Karlsruhe Institute of Technology

wave phenomena
analysis and numerics

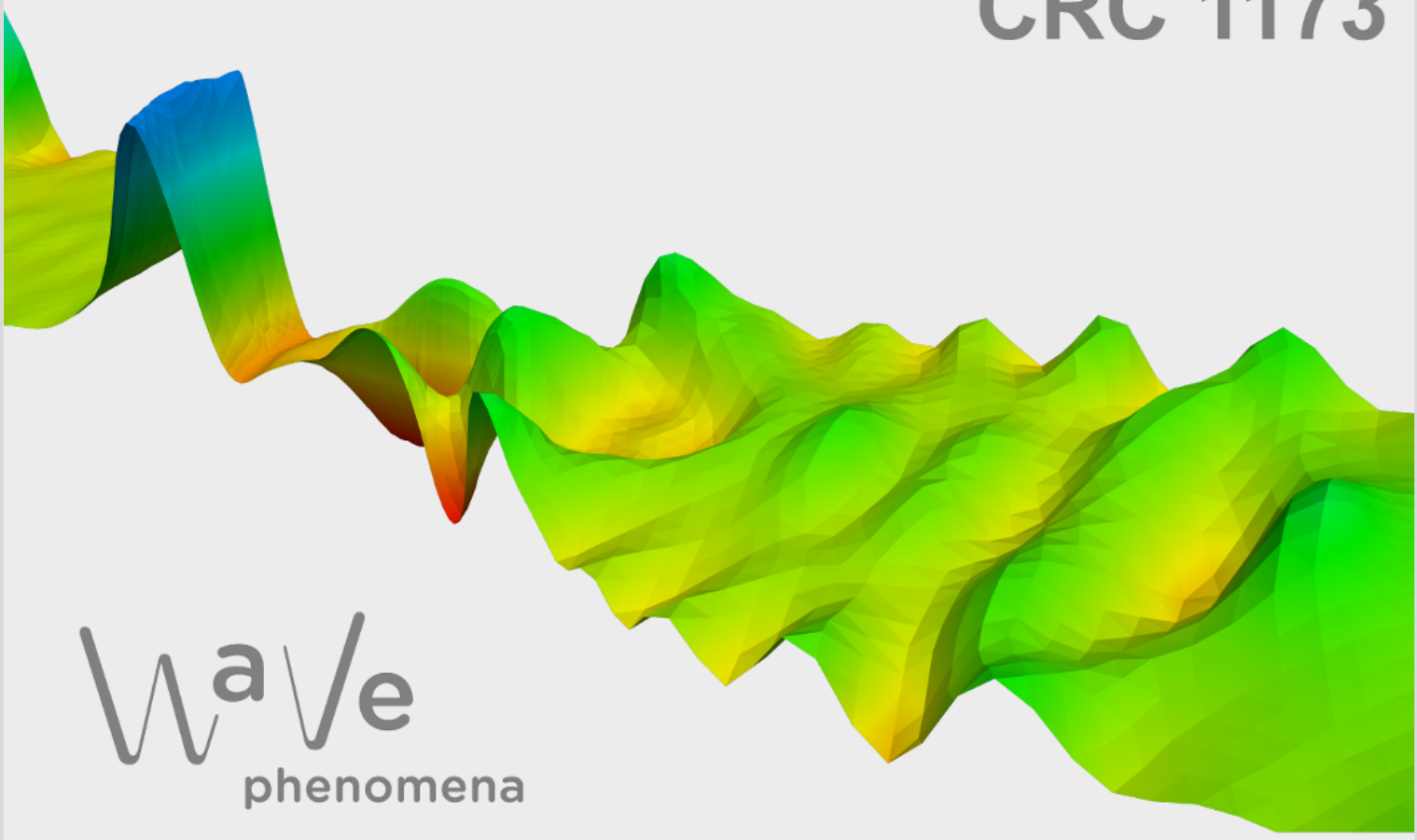
Complexity bounds on neural networks for the solution of structured linear systems of equations

Benjamin Dörich, Roland Maier, Lukas Ullmer

CRC Preprint 2026/12, April 2026

KARLSRUHE INSTITUTE OF TECHNOLOGY

CRC 1173



Participating universities

Funded by

ISSN 2365-662X

COMPLEXITY BOUNDS ON NEURAL NETWORKS FOR THE SOLUTION OF STRUCTURED LINEAR SYSTEMS OF EQUATIONS*

BENJAMIN DÖRICH†, ROLAND MAIER†, AND LUKAS ULLMER

Abstract. We derive upper bounds on the complexity of ReLU neural networks approximating the solution of a linear system given the matrix and the right-hand side. We focus on matrices which are symmetric positive definite and sparse, as they appear in the context of finite difference and finite element methods. For such matrices, we extend available results for the matrix inversion to the task of solving a linear system, where we leverage favorable properties of classical methods such as the modified Richardson and the conjugate gradient method. Our bounds on the number of layers and neurons are not only explicit with respect to the size of the matrices, but also with respect to their condition numbers.

Key words. Approximation theory, neural networks, iterative methods for linear systems

MSC codes. 35A35, 41A25, 41A46, 65N30, 68T07,

1. Introduction.

In this work, we investigate the complexity of neural networks for the approximate solution of linear systems of the form

$$Ax = r, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $n \in \mathbb{N}$ is a (possibly large) matrix with a certain structure, $r \in \mathbb{R}^n$ a given vector, and $x \in \mathbb{R}^n$ the (unknown) solution to the system. Specifically, we have in mind linear systems that arise in the context of finite element or finite difference methods. These have the property that A is typically sparse with a known sparsity structure, symmetric, and in general have a large condition number. The main goal is to understand the realization of the mapping $(A, r) \mapsto x = A^{-1}r$ via classical feed-forward neural networks. While we do not aim at practically realizing such a construction, the main goal is to understand the complexity of a possible realization and, particularly, its dependence on the condition number and the sparsity pattern. In essence, this shall provide a rough estimation on the necessary size of a neural network to realize the solution of a linear system of the form (1.1) for a specific class of matrices.

The study of approximation properties and sufficient complexity bounds on neural networks for different tasks is an active field of research. First theoretical results on approximation capabilities for certain classes of functions have been presented in [4, 9] but without explicit dependencies on the size of the networks. An approximation result with explicit rates has first been obtained in [2] and important findings on the approximation of a scalar multiplication operator based on the polarization formula are presented in [23], which have been employed to approximate smooth functions with ReLU neural networks in the L^∞-norm. These findings can be extended to approximation results in Sobolev norms, see [7]. In [18], a framework for studying approximation properties of neural networks has been presented that is based on the

*Submitted to the editors DATE.

Funding: B. Dörich and R. Maier acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 258734477 – SFB 1173. Parts of this work were conducted during B. Dörich's and R. Maier's stay at the Hausdorff Research Institute for Mathematics funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2047/2 – 390685813.

†Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology, 76149 Karlsruhe, Germany (benjamin.doerich@kit.edu, roland.maier@kit.edu).

idea of using smaller neural networks as building blocks for certain, more complicated, approximation tasks. It has been used to show approximation results in the context of, e.g., higher-order finite elements [16, 17] and parametrized/high-dimensional partial differential equations [5, 13, 15] (see also [6] for a corresponding numerical study on the complexity). An overview on the expressivity of neural networks can be found in [8].

The main goal of this work is to approximate the solution process of a potentially sparse linear systems of equations by mimicking iterative algorithms. In [13], the inversion of a matrix is approximated up to arbitrary precision by a neural network based on the Neumann series. Further, upper bounds on the size of the neural network are presented in terms of depth and number of non-zero neurons. While this approach could be directly used to construct a neural network that implements the solution of a linear system, the overall complexity scales worse than n^3 for solving a system with n unknowns, which even exceeds the effort for inverting the matrix directly. Note that the main motivation in [13] are rather small matrices and uniformly bounded condition numbers in the context of a reduced basis method, where the cubic dependence is not severe. In contrast, we have large matrices in mind, where such a complexity is prohibitively expensive. Particularly for sparse linear systems in the context of, e.g., finite element methods, the complexity bounds appear suboptimal, as well-known classical iterative methods exist that do not require the inversion of the matrix and can even solve such systems with almost linear complexity. Note that the dominant cubic scaling in [13] can be improved to $n^{\log_2 7} \approx n^{2.8}$ as recently proposed in [20]. This is possible by constructing the networks for the matrix-matrix multiplication via the *Strassen algorithm* [22] and implementing those in the Neumann series. For sparse linear systems, this is still suboptimal and to the best of our knowledge, there are no other works studying the complexity analysis of the solution of linear systems of equations specifically.

In this work, we aim at constructing a neural network that realizes the solution of a linear system of equations and which makes use of iterative algorithms, particularly for sparse systems. Exemplarily, we construct neural networks that realize the modified Richardson and the cg-algorithm. Compared to the mentioned works, we pay special attention to the role of the condition number and, in particular, prove our estimates with an explicit tracing of the condition number. One of the main findings is that the complexity suffers from large condition numbers, mainly since the underlying iterative methods deteriorate in this case. Note, however, that our approach opens up the possibility to study the combination of iterative methods with pre-conditioners. This can enable the construction of upper bounds on the necessary size of neural networks which are less sensitive to large condition numbers.

Besides the interesting question of how the (worst possible) condition of a matrix influences the construction of a neural network and in particular its size, the considerations of this work may also be applied in the context of more involved constructions to understand the complexity of neural networks for specific tasks. For instance, in [12], the complexity estimate in [13] directly enters in the context of replacing bottleneck computations in finite element-based multiscale methods by appropriate neural networks. The suboptimal scaling of the size of required neural networks therein is automatically and substantially improved with our construction.

Outline of the paper. We first introduce our notation in Section 2 and present for reference the important result on the approximate matrix inversion established in [13] adapted to our setting. In Section 3, we present the overall framework as

well as our main results. Further, we discuss the application to stiffness matrices from a finite element background and the implications on other works that use neural networks in their algorithms. Several useful results, which are used throughout the analysis, are recalled in Section 4 and the building block for our main theorems are shown. Sections 5 and 6 are devoted to the proofs of our main results employing the Richardson and the cg-approach, respectively.

2. Notation. In this section, we fix the notation of several spaces and objects which are used throughout this work. First, we note that the dimension n of the system in (1.1) is fixed throughout the paper. We denote the space of admissible matrices and vectors appearing in (1.1) by

$$\mathcal{K}^z := \{A \in \mathbb{R}^{n\times n} \mid \|A\|_2 \le z\}, \qquad k^z := \{r \in \mathbb{R}^n \mid \|r\|_2 \le z\},$$

for some $z > 0$, where $\|\cdot\|_2$ denotes the standard Euclidean norm for vectors or the spectral norm for matrices, respectively. In addition, we need the class of symmetric, positive definite matrices with bounded spectrum, which we denote by

$$\Sigma^{\lambda,\Lambda} := \{A \in \mathbb{R}^{n\times n} \mid A = A^\top, \lambda \le \lambda_{\min}(A) \le \lambda_{\max}(A) \le \Lambda\},$$

where $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the smallest and largest eigenvalue of a matrix A, respectively. We denote the condition number of A by $\kappa(A)$ and recall that for any $A \in \Sigma^{\lambda,\Lambda}$ we have the uniform bound

$$\kappa(A) \le \frac{\Lambda}{\lambda} =: \widehat{\kappa}. \tag{2.1}$$

This bound is used for $\alpha \in \{\frac{1}{2}, 1\}$ in the context of the function

$$\rho_\alpha = \rho_\alpha(\widehat{\kappa}) = \frac{\widehat{\kappa}^\alpha - 1}{\widehat{\kappa}^\alpha + 1} \in [0, 1), \tag{2.2}$$

which will later relate to the convergence rate of our methods.

We further introduce the space of matrices with a special sparsity pattern. We define the list $\boldsymbol{\eta} = (\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_n)$, where $\boldsymbol{\eta}_i \in \{1, \dots, n\}^{\eta_i}$ are vectors of length η_i, respectively. $\boldsymbol{\eta}_i$ encodes the columns in the ith row of a matrix, where entries are allowed to be different from zero. In particular, if j is *not* an element of $\boldsymbol{\eta}_i$, then the entry $A_{i,j} = 0$ (but the reverse statement is not necessarily true). Matrices of a sparsity pattern contained in $\boldsymbol{\eta}$ are collected in the set

$$\mathcal{S}^{\boldsymbol{\eta}} = \{A \in \mathbb{R}^{n\times n} \text{ s.t. the sparsity pattern of } A \text{ is contained in } \boldsymbol{\eta}\ \}, \tag{2.3}$$

and set $\eta := \sum_{i=1}^n \eta_i$ as the maximal number of non-zero elements, and note that $n \le \eta \le n^2$. Further, we define

$$\eta_{\max} = \max_{i=1,\dots,n} \eta_i.$$

For the application we have in mind, it is reasonable to assume an a-priori known super-set of the sparsity pattern, and we discuss this after our main results. In addition, we discuss possible extension in Section 7 below.

Remark 2.1. Within this work, we assume that the matrix A is given in a COO-type format, i.e., there is a vector A^{v}, which contains the η values and two vectors

which contain the corresponding row and column index, respectively. However, since we have a fixed sparsity pattern, the two vectors for the row and column indices do not change and can be neglected.

In particular, this allows us to abuse the notation as follows: we will interchangeably use the matrix A and the vector with its (relevant) values A^{v}. That is, a matrix A as input of a neural network (cf. Definition 2.2 below) makes sense due to its (compressed) vector representation. Multiple matrices and/or vectors as inputs have to be understood as a long vector containing all the input values.

Next, we present the type of neural network which we consider in this work.

DEFINITION 2.2. *Let $L \in \mathbb{N}$. A neural network Φ of depth L consists of a list of matrix-vector-tupels*

$$\Phi = [(W_1, b_1), (W_2, b_2), \dots, (W_L, b_L)],$$

where the weights W_ℓ are a matrix in $\mathbb{R}^{N_\ell \times N_{\ell-1}}$ and the corresponding bias b_ℓ is a vector in $\mathbb{R}^{N_\ell}$ with $N_0, \dots, N_L \in \mathbb{N}$. We call $\dim_{\mathrm{in}}(\Phi) := N_0$ the input dimension and $\dim_{\mathrm{out}}(\Phi) := N_L$ the output dimension. Together with the ReLU activation function $\varrho\colon \mathbb{R} \to \mathbb{R}$, $x \mapsto \max\{0, x\}$ and the input vector x, the neural network is recursively defined via

$$\begin{aligned} \mathrm{x}_0 &:= \mathrm{x}, \\ \mathrm{x}_\ell &:= \varrho(W_\ell \mathrm{x}_{\ell-1} + b_\ell) \quad \text{for } \ell = 1, \dots, L-1, \\ \mathrm{x}_L &:= W_L \mathrm{x}_{L-1} + b_L, \end{aligned}$$

where ϱ acts component-wise on vectors by convention. Overall, for a given input x_0 the output x_L of Φ is given by the relation

$$\Phi(\mathrm{x}_0) = \mathrm{x}_L$$

and the neural network Φ can be understood as a map $\mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$. The network is said to have L layers and N_j is the number of neurons in the j-th layer. With $\|A\|_0$ the number of non-zero entries of A, for $\ell \leq L$ we denote by

$$M_\ell := \|W_\ell\|_0 + \|b_\ell\|_0$$

the number of weights in the ℓ-th layer and by

$$M := \sum_{\ell=1}^{L} M_\ell$$

the total number of weights of Φ. In addition, we use the functions $\mathfrak{L}$ and $\mathfrak{M}$ to assign a neural network Φ its number of layers and weights, i.e., $\mathfrak{L}(\Phi) = L$ and $\mathfrak{M}(\Phi) = M$.

As a reference result, we present the one from the seminal work [13], where the authors construct a neural network for a general matrix $A \in \mathbb{R}^{n \times n}$ without any sparsity pattern and under the assumption of a uniformly bounded condition number. As an important reference, we recall the essential parts of the result using our notation introduced above.

THEOREM 2.3 (cf. [13, Thm. 3.8]). *Let $\delta \in (0, 1)$. Then, for any $\epsilon \in (0, \frac{1}{4})$ there*

exists a neural network Φ^{inv} *such that*

$$\sup_{A\in\mathcal{K}^{1-\delta}} \left\| \big(\mathrm{I} - A\big)^{-1} - \Phi^{\text{inv}}\big(A\big) \right\|_2 \leq \epsilon,$$

where $\dim_{\text{in}}(\Phi^{\text{inv}}) = n^2$ *and* $\dim_{\text{out}}(\Phi^{\text{inv}}) = n^2$. *Further, with*

$$m_{\text{N}} = m_{\text{N}}(\epsilon,\delta) := \left\lceil \frac{\log_2(0.5\epsilon\delta)}{\log_2(1-\delta)} \right\rceil,$$

there exists a constant $C_{\text{inv}} > 0$ *which is independent of* m_{N}, n, ϵ, *and* δ *such that*

(i) $\mathfrak{L}\Big(\Phi^{\text{inv}}\Big) \leq C_{\text{inv}} \log_2(m_{\text{N}})\Big(\log_2(\frac{1}{\epsilon}) + \log_2(m_{\text{N}}) + \log_2(n)\Big)$,

(ii) $\mathfrak{M}\Big(\Phi^{\text{inv}}\Big) \leq C_{\text{inv}} m_{\text{N}} \log_2^2(m_{\text{N}})\Big(\log_2(\frac{1}{\epsilon}) + \log_2(m_{\text{N}}) + \log_2(n)\Big) n^3$.

The result in Theorem 2.3 is the starting point for our investigations. In the upcoming section, we aim to exploit the knowledge of a certain pattern that matrices in $\mathcal{S}^{\boldsymbol{\eta}}$ have; cf. (2.3). The goal is to reduce the computational complexity (in the sense of number of layers and neurons) also for matrices with a large condition number. In particular, we aim for a better scaling of the number of weights in the dimension n.

3. Main results. In this section, we first present our main results and apply them to a class of matrices as they usually appear in the context of finite element or finite difference methods. In particular, we are interested in symmetric and positive definite matrices with specific sparsity patterns. As an example of special interest and based on the notation in (2.3), we will mainly consider the case

$$\eta = \mathcal{O}(n), \quad \kappa(A) \gg 1.$$

In the context of finite elements, η_i represents the number of neighboring nodes of the ith node a_i in the mesh, which is a uniformly bounded number even under refinement. This directly implies the scaling of η. However, we could also account for systems with linear constraints such as the Poisson problem with Neumann boundary conditions, where the last row consists of $\mathcal{O}(n)$ non-zero elements. Since we aim at solving linear systems of the form (1.1), i.e.,

$$Ax = r,$$

an important aspect of our work is that we do not approximate the map $A \mapsto A^{-1}$, but rather $(A, r) \mapsto x = A^{-1}r$, which can be realized by an adaptation of well-known iterative methods.

3.1. Abstract results. We first present a result which resembles the modified Richardson iteration, and is thus closely related to the result from [13]. In addition, we employ a cg-type approach by constructing the optimal polynomial related to the required number of iteration steps to obtain a given tolerance.

Richardson method. For our first result, we briefly revisit the modified Richardson method [19]. Starting with the original system in (1.1), we rewrite the problem for $A \in \Sigma^{\lambda,\Lambda}$ as a fixed-point problem

$$x = (I - \omega A)x + \omega r \quad \text{with} \quad \omega = \frac{2}{\lambda + \Lambda}. \tag{3.1}$$

For an eigenvalue λ_i of A, we observe with $\widehat{\kappa}$ defined in (2.1) that

$$1 - \omega\lambda_i = \frac{\Lambda - 2\lambda_i + \lambda}{\lambda + \Lambda} = \frac{\widehat{\kappa} - 2\frac{\lambda_i}{\lambda} + 1}{\widehat{\kappa} + 1}.$$

Since $1 \leq \frac{\lambda_i}{\lambda} \leq \widehat{\kappa}$, the definition of ρ_1 in (2.2) gives $I - \omega A \in \mathcal{K}^{\rho_1}$. By assumption it holds $\rho_1 < 1$, and we have convergence due to the Banach fixed-point theorem. In addition, we obtain for $x^0 = 0$ the explicit formula

$$x^{m+1} = \sum_{\ell=0}^{m} (I - \omega A)^{\ell}(\omega r). \tag{3.2}$$

Thus, the construction is mainly based on two important steps: First, we pre-compute $\widehat{A} = I - \omega A$ and $\widehat{r} = \omega r$, and second, approximate the Neumann series (3.2) with a neural network, adapting the results in [13]. More precisely, the matrix-matrix products are replaced by matrix-vector products, which immediately reduces the complexity of the network. Further, we reduce the effort by taking the sparsity pattern of A into account. With this strategy, we are able to prove the following result.

THEOREM 3.1 (Approximation via modified Richardson iteration).
Let $0 < \lambda < \Lambda$ and let ω be given as in (3.1) and ρ_1 as in (2.2). Then, for any $\epsilon \in (0,1)$ and $c_{\mathrm{sc}} \in [1, \frac{1+\widehat{\kappa}}{2}]$ there exists a neural network Φ^{Ric} such that

$$\sup_{A \in \mathcal{S}^{\eta} \cap \Sigma^{\lambda,\Lambda},\, r \in k^{c_{\mathrm{sc}}\lambda}} \left\| A^{-1}r - \Phi^{\mathrm{Ric}}(A, r) \right\|_2 \leq \epsilon,$$

where $\dim_{\mathrm{in}}(\Phi^{\mathrm{Ric}}) = n(n+1)$ and $\dim_{\mathrm{out}}(\Phi^{\mathrm{Ric}}) = n$. Further, we define

$$m_{\mathrm{Ric}} = m_{\mathrm{Ric}}(\epsilon, c_{\mathrm{sc}}, \rho_1) = \left\lceil \frac{|\log_2(\frac{\epsilon}{2c_{\mathrm{sc}}})|}{|\log_2(\rho_1)|} \right\rceil.$$

Then there exists a constant $C_{\mathrm{Ric}} > 0$ independent of m_{Ric}, n, η, and ϵ, such that
(i) $\mathfrak{L}(\Phi^{\mathrm{Ric}}) \leq C_{\mathrm{Ric}} m_{\mathrm{Ric}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{Ric}})\big)$,
(ii) $\mathfrak{M}(\Phi^{\mathrm{Ric}}) \leq C_{\mathrm{Ric}} m_{\mathrm{Ric}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{Ric}})\big)\eta$.

Proof. The proof is postponed to Section 5. ∎

cg-type method. Next, we consider the approximation via a cg-type approach. To this end, we rescale $\widehat{A} = \frac{1}{\Lambda}A \in \mathcal{K}^1$ and $\widehat{r} = \frac{1}{\Lambda}r \in k^1$ for $r \in k^{\Lambda}$. For the construction, we decompose the approximation error as

$$A^{-1}r - \Phi^{\mathrm{cg}}(A, r) = \big(A^{-1}r - q_{m-1}(\widehat{A})\widehat{r}\big) + \big(q_{m-1}(\widehat{A})\widehat{r} - \Phi^{\mathrm{cg}}(A, r)\big), \tag{3.3}$$

with an appropriate polynomial $q_{,m-1} \in \mathcal{P}_{m-1}$, where $\mathcal{P}_{m-1}$ denotes the space of polynomials up to degree $m-1$. The first term on the right-hand side of (3.3) is estimated by classical numerical linear algebra results to obtain a suitable parameter $m_{\mathrm{cg}} = m_{\mathrm{cg}}(\epsilon)$. The network Φ^{cg} is then constructed in such a way that it approximates $q_{m-1}(\widehat{A})\widehat{r}$ up to the desired tolerance. This allows us to prove the following result.

THEOREM 3.2 (Approximation via cg-type iteration).
Let $0 < \lambda < \Lambda$ and let ω be given as in (3.2) and ρ_1 as in (2.2). Then, for any $\epsilon \in (0,1)$ and $c_{\mathrm{sc}} \in [1, \widehat{\kappa}]$, there exists a neural network Φ^{cg} such that

$$\sup_{A \in \mathcal{S}^{\eta} \cap \Sigma^{\lambda,\Lambda},\, r \in k^{c_{\mathrm{sc}}\lambda}} \left\| A^{-1}r - \Phi^{\mathrm{cg}}(A, r) \right\|_2 \leq \epsilon,$$

where $\dim_{\text{in}}(\Phi^{\text{cg}}) = n(n+1)$ *and* $\dim_{\text{out}}(\Phi^{\text{cg}}) = n$. *Further, define*

$$m_{\text{cg}} = m_{\text{cg}}(\epsilon, c_{\text{sc}}, \rho_{1/2}) = \left\lceil \frac{|\log_2(\frac{\epsilon}{4c_{\text{sc}}})|}{|\log_2(\rho_{1/2})|} \right\rceil.$$

Then, there exists a constant $C_{\text{cg}} > 0$ *independent of* m_{cg}, n, η, *and* ϵ, *such that*

(i) $\mathfrak{L}(\Phi^{\text{cg}}) \le C_{\text{cg}} m_{\text{cg}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\text{cg}})\big)$,

(ii) $\mathfrak{M}(\Phi^{\text{cg}}) \le C_{\text{cg}} m_{\text{cg}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m)\big)\eta$.

Proof. The proof is postponed to Section 6. ∎

Remark 3.3. The range of c_{sc} in Theorems 3.1 and 3.2 ensures that the rescaled version of the right-hand side fulfills $\|\widehat{r}\| \le 1$, which simplifies several computations in the proof. To be precise, we observe for the rescaling $\widehat{r} = \omega r$ in Theorem 3.1 that $\lambda c_{\text{sc}} \le \omega^{-1}$ and for $\widehat{r} = \frac{1}{\Lambda} r$ in Theorem 3.2 we have the relation $\lambda c_{\text{sc}} \le \Lambda$. Larger right-hand sides could be treated as well, which would lead to additional (logarithmic) terms in the complexity bounds, but we do not further discuss this here for the sake of readability.

Remark 3.4. For finite element methods, the influence of the condition number (through ρ_α in (2.2)) is important in determining the number of required iterations when using iterative algorithms. Therefore, the asymptotic scaling of terms involving ρ_α is essential.

We have $\frac{1}{|\log_2 \rho_\alpha|}$ in the denominator of both m_{Ric} and m_{cg}, which is the main difference in the size of both networks. Using the definition of ρ_α, we obtain

$$\rho_\alpha^{-1} = 1 + \tfrac{2}{\widehat{\kappa}^\alpha - 1}$$

and thus by the Taylor approximation of the logarithm

$$|\log_2 \rho_\alpha|^{-1} = |\log_2\big(1 + \tfrac{2}{\widehat{\kappa}^\alpha - 1}\big)|^{-1} \lesssim \widehat{\kappa}^\alpha$$

for $\widehat{\kappa}$ sufficiently large. Thus, we can see that for large values of $\widehat{\kappa}$, the cg-type network can be chosen to be much smaller than the Richardson-type one.

Remark 3.5. To compare our result with Theorem 2.3, we note that in the derivation in Section 2 of [13], the matrix A stems from a rescaled system, and the parameter δ is approximately given by $\frac{\lambda_{\min}(A)}{\lambda_{\max}(A)} = \frac{1}{\kappa} \in (0,1)$. In particular, we again observe the scaling

$$|\log_2(1-\delta)|^{-1} = |\log_2(1 + \tfrac{1}{\kappa - 1})|^{-1} \sim \kappa,$$

which resembles the scaling of our Richardson approach. However, let us emphasize that in [13] the network was applied to matrices which are "pre-conditioned" in a dimension-reduction step, such that therein the condition numbers are considered moderate.

3.2. Application to finite element methods. In the following, we discuss our results for the special case of matrices stemming from a finite element discretization of some partial differential equation $\mathrm{L}u = f$ for some second-order elliptic differential operator L, right-hand side $f \in L^2(\Omega)$, and bounded domain Ω.

Condition number. We denote the mesh width by h, which is the largest diameter over all elements. It is well-known that for quasi-uniform meshes, the scaling

of the eigenvalues implies

$$\lambda \sim h^d, \quad \Lambda \sim h^{d-2}.$$

Further, we note the relation $n \sim h^{-d}$ of the degrees of freedom and the mesh size, which allows us to deduce the scaling

$$\widehat{\kappa} \sim h^{-2} \sim n^{2/d},$$

for $\widehat{\kappa}$ defined in (2.1).

Band width. Focusing on regular meshes, we denote by N the number of degrees of freedom per spatial direction, i.e., we set $N = n^{1/d}$. In the simplest case, the sparsity pattern can be chosen to be a band of the matrix. For one-dimensional problems this allows to choose $\eta_i = 3$, and thus $\eta \leq 3n$, in dimension two $\eta_i \sim N$, and thus $\eta \lesssim nN \sim n^{3/2}$ and in dimension three $\eta_i \sim N^2$, and thus $\eta \lesssim nN^2 \sim n^{5/3}$.

Including more knowledge on the mesh structure, one can in principle also exploit that there is only a moderate number of non-zero entries per row, allowing $\eta_i \leq C$ for a uniform constant C depending on the maximal number of incident elements at each node. If this information is available for the whole class of considered problems, one can also obtain $\eta \lesssim n$, which is the typical case for finite element methods.

Load vector. Assembling the load vector r is performed by computing the integrals

$$r_i = \int_\Omega f\varphi_i \, dx$$

where φ_i are the standard ansatz function of the finite element method. Denoting the support of φ_i by $S_i = \operatorname{supp} \varphi_i$, we obtain for $f \in L^p(\Omega)$, $p \geq 2$ and conjugate $q = p/(p-1) \leq 2$,

$$|r_i| = |\int_{S_i} f\varphi_i \, dx| \leq \|f\|_{L^p(S_i)} \, \|\varphi_i\|_{L^q(S_i)} \lesssim \|f\|_{L^p(S_i)} \, h^{d/q}.$$

Hence, we have

$$\|r\|_2 \leq n^{1/2-1/p} \, \|r\|_p = n^{1/2-1/p} \big(\sum_{i=1}^n |r_i|^p \big)^{1/p} \lesssim n^{1/2-1/p} h^{d/q} \, \|f\|_{L^p(\Omega)} \,,$$

and by the relation $n = h^{-d}$ and $\frac{1}{q} = \frac{p-1}{p}$, we obtain

$$\|b\|_2 \lesssim h^{d/2} \, \|f\|_{L^p(\Omega)} \lesssim h^{-d/2} \lambda \, \|f\|_{L^p(\Omega)} \,.$$

In view of Theorem 3.1 and Theorem 3.2, this implies that $c_{\mathrm{sc}} \sim h^{-d/2} \sim n^{2/d}$, and thus the iteration numbers m_{Ric} and m_{cg} include an additional logarithmic scaling with respect to the degrees of freedom n.

Implication for constructed networks in the literature. In the specific setting that not directly the inversion of the matrix is of interest but rather the solution of a linear system, our results directly improve the complexity bounds that immediately follow from an application of the results in [13], see Theorem 2.3 above. Note that this is not a surprise as the results in [13] are much more general and

the main motivation therein is the solution of smaller systems in the context of a reduced basis approach. Our results are particularly advantageous in the case of large sparse matrices with a prescribed sparsity pattern, which is relevant in the context of, e.g., finite element or finite difference methods. In that regard, our results improve the complexity bounds for the construction in [12]. Therein, suitable (worst-case) complexity bounds are derived for the neural network realization of the multiscale method known as *localized orthogonal decomposition*, see also [11] and [14, 1] for an overview on the original methodology. The construction sets up stiffness matrices by using locally constructed neural networks on certain patches. More precisely, on a coarse mesh on the scale of interest $H > 0$ with $N = H^{-d}$ degrees of freedom a finite element-type method is constructed that takes into account fine-scale features from a much finer scale $h < H$. This comes at the cost of a slightly increased sparsity pattern with $|\log_2(H)|$ entries per row. While [12, Thm. 4.3] suggests a scaling of the number of layers and overall neurons of $\mathcal{O}(|\log_2(h)|^2)$ and $\mathcal{O}(h^{-2}|\log_2(h)|^4(|\log_2(H)|H/h)^{3d})$, respectively, by using the result in [13] (see Theorem 2.3 above), our construction in, e.g., Theorem 3.2 works with only $\mathcal{O}(h^{-1}|\log_2(h)|(|\log_2(H)|^2H/h)^d)$ neurons. The construction is deeper, though, requiring $\mathcal{O}(h^{-1}|\log_2(h)|^2)$ layers. With appropriate pre-conditioning, one may potentially even avoid the pre-factor h^{-1} for an almost optimal scaling (up to logarithmic terms), see also the discussion in Section 7.

4. Preliminaries for the proofs. The main goal of this section is to establish the construction and analysis of networks which can realize elementary maps that are essential for the two iterative methods. First, we consider the map used for the Richardson iteration given by

$$R\colon \mathbb{R}^\eta \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^\eta \times \mathbb{R}^n \times \mathbb{R}^n \quad (A, r, c) \mapsto (A, Ar, r + c), \tag{4.1}$$

which will be the cornerstone of the construction of the network in Theorem 3.1.

THEOREM 4.1. *Let $\epsilon \in (0, 1)$. Then, there exists a neural network Φ^R with $\dim_{\text{in}}(\Phi^R) = \eta + 2n$ and $\dim_{\text{out}}(\Phi^R) = \eta + 2n$ such that for $z \geq 1$*

$$\sup_{A \in \mathcal{S}^\eta \cap \mathcal{K}^1,\, r \in k^z,\, c \in \mathbb{R}^n} \|R(A, r, c) - \Phi^R(A, r, c)\|_2 \leq \epsilon$$

Further, there exists a generic constant $C_R > 0$ independent of n, A, r, ϵ, η, and c such that

(i) $\mathfrak{L}\Big(\Phi^R\Big) \leq C_R\Big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big)$,

(ii) $\mathfrak{M}\Big(\Phi^R\Big) \leq C_R\Big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big)\eta$.

The proof of the theorem is given at the end of this section. Second, for the cg-type iteration we construct the map

$$S_\alpha\colon \mathbb{R}^\eta \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^\eta \times \mathbb{R}^n \times \mathbb{R}^n, \quad (A, r, c) \mapsto (A, \alpha\mathbf{1} + 2Ar - c, r) \tag{4.2}$$

for some fixed $\alpha \leq 1$ and $\mathbf{1} \coloneqq (1, \dots, 1)^\top$. This map is the basis to perform a neural network version of the Clenshaw algorithm and thus to prove Theorem 3.2.

THEOREM 4.2. *Let $\epsilon \in (0, 1)$. Then, there exists a neural network Φ^{S_α} with $\dim_{\text{in}}(\Phi^{S_\alpha}) = \eta + 2n$ and $\dim_{\text{out}}(\Phi^{S_\alpha}) = \eta + 2n$ such that for $z \geq 1$*

$$\sup_{A \in \mathcal{S}^\eta \cap \mathcal{K}^1,\, r \in k^z,\, c \in \mathbb{R}^n} \|S_\alpha(A, r, c) - \Phi^{S_\alpha}(A, r, c)\|_2 \leq \epsilon$$

Further, there exists a generic constant $C_S > 0$ independent of n, A, r, c, ϵ, η, and α such that

(i) $\mathfrak{L}\left(\Phi^{S_\alpha}\right) \leq C_S\left(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\right)$,

(ii) $\mathfrak{M}\left(\Phi^{S_\alpha}\right) \leq C_S\left(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\right)\eta$.

The proof is also provided at the end of this section. Note that one could as well use an adaption of Theorem 4.1 for the Chebyshev polynomials, replacing the third entry of R defined in (4.1) by $\alpha r + c$. However, this results in the multiplication with different large coefficients α up to size $\sim 2^m$ and thus induces additional factors of m in the scaling of the layers and neurons.

We now provide all the auxiliary results to eventually prove the two theorems on the approximation of R and S_α, starting with the identity network.

LEMMA 4.3 (identity network, cf. Lem. 2.3 and Rem. 2.4 in [18]). *For any $L \in \mathbb{N}$, $L \geq 2$, there exists a neural network Φ^{id} that realizes the identity in $\mathbb{R}^k$ with* $\dim_{\mathrm{in}}(\Phi^{\mathrm{id}}) = \dim_{\mathrm{out}}(\Phi^{\mathrm{id}}) = k$

(i) $\mathfrak{L}\left(\Phi^{\mathrm{id}}\right) = L$,

(ii) $\mathfrak{M}\left(\Phi^{\mathrm{id}}\right) \leq 2kL$.

The identity network is an important ingredient to define appropriate (sparse) concatenations of neural networks. A main issue with a classical concatenation of two neural networks Φ^1 and Φ^2 (denoted $\Phi^1 \bullet \Phi^2$) is that it does not easily allow for an estimation of $M(\Phi^1 \bullet \Phi^2)$ which depends linearly on $M(\Phi^1)$ and $M(\Phi^2)$ (due to the merge of two layers from different networks). This issue can be resolved by the introduction of an intermediate identity network.

DEFINITION 4.4 (sparse concatenation). *Let the networks*

$$\Phi^1 = \left[(W_1^1, b_1^1), \dots, (W_{L_1}^1, b_{L_1}^1)\right], \quad \Phi^2 = \left[(W_1^2, b_1^2), \dots, (W_{L_2}^2, b_{L_2}^2)\right]$$

with $\dim_{\mathrm{in}}(\Phi^1) = \dim_{\mathrm{out}}(\Phi^2)$ *be given. Further, denote with Φ^{id} the identity network with input and output dimension* $\dim_{\mathrm{in}}(\Phi^1) = \dim_{\mathrm{out}}(\Phi^2)$. *Then we call*

$$\Phi^1 \circ \Phi^2 \coloneqq \Phi^1 \bullet \Phi^{\mathrm{id}} \bullet \Phi^2$$

the sparse concatenation of Φ^1 and Φ^2.

DEFINITION 4.5 (parallelization, cf. [18, 5]). *Let $\Phi^1, \dots, \Phi^j$ be neural networks with L layers each. Then the neural network*

$$P(\Phi^1, \dots, \Phi^j) \coloneqq \left[\left(\begin{pmatrix} W_1^1 & & \\ & \ddots & \\ & & W_1^j \end{pmatrix}, \begin{pmatrix} b_1^1 \\ \vdots \\ b_1^j \end{pmatrix}\right), \dots, \left(\begin{pmatrix} W_L^1 & & \\ & \ddots & \\ & & W_L^j \end{pmatrix}, \begin{pmatrix} b_L^1 \\ \vdots \\ b_L^j \end{pmatrix}\right)\right]$$

realizes the parallelization of $\Phi^1, \dots, \Phi^j$ (without inter-network communication). The definition naturally extends to networks that do not have the same number of layers by bridging missing layers with identity networks, which will be the case in the following.

The next result follows directly from a repeated (recursive) application of [5, Lem. 5.3]. Note that the result is typically not sharp.

LEMMA 4.6 (sparse concatenation). *Let $\Phi^1, \dots, \Phi^j$ be j neural networks with* $\dim_{\mathrm{in}}(\Phi^i) = \dim_{\mathrm{out}}(\Phi^{i+1})$, $i = 1, \dots, j-1$. *Then the sparse concatenation $\Phi \coloneqq \Phi^1 \circ \cdots \circ \Phi^j$ as defined in Definition 4.4 satisfies*

(i) $\mathfrak{L}(\Phi) \leq \sum_{i=1}^{j} \mathfrak{L}(\Phi^i)$,
(ii) $\mathfrak{M}(\Phi) \leq 3 \sum_{i=1}^{j} \mathfrak{M}(\Phi^i)$.

We also have a result on the parallelization of networks, see [5, Lem. 5.4].

LEMMA 4.7 (parallelization). *Let $\Phi^1, \dots, \Phi^j$ be neural networks and define the network $\Phi := P(\Phi^1, \dots, \Phi^j)$ as their parallelization according to Definition 4.5. Then*
(i) $\mathfrak{L}(\Phi) \leq \max_{i=1,\dots,j} \mathfrak{L}(\Phi^i)$,
(ii) $\mathfrak{M}(\Phi) \leq 2\left(\sum_{i=1}^{j} \mathfrak{M}(\Phi^i)\right) + 4\left(\sum_{i=1}^{j} \dim_{\text{out}}(\Phi^i)\right) \max_{i=1,\dots,j} \mathfrak{L}(\Phi^i)$.

The following lemma follows directly from [13, Prop. 3.7] and is adjusted for our purposes as the special case of a matrix-matrix product.

PROPOSITION 4.8 (Scalar product). *Let $\epsilon \in (0,1)$ and $k \in \mathbb{N}$. There exists a neural network $\Phi^{\text{sca},k}$ such that for $z \geq 1$*

$$\sup_{x,y \in \mathbb{R}^k, \|x\|_2 \leq 1, \|y\|_2 \leq z} |y^\top x - \Phi^{\text{sca},k}(y,x)| \leq \epsilon.$$

Further, there exists a generic constant $C_{\text{sca}} > 0$ independent of k, y, x, ϵ, and z such that
(i) $\mathfrak{L}(\Phi^{\text{sca},k}) \leq C_{\text{sca}}\Big(\log_2(\frac{1}{\epsilon}) + \log_2(k) + \log_2(z)\Big)$,
(ii) $\mathfrak{M}(\Phi^{\text{sca},k}) \leq C_{\text{sca}}\Big(\log_2(\frac{1}{\epsilon}) + \log_2(k) + \log_2(z)\Big)k$.

With Proposition 4.8, we may deduce the generalization to matrix-vector multiplication for sparse matrices, which is the essential ingredient for approximating the maps R and S_α.

THEOREM 4.9 (sparse matrix-vector multiplication). *Let $\epsilon \in (0,1)$. Then there exists a neural network Φ^{mult} with $\dim_{\text{in}}(\Phi^{\text{mult}}) = \eta + n$ and $\dim_{\text{out}}(\Phi^{\text{mult}}) = n$ such that for $z \geq 1$*

$$\sup_{A \in \mathcal{S}^\eta \cap \mathcal{K}^1,\, r \in k^z} \|Ar - \Phi^{\text{mult}}(A,r)\|_2 \leq \epsilon.$$

Further, there exists a generic constant $C_{\text{mult}} > 0$ independent of n, A, x, ϵ, and z such that
(i) $\mathfrak{L}\Big(\Phi^{\text{mult}}\Big) \leq C_{\text{mult}}\Big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big)$,
(ii) $\mathfrak{M}\Big(\Phi^{\text{mult}}\Big) \leq C_{\text{mult}}\Big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big)\eta$.

Proof. Recall that the exact sparsity pattern of A is given. For the ith row, we first restrict r to the relevant entries with a one-layer network with $\mathcal{O}(\eta_i)$ non-zero entries (a classical restriction matrix). Then, we approximate the sparse scalar product of the η_i entries of $e_i^\top A$, where e_i denotes the i-th unit vector, with the corresponding restricted vector of r-values, which altogether realizes $(Ar)_i$. With Proposition 4.8, this is possible up to accuracy $\epsilon/\sqrt{n}$ with a network Φ^{sca,η_i} fulfilling

$$\mathfrak{L}\Big(\Phi^{\text{sca},\eta_i}\Big) \leq C_{\text{sca}}\Big(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big),$$
$$\mathfrak{M}\Big(\Phi^{\text{sca},\eta_i}\Big) \leq C_{\text{sca}}\Big(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(z)\Big)\eta_i.$$

Note that we have used that $\eta_i \leq n$.

Computing the $(Ar)_i$ in parallel, we obtain with Lemma 4.7 for the final network Φ^{mult} the desired bounds, using that $\eta = \sum_{i=1}^n \eta_i$ and

$$\|Ar - \Phi^{\text{mult}}(A, r)\|_2^2 = \sum_{i=1}^n |(Ar)_i - \Phi^{\text{sca},\eta_i}(e_i^\top A, r)|^2 \leq n\epsilon^2/n = \epsilon^2.$$

Note that we slightly misuse the notation by implicitly treating $e_i^\top A$ and r as vectors of length η_i. ∎

The next result is again important for the construction of both R and S_α.

LEMMA 4.10. *Let $\alpha \in \mathbb{R}$ and $n \in \mathbb{N}$. Then, there exists a neural network Φ^{scale}, that realizes the map*

$$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n, \quad (x, y) \mapsto \alpha x + y.$$

Further, we have

(i) $\mathfrak{L}\left(\Phi^{\text{scale}}\right) \leq 2$,

(ii) $\mathfrak{M}\left(\Phi^{\text{scale}}\right) \leq 8n$.

Proof. The result follows directly from using Lemma 4.3 with $L = 2$ for x (scaled by the factor α) and y in parallel, and a summation in the last layer. ∎

Finally, we conclude this section by assembling all the ingredients for the proofs of Theorem 4.1 and Theorem 4.2.

Proof of Theorem 4.1. For the construction of the network to approximate the operator R, we parallelize the networks that

- realize the identity with input A based on Lemma 4.3,
- approximate Ar with accuracy ϵ according to Theorem 4.9,
- calculate $r + c$ based on Lemma 4.10.

Clearly, the approximation of Ar requires the deepest network. Therefore, we directly get with Lemma 4.7 and a slight adjustments of the constants in Theorem 4.9

$$\mathfrak{L}\left(\Phi^R\right) \leq C_R\left(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(z)\right)$$

and

$$\mathfrak{M}\left(\Phi^R\right) \leq C_R\left(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(z)\right)\eta.$$

Note that we have used that $n \leq \eta$. Therefore, the effort for computing the approximation of Ar dominates the overall complexity. This concludes the proof. ∎

Proof of Theorem 4.2. The construction is very similar to the one in the proof of Theorem 4.1. We parallelize the networks that

- realize the identity with input A using Lemma 4.3,
- approximate Ar with accuracy ϵ according to Theorem 4.9 followed by applications of Lemma 4.10.
- realize the identity with input r using once more Lemma 4.3,

Again, the dominant contribution comes from the approximation of Ar and with Lemma 4.7 the bounds from Theorem 4.1 are resembled (up to constants). ∎

5. Richardson method - proof of Theorem 3.1. In this section, we provide the construction of the network stated in Theorem 3.1. In the first step, we determine the necessary number of iterations m in (3.2) to achieve the required accuracy.

LEMMA 5.1. *Let $\epsilon > 0$, $c_{\mathrm{sc}} \geq 1$, and let ω be given as in (3.2). Then, there exists an integer $m \geq 1$ such that*

$$\sup_{A \in \mathcal{S}^{\eta} \cap \Sigma^{\lambda,\Lambda},\, r \in k^{c_{\mathrm{sc}}\lambda}} \left\| A^{-1} r - \sum_{\ell=0}^{m} (I - \omega A)^{\ell} (\omega r) \right\|_2 \leq \frac{\epsilon}{2},$$

where $m = m(\epsilon, c_{\mathrm{sc}}, \rho_1)$ satisfies

$$m \geq \frac{|\log_2(\frac{\epsilon}{2c_{\mathrm{sc}}})|}{|\log_2(\rho_1)|}$$

with ρ_1 defined in (2.2).

Proof. We use the same technique as for the error bound of the Richardson method, see for example [21, Sec. 4.2.1], and recall the approximation sequence from (3.2) with $x^0 = 0$. Using that $x = A^{-1}r = (\omega A)^{-1}\omega r$ and defining the error $e^m = x - x^m$, we obtain the recursion

$$e^m = (I - \omega A) e^{m-1} = (I - \omega A)^m x.$$

Since $\|I - \omega A\|_2 \leq \rho_1$, we conclude

$$\|e^m\|_2 \leq \rho_1^m \|x\|_2 \leq \rho_1^m \|A^{-1}\|_2 \|r\|_2 \leq \rho_1^m \lambda^{-1} c_{\mathrm{sc}} \lambda = \rho_1^m c_{\mathrm{sc}}.$$

We can then guarantee the claimed bound if

$$\rho_1^m c_{\mathrm{sc}} \leq \frac{\epsilon}{2} \qquad \text{or, equivalently,} \qquad (\rho_1)^{-m} \geq \frac{2c_{\mathrm{sc}}}{\epsilon}$$

as stated. ∎

Finally, we provide the construction of the neural network which approximates the polynomial in (3.2).

Proof of Theorem 3.1. The main idea of the proof is to approximate the polynomial construction in Lemma 5.1 with the optimal $m = m_{\mathrm{Ric}}$, followed by a triangle inequality.

In the first step, we construct a network to perform the rescaling $\widehat{A} = I - \omega A$ and $\widehat{r} = \omega r$, using Lemma 4.10. The complexity for this operation scales linearly and is therefore negligible for the overall complexity bounds. The same statement holds true for the final rescaling in the very end.

We focus on the main scaling, which is introduced via the realization of the polynomial in Lemma 5.1. We express the desired polynomial via the map of Theorem 4.1, and note that

$$R^k(X^0) = R \circ \ldots \circ R(X^0) = \Big(\widehat{A}, \widehat{A}^m \widehat{r}, \sum_{\ell=0}^{k-1} \widehat{A}^{\ell} \widehat{r}\Big), \quad X^0 = (\widehat{A}, \widehat{r}, 0), \tag{5.1}$$

where we also use $\circ$ here for the standard concatenation of operators. We approximate (5.1) with $k-1$ sparse concatenations of Φ^R with itself and abbreviate $[\Phi^R]^k =$

$\Phi^R \circ \ldots \circ \Phi^R$. Note that Φ^R is constructed with accuracy $\delta = \delta(\epsilon) = \epsilon/(2m^2) > 0$ for input variables of size $z \leq m+2$.

(a) We now show by induction that for $i = 0, \ldots, m$

$$\|R^i(X^0) - [\Phi^R]^i(X^0)\|_2 \leq \tfrac{\epsilon}{2}, \quad \|[\Phi^R]^i(X^0)\|_2 \leq i+3, \tag{5.2}$$

and note that the induction is trivially anchored at $i = 0$, if we define in this case both operators as the identity.

Now let $1 \leq i \leq m$, assume that (5.2) holds for $0, \ldots, i-1$, and compute

$$\begin{aligned}
&\|R^i(X^0) - [\Phi^R]^i(X^0)\|_2 \\
&\qquad \leq \sum_{\ell=0}^{i-1} \|R^{i-\ell}([\Phi^R]^\ell(X^0)) - R^{i-\ell-1}([\Phi^R]^{\ell+1}(X^0))\|_2 \\
&\qquad = \sum_{\ell=0}^{i-1} \|R^{i-\ell-1}(R([\Phi^R]^\ell(X^0))) - R^{i-\ell-1}(\Phi^R([\Phi^R]^\ell(X^0)))\|_2 \\
&\qquad = \sum_{\ell=0}^{i-1} \|R^{i-\ell-1}(R(X^\ell)) - R^{i-\ell-1}(\Phi^R(X^\ell))\|_2
\end{aligned}$$

for $X^\ell = [\Phi^R]^\ell(X^0)$. Further, note that the first and last argument of $R(X^\ell)$ and $\Phi^R(X^\ell)$ are identical, and we denote the second arguments by b and $\widetilde{b}$, respectively. We now have to consider the stability of R^k for such arguments. We first compute

$$\|R^k(\widehat{A}, b, c) - R^k(\widehat{A}, \widetilde{b}, c)\|_2 = \Big\|(0, \widehat{A}^k(b-\widetilde{b}), \sum_{j=0}^{k-1} \widehat{A}^j(b-\widetilde{b})\Big\|_2 \leq (k+1)\|b-\widetilde{b}\|_2,$$

using that $\|\widehat{A}\| \leq 1$. Since $\ell \leq i-1$, we can apply (5.2) to obtain the bounds on the argument X^ℓ, and we can use the approximation property in Theorem 4.1 to obtain $\|b-\widetilde{b}\|_2 \leq \delta$. Plugging this into the sum and using $i-1 \leq m$, we estimate

$$\|R^i(X^0) - [\Phi^R]^i(X^0)\|_2 \leq m^2\delta = \tfrac{\epsilon}{2}.$$

For the second part of the induction, we write

$$\|[\Phi^R]^i(X^0)\|_2 \leq \|R^i(X^0)\|_2 + \|R^i(X^0) - [\Phi^R]^i(X^0)\|_2 \leq i + 2 + \tfrac{\epsilon}{2} \leq i+3,$$

using $\|\widehat{r}\| \leq 1$ and the stability of R^i based on (5.1). This closes the induction.

(b) For the size of the network, we first observe that with $m = m_{\mathrm{Ric}}$

$$\log_2(\tfrac{1}{\delta}) \sim \log_2(\tfrac{1}{\epsilon}) + \log_2(m_{\mathrm{Ric}})$$

and the estimates in Theorem 4.1 then gives

(i) $\mathfrak{L}\big(\Phi^{\mathrm{Ric}}\big) \leq C_{\mathrm{Ric}} m_{\mathrm{Ric}} \big(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{Ric}})\big)$,

(ii) $\mathfrak{M}\big(\Phi^{\mathrm{Ric}}\big) \leq C_{\mathrm{Ric}} m_{\mathrm{Ric}} \big(\log_2(\tfrac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{Ric}})\big)\eta$,

which is precisely the claimed network size.

Finally, we combine Lemma 5.1 with the above construction, which gives the result using the triangle inequality. ∎

6. Cg-type method - proof of Theorem 3.2. In this section, we prove our second main result in Theorem 3.2, for which we recall the decomposition in (3.3). The first estimate can be deduced from classical estimates, which we collect in the following lemma, see for example [21].

LEMMA 6.1. *Let $\epsilon > 0$. Then, there exists a polynomial $q_{m-1} \in \mathcal{P}_{m-1}$ such that*

$$\sup_{A \in \mathcal{S}^{\boldsymbol{\eta}} \cap \Sigma^{\lambda,\Lambda},\, r \in k^{c_{\mathrm{sc}}\lambda}} \left\| A^{-1} r - q_{m-1}(\tfrac{1}{\Lambda} A)\tfrac{1}{\Lambda} r \right\|_2 \leq \frac{\epsilon}{2}$$

where $m = m(\epsilon, c_{\mathrm{sc}}, \rho_{1/2})$ satisfies

$$m \geq \frac{|\log_2(\frac{\epsilon}{4c_{\mathrm{sc}}})|}{|\log_2(\rho_{1/2})|}$$

with $\rho_{1/2}$ defined in (2.2).

Proof. The idea is to start with the (matrix) polynomial which is used in order to prove the approximation property of the cg method. Let us denote $\widehat{A} = \frac{1}{\Lambda} A$ and $\widehat{r} = \frac{1}{\Lambda} r$. For the convergence proof of the cg method, one solves the minimization problem over all polynomial $p_m \in \mathcal{P}_m$ with $p_m(0) = 1$ such that the induced energy norm

$$\|p_m(\widehat{A})\widehat{r}\|_{\widehat{A}}$$

becomes small. Here, the optimal choice is the rescaled Chebyshev polynomial given as

$$p_m(z) = \frac{T_m(\sigma(z))}{T_m(\sigma(0))}, \quad \sigma(z) = \frac{\widehat{\kappa}+1}{\widehat{\kappa}-1} - \frac{2}{\Lambda - \lambda} z.$$

Since $p_m(0) = 1$, we can define the polynomial of degree $m-1$ by $q_{m-1}(z) = \frac{p_m(z)-1}{z}$. This gives for $x_m = q_{m-1}(\frac{1}{\Lambda} A)\frac{1}{\Lambda} r$ the estimate in the induced energy norm, i.e.,

$$\|x - x_m\|_{\widehat{A}} \leq 2\rho_{1/2}^m \|x\|_{\widehat{A}},$$

see [21, Sec. 6.11.3]. This allows us to conclude

$$\begin{aligned} \|x - x_m\|_2 &\leq \lambda_{\min}^{-1/2}(\widehat{A}) \|x - x_m\|_{\widehat{A}} \leq \lambda_{\min}^{-1/2}(\widehat{A}) 2\rho_{1/2}^m \|\widehat{A}^{-1/2}\widehat{A}x\|_2 \\ &\leq \lambda_{\min}^{-1}(\widehat{A}) 2\rho_{1/2}^m \|\widehat{r}\|_2 = \lambda_{\min}^{-1}(A) 2\rho_{1/2}^m \|r\|_2 \leq 2c_{\mathrm{sc}}\rho_{1/2}^m, \end{aligned}$$

and hence we can ensure $\|x - x_m\|_2 \leq \frac{\epsilon}{2}$ by the above choice of m. ∎

Before we eventually present the proof of Theorem 3.2, we need to recall some results on Chebyshev polynomials. Besides the ones of first kind, which we denoted by T_j, we further need the Chebyshev polynomials of second kind, denoted by U_j. They are defined via the same three-term recurrence, starting however with $U_0(x) = 1$ and $U_1(x) = 2x$. The crucial identity for our construction is given by

$$\frac{T_m(x) - T_m(x_0)}{x - x_0} = \sum_{\ell=0}^{m-1} \alpha_\ell(x_0) U_\ell(x), \tag{6.1}$$

with the coefficients

$$\alpha_\ell(x_0) = \begin{cases} T_{m-1-\ell}(x_0) & \text{for } \ell = 0 \text{ and } \ell = m-1, \\ 2T_{m-1-\ell}(x_0) & \text{for } 1 \leq \ell \leq m-2, \end{cases}$$

see [10, Thm. 5]. We can thus rewrite the polynomial q_{m-1} as

$$q_{m-1}(z) = \frac{T_m(\sigma(z)) - T_m(\sigma(0))}{T_m(\sigma(0))z} = -\frac{2}{(\Lambda - \lambda)T_m(\sigma(0))} \frac{T_m(\sigma(z)) - T_m(\sigma(0))}{\sigma(z) - \sigma(0)},$$

such that we can plug in (6.1) for $x = \sigma(z)$ and $x_0 = \sigma(0)$. Further, if we define

$$\alpha_{\max} = \max_{\ell=0,\dots,m-1} \alpha_\ell(\sigma(0)),$$

the main effort in the network can be traced back to the construction of an approximation of

$$\sum_{\ell=0}^{m-1} \frac{\alpha_\ell(\sigma(0))}{\alpha_{\max}} U_\ell(\sigma(\widehat{A}))\widehat{r}, \tag{6.2}$$

which is a Chebyshev series that can be evaluated by the Clenshaw algorithm [3] with m matrix-vector multiplications. Indeed, for $p(x) = \sum_{\ell=0}^{m-1} \alpha_\ell U_\ell(x)$, the iteration reads

$$b_k = \alpha_k + 2xb_{k+1} - b_{k+2}, \quad k = m-1, \dots, 0, \quad \text{with} \quad b_m = b_{m+1} = 0,$$

and the result is given by $p(x) = b_0$. This is realized by the concatenation of the maps S_{α_k} defined in (4.2), and approximated by the neural networks constructed in Theorem 4.2.

Proof of Theorem 3.2. The proof is similar to the one of Theorem 3.1. We approximate the construction in Lemma 6.1 for $m = m_{\text{cg}}$ by a neural network and then apply the triangle inequality. As before, we first need to perform the rescaling of $\widehat{A} = \frac{1}{\Lambda}A$ and $\widehat{r} = \frac{1}{\Lambda}r$. Further, we need to compute $\sigma(\widehat{A})$. Again, this is done in linear complexity and can thus be neglected. After computing (6.2), we need also need to rescale in linear complexity.

Thus, we focus on (6.2). Note that the desired polynomial can be expressed via

$$S_0 \circ \dots \circ S_{m-1}(X^0), \quad X^0 = (\widehat{A}, 0, 0), \tag{6.3}$$

where $S_k := S_{\alpha_k}$, cf. (4.2). Further, we set $S_{i,j} = S_i \circ \dots \circ S_j$ for $j > i$, $S_{i,j} = S_j$ for $i = j$, and $S_{i,j} = \text{id}$ the identity map for $j < i$. We approximate (6.3) with the sparse concatenation $\Phi^S_{i,j} = \Phi^S_i \circ \dots \circ \Phi^S_j$, where each Φ^S_k approximates S_k as constructed in Theorem 4.2 with accuracy $\delta = \delta(\epsilon) = \epsilon/(2(m+1)^2) > 0$ for input variables of size $z \leq 3m^2$. Note that we use same convention on i, j for $\Phi^S_{i,j}$ as for $S_{i,j}$.

(a) We now show by induction that for $i = m, \dots, 0$

$$\|S_{i,m-1}(X^0) - \Phi^S_{i,m-1}(X^0)\|_2 \leq \tfrac{\epsilon}{2}, \quad \|\Phi^S_{i,m-1}(X^0)\|_2 \leq 3m^2, \tag{6.4}$$

and note that the induction is trivially anchored at $i = m$.

Now let $0 \leq i \leq m-1$, assume that (6.4) holds for $m, \dots, i+1$, and compute

$$\begin{aligned}
&\|S_{i,m-1}(X^0) - \Phi^S_{i,m-1}(X^0)\|_2 \\
&\qquad \leq \sum_{\ell=i}^{m-1} \|S_{i,\ell-1}(\Phi^S_{\ell,m-1}(X^0)) - S_{i,\ell}(\Phi^S_{\ell+1,m-1}(X^0))\|_2 \\
&\qquad = \sum_{\ell=i}^{m-1} \|S_{i,\ell-1}(\Phi_\ell(\Phi^S_{\ell+1,m-1}(X^0))) - S_{i,\ell-1}(S_\ell(\Phi^S_{\ell+1,m-1}(X^0)))\|_2 \\
&\qquad = \sum_{\ell=i}^{m-1} \|S_{i,\ell-1}(\Phi^S_\ell(X^{m-1-\ell})) - S_{i,\ell-1}(S_\ell(X^{m-1-\ell}))\|_2
\end{aligned}$$

for $X^{m-1-\ell} = \Phi^S_{\ell+1,m-1}(X^0)$. We emphasize that the first and last argument of $S_\ell(X^{m-1-\ell})$ and $\Phi^S_\ell(X^{m-1-\ell})$ are identical, and as above we denote the second arguments by b and $\widetilde{b}$, respectively. Thus, we have to consider the stability of $S_{i,\ell-1}$ for such arguments. We first compute for $B = \sigma(\widehat{A})$

$$\begin{aligned}
\|S_{i,\ell-1}(B,b,c) - S_{i,\ell-1}(B,\widetilde{b},c)\|_2 &= \|(0, U_{\ell-i}(B)(b-\widetilde{b}), U_{\ell-i-1}(B)(b-\widetilde{b}))\|_2 \\
&\leq (2\ell+1)\|b-\widetilde{b}\|_2,
\end{aligned}$$

with two applications of the estimate $|U_j(x)| \leq j+1$ for $x \in [-1,1]$. Since $i+1 \leq \ell+1 \leq m$, we can apply (6.4) to obtain the bounds on all the inputs $X^{m-1-\ell}$. Further, we thus can use the approximation property in Theorem 4.2 to obtain $\|b-\widetilde{b}\|_2 \leq \delta$ and conclude by the summation over ℓ that

$$\|S_{i,m-1}(X^0) - \Phi^S_{i,m-1}(X^0)\|_2 \leq (m+1)^2\delta = \tfrac{\epsilon}{2}.$$

For the second part of the induction, we write

$$\begin{aligned}
\|\Phi^S_{i,m-1}(X^0)\|_2 &\leq \|S_{i,m-1}(X^0)\|_2 + \|S_{i,m-1}(X^0) - \Phi^S_{i,m-1}(X^0)\|_2 \\
&\leq m^2 + 1 + \tfrac{\epsilon}{2} \leq 3m^2,
\end{aligned}$$

where we employed the estimate

$$\|S_{i,m-1}(\widehat{A},0,0)\|_2 \leq m^2 + 1.$$

The latter inequality follows from

$$\|S_{\beta_k} \circ \ldots \circ S_{\beta_1}(A,0,0)\|_2 = \Big\| \Big(A, \sum_{\ell=1}^{k} U_{k-\ell}(A)\beta_\ell, \sum_{\ell=1}^{k-1} U_{k-\ell-1}(A)\beta_\ell\Big) \Big\|_2 \leq k^2 + 1,$$

for arbitrary coefficients $|\beta_j| \leq 1$, using once again the estimates on U_j. Since we can always choose $k \leq m$, the induction is closed.

(b) For the size of the network, we first observe that with $m = m_{\mathrm{cg}}$

$$\log_2(\tfrac{1}{\delta}) \sim \log_2(\tfrac{1}{\epsilon}) + \log_2(m_{\mathrm{cg}})$$

and the estimates in Theorem 4.2 lead to

(i) $\mathfrak{L}\big(\Phi^{\mathrm{cg}}\big) \leq C_{\mathrm{cg}} m_{\mathrm{cg}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{cg}})\big)$,

(ii) $\mathfrak{M}\big(\Phi^{\mathrm{cg}}\big) \leq C_{\mathrm{cg}} m_{\mathrm{cg}} \big(\log_2(\frac{1}{\epsilon}) + \log_2(n) + \log_2(m_{\mathrm{cg}})\big)\eta$,

which is precisely the claimed network size.

Finally, we combine Lemma 6.1 with the above construction, which gives the result using the triangle inequality. ∎

7. Possible extensions. In this last section, we discuss three possible extensions of the presented approach. We only provide rough ideas and leave a rigorous treatment for future research.

Pre-processing. An important assumption of our constructions is that matrices are given with a pre-scribed (maximal) sparsity pattern. As the networks can only process inputs of a fixed size, matrices A with even fewer entries (resulting in a shorter vector A^{v} containing the non-zero values) need to be prolonged to fit the maximal sparsity pattern. This requires a classical prolongation matrix, which can be set up with linear complexity. Such a pre-processing step could as well be realized by a neural network, which, however, then depends on the exact positions of the non-zero entries.

Pre-conditioning. The dependence on condition numbers in classical Richardson or cg iterations directly transfers over to our results. In particular, the number of required iterations (also in the network construction) suffers from large condition numbers, see also the discussion in Section 3.2 in the context of finite element-based methods. A natural procedure to avoid the problematic scaling are pre-conditioning techniques. To include pre-conditioning into our construction, the necessary operations to set up appropriate well-conditioned systems have to be approximated by neural networks as well. In case that the technique provably lowers the condition number for the classical algorithm, such results would also transfer to corresponding neural network approximations. However, a precise study is beyond the scope of this article.

Variable sparsity patterns. One of the drawbacks of our construction is the necessary a-priori knowledge on the sparsity pattern of the class of matrices. In principle, it is also possible to only keep the number of non-zero entries fixed and derive information about the precise sparsity pattern of a matrix A directly from the input. This requires besides A^{v} also information about the matrix entries (as usually given for, e.g., COO-type formats with vectors A^{r} and A^{c} containing the row and column indices of the non-zero entries, respectively). In order to realize the mapping $(A, r) \mapsto Ar$ that is relevant for all the presented iterative procedures, for each row of A we need a mapping that reduces the entries of r to the corresponding non-zero entries in the corresponding row of A. Realizing such a mapping for all rows and all possible sparsity patterns, however, exceeds the derived complexity bounds. Therefore, we refrain from discussing possible constructions in more detail.

Acknowledgments. Preliminary results for this work have been developed in the Master thesis of L. Ullmer at Karlsruhe Institute of Technology. Finally, we thank Matthias Deiml for fruitful discussions.

References.

[1] R. Altmann, P. Henning, and D. Peterseim, *Numerical homogenization beyond scale separation*, Acta Numer., 30 (2021), pp. 1–86, https://doi.org/10.1017/S0962492921000015.

[2] A. R. Barron, *Universal approximation bounds for superpositions of a sigmoidal function*, IEEE Trans. Inform. Theory, 39 (1993), pp. 930–945, https://doi.org/10.1109/18.256500.

[3] C. W. Clenshaw, *A note on the summation of Chebyshev series*, Math. Tables Aids Comput., 9 (1955), pp. 118–120.

[4] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Systems, 2 (1989), pp. 303–314, https://doi.org/10.1007/BF02551274.

[5] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab, *DNN expression rate analysis of high-dimensional PDEs: application to option pricing*, Constr. Approx., 55 (2022), pp. 3–71, https://doi.org/10.1007/s00365-021-09541-6.

[6] M. Geist, P. Petersen, M. Raslan, R. Schneider, and G. Kutyniok, *Numerical solution of the parametric diffusion equation by deep neural networks*, J. Sci. Comput., 88 (2021), pp. Paper No. 22, 37, https://doi.org/10.1007/s10915-021-01532-w.

[7] I. Gühring, G. Kutyniok, and P. Petersen, *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms*, Anal. Appl. (Singap.), 18 (2020), pp. 803–859, https://doi.org/10.1142/S0219530519410021.

[8] I. Gühring, M. Raslan, and G. Kutyniok, *Expressivity of deep neural networks*, in Mathematical aspects of deep learning, Cambridge Univ. Press, Cambridge, 2023, pp. 149–199.

[9] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Netw., 2 (1989), pp. 359–366, https://doi.org/10.1016/0893-6080(89)90020-8.

[10] S. H. Kim and J. H. Lee, *On difference quotients of Chebyshev polynomials*, Bull. Korean Math. Soc., 53 (2016), pp. 373–386, https://doi.org/10.4134/BKMS.2016.53.2.373.

[11] F. Köpfl, R. Maier, and D. Peterseim, *Operator compression with deep neural networks*, Adv. Contin. Discrete Models, Paper No. 29, 23 (2022), https://doi.org/10.1186/s13662-022-03702-y.

[12] F. Köpfl, R. Maier, and D. Peterseim, *Neural network approximation of coarse-scale surrogates in numerical homogenization*, Multiscale Model. Simul., 21 (2023), pp. 1457–1485, https://doi.org/10.1137/22M1524278.

[13] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider, *A theoretical analysis of deep neural networks and parametric PDEs*, Constr. Approx., 55 (2022), pp. 73–125, https://doi.org/10.1007/s00365-021-09551-4.

[14] A. Målqvist and D. Peterseim, *Numerical homogenization by localized orthogonal decomposition*, vol. 5 of SIAM Spotlights, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2020.

[15] C. Marcati and C. Schwab, *Exponential convergence of deep operator networks for elliptic partial differential equations*, SIAM J. Numer. Anal., 61 (2023), pp. 1513–1545, https://doi.org/10.1137/21M1465718.

[16] J. A. A. Opschoor, P. C. Petersen, and C. Schwab, *Deep ReLU networks and high-order finite element methods*, Anal. Appl., 18 (2020), pp. 715–770, https://doi.org/10.1142/S0219530519410136.

[17] J. A. A. Opschoor and C. Schwab, *Deep ReLU networks and high-order finite element methods II: Chebyšev emulation*, Comput. Math. Appl., 169 (2024), pp. 142–162, https://doi.org/10.1016/j.camwa.2024.06.008.

[18] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Netw., 108 (2018), pp. 296–330, https://doi.org/10.1016/j.neunet.2018.08.019.

[19] L. F. Richardson, *IX. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam*, Philos. Trans. R. Soc. Lond. A., 210 (1911), pp. 307–357, https://doi.org/10.1098/rsta.1911.0009.

[20] G. Romera and J. A. Bárcena-Petisco, *A theoretical analysis on the inversion of matrices via neural networks designed with Strassen algorithm*, ArXiv Preprint, 2501.06539 (2026).
[21] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003, https://doi.org/10.1137/1.9780898718003.
[22] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math., 13 (1969), pp. 354–356, https://doi.org/10.1007/BF02165411.
[23] D. Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Netw., 94 (2017), pp. 103–114, https://doi.org/10.1016/j.neunet.2017.07.002.