Learning to Optimize Multigrid PDE Solvers

Daniel Greenfeld [1] Meirav Galun [1] Ron Kimmel [2] Irad Yavneh [2] Ronen Basri [1]

Abstract

Constructing fast numerical solvers for partial differential equations (PDEs) is crucial for many scientific disciplines. A leading technique for solving large-scale PDEs is using multigrid methods. At the core of a multigrid solver is the prolongation matrix, which relates between different scales of the problem. This matrix is strongly problem-dependent, and its optimal construction is critical to the efficiency of the solver. In practice, however, devising multigrid algorithms for new problems often poses formidable challenges. In this paper we propose a framework for learning multigrid solvers. Our method learns a (single) mapping from a family of parameterized PDEs to prolongation operators. We train a neural network once for the entire class of PDEs, using an efficient and unsupervised loss function. Experiments on a broad class of 2D diffusion problems demonstrate improved convergence rates compared to the widely used Black-Box multigrid scheme, suggesting that our method successfully learned rules for constructing prolongation matrices.

1. Introduction

Partial Differential Equations (PDEs) are a key tool for modeling diverse problems in science and engineering. In all but very specific cases, the solution of PDEs requires carefully designed numerical discretization methods, by which the PDEs are approximated by algebraic systems of equations. Practical settings often give rise to very large ill-conditioned problems, e.g., in predicting weather systems, oceanic flow, image and video processing, aircraft and auto design, electromagnetics, to name just a few. Developing efficient solution methods for such large systems has therefore been an active research area since many decades ago.

Multigrid methods are leading techniques for solving large-scale discretized PDEs, as well as other large-scale problems (for textbooks see, e.g., (Briggs et al., 2000; Trottenberg et al., 2001)). Introduced about half a century ago as a method for fast numerical solution of scalar elliptic boundary-value problems, multigrid methods have since been developed and adapted to problems of increasing generality and applicability. Despite their success, however, applying off-the-shelf multigrid algorithms to new problems is often non-optimal. In particular, new problems often require expertly devised prolongation operators, which are critical to constructing efficient solvers. This paper demonstrates that machine learning techniques can be utilized to derive suitable operators for wide classes of problems.

We introduce a framework for learning multigrid solvers, which we illustrate by applying the framework to 2D diffusion equations. At the heart of our method is a neural network that is trained to map discretized diffusion PDEs to prolongation operators, which in turn define the multigrid solver. The proposed approach has three main attractive properties:

Scope. We train a *single* deep network *once* to handle *any* diffusion equation whose (spatially varying) coefficients are drawn from a given distribution. Once our network is trained it can be used to produce solvers for any such equation. Our goal in this paper, unlike existing paradigms, is not to learn to solve a given problem, but instead to learn compact *rules* for constructing solvers for many different problems.

Unsupervised training. The network is trained with no supervision. It will not be exposed to ground truth operators, nor will it see numerical solutions to PDEs. Instead, our training is guided by algebraic properties of the produced operators that allude to the quality of the resulting solver.

Generalization. While our method is designed to work with problems of arbitrary size, it will suffice to train our system on quite small problems. This will be possible due to the local nature of the rules for determining the prolongation operators. Specifically, we train our system on block periodic problem instances using a specialized block Fourier mode analysis to achieve efficient training. At test time we generalize for size (train on 32×32 grid and test on a 1024×1024 grid), boundary conditions (train with periodic BCs and test with Dirichlet), and instance types (train on block periodic

[1]Weizmann Institute of Science, Rehovot, Israel. [2]Technion, Israel Institute of Technology, Haifa, Israel. Correspondence to: Daniel Greenfeld <daniel.greenfeld@weizmann.ac.il>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

instances and test on general problem instances). We compare our method to the widely used Black Box multigrid scheme (Dendy (Jr.), 1982) for selecting operator-dependent prolongation operators, demonstrating superior convergence rates under a variety of scenarios and settings.

1.1. Previous efforts

A number of recent papers utilized NN to numerically solve PDEs, some in the context of multigrid methods. Starting with the classical paper of (Lagaris et al., 1998), many suggested to design a network to solve specific PDEs (Hsieh et al., 2019; Baque et al., 2018; Baymani et al., 2010; Berg & Nyström, 2018; Han et al., 2017; 2018; Katrutsa et al., 2017; Mishra, 2018; Sirignano & Spiliopoulos, 2018; Sun et al., 2003; Tang et al., 2017; Wei et al., 2018), generalizing for different choices of right hand sides, boundary conditions, and in some cases to different domain shapes. These methods require separate training for each new equation.

Some notable approaches in this line of work include (Tang et al., 2017), who learn to solve diffusion equations on a fixed grid with variable coefficients and sources drawn randomly in an interval. A convolutional NN is utilized, and its depth must grow (and it needs to be retrained) with larger grid sizes. (Hsieh et al., 2019) proposes an elegant learning based approach to accelerate existing iterative solvers, including multigrid solvers. The method is designed for a specific PDE and is demonstrated with the Poisson equation with constant coefficients. It is shown to generalize to domains which differ from the training domain. (Berg & Nyström, 2018) handle complex domain geometries by penalizing the PDE residual on collocation points. (Han et al., 2018; Sirignano & Spiliopoulos, 2018) introduce efficient methods for solving specific systems in very high dimensions. (Mishra, 2018) aims to reduce the error of a standard numerical scheme over a very coarse grid. (Sun et al., 2003) train a neural net to solve the Poisson equation over a surface mesh. (Baque et al., 2018) learn to simulate computational fluid dynamics to predict the pressures and drag over a surface. (Wei et al., 2018) apply deep reinforcement learning to solve specific PDE instances. (Katrutsa et al., 2017) use a linear NN to derive optimal restriction/prolongation operators for solving a single PDE instance with multigrid. The method is demonstrated on 1D PDEs with constant coefficients. The tools suggested, however, do not offer ways to generalize those choices to other PDEs without re-training.

More remotely, several recent works, e.g., (Chen et al., 2019; Haber et al., 2018; Chang et al., 2018) suggest an interpretation of neural networks as dynamic differential equations. Under this continuous representation, a multilevel strategy is employed to accelerate training in image classification tasks.

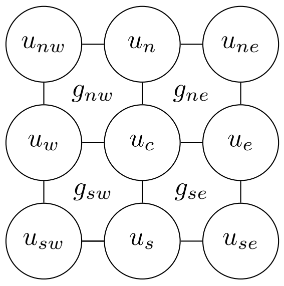

Figure 1. Sub-grid of 3×3. The discrete diffusion coefficients g are defined at cell centers. The discrete solution u and the discrete right hand side f are located at the vertices of the grid. The discrete equation for u_c has nine non-zero coefficients multiplying the unknowns u_c and its eight neighbors.

2. Multigrid background and problem setting

We focus on the classical second-order elliptic diffusion equation in two dimensions,

$$-\nabla \cdot (\mathbf{g} \nabla \mathbf{u}) = \mathbf{f}, \tag{1}$$

over a square domain, where $\mathbf{g}$ and $\mathbf{f}$ are given functions, and the unknown function $\mathbf{u}$ obeys some prescribed boundary conditions, for example, Dirichlet boundary conditions whereby $\mathbf{u}$ is given at every point on the boundary. The equation is discretized on a square grid of $n \times n$ grid cells with uniform mesh-size h. The discrete diffusion coefficients g are defined at cell centers, while the discrete solution vector u and the discrete right-hand side vector f are located at the vertices of the grid, as illustrated in the 3×3 sub-grid depicted in Fig. 1.

Employing bilinear finite element discretization, we obtain the following equation associated with the variable u_c,

$$\begin{aligned}
&-\frac{1}{3h^2}(g_{nw}u_{nw} + g_{ne}u_{ne} + g_{se}u_{se} + g_{sw}u_{sw}) \\
&-\frac{1}{6h^2}\big((g_{nw} + g_{ne})u_n + (g_{ne} + g_{se})u_e + \\
&(g_{se} + g_{sw})u_s + (g_{sw} + g_{nw})u_w\big) \\
&+\frac{2}{3h^2}(g_{nw} + g_{ne} + g_{se} + g_{sw})u_c = f_c\,.
\end{aligned} \tag{2}$$

Arranging these equations in matrix-vector form, we obtain a linear system

$$Au = f, \tag{3}$$

where $A_{c,j}$ is the coefficient multiplying u_j in the discrete equation associated with u_c. The term "the stencil of u_c" will refer to the 3×3 set of coefficients associated with the equation for u_c.

The discretization matrix A is symmetric positive semi-definite (and strictly positive definite in the case of Dirichlet boundary conditions) and sparse, having at most nine nonzero elements per row, corresponding to the nine stencil elements. The size of u, i.e., the number of unknowns, is approximately n^2 (with slight variations depending on whether or not boundary values are eliminated), while the size of A is approximately $n^2 \times n^2$. For large n, these properties of A render iterative methods attractive. One simple option is the classical Gauss-Seidel relaxation, which is induced by the splitting $A = L + U$, where L is the lower triangular part of A, including the diagonal, and U is the upper triangular part of A. The resulting iterative scheme,

$$u^{(k)} = u^{(k-1)} + L^{-1}\left(f - Au^{(k-1)}\right), \tag{4}$$

is convergent for symmetric positive definite matrices. Here, (k) denotes the iteration number. The error after iteration k, $e^{(k)} = u - u^{(k)}$, is related to the error before the iteration by the error propagation equation,

$$e^{(k)} = Se^{(k-1)}, \tag{5}$$

where $S = I - L^{-1}A$ is the error propagation matrix of the Gauss-Seidel relaxation, with I denoting the identity matrix of the same dimension as A.

Although the number of elements of A is $O(n^4)$, Gauss-Seidel iteration requires only $O(n^2)$ arithmetic operations because A is extremely sparse, containing only $O(n^2)$ nonzero elements. Nevertheless, as a stand-alone solver Gauss-Seidel is very inefficient for large n because the matrix A is highly ill-conditioned resulting in slow convergence. However, Gauss-Seidel is known to be very efficient for *smoothing* the error. That is, after a few Gauss-Seidel iterations, commonly called relaxation sweeps, the remaining error varies slowly relative to the mesh-size, and it can therefore be approximated well on a coarser grid. This motivates the multigrid algorithm, which is described next.

2.1. Multigrid Cycle

A coarse grid is defined by skipping every other mesh point in each coordinate, obtaining a grid of $\frac{n}{2} \times \frac{n}{2}$ grid cells and mesh-size $2h$. A prolongation operator P is defined and it can be represented as a sparse matrix whose number of rows is equal to the size of u and the number of columns is equal to the number of coarse-grid variables, approximately $\left(\frac{n}{2}\right)^2$. The two-grid version of the multigrid algorithm proceeds by applying one or more relaxation sweeps on the fine grid, e.g., Gauss-Seidel, obtaining an approximation $\tilde{u}$ to u, such that the remaining error, $u - \tilde{u}$ is smooth and can therefore be approximated well on the coarse grid. The linear system for the error is then projected to the coarse grid by the Galerkin method as follows. The coarse grid operator is defined as P^TAP and the right-hand-side is the restriction of the residual to the coarse grid, i.e., $P^T(f - A\tilde{u})$. Then, the coarse-grid system is solved directly in the two-grid algorithm, recursively in multigrid, and the resulting solution is transferred by the prolongation P to the fine grid and added to the current approximation. This is typically followed by one or more additional fine-grid relaxation sweeps. This entire process comprises a single two-grid iteration as formally described in Algorithm 1.

Algorithm 1 Two-Grid Cycle

```
1: Input: Discretization matrix $A$, initial approximation $u^{(0)}$, right-hand side $f$, prolongation matrix $P$, a relaxation scheme, $k = 0$, residual tolerance $\delta$
2: repeat
3:     Perform $s_1$ relaxation sweeps starting with the current approximation $u^{(k)}$, obtaining $\tilde{u}^{(k)}$
4:     Compute the residual: $r^{(k)} = f - A\tilde{u}^{(k)}$
5:     Project the error equations to the coarse grid and solve the coarse grid system: $P^TAPv^{(k)} = P^Tr^{(k)}$
6:     Prolongate and add the coarse grid solution: $\tilde{u}^{(k)} = \tilde{u}^{(k)} + Pv^{(k)}$
7:     Perform $s_2$ relaxation sweeps obtaining $u^{(k+1)}$
8:     $k = k + 1$
9: until $r^{(k-1)} < \delta$
```

In the multigrid version of the algorithm, Step 5 is replaced by one or more recursive calls to the two-grid algorithm, employing successively coarser grids. A single recursive call yields the so-called multigrid V cycle, whereas two calls yield the W cycle. These recursive calls are repeated until reaching a very coarse grid, where the problem is solved cheaply by relaxation or an exact solve. The entire multigrid cycle thus obtained has linear computational complexity. The W cycle is somewhat more expensive than the V cycle but may be cost-effective in particularly challenging problems.

The error propagation equation of the two-grid algorithm is given by

$$e^{(k)} = Me^{(k-1)}, \tag{6}$$

where $M = M(A, P; S, s_1, s_2)$ is the two-grid error propagation matrix

$$M = S^{s_2}CS^{s_1}. \tag{7}$$

Here, s_1 and s_2 are the number of relaxation sweeps performed before and after the coarse-grid correction phase, and the error propagation matrix of the coarse grid correction is given by

$$C = (I - P\left[P^TAP\right]^{-1}P^TA). \tag{8}$$

For a given operator A, the error propagation matrix M defined in (7) governs the convergence behavior of the two-grid (and consequently multigrid) cycle. The cycle efficiency relies on the complementary roles of the relaxation

S and the coarse-grid correction C; that is, the error propagation matrix of the coarse grid correction phase, C, must reduce significantly any error which is not reduced by S, called *algebraically smooth error*.

For symmetric positive definite A and full-rank P, as we assume throughout this discussion, the matrix $P\left[P^TAP\right]^{-1}P^TA$ in (8) is an A-orthogonal projection onto the range of P (i.e., the subspace spanned by the columns of P). Thus, C, the error propagation matrix of the coarse grid correction phase (8), essentially subtracts off the component of the error that is in the range of P. This requires that the algebraically smooth error will approximately be in the range of P. The task of devising a good prolongation is challenging, because P also needs to be very sparse for computational efficiency.

Commonly, a specific relaxation scheme, such as Gauss-Seidel, is preselected, as are the number of relaxation sweeps per cycle, and therefore the efficiency of the cycle is governed solely by the prolongation operator P. The challenging task therefore is to devise effective prolongation operators. A common practice for diffusion problems on structured grids is to impose on P the sparsity pattern of bilinear interpolation[1] and then to skillfully select values of the nonzero elements of P based locally on the elements of the discretization matrix A. In contrast, our approach is to automatically learn the local rules for determining the prolongation coefficients by training a single neural network, which can be applied to the entire class of diffusion equations discretized by 3×3 stencils.

3. Method

We propose a scheme for learning a mapping from discretization matrices to prolongation matrices. We assume that the diffusion coefficients are drawn from some distribution, yielding a distribution $\mathcal{D}$ over the discretization matrices. A natural objective would be to seek a mapping that minimizes the expected spectral radius of the error propagation matrix $M(A, P)$ defined in (7), which governs the asymptotic convergence rate of the multigrid solver. Concretely, we represent the mapping with a neural network parameterized by θ that maps discretization matrices A to prolongations $P_\theta(A) \in \mathcal{P}$ with a predefined sparsity pattern. The relaxation scheme S is fixed to be Gauss-Seidel, and the parameters s_1, s_2 are set to 1. Thus, we arrive at the following learning problem:

$$\min_{P_\theta \in \mathcal{P}} \mathbf{E}_{A\sim\mathcal{D}} \;\; \rho(M(A, P_\theta(A))), \tag{9}$$

[1]Assume that u_c in the subgrid diagram in Fig. 1 coincides with a coarse-grid point U_c. Then the column of P corresponding to U_c contains nonzero values only at the rows corresponding to the nine fine-grid variables appearing in the diagram.

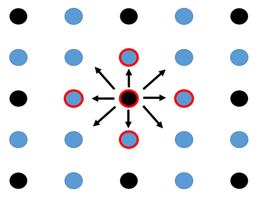

Figure 2. The input and the output of the network. The discs denote the (fine) grid points, where the black discs mark the subset of points selected as coarse grid points. The input of the network consists of the 3×3 stencils of the five points, denoted by the red cycles. The black arrows illustrate the output of the network, i.e., the contribution of the prolongation of one coarse point to its eight fine grid neighbors.

where $\rho(M)$ is the spectral radius of the matrix M, and $\mathcal{D}$ is the distribution over the discretization matrices A.

3.1. Inferring P from local information

The network we construct receives an input vector of size 45, consisting of a local subset of the discretization matrix A, and produces an output that consists of 4 numbers, which in turn determine the 9 nonzero entries of one column of the prolongation matrix P. Existing multigrid solvers for diffusion problems on structured grids (e.g., (Alcouffe et al., 1981; de Zeeuw, 1990; Dendy (Jr.), 1982)), infer the prolongation weights from local information. Following their approach, we construct our network to determine each column j of P from five 3×3 stencils. Specifically, the input to the network is composed of the stencil of the fine grid point coinciding with coarse point j, and the stencils of its four immediate neighbors, marked by the red circles in Fig. 2.

For the output we note that the sparsity pattern imposed on P implies that each column has at most nine non-zero elements, where each non-zero element P_{ij} is the prolongation weight of the coarse grid point j to a nearby fine grid point i. Geometrically, this means that a coarse grid point contributes only to the fine-grid point with which it coincides (and the corresponding prolongation coefficient is set to 1) and to its eight fine-grid neighboring points, as illustrated in Fig. 2. Only the four prolongation coefficients corresponding to the nearest neighbors are learned; the four remaining prolongation coefficients, marked by diagonal arrows in Fig. 2, are then calculated such that any grid function u obtained by prolongation from the coarse grid satisfies $Au = 0$ at these four grid points. The complete prolongation matrix P is constructed by applying the same network repeatedly to all the coarse points.

The inference from local information maintains the efficiency of the resulting multigrid cycle, as the mapping has constant time computation per coarse grid point, and we construct P by applying the network repeatedly to all coarse grid points. Moreover, the local nature of the inference allows application of the network on different grid-sizes. Further details are provided in Section 4.

3.2. Fourier analysis for efficient training

The fact that the network determines P locally does not mean that it suffices to train on very small grids. Because the method is to be used for large problems, it is critical that the subspace spanned by the columns of P will approximate well all algebraically smooth errors of large problems, as discussed, e.g., in (Falgout, 2006). This implies that such errors should be encompassed in the loss function of the training phase. In practice, our experiments show that good performance on large grids is already obtained after training only on a 32×32 grid, which is not very large but still results in an error propagation matrix M of size 1024×1024.

The main computational barrier of the loss (9) is due to the coarse-grid correction matrix C (8), whose computation requires inversion of the matrix $P^T AP$ of size $(n/2)^2 \times (n/2)^2$ elements. To overcome this prohibitive computation, we introduce two surrogates. First, we relax the spectral radius of the error propagation matrix with its squared Frobenious norm, relying on the fact that the Frobenious norm bounds the spectral radius from above, yielding a differentiable quantity without the need for (expensive) spectral decomposition. Secondly, we train on a relatively limited class of discretization matrices, A, which are called block-circulant matrices, allowing us to train efficiently on large problems, because it requires inversion only of small matrices, as explained below. Due to the local dependence of P on A, we expect that the resulting trained network would be equally effective for general (non block-periodic) A, and this is indeed borne out in our experiments.

The block-periodic framework allows us to train efficiently on large problems. To do so, we exploit a block Fourier analysis technique that was recently introduced independently in several variants and for different applications (Bolten & Rittich, 2018; Brown et al., 2018; Kumar et al., 2018). Classical Fourier analysis has been employed for quantitative prediction of two-grid convergence factors since the 1970s. This technique, however, is exact only in very special cases of constant-coefficient operators and simple boundary conditions. Here, in contrast, we need to cater to arbitrary discrepancies in the values of the diffusion coefficients of neighboring grid cells, which imply strongly varying coefficients in the matrix A, so classical Fourier analysis is not appropriate.

To apply the new block Fourier analysis, we partition our $n \times n$ grid into equal-sized square blocks of $c \times c$ cells each, such that all the $\frac{n}{c} \times \frac{n}{c}$ blocks are identical with respect to their cell g values, but within the block the g values vary arbitrarily, according to the original distribution. This can be thought of as tiling the domain by identical blocks of $c \times c$ cells. Imposing periodic boundary conditions, we obtain a discretization matrix A that is block-circulant. Furthermore, due to the dependence of P on A, the matrix M itself is similarly block-circulant and can be written as

$$M = \begin{bmatrix} M_0 & M_1 & \dots & M_{b-2} & M_{b-1} \\ M_{b-1} & M_0 & M_1 & \dots & M_{b-2} \\ M_{b-2} & M_{b-1} & M_0 & \dots & M_{b-3} \\ \dots & \dots & \dots & \dots & \dots \\ M_1 & \dots & M_{b-2} & M_{b-1} & M_0 \end{bmatrix}, \quad (10)$$

where $M_j, j = 0, \dots, b-1$, are $c^2 \times c^2$ blocks and $b = \frac{n^2}{c^2}$. This special structure has the following important implication. M can easily be block-diagonalized in a way that each block of size $c^2 \times c^2$ on the diagonal has a simple closed form that depends on the elements of A and a single parameter associated with a certain Fourier component. As a result, the squared Frobenius norm of the matrix M, which constitutes the loss for our network, can be decomposed into a sum of squared Frobenius norms of these small easily computed blocks, requiring only the inversion of relatively small matrices.

The theoretical foundation of this essential tool is summarized briefly below. For further details, we refer the reader to the supplemental material and to (Bolten & Rittich, 2018; Brown et al., 2018).

Block diagonalization of block circulant matrices Let the $n \times n$ matrix K be block-circulant, with b blocks of size k. That is, $n = bk$, and the elements of K satisfy:

$$K_{l,j} = K_{\mathrm{mod}(l-k,n),\mathrm{mod}(j-k,n)}, \quad (11)$$

with rows, column, blocks, etc., numbered starting from 0 for convenience. Here, we are adopting the MATLAB form $\mathrm{mod}(x, y)$ = "x modulo y", i.e., the remainder obtained when dividing integer x by integer y. Below, we continue to use l and j to denote row and column numbers, respectively, and apply the decomposition:

$$l = l_0 + tk, \quad j = j_0 + sk, \quad (12)$$

where $l_0 = \mathrm{mod}(l, k)$, $t = \lfloor \frac{l}{k} \rfloor$, $j_0 = \mathrm{mod}(j, k)$, $s = \lfloor \frac{j}{k} \rfloor$. Note that $l, j \in \{0, ..., n-1\}$; $l_0, j_0 \in \{0, ..., k-1\}$; $t, s \in \{0, ..., b-1\}$.

Let the column vector

$$v_m = \left[1, e^{i\frac{2\pi m}{n}}, \dots, e^{i\frac{2\pi mj}{n}}, \dots, e^{i\frac{2\pi m(n-1)}{n}}\right]^*$$

denote the unnormalized mth Fourier component of dimension n, where $m = 0, \dots, n-1$. Finally, let W denote the

$n \times n$ matrix whose nonzero values are comprised of the elements of the first b Fourier components as follows:

$$W_{l,j} = \frac{1}{\sqrt{b}} \delta_{l_0, j_0} v_s(l) \,, \tag{13}$$

where $v_s(l)$ denotes the lth element of v_s, and δ is the Kronecker delta. Then we have:

Theorem 1. *W is a unitary matrix, and the similarity transformation $\hat{K} = W^* K W$ yields a block-diagonal matrix with b blocks of size $k \times k$, $\hat{K} = \mathrm{blockdiag}\left(\hat{K}^{(0)}, ..., \hat{K}^{(b-1)}\right)$. Furthermore, if K is band-limited modulo n such that all the nonzero elements in the lth row of K, $l = 0, ..., n-1$, are included in $\{K_{l,\mathrm{mod}(l-\alpha,n)}, ..., K_{l,l}, ..., K_{l,\mathrm{mod}(l+\beta,n)}\}$, and $\beta + \alpha + 1 \le k$, then the nonzero elements of the blocks are simply*

$$\begin{aligned} \hat{K}^{(s)}_{l_0,\mathrm{mod}(l_0+m,k)} &= e^{-i\frac{2\pi s m}{n}} K_{l_0,\mathrm{mod}(l_0+m,n)} \,, \\ l_0 &= 0, ..., k-1, \;\; m = -\alpha, ..., \beta \,. \end{aligned}$$

The proof is in the supplementary material.

By applying Theorem 1 recursively, we can block diagonalize M (10) for our 2D problems. In practice, for computational efficiency, we perform an equivalent analysis using Fourier symbols for each of the multigrid components as is commonly done in multigrid Fourier analysis (see, e.g., (Wienands & Joppich, 2004)). We finally compute the loss

$$\|M\|_F^2 = \|\hat{M}\|_F^2 = \sum_{s=0}^{b-1} \|\hat{M}^{(s)}\|_F^2,$$

where $\hat{M} = \mathrm{blockdiag}\left(\hat{M}^{(0)}, ..., \hat{M}^{(b-1)}\right)$. Note that, $\|\hat{M}^{(s)}\|_F^2$ is cheap to compute since $\hat{M}^{(s)}$ is of size $c^2 \times c^2$ ($c = 8$ in our experiments).

To summarize, Theorem 1 allows us to train on block- periodic problems with grid size of $n \times n$ using $\frac{n^2}{c^2}$ matrices of size $c^2 \times c^2$ instead of a matrix of size $n^2 \times n^2$.

4. Experiments

For evaluating our algorithm several measures are employed, and we compare the performance of our network based solver to the classical and widely used Black Box multigrid scheme (Dendy (Jr.), 1982). To the best of our knowledge, this is the most efficient scheme for prolongation construction for diffusion problems. We train and test the solver for the diffusion coefficients g sampled from a log-normal distribution, which is commonly assumed, e.g., in modeling flow in porous media (cf. (Moulton et al., 1998)), where Black Box prolongation is used for homogenization in this regime). As explained above, the network is trained to minimize the Frobenious norm of the error propagation matrix of rather small grids comprised of circulant blocks and periodic boundary conditions. However, the tests are performed for a range of grid sizes, general non block-periodic g, Dirichlet boundary conditions, and even a different domain. Finally, we remark that the run-time per multigrid cycle of the network based algorithm is the same as that of Black Box multigrid, due to the identical sparsity pattern. However, the once-per-problem setup phase of the network based algorithm is more expensive than that of Black Box scheme because the former uses the trained network to determine P whereas the latter uses explicit formulas.

Network details The inputs and outputs to our network are specified in Sec. 3.1. We train a residual network consisting of 100 fully-connected layers of width 100 with RELU activations. Note that all matrix-dependent multigrid methods, including Black-Box, apply local nonlinear mappings to determine the prolongation coefficients.

Handling the singularity Employing block Fourier analysis, as we do for efficiency, requires training with periodic boundary conditions. This means that our discretization matrices A are singular, with null space comprised of the constant vector. This in turn means that $P^T AP$ is also singular and cannot be inverted, so M cannot be explicitly computed. We overcome this problem by taking two measures. First, we impose that the sum of each row of P be equal to 1. This ensures that the null space of the coarse-grid matrix $P^T AP$ too is comprised of the (coarse-grid) constant vector. Second, when computing the loss with the block Fourier analysis, we ignore the undefined block which corresponds to the zeroth Fourier mode (i.e., the constant vector). To force the rows of the prolongation to sum to one, we simply normalize the rows of P that are learned by the network (left, right, above and below each coarse-grid point) before completing the construction of P as described in Section 3.1. When dealing with Dirichlet boundary conditions, this constraint is not feasible for rows corresponding to points near the boundary. For those points, we use the prolongation coefficients proposed by the Black Box algorithm.

Training details Training is performed in three stages. First, the network was trained for two epochs on 163840 diffusion problems with grid-size 16×16 composed of 8×8 doubly-periodic core blocks and with doubly periodic boundary conditions. This results in an tentative network, which is further trained as follows. The tentative network was used to create prolongation matrices for 163840 non block-periodic diffusion problems with grid-size 16×16 and periodic boundary conditions. Then, using Galerkin coarsening $P^T AP$, this resulted in 163840 8×8 blocks corresponding to coarse level blocks, which were used as core blocks for generating 16×16 block periodic problems.

Table 1. Spectral radius of the two-grid error propagation matrix M for a 64×64 grid with Dirichlet boundary conditions (smaller is better).

METHOD	SPECTRAL RADIUS
BLACK BOX	0.1456 ± 0.0170
NETWORK	0.1146 ± 0.0168

Now, at the second stage, the new training set which consists of 2×163840 problems, was used for additional two epochs. After that, at the last stage, those 8×8 core blocks were used to compose problems of grid-size 32×32, and the training continued for two additional epochs. The second stage was done to facilitate good performance on coarse grids as well, since in practice a two grid scheme is too expensive and recursive calls are made to solve the coarse grid equation. The network was initialized using the scheme suggested in (Zhang et al., 2019). Throughout the training process, the optimizer used was Adam, with an initial learning rate drawn from $10^{-U([4,6])}$.

4.1. Evaluation

Spectral radius As a first evaluation, we present the spectral radius of the two-grid error propagation matrix obtained with our network on 64×64 grid problems with Dirichlet boundary conditions, where the diffusion coefficients were drawn from a log-normal distribution. Table 1 shows the results, averaged over 100 instances. We observe that the network based algorithm clearly outperforms Black Box multigrid by this measure, achieving a lower average $\rho(M)$, despite the discrepancies between the training and testing conditions (block-periodic g, Frobenius norm minimization and smaller grid in the training, versus general g, Dirichlet boundary conditions, spectral radius and larger grid in the tests).

Multigrid cycles Numerical experiments are performed with V and W cycles. In each experiment, we test 100 instances with Dirichlet boundary conditions, and the diffusion coefficients in each instance are drawn from a log-normal distribution. We solve the homogenous problem $Au = 0$, with the initial guess for the solution drawn from a normal distribution[2]. In each experiment we run 40 multigrid cycles and track the error norm reduction factor per cycle, $\frac{||e^{(k+1)}||_2}{||e^{(k)}||_2}$. We consider the ratio in the final iteration to be the asymptotic value.

[2]Due to the linearity of the problem and the algorithm, the convergence behavior is independent of f and of the Dirichlet boundary values; we choose the homogeneous problem in order to allow us to run many cycles and reach the worst-case asymptotic regime without encountering roundoff errors when the absolute error is on the order of machine accuracy.

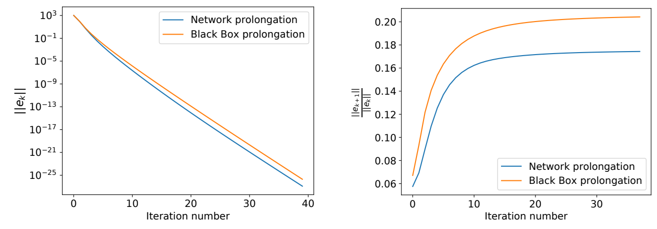

Figure 3. W-cycle performance, averaged over 100 problems with grid size 1024×1024 and Dirichlet Boundary conditions. Left: error norm as a function of iterations (W cycles). Right: error norm reduction factor per iteration.

Figure 3 (left) shows the norm of the error as a function of the iteration number for a W cycle, where the fine grid-size is 1024×1024 and nine grids are employed in the recursive multigrid hierarchy. Both algorithms exhibit the expected fast multigrid convergence. Figure 3 (right) shows the error reduction factor per iteration for this experiment. We see that the mean convergence rates increase with the number of iterations but virtually level off at asymptotic convergence factors of about 0.2 for Black Box multigrid and about 0.16 for the network-based method.

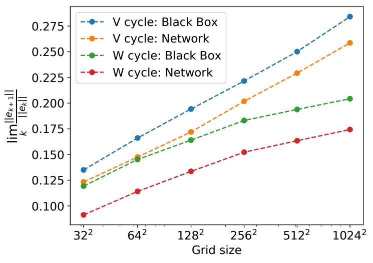

Figure 4. V cycle and W cycle average asymptotic error norm reduction factor per iteration.

Figure 4 shows the asymptotic error norm convergence factors per cycle of V and W cycles with fine-grid sizes ranging from 32×32 to 1024×1024. Additionally, Table 2 shows the success rate of the network based method, defined as the percentage of instances in which it outperformed the Black Box algorithm in terms of asymptotic convergence factor. Evidently, the network based method is superior by this measure, and we see no significant deterioration for larger grids, even though the training was performed on relatively small grids and with block-periodic g.

Uniform distribution As a test of robustness with respect to the diffusion coefficient distribution, we evaluate the network trained with log-normal distribution on a different distribution of the g values. Here, we present the results of applying multigrid cycles as in the previous experiment, except that in these tests the diffusion coefficients are drawn

Table 2. Success rate of V cycle and W cycle with log-normal g distribution.

GRID SIZE	V-CYCLE	W-CYCLE
32×32	83 %	100 %
64×64	92 %	100 %
128×128	91 %	100 %
256×256	84 %	99 %
512×512	81 %	99 %
1024×1024	83 %	98 %

Table 3. Success rate of V cycle and W cycle with uniform g distribution.

GRID SIZE	V-CYCLE	W-CYCLE
32×32	60 %	90 %
64×64	54 %	90 %
128×128	66 %	91 %
256×256	79 %	91 %
512×512	81 %	88 %
1024×1024	81 %	96 %

from the uniform distribution over $[0, 1]$. The results are shown in Figure 5, with Table 3, as before, showing the success rate of the network in these tests. Evidently, the advantage of the network based method is narrower in this case, due to the mismatch of distributions, but it still exhibits superior convergence factors.

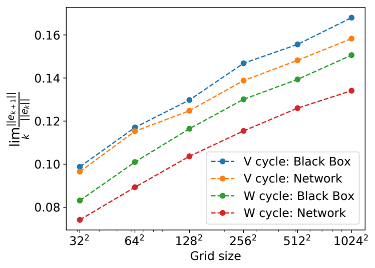

Figure 5. V cycle and W cycle average asymptotic error norm reduction factor per iteration tested with uniform g distribution, with network trained on log-normal distribution.

Non-square domain In the next experiment, we test our network on diffusion problems specified on a domain consisting of a two-dimensional disk. Our method achieves a better convergence rate in this case too, see Table 4.

Diagonally dominant problems In the final experiment, we evaluate the algorithms for a variant of the problem where a positive constant ε has been added to the diagonal, corresponding to the PDE

$$-\nabla(g \cdot \nabla u) + \varepsilon u = f. \tag{14}$$

Table 4. Asymptotic error reduction factor per cycle on a $2D$ disk with a diameter of 64 grid points, averaged over 100 instances.

METHOD	V-CYCLE	W-CYCLE
BLACK BOX	0.1969 ± 0.0290	0.1639 ± 0.0169
NETWORK	0.1868 ± 0.0296	0.1352 ± 0.0155

This test is relevant, in particular, to time-dependent parabolic PDE, where the diagonal term stems from discretization of the time derivative. For this experiment, we trained a second network, following the same training procedure as before, where for the training instances we used $\varepsilon h^2 = 10^{-8}$. Figure 6 indicates that the network based algorithm retains its advantage in those kind of problems also, and is able to perform well on different values of εh^2.

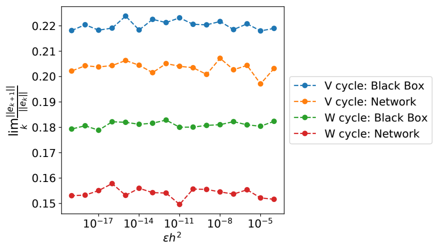

Figure 6. Experiments with varying values of εh^2 added to the diagonal. The graphs show the asymptotic error norm reduction factor of the V cycle and W cycles per iteration, averaged over 100 experiments with grid size 256×256 ($h^2 = 1/65536$).

5. Conclusion

In this work we introduced a framework for devising multigrid solvers for parametric families of PDEs. Posed as a learning problem, this task is approached by learning a single mapping from discretization matrices to prolongation operators, using an efficient and unsupervised learning procedure. Experiments on 2D diffusion equations show improved convergence rates compared to the classical Black Box scheme, which has withstood the test of time for decades. Moreover, the experiments show generalization properties with respect to the problem size, boundary conditions and to some extent, its underlying distribution. Extending our work to triangulated and unstructured grids is an exciting direction we intend to pursue, as well as exploring simpler regression models which will allow for faster inference.

References

Alcouffe, R. E., Brandt, A., Dendy, J. E., and Painter, J. W. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat.*

Comput., 2:430–454, 1981.

Baque, P., Remelli, E., Fleuret, F., and Fua, P. Geodesic convolutional shape optimization. *arXiv:1802.04016 [cs.CE]*, 2018.

Baymani, M., Kerayechian, A., and Effati, S. Artificial neural networks approach for solving stokes problem. *Applied Mathematics*, 1(04):288, 2010.

Berg, J. and Nyström, K. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

Bolten, M. and Rittich, H. Fourier analysis of periodic stencils in multigrid methods. *SIAM J. Sci. Comput.*, 40 (3):A1642–A1668, 2018.

Briggs, W. L., Henson, V. E., and McCormick, S. F. *A multigrid tutorial*. SIAM, second edition, 2000.

Brown, J., He, Y., and Maclachlan, S. Local Fourier analysis of BDDC-like algorithms. *Submitted*, 2018.

Chang, B., Meng, L., Haber, E., Tung, F., and Begert, D. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations*, 2018.

Chen, R., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *CoRR*, abs/1806.07366, 2019.

de Zeeuw, P. M. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *J. Comput. Appl. Math.*, 33:1–27, 1990.

Dendy (Jr.), J. E. Black box multigrid. *J. Comput. Phys.*, 48:366–386, 1982.

Falgout, R. D. An introduction to algebraic multigrid. *IEEE: Computing in Science and Engineering*, 8:24–33, 2006.

Haber, E., Ruthotto, L., Holtham, E., and Jun, S.-H. Learning across scales—multiscale methods for convolution neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Han, J., Jentzen, A., and Weinan, E. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning. *CoRR*, abs/1707.02568, 2017.

Han, J., Jentzen, A., and Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Hsieh, J., Zhao, S., Eismann, S., Mirabella, L., and Ermon, S. Learning neural PDE solvers with convergence guarantees. *ICLR*, 2019.

Katrutsa, A., Daulbaev, T., and Oseledets, I. Deep multigrid: learning prolongation and restriction matrices. *arXiv:1711.03825v1 [math.NA]*, 2017.

Kumar, P., Rodrigo, C., Gaspar, F. J., and Oosterlee, C. W. On cell-centered multigrid methods and local Fourier analysis for PDEs with random coefficients. *arXiv:1803.08864 [math.NA]*, 2018.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 1998.

Mishra, S. A machine learning framework for data driven acceleration of computations of differential equations. *arXiv:1807.09519 [math.NA]*, 2018.

Moulton, J. D., Dendy, J. E., and Hyman, J. M. The black box multigrid numerical homogenization algorithm. *J. Comput. Phys.*, 142(1):80–108, 1998.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *arXiv:1708.07469 [q-fin.MF]*, 2018.

Sun, M., Yan, X., and Sclabassi, R. J. Solving partial differential equations in real-time using artificial neural network signal processing as an alternative to finite-element analysis. In *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, volume 1, pp. 381–384. IEEE, 2003.

Tang, W., Shan, T., Dang, X., Li, M., Yang, F., Xu, S., and Wu, J. Study on a poissons equation solver based on deep learning technique. *EDAPS conference*, 2017.

Trottenberg, U., Oosterlee, C., and Schüller, A. *Multigrid*. Academic Press, London and San Diego, 2001.

Wei, S., Jin, X., and Li, H. General solutions for nonlinear differential equations: a deep reinforcement learning approach. *CoRR*, abs/1805.07297, 2018.

Wienands, R. and Joppich, W. *Practical Fourier analysis for multigrid methods*. Chapman and Hall/CRC, 2004.

Zhang, H., Dauphin, Y. N., and Ma, T. Residual learning without normalization via better initialization. In *International Conference on Learning Representations*, 2019.