop an error by 0.1 in the arbitrary norm, i.e., $\left\|e^{n+N}\right\|/\left\|e^n\right\| \leq 0.1$.

To obtain a convenient form of BPX preconditioner, we introduce a hierarchy of meshes

$$M_l = \left\{x_j^l = j/2^l : j = 0, 1, \ldots, 2^l - 1, 2^l\right\}, \; l = 1, \ldots, L \tag{13}$$

such that each next mesh contains a previous one, that is, $M_l \subset M_{l+1}$. For each mesh, we define a set of basis functions $\phi_i^l(x) = \phi^l(x - x_i), i = 0, \ldots, 2^l$, which are rescaled and translated copies of a tent function $\phi^l(x) = \left(1 + x/2^l\right)\mathrm{Ind}\left[-1/2^l \leq x \leq 0\right] + \left(1 - x/2^l\right)\mathrm{Ind}\left[0 < x \leq 1/2^l\right]$, where $\mathrm{Ind}\left[x\right]$ is 1 if x holds and 0 otherwise. Basis functions $\left\{\phi_i^L(x) : i = 0, \ldots, 2^L\right\}$ are used to perform standard finite element discretization [14] of elliptic problem (3) for $x \in [0, 1]$. For higher dimensions, we use M_l and ϕ_i^l that are direct products of unidimensional meshes and basis functions.

In article [4], authors show that for equation (3) in $D = 1$ with uniform Dirichlet boundary condition at $x = 0$ and uniform Neumann boundary condition at $x = 1$ discretized as we just described, BPX preconditioner has the following form

$$\mathcal{B} = \sum_{k=1}^{L} \alpha_k B_k^L B_L^k, \; B_l^L = I_l \otimes \eta_{L-l} + S_l \otimes \left(\xi_{L-l} - \eta_{L-l}\right), B_L^l = \left(B_l^L\right)^T, \; \alpha_k = 1$$
$$(\eta_k)_i = i/2^k, (\xi_k)_i = 1, \; (S_l)_{ij} = \delta_{ij+1}, (I_l)_{ij} = \delta_{ij}, \; i, j = 1, \; \ldots, \; 2^l. \tag{14}$$

If $D = 2$ matrices B_L^k are replaced with $B_L^k \otimes B_L^k$ and α_k are with ratio of grid spacings h_L/h_k. The proof of the optimality of symmetric preconditioner (14) can be found in [4, Appendix A].

It is easy to see that components of η_{L-l} and $\xi_{L-l} - \eta_{L-l}$ contains scalar products $\left(\phi^L, \phi^l\right)$. Using this observation, one can extend (14) on other boundary conditions:

Proposition 2. *For equation* (3) *in* $D = 1$ *discretized with linear finite elements, symmetric BPX preconditioner has a form* $\mathcal{B} = \sum_{k=1}^{L} \alpha_k B_k^L B_L^k$, *where matrices* B_l^L *depend on boundary conditions as follows:*

- *Dirichlet-Neumann:* $B_l^L = I_l \otimes \eta_{L-l} + S_l \otimes \left(\xi_{L-l} - \eta_{L-l}\right)$;

- *Neumann-Dirichlet:* $B_l^L = I_l \otimes \eta_{L-l}^r + (S_l)^T \otimes \left(\xi_{L-l} - \eta_{L-l}^r\right)$, $(\eta_k^l)_i = (\eta_k)_{2^k - i + 1}$;

- *Neumann-Neumann:* $B_l^L = \begin{pmatrix} 1 & 0_{1 \times 2^l} \\ e_l \otimes \left(\xi_{L-l} - \eta_{L-l}\right) & I_l \otimes \eta_{L-l} + S_l \otimes \left(\xi_{L-l} - \eta_{L-l}\right) \end{pmatrix}$;

- *Dirichlet-Dirichlet:* $B_l^L = \left[I_l \otimes \eta_{L-l} + S_l \otimes \left(\xi_{L-l} - \eta_{L-l}\right)\right]_{\text{last row and column are removed}}$.

All boundary conditions are uniform and vectors ξ_{L-l}, η_{L-l} *are defined as in* (14).

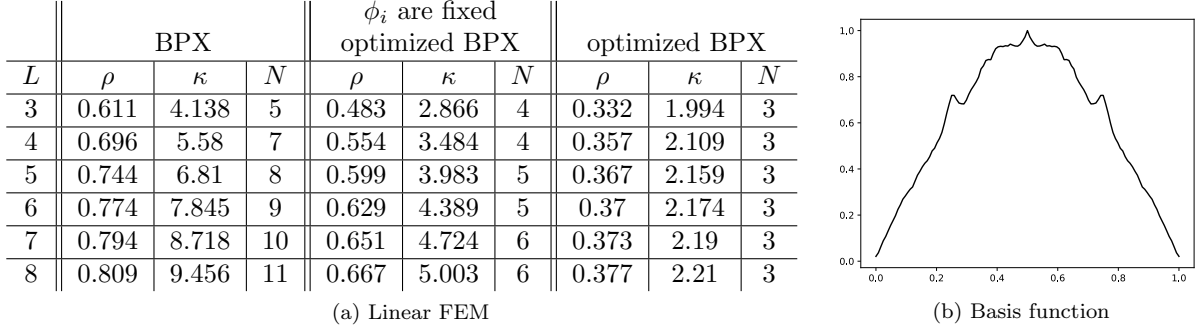| L	BPX			ϕ_i are fixed optimized BPX			optimized BPX		
	ρ	κ	N	ρ	κ	N	ρ	κ	N
3	0.611	4.138	5	0.483	2.866	4	0.332	1.994	3
4	0.696	5.58	7	0.554	3.484	4	0.357	2.109	3
5	0.744	6.81	8	0.599	3.983	5	0.367	2.159	3
6	0.774	7.845	9	0.629	4.389	5	0.37	2.174	3
7	0.794	8.718	10	0.651	4.724	6	0.373	2.19	3
8	0.809	9.456	11	0.667	5.003	6	0.377	2.21	3

(a) Linear FEM (b) Basis function

Figure 3: Results of optimization and basis function for 1D Poisson equation (16).

Based on (14) and Proposition 2, we put forward the following parametrization

$$\widetilde{\mathcal{B}} = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2 \, \widetilde{B}_k^L \widetilde{B}_L^k, \ \ \widetilde{B}_l^L = I_l \otimes \widetilde{\eta}_{L-l} + S_l \otimes \widetilde{\xi}_{L-l}, \left(\widetilde{\xi}_{L-l}\right)_{2^l} = 0, \ \widetilde{\eta}_0 = 1, \ \widetilde{\alpha}_L = 1, \tag{15}$$

where $\widetilde{\alpha}_k, \widetilde{\eta}_{L-k}$ and $\widetilde{\xi}_{L-k}$ are free parameters that correspond to ω in Algorithm 1 and Algorithm 2. Chosen parametrization differs from (14) in two respects. First, we use $\widetilde{\xi}_{L-k}$ in place of $\widetilde{\xi}_{L-k} - \widetilde{\eta}_{L-k}$. Since both $\widetilde{\eta}_{L-k}$ and $\widetilde{\xi}_{L-k}$ are free parameters, both options lead to the same family of preconditioners. Second, we use $(\widetilde{\alpha}_k)^2$ in place of $\widetilde{\alpha}_k$. This choice among with conditions $\widetilde{\eta}_0 = 1$ and $\widetilde{\alpha}_L = 1$ guarantee that $\widetilde{\mathcal{B}}$ is positive definite regardless of the choice of other parameters. Indeed, $\widetilde{\mathcal{B}}$ has a form $I + \sum_{k=1}^{L-1} (\widetilde{\alpha}_k)^2 \left(B_L^k\right)^T B_L^k$, that is, the sum of positive definite and positive semidefinite matrices. Because of that, conditions of Proposition 1 apply and we can use parametric family (15) to optimize condition number with Algorithm 1 and Algorithm 2. The last condition $\left(\widetilde{\xi}_{L-l}\right)_{2^l} = 0$ ensures that basis functions on level l have the same support as the ordinary tent functions.

5 Experiments

Here we present the results of the optimization for a set of test problems. First, we give an overview of model equations and the discretization used and then comment on the performance of optimized BPX preconditioners.

5.1 Model equations

5.1.1 Poisson equation

Poisson equation appears in a variety of contexts, from continuum mechanics [37, Sections 4.3, 5.1] to electrodynamics [28, Section 1.7]. It is also a standard test equation for multilevel solvers and preconditioners [47, Section 1.4]. The continuum boundary value problem reads

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x), \ x, y \in [0,1]^2, \ \ u(x,y)|_{\partial \Gamma} = 0, \tag{16}$$

here Γ represents a domain, and $\partial\Gamma$ is a boundary. We use standard bilinear finite element discretization in $D = 1$ and $D = 2$ (see Section 4), and also employ a high order compact scheme known as Mehrstellen [15, Table VI]. Mehrstellen discretization corresponds to the stencil

$$s = \begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}, \tag{17}$$

7

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.621</td><td>4.277</td><td>5</td><td>0.316</td><td>1.922</td><td>2</td></tr>
<tr><td>4</td><td>0.701</td><td>5.678</td><td>7</td><td>0.385</td><td>2.254</td><td>3</td></tr>
<tr><td>5</td><td>0.746</td><td>6.867</td><td>8</td><td>0.431</td><td>2.515</td><td>3</td></tr>
<tr><td>6</td><td>0.774</td><td>7.866</td><td>10</td><td>0.457</td><td>2.685</td><td>3</td></tr>
</tbody>
</table>

(a) $k^2h = 0.01$

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">$s = 1$
optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.919</td><td>23.719</td><td>28</td><td>0.592</td><td>3.9</td><td>5</td></tr>
<tr><td>4</td><td>0.956</td><td>44.127</td><td>51</td><td>0.625</td><td>4.339</td><td>5</td></tr>
<tr><td>5</td><td>0.967</td><td>60.262</td><td>70</td><td>0.653</td><td>4.766</td><td>6</td></tr>
<tr><td>6</td><td>0.973</td><td>72.413</td><td>84</td><td>0.68</td><td>5.25</td><td>6</td></tr>
</tbody>
</table>

(b) $\epsilon = 10$

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.621</td><td>4.277</td><td>5</td><td>0.316</td><td>1.922</td><td>2</td></tr>
<tr><td>4</td><td>0.701</td><td>5.678</td><td>7</td><td>0.385</td><td>2.254</td><td>3</td></tr>
<tr><td>5</td><td>0.746</td><td>6.867</td><td>8</td><td>0.431</td><td>2.515</td><td>3</td></tr>
<tr><td>6</td><td>0.774</td><td>7.866</td><td>10</td><td>0.457</td><td>2.685</td><td>3</td></tr>
</tbody>
</table>

(c) $k^2h = 0.1$

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">$s = 2$
optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.974</td><td>75.467</td><td>87</td><td>0.679</td><td>5.235</td><td>6</td></tr>
<tr><td>4</td><td>0.991</td><td>216.104</td><td>249</td><td>0.704</td><td>5.763</td><td>7</td></tr>
<tr><td>5</td><td>0.996</td><td>468.362</td><td>540</td><td>0.71</td><td>5.9</td><td>7</td></tr>
<tr><td>6</td><td>0.997</td><td>753.064</td><td>867</td><td>0.754</td><td>7.145</td><td>9</td></tr>
</tbody>
</table>

(d) $\epsilon = 100$

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.616</td><td>4.213</td><td>5</td><td>0.317</td><td>1.928</td><td>3</td></tr>
<tr><td>4</td><td>0.698</td><td>5.612</td><td>7</td><td>0.387</td><td>2.264</td><td>3</td></tr>
<tr><td>5</td><td>0.744</td><td>6.808</td><td>8</td><td>0.432</td><td>2.519</td><td>3</td></tr>
<tr><td>6</td><td>0.773</td><td>7.817</td><td>9</td><td>0.457</td><td>2.683</td><td>3</td></tr>
</tbody>
</table>

(e) $k^2h = 1$

<table>
<thead>
<tr><th></th><th colspan="3">BPX</th><th colspan="3">$s = 2$
optimized BPX</th></tr>
<tr><th>L</th><th>ρ</th><th>κ</th><th>N</th><th>ρ</th><th>κ</th><th>N</th></tr>
</thead>
<tbody>
<tr><td>3</td><td>0.983</td><td>118.948</td><td>137</td><td>0.694</td><td>5.531</td><td>7</td></tr>
<tr><td>4</td><td>0.997</td><td>578.133</td><td>666</td><td>0.735</td><td>6.554</td><td>8</td></tr>
<tr><td>5</td><td>0.999</td><td>1713.449</td><td>1973</td><td>0.763</td><td>7.454</td><td>9</td></tr>
<tr><td>6</td><td>1.0</td><td>4032.087</td><td>4643</td><td>0.807</td><td>9.348</td><td>11</td></tr>
</tbody>
</table>

(f) $\epsilon = 1000$

Figure 4: First column (a, c, e) results for Helmholtz equation (18), second column (b, d, f) results for anisotropic Poisson equation (19); s refers to semicoarsening (29).

which can be used to construct a fourth and sixth-order accurate approximation to the Poisson equation if boundary conditions and right-hand side are sufficiently smooth [41].

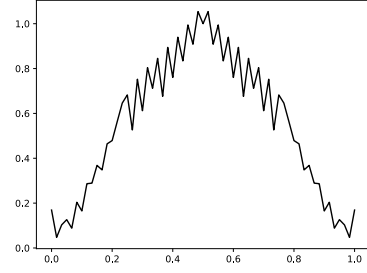### 5.1.2 Helmholtz equation

Helmholtz equation

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} - k^2 u(x,y) = f(x),\ x,y \in [0,1]^2,\ \ u(x,y)|_{\partial\Gamma} = 0, \tag{18}$$

appears in the context of wave propagation problems [17, Section 2.1]. For example, the Helmholtz equation needs to be solved at each time step in the semi-implicit discretization of governing equation of non-hydrostatic weather prediction models [43, Section 4.1].

Because of the term $-k^2 u(x,y)$, bilinear finite element discretization can result in an indefinite matrix, especially for large k, which renders our method inapplicable. However, the value of k can not be arbitrary on a given grid because of the pollution problem [3]. More precisely, unless k^2h is sufficiently small, the solution to a discrete problem is of no use because it does not approximate an exact solution. Having this condition in mind, we choose k small enough to have a positive definite problem.

	BPX			optimized BPX		
L	ρ	κ	N	ρ	κ	N
3	0.878	15.367	18	0.846	11.984	14
4	0.96	48.717	57	0.878	15.33	18
5	0.988	167.576	193	0.899	18.9	22
6	0.997	617.095	711	0.945	35.073	41

(a) 13-point stencil

(b) Basis function

Figure 5: Results of optimization and basis function for the Biharmonic equation (20).

5.1.3 Anisotropic Poisson equation

Anisotropic version of Poisson equation

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \epsilon \frac{\partial^2 u(x,y)}{\partial y^2} = f(x), \ x,y \in [0,1]^2, \ \ u(x,y)|_{\partial\Gamma} = 0, \tag{19}$$

arises naturally in computational fluid dynamics when a refined or stretched grid is used to resolve a boundary layer, shock, or some other singularity [32, Chapter 4], [47, Section 5.1.2]. Parameter ϵ can also be related to the anisotropy of the physical system. For example, a crystal's permittivity can depend on the direction [36, Chapter 9], so electrostatic boundary-value problems lead to an anisotropic Poisson equation.

5.1.4 Biharmonic equation

The only fourth-order equation we consider is biharmonic:

$$\frac{\partial^4}{\partial x^4} u(x,y) + 2\frac{\partial^2}{\partial x \partial y} u(x,y) + \frac{\partial^4}{\partial y^4} u(x,y) = f(x,y), \ x,y \in [0,1]^2, \ \ u(x,y)|_{\partial\Gamma} = 0, \ \ \partial_n u(x,y)|_{\partial\Gamma} = 0, \tag{20}$$

here ∂_n is a derivative along the normal direction to the boundary $\partial\Gamma$. Applications of the Biharmonic equation include a description of fluid flows [11], vibrating plates, Chladni figures [19], gravitation theory, and quantum mechanics [34, Introduction]. To discretize this equation, we use centered second-order finite difference approximation given by a 13 point stencil

$$s = \begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix}, \tag{21}$$

which should be modified appropriately near the boundaries [46, Section 4] (see also [24] and [6]).

5.1.5 Convection-diffusion equation

When convective transport is present, the original diffusion equation needs to be modified as follows

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} + v_x u(x,y) + v_y u(x,y) = f(x,y), \ x,y \in [0,1]^2, \ \ u(x,y)|_{\partial\Gamma} = 0. \tag{22}$$

The presence of v_x and v_y results in nonsymmetric matrix. This means Proposition 1 does not hold, but Algorithm 1 can be applied to optimize modified Richardson iteration. Since we employ bilinear finite element discretization (centered difference approximation), the stability restriction is given by Peclet condition $\max(|v_x|, |v_y|) \le 2/h$.

	BPX		optimized BPX			BPX		optimized BPX			BPX		optimized BPX	
L	ρ_3	N	ρ_3	N	L	ρ_3	N	ρ_3	N	L	ρ_3	N	ρ_3	N
3	0.629	5	0.398	3	3	0.787	10	0.574	5	3	0.855	15	0.693	7
4	0.741	8	0.554	4	4	0.830	13	0.711	7	4	0.869	17	0.743	8
5	0.797	11	0.649	6	5	0.864	16	0.737	8	5	0.872	17	0.785	10
6	0.829	13	0.690	7	6	0.874	18	0.743	8	6	0.874	18	0.792	10

(a) $v_x = -v_y = 1/h$ (b) $v_x = -v_y = 2/h$ (c) $v_x = -v_y = 3/h$

Figure 6: Results of optimization for convection-diffusion equation (22), h is a distance between grid points on the finest grid. Note, that the value of a loss function (9) is listed, not an "exact" spectral radius.

5.1.6 Diffusion with discontinuous coefficients

In some situations, diffusion coefficient $a(x, y)$ in equation

$$-\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u(x,y)}{\partial x}\right) - \frac{\partial}{\partial y}\left(a(x,y)\frac{\partial u(x,y)}{\partial y}\right) = f(x,y), \ x,y \in [0,1]^2, \ u(x,y)|_{\partial\Gamma} = 0, \qquad (23)$$

is discontinuous along some curve or surface inside the computational domain. For example, this is the case in reservoir simulation [47, Section 7.7.1], and the description of the neutron diffusion [1]. For our experiments, we take

$$a(x,y) = g(x) + g(y), \ g(x) = \sigma^{-1}\mathsf{Ind}\left[x < 1/2\right] + \sigma\mathsf{Ind}\left[x \geq 1/2\right], \qquad (24)$$

where σ is a parameter that controls the magnitude of the jump. The discretization we used is, again, FEM.

5.1.7 Mixed derivative

Another problem of interest is a Poisson equation with mixed derivative

$$-\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} - 2\tau\frac{\partial^2 u(x,y)}{\partial x \partial y} = f(x), \ x,y \in [0,1]^2, \ u(x,y)|_{\partial\Gamma} = 0. \qquad (25)$$

For $|\tau| > 1$ the equation becomes hyperbolic, so it is interesting to look how optimization works for $\tau \simeq 1$.

5.1.8 Implicit scheme for the heat equation

The last equation that we consider comes from the trapezoidal discretization (in time) of the heat equation

$$\frac{\partial u(x,y,t)}{\partial t} = \frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2}, \ x,y \in [0,1]^2, \ t \in [0,+\infty),$$

$$u(x,y,t)|_{t=0} = \phi(x,y), \ u(x,y)|_{\partial\Gamma} = 0. \qquad (26)$$

Let A be a matrix that corresponds to a spatial FEM discretization of the right-hand side operator. It results in a system of ordinary differential equations

$$\frac{du_i(t)}{dt} = \sum_j A_{ij}u_j(t), \ u_i(0) = \phi_i. \qquad (27)$$

Application of the trapezoidal rule leads to an unconditionally stable iteration

$$\sum_j \left(I - \frac{\widetilde{\mu}}{2}A\right)_{ij} u_j^{n+1} = \sum_j \left(I + \frac{\widetilde{\mu}}{2}A\right)_{ij} u_j^n \qquad (28)$$

10

known as Crank-Nicolson scheme [27, Section 16.4]. Here $\widetilde{\mu} = \Delta t$ is related to the Courant number $\mu = \Delta t / \Delta x^2$. Since matrix $\left(I - \frac{\widetilde{\mu}}{2} A \right)$ is symmetric positive definite for $\widetilde{\mu} \geq 0$ that needs to be inverted during each iteration, we test our preconditioner on this problem.

5.2 Optimization results

For all equations we use a symmetric form of both BPX (14) and modified BPX (15) preconditioners. To access the results of optimization we list three related numbers: $\rho = \lambda_{\max} \left(I - \theta_{\mathrm{opt}} BAB \right)$ – a spectral radius of the optimal Richardson iteration for a given preconditioner, $\kappa = \lambda_{\max}(BAB) / \lambda_{\min}(BAB)$ – spectral condition number, and N – the number of iteration needed to drop an error by 0.1 with the optimal modified Richardson iteration in an arbitrary chosen norm, i.e., $\left\| e^{n+N} \right\| / \left\| e^n \right\| \leq 0.1$. The number of iterations N is computed as $\left\lceil -1 / \log_{10} \rho \right\rceil$, where $\lceil \cdot \rceil$ is the ceiling function. [1]

In all cases, we use Dirichlet boundary conditions. Value of L fixes the number of points along each direction to be $2^L - 1$.

For all examples we employed Algorithm 1 with the loss function (9) ($N_{\mathrm{batch}} = 10$, $k = 10$), ADAM optimizer [30], $N_{\mathrm{epoch}} = 500$, $N_{\mathrm{inner}} = 1$. Initial parameters $\widetilde{\alpha}, \widetilde{\eta}, \widetilde{\xi}$ of the modified BPX preconditioner (15) were chosen such that the resulting matrix $\widetilde{\mathcal{B}}$ coincides with the BPX preconditioner (14).

All algorithms were implemented in Julia [5] and available in a public repository `https://github.com/VLSF/neuralBPX`.

5.2.1 Poisson equation

We can see on Figure 2 that for the 2D Poisson equation optimization successfully decreases the condition number. Moreover, it seems to grow slower compared to the original BPX preconditioner as the number of points increases [2]. To assess the contribution of the optimized basis functions, we perform additional optimization in $D = 1$ with fixed basis functions. Results, given in Figure 3, indicate that optimization of the basis function leads to twice as small spectral radius compare to the situation when only scales are being optimized. The basis function itself is depicted in Figure 3b. We can see that it is self-similar and seems to be well defined (in a sense that a subsampled basis function for $L_1 > L_2$ is a good basis function for L_2). We can deduce that this function is a limit of some subdivision scheme [40], but we could not reliably define subdivision weights from our numerical experiments.

5.2.2 Helmholtz equation

The first column in Figure 4 contains the results for Helmholtz equation (18) with $k^2 h$ equal to 0.01, 0.1 and 1. The results are similar to the one for the Poisson equation. However, if we further increase the number of points or k, the resulting matrix becomes indefinite, and the optimization breaks down. That means that with our approach, we cannot construct preconditioners for the Helmholtz equation. It is known that preconditioners for the Helmholtz equation significantly differ from preconditioners for Poisson-like equations (see [17] for the review), so this result is not surprising.

5.2.3 Anisotrpoic Poisson equation

The second column in Figure 4 contains the results for anisotropic Poisson equation (19) with ϵ equal to 10, 100 and 1000. To cope with the anisotropy, we apply semicoarsening [47, Section 5.1]. Without semicoarsening a "projector" on the grid $M_k \times M_k$ (M_k is as in (13)) reads $\widetilde{B}_L^k \otimes \widetilde{B}_L^k$. For semicoarsening the hierarchy of grids is modified, that is, in place of $M_k \times M_k$ we project on $M_{\min(k-s,0)} \times M_k$, where s

---

[1] This definition of N guarantees $\left\| e^{n+N} \right\| / \left\| e^n \right\| \leq 0.1$ for normal iteration matrix $M(\omega, A)$. If $M(\omega, A)$ is not normal, N holds as an estimation (see the discussion in Section 2 after equation (6)).

[2] To estimate the growth rate we fit data using ordinary least squares with the model $\kappa(L) = c_1 + c_2 L$. For BPX preconditioner $(c_1, c_2) = (0.792, 1.196)$, and for the optimized BPX $(c_1, c_2) = (1.164, 0.264)$.

L	BPX (r) ρ	κ	N	optimized BPX (r) ρ	κ	N
3	0.727	6.337	8	0.657	4.834	6
4	0.898	18.524	22	0.616	4.213	5
5	0.964	54.813	64	0.652	4.746	6
6	0.986	145.244	168	0.744	6.811	8

(a) $\sigma = 10$

L	BPX (r) ρ	κ	N	optimized BPX (r) ρ	κ	N
3	0.753	7.082	9	0.614	4.177	5
4	0.912	21.831	26	0.692	5.504	7
5	0.97	66.626	77	0.739	6.652	8
6	0.989	186.977	216	0.809	9.485	11

(b) $\sigma = 100$

L	BPX ρ	κ	N	optimized BPX ρ	κ	N
3	0.68	5.255	6	0.45	2.638	3
4	0.751	7.044	9	0.511	3.086	4
5	0.79	8.51	10	0.553	3.479	4
6	0.813	9.685	12	0.577	3.731	5

(c) $\tau = 0.5$

L	BPX ρ	κ	N	optimized BPX ρ	κ	N
3	0.817	9.93	12	0.697	5.599	7
4	0.89	17.24	20	0.814	9.781	12
5	0.922	24.787	29	0.864	13.752	16
6	0.935	29.933	35	0.894	17.811	21

(d) $\tau = 0.9$

L	BPX ρ	κ	N	optimized BPX ρ	κ	N
3	0.908	20.723	24	0.067	1.144	1
4	0.979	94.196	109	0.033	1.069	1
5	0.995	407.671	470	0.016	1.033	1
6	0.99	1702.583	1961	0.017	1.031	1

(e) $\widetilde{\mu} = h/2$

L	BPX ρ	κ	N	optimized BPX ρ	κ	N
3	0.641	4.57	6	0.307	1.885	2
4	0.729	6.383	8	0.391	2.287	3
5	0.789	8.47	10	0.505	2.919	4
6	0.845	11.876	14	0.709	5.875	7

(f) $\widetilde{\mu} = 2/h$

Figure 7: Results of optimization for: first row (a, b) diffusion with discontinuous coefficients (24) ((r) refers to rescaled version (30)), second row (c, d) Laplace operator with mixed derivative (25), last row (e, f) matrix from Crank-Nicolson scheme (28).

quantifies the extent to which the grid along one direction is denser than a grid in the other direction. With this modification, a preconditioner itself takes a form

$$\widetilde{\mathcal{B}}_s = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2 \left(\widetilde{B}_{\min(k-s,0)}^L \otimes \widetilde{B}_k^L \right) \left(\widetilde{B}_L^{\min(k-s,0)} \otimes \widetilde{B}_L^k \right). \tag{29}$$

As a result, the coarsening is delayed for y because $\epsilon > 1$ in (19), i.e., y is a direction of the strong coupling. Note that in Figure 4 we the compare (29) with original BPX preconditioner. If semicoarsening is applied to the BPX preconditioner, the weights α_k need to be modified. Original weights α_k combined with semicoarsening lead to worse performance. We can see that the optimization was able to fix the weights correctly. Moreover, comparing to semicoarsening applied in the context of filtering preconditioners [46] we were able to perform more aggressive coarsening, i.e., to decrease the number of floating-point operations.

5.2.4 Biharmonic equation

Results for the biharmonic equation are given in Figure 5. We can see that the BPX preconditioner is relatively inefficient. It was able to substantially decrease the condition number compared to the original matrix (this condition number is not listed), but still, the condition number is large and grows like $\kappa_{L+1} \simeq 4\kappa_L$. Condition number for the optimized BPX preconditioner is not only smaller but grows like $\kappa_{L+1} \simeq 2\kappa_L$. The basis function on Figure 5b does not seem to be stable in this case. Authors in [46] were managed to obtain a better preconditioner for the biharmonic equation using larger filters. The same applies to the case of multigrid solvers, where orders of interpolation n_i and restriction n_r operators should fulfill $n_i + n_r > n_l$ [47, Remark 2.7.1], where n_l is the order of the linear operator (4 in the case of biharmonic equation). Given that, we can suggest that by increasing the basis function's support, one can achieve a better condition number. We will study this elsewhere.

5.2.5 Convection-diffusion equation

Convection-diffusion equation leads to a non-symmetric matrix. Because of this, we do not list spectral condition number in Figure 6. Here optimization results in about twice as efficient solver, but the improvement becomes less pronounced for larger convection coefficient values.

5.2.6 Diffusion with discontinuous coefficients

Because neither BPX nor modified BPX account for the variation of coefficients, we used a rescaled version of preconditioner

$$\widetilde{\mathcal{B}}_r = \sum_{k=1}^{L} (\widetilde{\alpha}_k)^2 \, \widetilde{B}_k^L D \left(B_L^k A B_k^L \right)^{-1/2} \widetilde{B}_L^k, \tag{30}$$

where $D(\cdot)$ denotes the diagonal part of the matrix. For the original BPX preconditioner we again insert a diagonal part in-between "projectors" and use α_k as in (14). Results are given in the first row of Figure 7. It is evident that it is enough to recover the correct scales $\widetilde{\alpha}_k$. This was achieved by optimization which produces a good preconditioner regardless of scale.

The other option would be to perform a Jacobi preconditioning step $A \rightarrow D(A)^{-1/2} A D(A)^{-1/2}$ as explained in [8, discussion after equation (5.2)] and (in relation to diffusion with discontinuous coefficients) in [49, Section 3.1]. If this kind of rescaling is performed, BPX becomes a reasonable preconditioner, and optimization leads to results similar to the observed ones for the Poisson equation.

5.2.7 Mixed derivative

Results can be found in the second row of Figure 7. We can see that optimization is better for smaller values of τ, but when τ becomes closer to one, optimization deteriorates.

5.2.8 Implicit scheme for heat equation

Results are in the third row of Figure 7. We study problem (28) in two regimes. The first one corresponds to small time steps $\widetilde{\mu} = h/2$ used when the transient dynamic is of interest. In this case $I - (\widetilde{\mu}/2) A \simeq I$ so the preconditioner is not needed. As a result, BPX applied in a naive manner increases the condition number. The alternative solution would be to apply BPX preconditioner to the second matrix only, i.e., $I - (\widetilde{\mu}/2) BAB$, which solves this problem. However, the goal was to access the optimization, so we keep this experiment. In the other regime $\widetilde{\mu} = 2/h$ and one is interested in steady-state. In this situation, optimization again helps to decrease the spectral condition number. The last regime related to the elliptic equation with a linear source (different sign compare to the Helmholtz equation) for which a robust preconditioner was constructed in [22] with the help of a sophisticated subspace splitting technique.

6 Conclusion

In this article, we study the direct optimization of the spectral condition number. We derive two new loss functions, demonstrate how they are related to the spectral condition number, and show how stochastic optimization can be used to construct locally optimal preconditioners. We test our approach on a parametric family of modified BPX preconditioners. Optimization results show that for a large class of linear equations, automatic construction of reasonable preconditioners is possible. We want to emphasize that for many equations above, other more specialized preconditioners are available. There are also robust Schwarz preconditioners that are applicable for a broad class of second-order elliptic problems (see [18], [16]). The proposed approach differs from the previous attempts in three respects. First, described algorithms allow for a black-box construction of preconditioners, should a suitable parametrization is available. That means it is theoretically possible to apply the proposed approach in the algebraic setting as well. Second, the resulting preconditioner is locally optimal. The technique developed in [16] undoubtedly leads to a robust preconditioner. However, there is no guarantee that the resulting preconditioner is optimal. Since we are using stochastic gradient descent to directly optimize the spectral condition number of a preconditioner system, we can be sure that we achieve locally optimal preconditioner.[3] Third, proposed algorithms can be potentially applied to a wider class of linear problems, f.e., different discretizations and higher-order equations. As a downside, our approach currently is not practically applicable for real problems because optimization includes thousands of matrix-vector products. However, it could be possible to transfer from optimization to learning, i.e., to construct a model that can be trained on small matrices and applied on larger matrices as it was done for the multigrid method [21]. This is the focus of our current investigations.

7 Acknowledgement

References

[1] Raymond E. Alcouffe, Achi Brandt, Joel E. Dendy, Jr, and James W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM Journal on Scientific and Statistical Computing*, 2(4):430–454, 1981.

[2] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.

[3] Ivo M Babuska and Stefan A Sauter. Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers? *SIAM Journal on numerical analysis*, 34(6):2392–2423, 1997.

---

[3] We can not guarantee global optimality within a giving family of preconditioners. The practical approach would be to use numerical continuation as explained in [29].

[4] Markus Bachmayr and Vladimir Kazeev. Stability of low-rank tensor representations and structured multilevel preconditioning for elliptic PDEs. *Foundations of Computational Mathematics*, pages 1–62, 2020.

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[6] James H Bramble. A second order finite difference analog of the first biharmonic boundary value problem. *Numerische Mathematik*, 9(3):236–249, 1966.

[7] James H Bramble, Joseph E Pasciak, and Jinchao Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22, 1990.

[8] Marian Brezina. *Robust iterative methods on unstructured meshes*. PhD thesis, University of Colorado at Denver, 1997.

[9] Jed Brown, Yunhui He, Scott MacLachlan, Matt Menickelly, and Stefan M Wild. Tuning multigrid methods with robust optimization. *arXiv preprint arXiv:2001.00887*, 2020.

[10] Tony F Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM journal on scientific and statistical computing*, 9(4):766–771, 1988.

[11] Guo Chen, Zhilin Li, and Ping Lin. A fast finite difference method for biharmonic equations on irregular domains. Technical report, North Carolina State University. Center for Research in Scientific Computation, 2004.

[12] Edmond Chow and Yousef Saad. Approximate inverse preconditioners for general sparse matrices. *Res. Rep. UMSI*, 94(1.01), 1994.

[13] Edmond Chow and Yousef Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19(3):995–1023, 1998.

[14] Philippe G Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[15] Lothar Collatz. *The numerical treatment of differential equations*, volume 60. Springer Science & Business Media, 2012.

[16] Yalchin Efendiev, Juan Galvis, Raytcho Lazarov, and Joerg Willems. Robust domain decomposition preconditioners for abstract symmetric positive definite bilinear forms. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(5):1175–1199, 2012.

[17] Yogi A Erlangga. Advances in iterative methods and preconditioners for the Helmholtz equation. *Archives of Computational Methods in Engineering*, 15(1):37–66, 2008.

[18] Juan Galvis and Yalchin Efendiev. Domain decomposition preconditioners for multiscale flows in high-contrast media. *Multiscale Modeling & Simulation*, 8(4):1461–1483, 2010.

[19] Martin J Gander and Felix Kwok. Chladni figures and the Tacoma bridge: motivating pde eigenvalue problems via vibrating plates. *SIAM Review*, 54(3):573–596, 2012.

[20] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.

[21] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid PDE solvers. In *International Conference on Machine Learning*, pages 2415–2423. PMLR, 2019.

[22] Michael Griebel and Peter Oswald. Tensor product type subspace splittings and multilevel iterative methods for anisotropic problems. *Advances in Computational Mathematics*, 4(1):171, 1995.

[23] Marcus J Grote and Thomas Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[24] Murli M Gupta and Ram P Manohar. Direct solution of the biharmonic equation using noncoupled approach. *Journal of Computational Physics*, 33(2):236–248, 1979.

[25] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95. Springer, 1994.

[26] A Hadjidimos. Successive overrelaxation (SOR) and related methods. *Journal of Computational and Applied Mathematics*, 123(1-2):177–199, 2000.

[27] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.

[28] John David Jackson. Classical electrodynamics, 1999.

[29] Alexandr Katrutsa, Talgat Daulbaev, and Ivan Oseledets. Black-box learning of multigrid parameters. *Journal of Computational and Applied Mathematics*, 368:112524, 2020.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Victor Kozyakin. On accuracy of approximation of the spectral radius by the Gelfand formula. *Linear Algebra and its Applications*, 431(11):2134–2141, 2009.

[32] Vladimir D Liseikin. *Grid generation methods*. Springer, 2017.

[33] Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh. Learning algebraic multigrid using graph neural networks. *arXiv preprint arXiv:2003.05744*, 2020.

[34] Man Kwong Mak, Chun Sing Leung, and Tiberiu Harko. Solving the nonlinear biharmonic equation by the laplace-adomian and adomian decomposition methods. *arXiv preprint arXiv:1810.09544*, 2018.

[35] Thomas A Manteuffel. Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration. *Numerische Mathematik*, 31(2):183–208, 1978.

[36] Robert E Newnham. *Properties of materials: anisotropy, symmetry, structure*. Oxford University Press on Demand, 2005.

[37] Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC press, 2012.

[38] John K Reid. A method for finding the optimum successive over-relaxation parameter. *The Computer Journal*, 9(2):200–204, 1966.

[39] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[40] Olivier Rioul. Simple regularity criteria for subdivision schemes. *SIAM Journal on Mathematical Analysis*, 23(6):1544–1576, 1992.

[41] J Barkley Rosser. Nine-point difference solutions for Poisson's equation. *Computers & Mathematics with Applications*, 1(3-4):351–360, 1975.

[42] Jonas Schmitt, Sebastian Kuckuk, and Harald Köstler. Optimizing geometric multigrid methods with evolutionary computation. *arXiv preprint arXiv:1910.02749*, 2019.

[43] J Steppeler, R Hess, U Schättler, and Luca Bonaventura. Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics*, 82(1):287–301, 2003.

[44] Gilbert Strang. A proposal for Toeplitz matrix calculations. *Studies in Applied Mathematics*, 74(2):171–176, 1986.

[45] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[46] Charles H Tong, Tony F Chan, and CC Jay Kuo. Multilevel filtering preconditioners: Extensions to more general elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 13(1):227–242, 1992.

[47] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.

[48] Evgenij E Tyrtyshnikov. Optimal and superoptimal circulant preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 13(2):459–473, 1992.

[49] Andrew J Wathen. Preconditioning. *Acta Numerica*, 24, 2015.

[50] Xuejun Zhang. Multilevel Schwarz methods. *Numerische Mathematik*, 63(1):521–539, 1992.