Preconditioning parametrized linear systems: hierarchical maps approach

Eric de Sturler and Michal Outrata

Abstract

We propose, analyze and numerically experiment with a new type of preconditioner maps for sequences of systems of linear algebraic equations. Similarly to [7], we propose to renovate a good-quality preconditioner for the first problem $A_1\mathbf{x}_1 = \mathbf{b}_1$ so that we obtain a comparably good preconditioner also for the follow-up problems $A_k\mathbf{x}_k = \mathbf{b}_k$ ($k \geq 2$) in the problem sequence. Complementary to [7] we use *data-sparsity* techniques, namely hierarchical matrix formats HODLR and HSS, and propose two different preconditioner maps. We demonstrate the quality as well as efficiency of the proposed preconditioner maps and try to further improve on these numerically on a model problem.

1 Introduction

Consider a sequence of linear problems

$$A_k\mathbf{x}_k = \mathbf{b}_k, \qquad \text{for } k = 1, 2, \ldots, \tag{1}$$

where $A_k \in \mathbb{R}^{N\times N}, \mathbf{b}_k \in \mathbb{R}^N$ and denote the update matrices by $E_{k,k+\ell}$, i.e.,

$$A_k = A_{k-\ell} + E_{k-\ell,k}, \qquad \text{for } \ell = 1, \ldots, k-1.$$

In many applications the *efficient* solution of each of these problems requires the use of iterative method such as Krylov methods coupled with *preconditioners*. Obtaining these can be computationally very demanding but overall still improves the running time significantly (see, e.g., [5] and the references therein). Assuming we have computed a high-quality preconditioner P_1 for the first system so that the GMRES method applied to

$$A_1P_1\mathbf{y}_1 = \mathbf{b}_1, \tag{2}$$

converges rapidly, we note that even for relatively small consecutive updates $E_{k-1,k}$, the GMRES convergence with fixed preconditioner P_1 can deteriorate drastically, see an illustrative example in Figure 2 in Section 4. This means that the preconditioner P_1 needs an update as well and in [7, Section 2], the authors proposed[1] a preconditioning map $\mathcal{P}$,

$$\mathcal{P} \;:\; (A_1, A_k) \mapsto P_k \approx A_k^{-1}A_1, \tag{3}$$

so that

$$A_kP_kP_1 \approx A_1P_1.$$

[1] A similar idea but for a more particular setting was already proposed in [1].

In words, we shift the focus from construction of a stand-alone preconditioner for the k-th system to construction of P_k such that the $A_k P_k$ system "resembles" the original one for which we already have a high-quality preconditioner. Alternatively, we can also think of this map as *renovating* the preconditioner P_1 into the precondtioner $P_k P_1$ with an important distinction that the product $P_k P_1$ is never assembled but rather kept as two separate mat-vec routines in the preconditioned GMRES algorithm – this is preferable both because in some situations P_1 is available itself only as a mat-vec routine and because of the computational burden of performing matrix-matrix product for large systems, which is often prohibitive. Moving forward we continue considering a right-preconditioner but the same ideas apply to the case of a left-preconditioner. The case of split preconditioning is more interesting and will be treated separately elsewhere.

In [7], the authors study a particular type of $\mathcal{P}$ in (3), using the sparse approximate inverse technique to obtain an efficient approximation and call the resulting $\mathcal{P}$ the *sparse approximate map* (SAM). The key point then lies in balancing the trade-off between the *(structural) sparsity pattern* of P_k, which controls the computational complexity of its application, and the error introduced by the *approximation*, which controls the quality of the map. In [7, Section 3], the authors experimentally show that even for modest sparsity patterns (there the authors considered the patterns of powers of the system matrix) and approximation error in (3), the SAM approach can be very effective. Naturally, the area of updating preconditioners for systems of problems (possibly with only right-hand sides changing or only system matrices changing) is much richer and we refer the interested reader to the literature cited in [7] but also to [24] and the references therein.

Our approach is complementary to that in [7] – we want to use *data-sparsity* instead of the *structural sparsity* to construct a suitable preconditioner map, starting with hierarchical matrix formats such as HODLR (a non-nested format) and HSS (a nested format) but with emphasis on the generality, so that other hierarchical formats can be easily put in the place of HODLR/HSS. As a result we use a different map than the one in [7]. We also analyze the GMRES convergence behavior for the preconditioned systems using our preconditioner map, doing so in a complementary way to [7], i.e., using the pseudospectra bounds, in contrast to the spectral bound in [7]. The analysis is not directly transferable but the approach is, i.e., our analysis approach can be applied to obtain analogous bounds also in [7] and, reversely, the spectral bound in [7] could be adapted to our preconditioner map as well.

The rest of the paper is structured as follows: we introduce the hierarchical approximate maps (HAMs) as well as other necessary terms in Section 2 and analyze the GMRES convergence behavior of the resulting preconditioners in Section 3. We explore this new approach numerically for a model problem in Section 4 – we start by looking at the proposed bounds in Section 4.1 and continue with exploring some basic ideas for making the preconditioner more efficient in Section 4.2. There we explore several different directions for improving the efficiency and, finally, in summarize the contributions in Section 5.

2 HAM: hierarchical approximate maps

We start by looking at the *optimal* preconditioner map for the second system – obtained by assuming equality in (3) instead of approximation – and obtain P_2^{opt} as

$$P_2^{\text{opt}} = A_2^{-1} A_1 = \left(A_1^{-1} A_2\right)^{-1} = \left(A_1^{-1}(A_1 + E_{1,2}\right)^{-1} = \left(I + A_1^{-1} E_{1,2}\right)^{-1}.$$

Replacing the solve with the matrix A_1 with the application of the preconditioner P_1 we obtain an *approximate* preconditioner map, which requires a solve with the matrix

$$\tilde{R}_2 := I + P_1 E_{1,2}.$$

To make this solve efficient we approximate $\tilde{R}_2$ in a *hierarchical matrix format* – here we will consider two examples of such formats – HODLR and HSS, see [15] and [28] for an introduction to these – but we avoid, on purpose, using any particular properties of these so that any hierarchical format can be easily substituted for these. Generally speaking, the main appeal of using hierarchical formats lies in the available fast hierarchical solvers with linear[2] ($\mathcal{O}(N)$) or almost linear ($\mathcal{O}(N \log^{\alpha}(N))$ for some $\alpha \gtrapprox 1$) complexity with respect to the system size N. Hence, using a hierarchical format we obtain an approximation to solving with $\tilde{R}_2$ – we denote these two operations by $\mathcal{H}()$ for the approximation in a hierarchical format and $\mathcal{H}_{solve}()$ for the hierarchical solve and get

$$P_2 = \mathcal{H}_{solve}(R_2), \quad \text{with} \quad R_2 := \mathcal{H}(I + P_1 E_{1,2}). \tag{4}$$

The preconditioner for A_2 then becomes $P_2 P_1$ – a product which is not assembled but rather applied piece by piece. The area of hierarchical matrices (and tensors) has yielded many very efficient approaches and algorithms in multiple different applications (see, e.g., [4, 14, 11, 3, 16, 2, 20] and also [13, 18, 29] among others) and has been a very active field of research for the last two decades. The efficiency is achieved via a multilevel scheme in which we approximate certain off-diagonal parts of the matrix in question on each level. Before generalizing to the k-th system, we recall that before computing a hierarchical approximation we have to make quite a few choices – the format (HODLR, HSS or some more involved formats, e.g., $\mathcal{H}$ and $\mathcal{H}^2$ matrices, see [16, Sections 6–8]), the structure (usually given by the so-called cluster trees – row and column – and admissibility condition), the accuracy (denoted by ϵ) or the hierarchical rank (denoted by r or by $\mathcal{H}$-rank()) and the minimal block size (denoted by β) to name the most common ones. Making a choice for all of these, we then obtain a hierarchical matrix of a particular *type*. As the set of all matrices of a particular type does not necessarily form a vector space with the standard algebraic operations, it is common to use the so-called *formatted* (or *hierarchical*) version of the algebraic operations, such as addition, multiplication or inversion. In the simplest form these correspond to performing the desired operation algebraically as we would usually do for two matrices and then calculating the best hierarchical approximation of the given type of the outcome of that operation, i.e., projecting back onto the set of all matrices of the given type. Unfortunately, this would destroy the efficiency for many operations and so more sophisticated algorithms have been proposed to calculate these formatted/hierarchical operations; see [16, Section 3 and onwards]. We denote these formatted operation using the o-notation, e.g., $\oplus$ or $\otimes$, but shall not go into any more details; we use the `hm-toolbox` implementation in MATLAB of the HODLR and HSS formats, which is freely available and is described in [21]. We elaborate on the effects of some of the above parameters later in Section 4.

For the k-th system in (1), the preconditioner becomes $P_k P_1$ with $P_k = \mathcal{H}_{solve}(R_k)$ and we see two possible ways of defining R_k as a generalization of (4), namely

$$R_k^{(1)} := \mathcal{H}(I + P_1 E_{1,k}) \quad \text{or} \quad R_k^{(2)} := R_{k-1} \oplus \mathcal{H}(P_1 E_{k-1,k}). \tag{5}$$

[2] As usual, there is a "catch" to this statement and we comment on this in more detail throughout the manuscript.

Once the application of P_1 onto the update matrix is evaluated the rest of the operations is again done using the formatted arithmetic. Formally, we introduce two *hierarchical approximate maps* (HAMs)

$$\begin{aligned} \mathcal{P}^{(1)} &: (P_1, E_{1,k}) \mapsto P_k^{(1)} := \mathcal{H}_{solve}\left(R_k^{(1)}\right), \\ \mathcal{P}^{(2)} &: (P_1, R_{k-1}, E_{k-1,k}) \mapsto P_k^{(2)} := \mathcal{H}_{solve}\left(R_k^{(2)}\right). \end{aligned} \tag{6}$$

First, we note that $\mathcal{P}^{(1)}$ is the natural analogue of the SAM approach in [7] but observe a key difference – evaluating the SAM map (i.e., obtaining the matrix R_k; in [7] denoted by N_k) does not involve P_1 at all.

Second, let us highlight that the construction of $R_k^{(1,2)}$ requires[3] (among other steps) the matrix P_1E, with $E = E_{1,k}$ or $E = E_{k-1,k}$. If the update matrices E would each have *many* non-zero columns, then the evaluation of P_1E requires many applications of P_1 and can become the bottleneck of the entire solution process. In some applications we expect our system matrices A_k to form a "cluster"[4] around A_1 so that both $E_{1,k}$ and $E_{k-1,k}$ contain roughly the same number of non-zero columns but for other applications the matrices A_k form a "string"[5] stretching away from A_1, where at each step the number of non-zero columns of $E_{k-1,k}$ is manageable but already after few steps the matrix $E_{1,k}$ accumulates large number of non-zero columns.

This highlights an important trade-off between the maps $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ – the map $\mathcal{P}^{(1)}$ accumulates the information in the update matrix in contrast to the map $\mathcal{P}^{(2)}$ that keeps some information from the previous steps in form of R_{k-1}. Moreover, $\mathcal{P}^{(1)}$ allows us to *change* any parameters of the hierarchical type for *each* system A_k. This is not the case for $\mathcal{P}^{(2)}$ as the formatted sum can be efficiently evaluated only if both matrices are of the same hierarchical type or there is an efficient conversion in between these. However the conversions are often far from efficient, depending on the hierarchical structures and formats involved.

Last but not least, we note that for $\mathcal{P}^{(2)}$ it seems natural to keep P_k already assembled and update it *directly* in contrast to storing and updating $R_k^{(2)}$ – let us denote this approach as $\mathcal{P}^{(3)}$. In context of approximate preconditioner maps this idea has been used in [1], where the update to R_k is rank one and P_k is updated directly using Sherman-Morrison-Woodbury formula. The main appeal is in reducing the computational complexities – for both HODLR and HSS formats (as well as for others) the cost of solving $H\mathbf{x} = \mathbf{b}$ with H in a hierarchical format of rank[6] r scales as r^2 while the cost of multiplying with the same H scales only as r, see (20) ahead. This approach is not possible with $\mathcal{P}^{(1)}$ – there is no update to speak of – but can be derived based on $\mathcal{P}^{(2)}$, e.g., if we update the hierarchical inverse or the hierarchical LU factorization in some way. The fair comparison – computational-complexity-wise – is then comparing $\mathcal{P}^{(3)}$ with using $\mathcal{P}^{(2)}$ to obtain

[3] We also note that many standard hierarchical formats can be built in a matrix-free fashion, i.e., without the need to assemble the matrix P_1E and only by having mat-vec routine $\mathbf{v} \mapsto P_1E\mathbf{v}$ (e.g., the HODLR format, see [21, Section 3.3]). However, the construction sometimes requires also the action of the transpose, i.e., $\mathbf{v} \mapsto (P_1E)^T\mathbf{v}$ (e.g., for the HSS format, see [21, Section 3.3]), which might not be always available, e.g., if P_1 is itself only available as a mat-vec routine $\mathbf{v} \mapsto P_1\mathbf{v}$. We shall not focus on this possibility here and will assume the necessity of the construction of the matrix. The matrix-free adaptation will be discussed elsewhere.

[4] In the sense that the number of nonzero columns in $E_{1,k}$ does not grow substantially with k.

[5] In the sense that the number of nonzero columns in $E_{1,k}$ does grow substantially faster than that of $E_{k-1,k}$ with k.

[6] As noted above, the hierarchical formats keep the of-diagonal portions of the matrix in a low-rank format. Intuitively speaking, the highest rank of these is what we call the hierarchical rank of the matrix. Precise definition can, however, differ from format to format and can be found in the literature cited before.

$R_k^{(2)}$ and then calculating the hierarchical approximations to the *inverses of the LU factors* – the so-called *hierarchical factored approximate inverse* method/preconditioner ($\mathcal{H}$-FAINV), see [18]. The specifics of the hierarchical arithmetic makes it so that these operations have the same asymptotics (with respect to the system size N and the hierarchical rank r) and the constants are comparable (see [18, Theorem 2 and Lemma 1 and below]). This means that the main appeal of the map $\mathcal{P}^{(3)}$ – the lower complexity of application of the preconditioner – can be achieved also by combining $\mathcal{P}^{(2)}$ with $\mathcal{H}$-FAINV and, moreover, the extra cost of $\mathcal{H}$-FAINV is comparable to one additional formatted matrix sum. All of that is, of course, under the assumption that both $\mathcal{P}^{(3)}$ and $\mathcal{P}^{(2)}$-with-$\mathcal{H}$-FAINV *achieve comparable accuracy with the same hierarchical rank*. We are not aware of any experimental results but to keep the present manuscript from growing too wide we will address this direction of updating P_k rather than R_k elsewhere. Next we turn our attention to analysis of the proposed preconditioner maps.

3 Analysis of the HAM approach

The preconditioner maps have two points where an approximation was introduced – replacing A_1^{-1} with P_1 and replacing the inverse/solve operation with its hierarchical approximation – and the analysis reflects that. The first is assumed to be under control thanks to having a very good preconditioner[7] while the other is under our control by a suitable choices of the hierarchical format, structure, accuracy and so on.

Looking at the k-th preconditioned system matrix $A_k P_k^{(1)} P_1$ for the map $\mathcal{P}^{(1)}$, we set

$$\tilde{R}_k := I + P_1 E_{1,k} \tag{7}$$

and write

$$A_k P_k^{(1)} P_1 = A_k \mathcal{H}_{solve}\left(R_k^{(1)}\right) P_1 = B_k^{(1)} + C_k^{(1)} + \tilde{D}_k \tag{8}$$

with

$$B_k^{(1)} := A_k \left(\mathcal{H}_{solve}\left(R_k^{(1)}\right) - \left(R_k^{(1)}\right)^{-1}\right) P_1, \quad C_k^{(1)} := A_k \left(\left(R_k^{(1)}\right)^{-1} - \tilde{R}_k^{-1}\right) P_1,$$
$$\tilde{D}_k := A_k \tilde{R}_k^{-1} P_1 = A_k \left(P_1^{-1} + E_{1,k}\right)^{-1},$$

quantifying the three core approximation errors introduced in $\mathcal{P}^{(1)}$. The first two are controlled by the choice of the hierarchical format – the error introduced by the hierarchical LU factorization[8] in $B_k^{(1)}$ (see, e.g., [14, Theorem 24]) and the error introduce by the hierarchical approximation of the matrix $P_1 E_{1,k}$ in $C_k^{(1)}$. As most of the hierarchical formats keeps the diagonal entries explicitly, we in fact only approximate the product $P_1 E_{1,k}$ and then add the identity along the diagonal, obtaining

$$C_k^{(1)} = A_k \left(R_k^{(1)}\right)^{-1} \left(\tilde{R}_k - R_k^{(1)}\right) \tilde{R}_k^{-1} P_1 = A_k \left(R_k^{(1)}\right)^{-1} \left(P_1 E_{1,k} - \mathcal{H}(P_1 E_{1,k})\right) \tilde{R}_k^{-1} P_1. \tag{9}$$

[7] Similarly to above, we do not mean to say that good preconditioners always provide a good approximations of the inverse of the system matrix but rather that good preconditioners in some sense capture the essence (or a fundamental part of it) of the problem. However, the approximation quality can serve as an indicator and can be useful in absence of more refined tools.

[8] Notice that the error is introduced only there, the ensuing type and backward substitutions with the hierarchical factors does not introduce further approximation errors.

For $\mathcal{P}^{(2)}$ we follow the same idea and notation but adjust the approximation error due to the hierarchical matrix approximation to reflect the accumulation, i.e., we write

$$A_k P_k^{(2)} P_1 = A_k \mathcal{H}_{solve}\left(R_k^{(2)}\right) P_1 = B_k^{(2)} + C_k^{(2)} + \tilde{D}_k \tag{10}$$

with

$$B_k^{(2)} := A_k \left(\mathcal{H}_{solve}\left(R_k^{(2)}\right) - \left(R_k^{(2)}\right)^{-1}\right) P_1, \quad C_k^{(2)} := A_k \left(\left(R_k^{(2)}\right)^{-1} - \tilde{R}_k^{-1}\right) P_1,$$
$$\tilde{D}_k := A_k \tilde{R}_k^{-1} P_1 = A_k \left(P_1^{-1} + E_{1,k}\right)^{-1},$$

and recalling that

$$\begin{aligned} R_k^{(2)} &= R_{k-1}^{(2)} \oplus \mathcal{H}(P_1 E_{k-1,k}) = R_{k-2}^{(2)} \oplus \mathcal{H}(P_1 E_{k-2,k-1}) \oplus \mathcal{H}(P_1 E_{k-1,k}) \\ &= I \oplus \mathcal{H}(P_1 E_{1,2}) \oplus \ldots \oplus \mathcal{H}(P_1 E_{k-1,k}), \end{aligned}$$

we reformulate $C_k^{(2)}$ analogously to (9) and obtain

$$C_k^{(2)} = A_k \left(R_k^{(2)}\right)^{-1} \left(\tilde{R}_k - R_k^{(2)}\right) \tilde{R}_k^{-1} P_1,$$

with

$$\begin{aligned} \tilde{R}_k - R_k^{(2)} &= \sum_{\ell=2}^{k} P_1 E_{\ell-1,\ell} - \bigoplus_{\ell=2}^{k} \mathcal{H}(P_1 E_{\ell-1,\ell}) \\ &= \sum_{\ell=2}^{k} \left(P_1 E_{\ell-1,\ell} - \mathcal{H}(P_1 E_{\ell-1,\ell})\right) + \sum_{\ell=2}^{k} \mathcal{H}(P_1 E_{\ell-1,\ell}) - \bigoplus_{\ell=2}^{k} \mathcal{H}(P_1 E_{\ell-1,\ell}). \end{aligned}$$

The matrix $C_k^{(2)}$ captures the error accumulation due to the hierarchical approximation over the sequence of the problems. The second part, i.e., the difference of the algebraic and hierarchical sums of the hierarchical matrices, can be handled analogously to (9) but the error there is due to the *$\mathcal{H}$-rank compression*[9] after the formatted sum. Altogether we see that $\tilde{R}_k - R_k^{(2)}$ has $2k-1$ terms that capture the hierarchical approximation error, compared to only one for $\tilde{R}_k - R_k^{(1)}$: k thanks to the k approximation errors of hierarchical format representations of $P_1 E$ and $k-1$ due to the re-compression after the formatted additions. For long problem sequences this might make $\mathcal{P}^{(2)}$ less appealing[10] or might require a HAM restart, where we compute a new preconditioner and start as if for a new system sequence.

For both $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ we see that the last term in (8) and (10) captures the approximation quality of P_1 and is independent of the chosen map, namely we get

$$\begin{aligned} \tilde{D}_k = A_k \left(P_1^{-1} + E_{1,k}\right)^{-1} = \left(A_1 + E_{1,k}\right) \left(P_1^{-1} + E_{1,k}\right)^{-1} &= I + \left(A_1 - P_1^{-1}\right) \left(P_1^{-1} + E_{1,k}\right)^{-1} \\ &= I + \left(A_1 P_1 - I\right) \left(I + E_{1,k} P_1\right)^{-1}. \end{aligned} \tag{11}$$

[9] As mentioned above, For most hierarchical formats the efficiency stems from keeping a relatively small hierarchical rank r. However, after any formatted operation, the rank can quickly grow and hence, in practice, we usually follow the formatted operation with a hierarchical rank-compression – procedure where the hierarchical matrix is (approximately) projected onto the set of hierarchical matrices with hierarchical rank less or equal than some $r_{\max}$, see [21, Section 4.5] and also [16, Sections 7.2 and 8.8] for more details.

[10] As mentioned above, this can be sometimes balanced off by the accumulation of information in the update matrix $E_{1,k}$ compared to the update matrices $E_{\ell-1,\ell}$ for $\ell = 2, \ldots, k$, which then results in a higher computational costs both for computing and applying $P_k^{(1)}$ (assuming we keep the desired accuracy fixed).

Before addressing these further, we would like to emphasize a key difference between the HAM and SAM approaches in the following remark.

Remark 1. *The SAM approach ([7]) and its analysis does not explicitly capture the interaction between the system change (here the update matrix E) and the original preconditioned system – this interaction is "hidden" in the least square problem minimization and as a result the error terms can be fully user controlled quantities, see [7, equation (2.5) and below]. In our case, however, this interaction seems unavoidable and as a result the error contains also terms which are* problem-dependent *and cannot be adjusted for by the user.*

Following up on Remark 1, since we only know that A_k is non-singular for all k, it is a natural starting point for analysis to assume that the magnitude of the update is smaller than the distance of A_1 to singularity, i.e., that $\|E_{1,k}A^{-1}\| < 1$ so that indeed A_k is non-singular. Assuming that the analogue holds also for the initial preconditioner, i.e., that

$$\|E_{1,k}P_1\| =: \gamma < 1, \tag{12}$$

we can expand the inverse in $\tilde{D}_k$ into its Neumann series, obtaining

$$\tilde{D}_k = I + (A_1P_1 - I)\left(I + E_{1,k}P_1\right)^{-1} = A_1P_1 + (A_1P_1 - I)\sum_{i=1}^{+\infty}\left(E_{1,k}P_1\right)^i =: A_1P_1 + D_k. \tag{13}$$

Alternatively, in many applications the update matrices $E_{1,k}$ have a fixed rank ν, i.e., we have

$$E_{1,k} = XY^T, \quad X, Y \in \mathbb{R}^{N\times\nu},$$

e.g., thanks to the localization of the update in the physical domain of the underlying PDE, see Section 4. Then, instead of using the Neumann series expansion, we use the Sherman-Morrison-Woodbury formula and, denoting $Z := P_1^TY$, obtain

$$\tilde{D}_k = I + (A_1P_1 - I)\left(I + XZ^T\right)^{-1} = I + (A_1P_1 - I)\left(I - X\left(I + Z^TX\right)^{-1}Z^T\right) = A_1P_1 + D_k, \tag{14}$$

or, equivalently,

$$\tilde{D}_k = I + (A_1P_1 - I)\left(I + XZ^T\right)^{-1} = A_1P_1 + D_k =: I + A_1P_1 - I + D_k. \tag{15}$$

Altogether, we can write

$$A_kP_k^{(1,2)}P_1 = A_1P_1 + B_k^{(1,2)} + C_k^{(1,2)} + D_k = I + B_k^{(1,2)} + C_k^{(1,2)} + A_1P_1 - I + D_k, \tag{16}$$

where both $B_k^{(1,2)}$ and $C_k^{(1,2)}$ are expected to be small due to the control of the hierarchical errors and we *hope* that the interaction of the update and the preconditioner captured by $\tilde{D}_k$ can be efficiently treated with either "small in norm" approach or "low-rank" approach – this is in general not guaranteed as we can have a moderate rank and $\gamma > 1$ at the same time even for an excellent preconditioner P_1, see Section 4. However, if either ν is small or $\gamma < 1$, then we can get a handle on the matrix $\tilde{D}_k$ which, as we will see, then leads to a pseudospectral bounds for the preconditioned GMRES convergence behavior.

We continue with some direct bounds of the norm of the relevant matrices. For lack of further structure of the sequence A_k, i.e., assuming only (12), the bounds for $B_k^{(1,2)}, C_k^{(1,2)}$ are quite crude based on the sub-multiplicativity of the standard matrix norms, and we use the Neumann series bound to bound the norm of D_k, obtaining

$$\begin{aligned}
\|B_k^{(1,2)}\| &\leq \|A_k\|\|P_1\| \left\| \mathcal{H}_{solve}\left(R_k^{(1,2)}\right) - \left(R_k^{(1,2)}\right)^{-1} \right\|, \\
\|C_k^{(1,2)}\| &\leq \|A_k \left(R_k^{(1,2)}\right)^{-1} \|\|\tilde{R}_k^{-1} P_1\| \left\| \tilde{R}_k - \mathcal{H}(\tilde{R}_k) \right\|, \\
\|D_k\| &\leq \frac{\gamma}{1-\gamma} \|A_1 P_1 - I\|,
\end{aligned} \tag{17}$$

where $\|E_{1,k} P_1\| = \gamma < 1$. We note that we expect $A_k \left(R_k^{(1,2)}\right)^{-1} \approx A_1$ and therefore using these bounds comes with a hefty prize as the hierarchical approximation error has to balance out a possibly large terms of $\|A_k\|$ and $\|A_k \left(R_k^{(1,2)}\right)^{-1} \| \approx \|A_1\|$. Assuming, in addition, that $\|P_1 E_{1,k}\| < 1$ we can write

$$\|\tilde{R}_k^{-1} P_1\| \leq \frac{\|P_1\|\|P_1 E_{1,k}\|}{1 - \|P_1 E_{1,k}\|},$$

using, again, the Neumann series expansion bound, see (7). We see that our understanding would improved if we get an insight into the interplay of the hierarchical approximation and the matrices A_k, P_1. We comment on this in the following remark.

Remark 2. *From practical point of view, the bounds in* (17) *can be a large overestimation and some of the reasons are, in our eyes, hard to address in general, e.g., the first inequality is unlikely to be close to equality. However, there are some direct improvements. For example, in* (8) *and* (10) *we can couple the matrices* $B_k^{(1,2)}$ *and* $C_k^{(1,2)}$ *obtaining*

$$A_k P_k^{(1,2)} P_1 = A_k \left(\mathcal{H}_{solve}\left(R_k^{(1,2)}\right) - \tilde{R}_k^{-1} \right) P_1 + \tilde{D}_k, \tag{18}$$

so that the possibly large factors $\|A_k \left(R_k^{(1,2)}\right)^{-1} \|, \|\tilde{R}_k^{-1} P_1\|$ *can be disregarded, while the difference of the matrices* $\mathcal{H}_{solve}\left(R_k^{(1)}\right)$ *and* $\tilde{R}_k^{-1}$ *is still controlled by appropriate choice of the hierarchical type. We choose to continue working with* (17) *as it highlights the important difference between* $\mathcal{P}_1, \mathcal{P}_2$ *but all of the results below can be restated based on* (18) *rather than on* (17).

Other reformulations that would address the terms $\|A_k\|, \|P_1\|$ *as well as relaxed the rather ambitious assumption on* γ *include studying the interaction of the eigenspaces of the relevant matrices and will be treated in full length in a separately.*

Hence, we proceed to give general convergence bounds for the preconditioned GMRES directly. Adapting these to more specialized settings where further information about the systems can be used seems like worthwhile direction for future research.

The GMRES convergence behavior for the type of systems as in (16) has been already studied using the pseudospectral bounds – the perturbed system matrix in [23] and the low-rank plus small-norm perturbation to the identity matrix in [8]. Before applying these results to our situation, we recall that for any $\delta > 0$ the δ-pseudospectrum of a matrix M, denoted by $\sigma_\delta(M)$, is defined as

$$\sigma_\delta(M) = \left\{ z \in \mathbb{C} \mid \|(zI - M)^{-1}\| > \frac{1}{\delta} \right\} = \left\{ z \in \sigma(M + E) \text{ for some } E \text{ with } \|E\| < \delta \right\},$$

and for any $\delta > 0$ forms a union of Jordan curves surrounding the spectrum of M, denoted by $\sigma(M)$, which can be recovered by taking $\delta = 0$. Importantly, the standard relative residual bound for GMRES convergence using the δ-pseudospectrum reads

$$\frac{\|\mathbf{r}_m\|}{\|\mathbf{r}_0\|} \leq \frac{L_\delta}{2\pi\delta} \min_{\substack{\deg(\varphi)\leq m \\ \varphi(0)=1}} \max_{z\in\sigma_\delta(M)} |\varphi(z)|,$$

where L_δ denotes the arc length of the boundary of the δ-pseudospectrum of M and $\varphi(z)$ is a polynomial of degree up to m and normalized so that $\varphi(0) = 1$; for more details on pseudospectra we refer the reader to [25] and references therein and for their use in understanding and predicting Krylov subspace methods behavior (GMRES in particular) we refer the reader to [19, Sections 4.9 and 5.7.3] but also to the recent manuscript [10, Section 2.3] and the literature cited in these. We follow the notation in [23] and denote the GMRES residuals corresponding to the initial preconditioned problem (2) by $\mathbf{r}_m$ $(m = 1, 2, \dots)$ while denoting the residuals for the system $A_k\mathbf{x}_k = \mathbf{b}_k$ with the preconditioner $P_k^{(1,2)}P_1$ by $\boldsymbol{\rho}_m^{(1,2)}(k) = \boldsymbol{\rho}_m^{(1,2)}$ $(m = 1, 2, \dots)$. Following the calculations in [23, 8], we obtain the following results.

Proposition 1 ([23, Theorem 2.1])**.** *Consider a sequence of linear systems $A_k\mathbf{x}_k = \mathbf{b}_k$ for $k = 1, 2, \dots$ and adopting the above notation, we fix some $k > 1$ and $\star \in \{1, 2\}$. Let us assume that $\|E_{1,k}P_1\| < \gamma < 1$, that the hierarchical type used for $P_k^\star$ was chosen so that*

$$\left\|\mathcal{H}_{solve}\left(R_k^\star\right) - \left(R_k^\star\right)^{-1}\right\| \leq \varepsilon_B \frac{1}{\|A_k\|\|P_1\|},$$
$$\|P_1E_{1,k} - \mathcal{H}(P_1E_{1,k})\| \leq \varepsilon_C \frac{1}{\|A_k\left(R_k^\star\right)^{-1}\|\|\tilde{R}_k^{-1}P_1\|},$$

and, moreover, that the initial preconditioner P_1 was such that

$$\|A_1P_1 - I\| \leq \varepsilon_D \frac{1-\gamma}{\gamma},$$

where $\varepsilon_{B,C,D} = \varepsilon_{B,C,D}(k, \star)$ but we omit the k and $\star$ dependency to make the exposition easier for orientation. Let $\varepsilon(k, \star) = \varepsilon := \varepsilon_B + \varepsilon_C + \varepsilon_D < 1$. Then for any $\delta/2 > \varepsilon$ and all $m = 1, 2, \dots$ we have

$$\frac{\|\boldsymbol{\rho}_m^\star\|}{\|\boldsymbol{\rho}_0^\star\|} \leq \frac{\|\mathbf{r}_m\|}{\|\mathbf{r}_0\|} + \varepsilon \cdot \frac{L_\delta}{\pi\delta^2} \max_{z\in\sigma_\delta(A_1P_1)} |p_m(z)|,$$

where L_δ denotes the arc length of the boundary of the δ-pseudospectrum of A_1P_1 and $p_m(z)$ is the GMRES polynomial for the initial preconditioned system (2) at iteration m.

Proposition 2 ([8, Equations 5–7])**.** *Consider a sequence of linear systems $A_k\mathbf{x}_k = \mathbf{b}_k$ for $k = 1, 2, \dots$ and adopting the above notation, we fix some $k > 1$ and $\star \in \{1, 2\}$. Let us assume that $\operatorname{rank}(E_{1,k}) = \nu \ll N$ and that the hierarchical type used for $P_k^\star$ was chosen so that*

$$\left\|\mathcal{H}_{solve}\left(R_k^\star\right) - \left(R_k^\star\right)^{-1}\right\| \leq \varepsilon_B \frac{1}{\|A_k\|\|P_1\|},$$
$$\|P_1E_{1,k} - \mathcal{H}(P_1E_{1,k})\| \leq \varepsilon_C \frac{1}{\|A_k\left(R_k^\star\right)^{-1}\|\|\tilde{R}_k^{-1}P_1\|},$$

and, moreover, that the initial preconditioner P_1 was such that

$$\|A_1P_1 - I\| \leq \varepsilon_D,$$

where $\varepsilon_{B,C,D} = \varepsilon_{B,C,D}(k, \star)$ but we omit the k and $\star$ dependency to make the exposition easier for orientation. Let $\varepsilon(k, \star) = \varepsilon := \varepsilon_B + \varepsilon_C + \varepsilon_D < 1$. Then for any $\delta/2 > \varepsilon$ and any $m = \nu+1, \nu+2, \ldots$ we have

$$\frac{\|\boldsymbol{\rho}_m^\star\|}{\|\boldsymbol{\rho}_0^\star\|} \leq \varepsilon \cdot \frac{L_\delta}{\pi\delta^2} \max_{z \in \sigma_\delta(A_1P_1)} |p_m(z)|,$$

where L_δ denotes the arc length of the boundary of the δ-pseudospectrum of A_1P_1 and $p_m(z)$ is the GMRES polynomial for the initial preconditioned system (2) *at iteration m.*

First, we highlight that in both Proposition 1 and 2 we describe the GMRES behavior as a function of that of the initial system. More precisely, we describe the *delay* behind the GMRES behavior for the initial preconditioned system (2). Moreover, this delay is spelled out *explicitly* as a function of ε but is, seemingly, only *implicit* as a function of the GMRES iteration m (as highlighted in [23, p. 1071–1072]). As a result, solid grasp on the convergence of the initial system is an important piece of information to make these results useful and we comment on this further in the following section. Also, both Proposition 1 and 2 rely on some quantification of the interaction of the initial preconditioner P_1 with the update matrix $E_{1,k}$, as highlighted in Remark 1. If these assumptions become too crude, e.g., $\gamma \lessapprox 1$, $\nu \gg 1$ or $\|A_k\| \gg 1$, then we expect the convergence bounds to also become quite crude, possibly in need of an improvement in the spirit of Remark 2.

Second, following [23], we note that both Proposition 1 and 2 give a *family* of bounds based on $\delta > 0$, rather than a single bound – see Figures 3 and 4 ahead. Increasing δ, from a certain δ_0 onward, the δ-pseudospectrum will contain the origin and due to the GMRES polynomial normalization the bounds become useless. The common wisdom is that larger values of $\delta < \delta_0$ tend to be more descriptive at the initial convergence phase while smaller values of δ give a more accurate prediction for later stages of GMRES, see Figures 3–4 ahead or the figures in [10, Section 3.1].

Next, we want to point out that the bounds also seems appropriate for the field of values bound (FoV), similarly to [7, Section 2, eqns. (2.10–2.11)] for the SAM approach; see [19, Section 5.7.3] and [10, Section 2.2] for further references on the FoV bounds for GMRES. We address this direction in more detail in Section 4.1 below.

Last but not least, we would like to emphasize that we use the *ideal GMRES* bound, which can fail to capture the observed GMRES behavior to arbitrary level based on the interaction of the system matrix, the right-hand side and the initial guess, see [19, Section 5.7.3] for more details. Although in practice these bounds can be very useful, it is necessary to keep their limitations in mind. We continue by numerically investigating the proposed preconditioner maps on a particular model example.

4 Numerical experiments

To illustrate and further explore the results of Section 3 we consider a sequence of 2D non-linear advection-diffusion problems on a unit square $\Omega := [0, 1] \times [0, 1]$ with piecewise constant coefficients that slowly change, namely

$$\begin{aligned} -(p_k \cdot u_{x_1})_{x_1} - (q_k \cdot u_{x_2})_{x_2} + r \cdot u_{x_1} + s \cdot u_{x_2} &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{19}$$

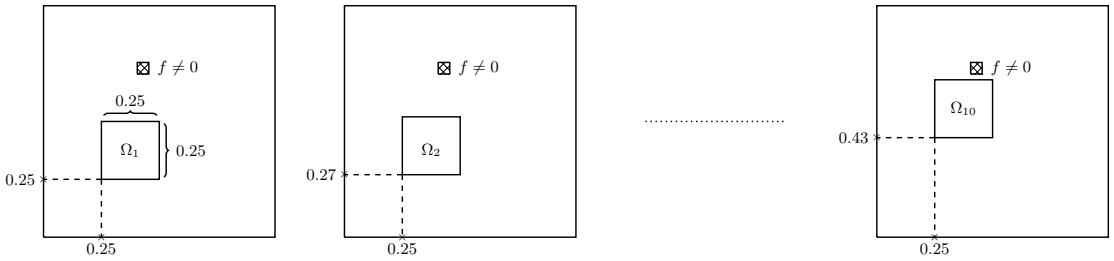

Figure 1: Geometry of the sequence of ten problems in (19).

with constant advection coefficients $r = s = -5$ and piecewise constant diffusion coefficients p_k, q_k

$$p_k(\mathbf{x}) = q_k(\mathbf{x}) = \begin{cases} = 5 & \text{if } \mathbf{x} \in \Omega_k, \\ = 0.1 & \text{otherwise,} \end{cases}$$

where for $k = 1, \ldots, 10$ we have a small square $\Omega_k \subset \Omega$ that is slowly moving in the x_2 direction upwards as k increase, see Figure 1. Having the fixed source terms f and g as

$$f(\mathbf{x}) = \begin{cases} = 10^4 & \text{if } \mathbf{x} \in [0.405, 0.455] \times [0.705, 0.755], \\ = 0 & \text{otherwise,} \end{cases}$$

$$g(\mathbf{x}) = \begin{cases} = 2\sin(8\pi x_1) & \text{if } x_2 = 0, \\ = 0 & \text{otherwise,} \end{cases}$$

we obtain a sequence of ten linear problems for $k = 1, \ldots, 10$.

We discretize (19) using the finite difference method with the standard 5-point stencil, obtaining a sequence of linear algebraic systems as in (1), and we notice that all of the update matrices $E_{\ell,k}$ have comparable number of non-zero columns, i.e., we both $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ are experimentally admissible. We calculate P_1 using the MATLAB's routine `ilu` with `options = 'crout'` and `droptol = 1e-6`, which we consider an expensive-to-calculate, high-quality preconditioner and we solve the problems $A_k\mathbf{x}_k = \mathbf{b}_k$ for $k \geq 1$ with the preconditioned GMRES method using P_1 for all ten problems. We show the number of iterations in Figure 2 (left) based on the number of unknowns N. We see that already for moderate problem sizes there is a large increase in number of iterations for $k \geq 2$, making this a good test case for our purposes – we wish to make a better use of P_1 for $k \geq 2$. Apart from this feature (which is common for many preconditioners), we do not consciously build upon any particular structure or nature of P_1, i.e., it is just a common choice that is meant to be easily replaceable by a user provided preconditioner for $A_k\mathbf{x}_1 = \mathbf{b}_1$.

For all our experiments, we use the MATLAB implementation of the HODLR and HSS formats given in [21] with necessary adjustments and unless specified otherwise, the parameter settings are left on their default. Using the predefined formats HODLR and HSS directly, we see in Figure 2 (left) that both $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ keep the number of iterations very low[11] (in fact constant) for all ten problems. Although this looks like a positive result, Figure 2 (right) shows the hierarchical rank r of the hierarchical matrices $R_k^{(\star)}$ for $k = 1, \ldots, 10$, which determines the efficiency of the application

[11] We show this only for the problem with the largest size – highlighted by the color – but this was true for any problem and size we have experimented with.

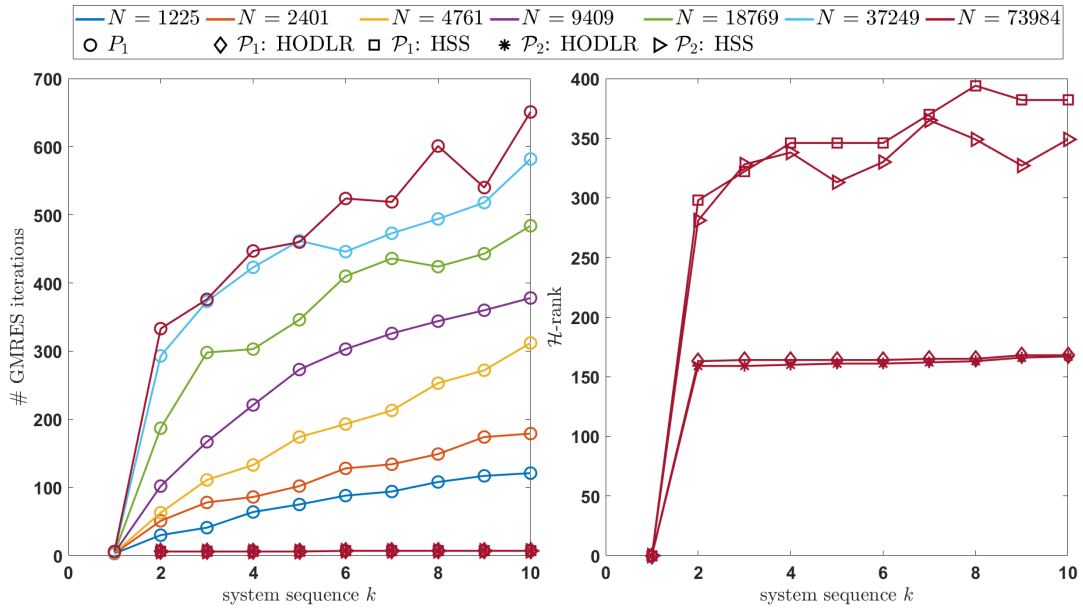

Figure 2: Left: the number of preconditioned GMRES iterations to reach the relative residual 10^{-10} using the preconditioner P_1 ($\circ$) for different N and also using the preconditioner maps $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}$ for hierarchical formats HODLR ($\Diamond$,$\star$) and HSS ($\square, \triangleright$). Right: the hierarchical rank of the matrices $P_k^{(1,2)}$ for the given format with $N = 73984$.

of these preconditioners. As the $\mathcal{H}_{solve}$ operation complexity scales as

$$\mathcal{O}\left(r^2 N \log^2(N)\right) \quad \text{and} \quad \mathcal{O}\left(r^2 N\right), \tag{20}$$

for the HODLR and HSS formats[12] (see [21, Table 3]) we see that applying these preconditioner is *not* efficient – in fact for both of these formats these complexities become comparable with $\mathcal{O}\left(N^2\right)$, which is the standard complexity for the backward and forward substitutions for a general factored preconditioner (such as P_1).

Remark 3. *We decided to use and present the hierarchical rank of the matrices in HODLR/HSS formats as the measurement of the efficiency of their application rather than resorting to timings of these. There are several reasons for that, starting with the computational environment of the cluster at Virginia Tech and ending with the interaction of the* `hm-toolbox` *and the MATLAB environment that requires a deep insight into both MATLAB as well as into the particularities of the toolbox implementation to guarantee efficiency. Recognizing that in practice we are much more likely to use the hierarchical format libraries in C or C++ (see, e.g., [6, 12]), we seek a reasonable proxy to measure the efficiency also in the MATLAB environment, where the analysis of the preconditioners is significantly easier, and the hierarchical rank is a clear indicator based on the analysis of the hierarchical formats, see, e.g., [16].*

We continue by looking first at the numerical results illustrating the above analysis in Section 4.1

[12]Notice that the favorable complexity for the HSS format is balanced out by the notably higher hierarchical rank. This is to be expected has been our experience in general and, as a result, makes both formats competitive.

and then continue in Section 4.2 by numerically investigating possible adjustments to the hierarchical types so that the considered preconditioners become more efficient.

4.1 HAM bounds

Pseudospectra We start with a closer look at the bounds in Proposition 1 and 2 – both of them contain the quantity

$$\frac{\varepsilon}{\delta/2}\tilde{S}_m(\delta), \quad \text{with } \tilde{S}_m(\delta) := \frac{L_\delta}{2\pi\delta} \max_{z\in\sigma_\delta(A_1P_1)} |p_m(z)|, \tag{21}$$

where L_δ denotes the arc length of the boundary of the δ-pseudospectrum of A_1P_1, $p_m(z)$ is the GMRES polynomial for the initial preconditioned system (2) at iteration m and $\tilde{S}_m(\delta)$ constitutes an upper bound on the preconditioned GMRES convergence for that problem. Clearly, this quantity needs to be evaluated or further estimated in order to obtain the desired convergence bounds as was already highlighted in [23, p. 1071–1072]. However, our set-up gives us a powerful tool – we have already run the preconditioned GMRES method for the initial system. In the context of the original work, this is not the case as one is interested in solving *only* the perturbed system, possibly not even having access to the unperturbed one. As a result we have at our disposal the $(m+1)$-by-m matrix $H_{m+1,m}$ (with entries h_{ij}) coming out of GMRES applied to (2).

If GMRES terminated at iteration $\tilde{\mu}$ with $h_{\tilde{\mu}+1,\tilde{\mu}} = 0$, then we found an invariant Krylov subspace, which by definition contains all relevant information for understanding the GMRES convergence and the system matrix in (21) (in our case A_1P_1) can be equivalently replaced by $H_{\tilde{\mu},\tilde{\mu}}$, i.e., the $\tilde{\mu}$-th leading principal submatrix of $H_{\tilde{\mu}+1,\tilde{\mu}}$. In practice, this is almost always ill-advised as we are satisfied with the approximate solution at some earlier iteration $\mu < \tilde{\mu}$ and as a result we have $h_{\mu+1,\mu} \neq 0$ when GMRES terminated. Nevertheless, we use the pseudospectra of $H_{\mu,\mu}$ (or $H_{\mu+1,\mu}$) as an *approximation* to those of the system matrix *before* $h_{\mu+1,\mu} = 0$, obtaining only a GMRES convergence *estimates* rather than bounds, see [10, Section 4 and Theorem 4.1] and the references therein. In practice, this becomes an efficient tool for evaluating (21) – from the beginning we aim to make good use of a very efficient preconditioner P_1, meaning that $\mu \ll N$ and hence $H_{\mu,\mu}$ (or $H_{\mu+1,\mu}$) are small matrices. As a result, calculating the harmonic Ritz values (which characterize the GMRES polynomial $p_\mu(z)$, see [19, Section 5.7.1]) as well as the δ-pseudospectrum[13] $\sigma_\delta(H_{\mu,\mu})$ (or $\sigma_\delta(H_{\mu+1,\mu})$) becomes *computationally insignificant* compared to the GMRES method to be run for $k = 2, 3, \ldots$ and so does the evaluation of (21). Hence, we can evaluate these *estimates* for a given ε *a-priori*, in theory allowing us to choose appropriate values of $\varepsilon_{B,C,D}(k, \star)$ so that an appropriate hierarchical type can be used.

We show the δ-pseudospectra of $H_{\mu,\mu}$ together with the estimates $S_m(\delta,\mu)$ of $\tilde{S}_m(\delta)$ with

$$S_m(\delta,\mu) := \frac{L_\delta}{2\pi\delta} \max_{z\in\sigma_\delta(H_{\mu,\mu})} |p_m(z)|, \quad m = 1, 2, \ldots,$$

[13] As of now, the EigTool package ([26]) used in the community as the default tool to calculate pseudospectra does not support calculation with large sparse matrices due to compatibility issues with newer releases of MATLAB and hence we cannot investigate how well the pseudospectra $\sigma_\delta(H_{\mu,\mu})$ (or $\sigma_\delta(H_{\mu+1,\mu})$) approximate $\sigma_\delta(A_1P_1)$. However, precisely this process, i.e., approximating the large sparse matrix with the small dense matrix $H_{m+1,m}$ coming out of m steps of the Arnoldi process, was the fundamental process on which the EigTool was built and it also has become a "default workaround" in the community to obtain the pseudospectra of large sparse matrices until the functionality of EigTool gets restored, see [10, 27, 26].

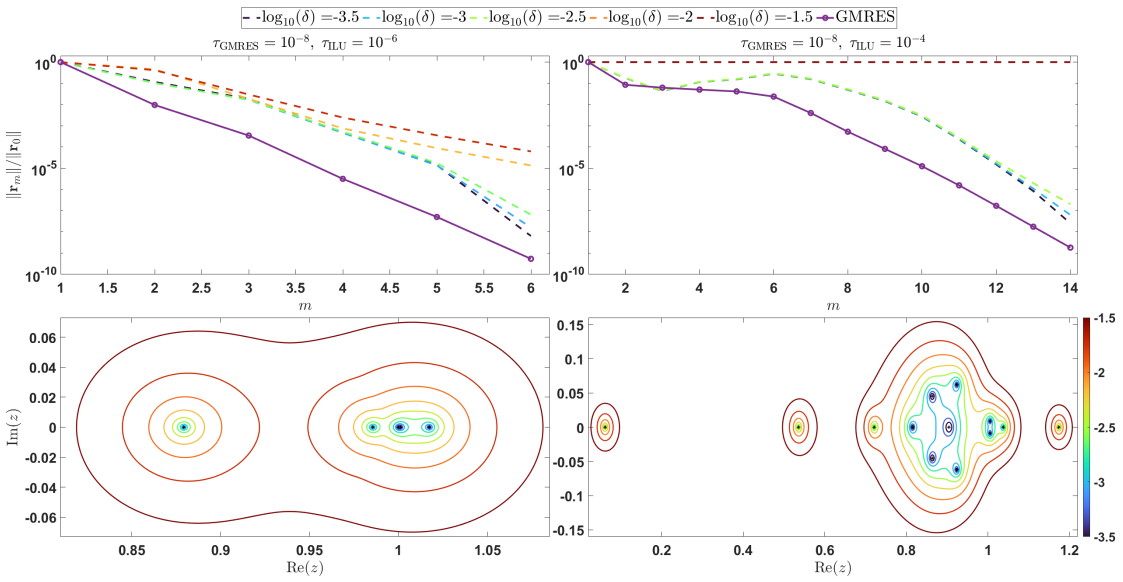

Figure 3: Top: the GMRES convergence together with the PSA bounds $C_m(\delta)$ for various δ; Bottom: the boundaries of the δ-pseudospectra inducing the bounds together with the Ritz values ($*$). We fixed $N = 73984$ and obtained P_1 via the MATLAB routine `ilu`(A_1) with `options.type = 'crout'` and `options.droptol` $= \tau_{\text{ILU}}$. The colorbar for the bottom row gives the decadic logarithm of δ, as suggested by the legend at the top.

where μ depends on the given GMRES relative residual tolerance τ_{GMRES}, in Figure 3, taking $\tau_{\text{GMRES}} = 10^{-8}$. We see that the estimates are generally reasonably accurate although they resulted in a slight underestimation of the relative residual at iteration $m = 3$ for $\tau_{\text{ILU}} = 10^{-4}$. Importantly, these estimates can be now inserted in Propositions 1 and 2 to obtain convergence estimates for the preconditioned GMRES convergence behavior for $k \geq 2$ and the better the estimates S_m are, the more accurately we will be estimating the behavior of our preconditioners.

We also note that even if we aim only for reducing the relative residual below, say, 10^{-8}, it might be worthwhile calculating few extra iterations beyond μ for the initial preconditioned system. At the cost of running few extra iterations for the system (2), we obtain more detailed information for evaluating the estimates $S_m(\delta, \mu)$ and as a result a better indicator for choosing ε in Propositions 1 and 2. For example, the estimates in Figure 3 cannot give a prediction of the required number of iterations to converge below τ_{GMRES} – because the bounds/estimates necessarily overestimate the relative residual and we stopped the GMRES iteration *precisely at the moment when the relative residual reached* τ_{GMRES}. Hence, the estimates necessarily stopped (well) above that threshold and they cannot be further extended (as we do not have the following Hessenberg matrices to calculate the pseudospectra and the optimal GMRES polynomials). Running couple of additional iterations of GMRES can give us this information so that we get estimates for the number of iterations to reach τ_{GMRES} and these can be then used in Propositions 1 and 2. To be more precise, we multiply these estimates by $2\varepsilon/\delta < 1$ as in (21) and then add this convergence curve to the observed convergence behavior of the system A_1P_1 in Proposition 1 or we consider this bound starting only at iteration ν in Proposition 2. We show an example in Figure 4, where we took extra 3 iterations (for $\tau_{ILU} = 10^{-6}$) and extra 7 iterations (for $\tau_{ILU} = 10^{-4}$), corresponding to the ill-advised case of τ_{GMRES} being the

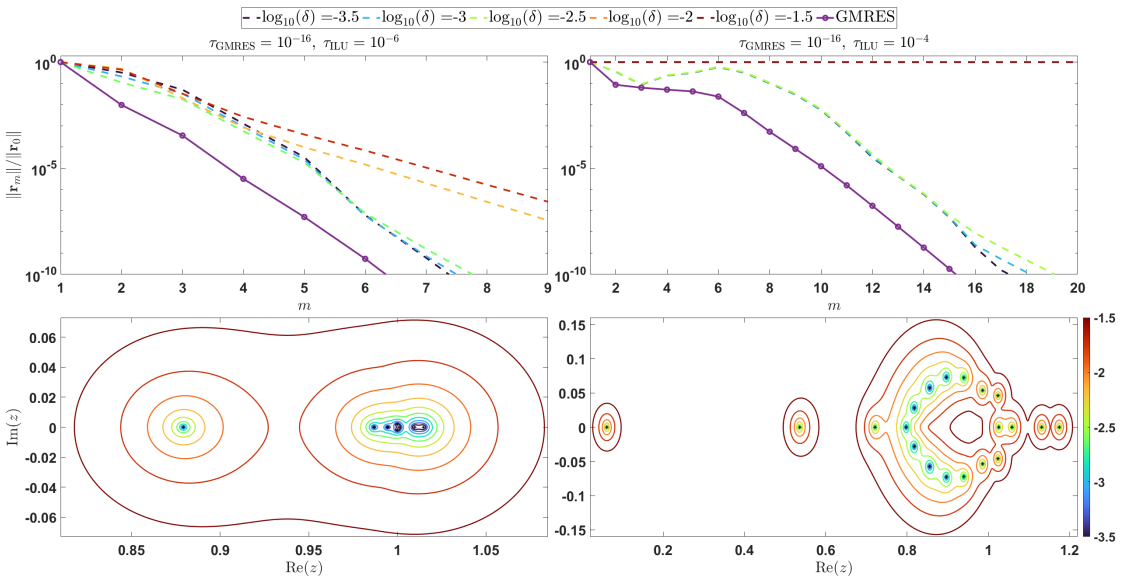

Figure 4: Top: the GMRES convergence together with the PSA bounds $C_m(\delta)$ for various δ; Bottom: the boundaries of the δ-pseudospectra inducing the bounds together with the Ritz values ($*$). We fixed $N = 73984$ and obtained P_1 via the MATLAB routine `ilu`(A_1) with `options.type = 'crout'` and `options.droptol` $= \tau_{\text{ILU}}$. The colorbar for the bottom row gives the decadic logarithm of δ, as suggested by the legend at the top.

machine precision, (which almost always corresponds to oversolving the algebraic system) – we see that indeed the estimates now give a concrete prediction for number of iterations of preconditioned GMRES so that the relative residual reaches the threshold 10^{-8}.

This example also highlights that when it comes to the estimates, the hurdle to overcome are the ε and/or ν parameters, just as predicted in Remark 2 – using Propositions 1 and 2 we have to settle for values that are overly restrictive. As highlighted in Section 3, the reason for the bounds being quite crude is the general scope of the analysis – by a direct calculation, we obtain $\gamma \gg 1$ and $\nu \approx 10^3$, meaning that the convergence bounds are no good in describing GMRES convergence behavior. Nevertheless, we still find the results useful in outlining the general idea of the HAM approach and improving on these is a subject of ongoing research, focusing on relaxing the assumptions in Propositions 1 and 2. Here, instead, we briefly touch upon the field of value bounds and then turn our attention to the issue of *efficiency* raised at the beginning of Section 4, thus finalizing the introduction of the HAM approach.

Field of values Building upon Section 3, we look at the contributions of the terms in (8) and (10) to enlarging the field of values of the preconditioned system. We use the MATLAB toolbox function `fv` to bound and visualize the field of values (FoVs) of matrices, see [17]. We show the FoVs of the three parts of (16) in Figure 5 for $\mathcal{P}^{(1)}$ and in Figure 6 for $\mathcal{P}^{(2)}$.

We see that the FoVs of $B_k^{(1,2)}, C_k^{(1,2)}$ are centered around the origin, as these capture the approximation error that converges to zero matrix as we decrease ϵ while the FoVs of D_k are centered around the point $1 + 0i$ as expected based on (11). However, we see that for both $\mathcal{P}^{(1,2)}$, the overall FoVs of the preconditioned system for $k \geq 2$ *includes the origin* for $\epsilon = 10^{-4}, 10^{-2}$ –

making the bound based on them useless. We note that, in theory, we could still try to make use of the bounds by the "cutting-out" technique of Crouziex and Greenbaum, see [9]. However, this approach is, in our opinion, *highly* unlikely to be successful here as the FoVs are dominated by the second line terms, i.e., by FoVs of $C_k^{(1,2)}$, which are *centered around the origin*. We also note that the dominating term has always been, in our experience, either the FoVs corresponding to $C_k^{(1,2)}$ – the term due to the error of the hierarchical approximation[14] – or to D_k – the term due to the quality of P_1.

Moreover, as N grows, the diameter of the FoVs of $C_k^{(1,2)}$ grows significantly as well, i.e., the term corresponding to D_k dominated only for problems of small size. We also note that changing the format to HSS either did not change the FoV meaningfully or even enlarged the diameters of the FoVs in all of our numerical experiments. A marginal improvements can be gained using the reformulation in the spirit of Remark 2 but, in principle, an improvement to these bounds needs to come from a more tailored approach to the analysis for the particular problem.

Although these results are negative, they are not completely surprising – the FoV bounds are generally expected to be at best as descriptive as the bounds based on pseudospectra but *often much less so*, see [10, Section 3.1 – No Example: Only FOV descriptive]. We also note that based on (8) and (10) we can give a proper bound, as suggested already in [7, Section 2, eqn. (2.8)] for SAM but based on the above insight, we see that such bound is not of much interest in general.

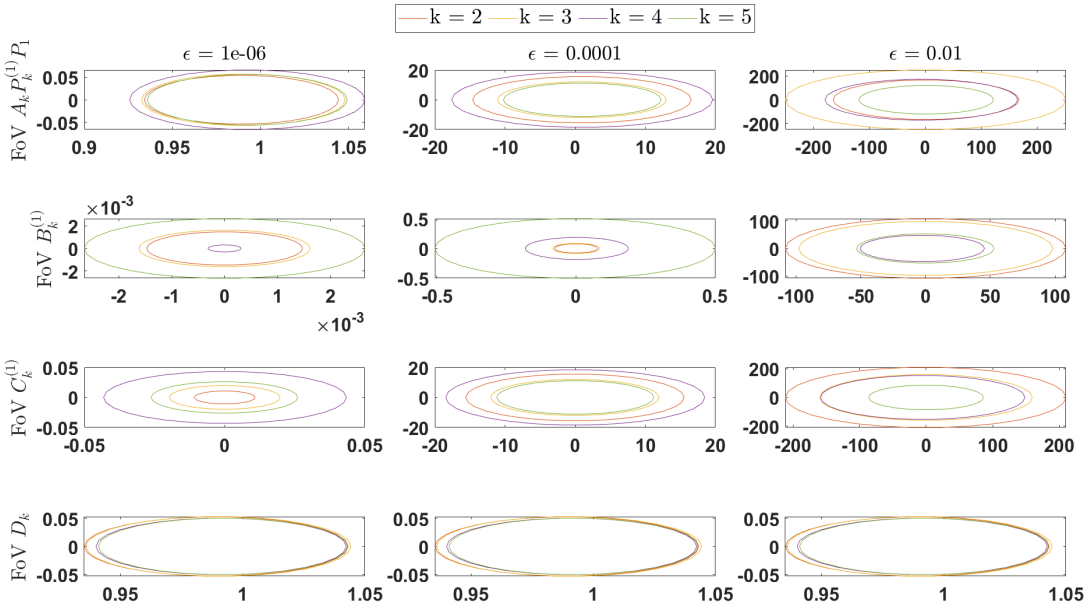

Figure 5: The bounds of FoVs for the three parts of the preconditioned systems for $k = 2, \ldots, 5$ for the map $\mathcal{P}^{(1)}$ different values of ϵ. We show the results only for the HODLR format and fixed $\beta = 128$ and $N = 18769$.

[14]Notice that this term differs meaningfully between the maps $\mathcal{P}^{(1,2)}$ – we predicted this in the analysis in Section 3 and now confirmed it in Figures 5 and 6.

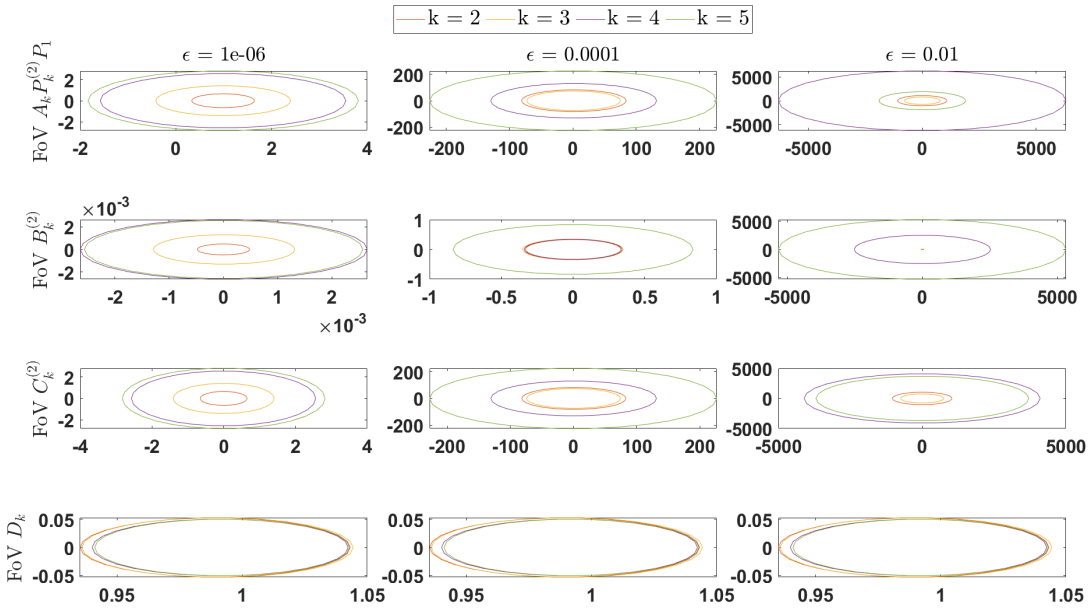

Figure 6: The bounds of FoVs for the three parts of the preconditioned systems for $k = 2, \ldots, 5$ for the map $\mathcal{P}^{(2)}$ different values of ϵ. We show the results only for the HODLR format and fixed $\beta = 128$ and $N = 18769$.

4.2 HAM efficiency

We return to the issue highlighted at the beginning of this section, namely that applying the HAM preconditioners $P^{(1,2)}$ is, complexity-wise, comparable to a Gaussian elimination for a pre-factored preconditioner. First, let us emphasize, that part of the problem that HAM is addressing is the fact that *we do not have* that pre-factored preconditioner but we acknowledge that such a high complexity is not feasible if we want to have a competitive preconditioner scheme. To understand why the hierarchical rank becomes so high, we take a closer look at the predefined settings of the used MATLAB `hm-toolbox` by Massei and Kressner, see [21].

In their implementation, both HODLR and HSS use the basic balanced binary tree as the cluster tree with the standard admissibility condition and therefore either of these formats is further characterized by only two parameters – the minimal block-size $\beta \in \mathbb{N}$ and the accuracy threshold ϵ. Increasing β corresponds to stopping the hierarchical blocking of the matrix at an earlier stage and therefore preserving larger diagonal blocks of the original matrix exact. Decreasing ϵ corresponds to requiring a more accurate overall approximation within the chosen format by virtue of increasing *the hierarchical rank* of the approximation (for more detailed description, we refer to the literature cited in Section 3 and to [21] and the references therein). The default values are set to $\beta = 256$ and $\epsilon = 10^{-12}$. While these might be perfectly reasonable as a standalone, the first natural step in our context is investigate the effect of relaxing these on the hierarchical rank of the resulting matrices $R_k^{(1,2)}$. We note that although there is some relevant theory linking ϵ with the hierarchical rank r, see, e.g., [4], to the best of our knowledge all analysis in this direction is considered *in the limit as* $\epsilon \to 0$ – the opposite of what we want to look at. Hence we resort to numerical investigation.

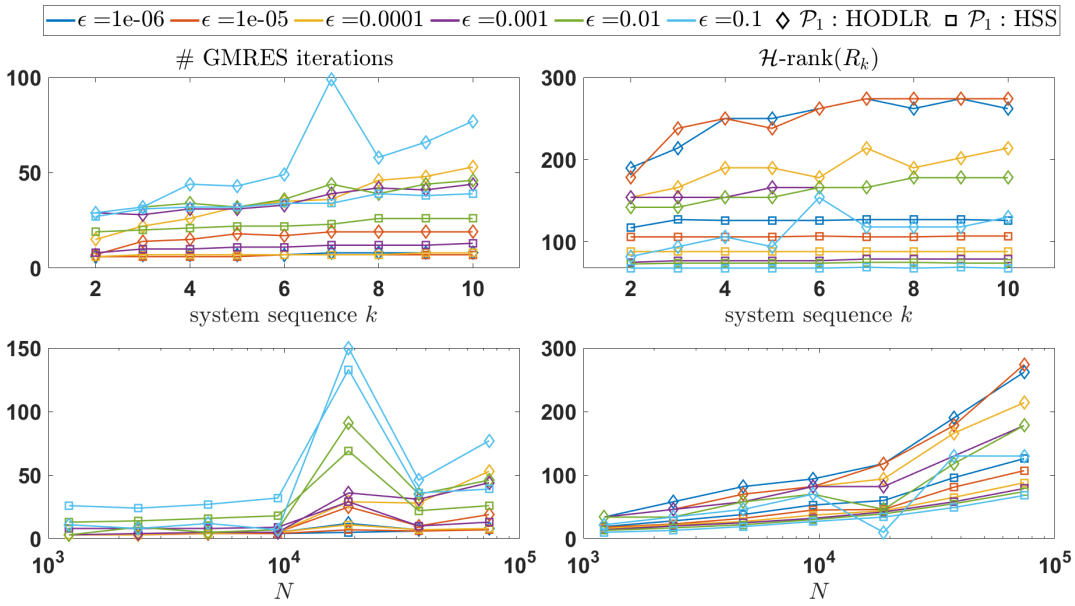

Figure 7: Number of iterations of the preconditioned GMRES (left) and the hierarchical rank of the matrix R_k (right) plotted against the system sequence (top) and system size (bottom) for the map $\mathcal{P}^{(1)}$ for both HODLR and HSS formats and various values of ϵ. Here we fixed $\beta = 256$ and $N = 73984$ for the top row and $k = 10$ for the bottom row.

Effect of β and ϵ First, we note that changing the block size β had little to no effect on the quality of the preconditioners. We ran extensive amount of experiments, varying both $\beta \in [32, 2048]$ as well as the system size $N \in (10^3, 10^5)$ for both maps $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}$ and for both formats HODLR and HSS and found that both the rank as well as the number of GMRES iterations are quite stable with respect to changing β, even though these do change a little more for $\beta = 1024, 2048$ – lowering the number of iterations and varying the hierarchical rank (notably but not significantly). Looking at the runtimes, our experience was that β and N should be kept *proportional* so that the cluster tree has more or less constant number of levels. We are, however, aware of implementation limitations of the `hm-toolbox` in MATLAB and more experiments and analysis is needed in this direction.

In contrast to that, changing the accuracy ϵ had a significant impact on both the number of iteration of preconditioned GMRES as well as on the hierarchical rank and we illustrate this in Figures 7–8. We see a notable difference for most of the values of ϵ – smaller values result in more accurate approximations and hence lower number of iterations but require a higher hierarchical rank. Altogether, we see that the number of preconditioned GMRES iterations remains more or less constant for $k = 1, \ldots, 10$ with much lower hierarchical rank compared to Figure 2 and with a *significant* improvement compared to using P_1, even for a quite poor accuracy $\epsilon = 0.1$, see Figure 2.

Comparing the maps $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$, we see that $\mathcal{P}^{(1)}$ results in lower number of GMRES iterations but requires a higher hierarchical rank to do so. Recalling (20), a fairer comparison is to compare the number of iterations times the cost per iteration. Denoting the number of GMRES iterations with the map $\mathcal{P}^{(i)}$ by $n^{(i)}$ and the resulting hierarchical rank by $r^{(i)}$, the map $\mathcal{P}^{(1)}$ becomes more efficient provided that

$$(n^{(2)}/n^{(1)})^{-1/2} < r^{(2)}/r^{(1)}. \tag{22}$$

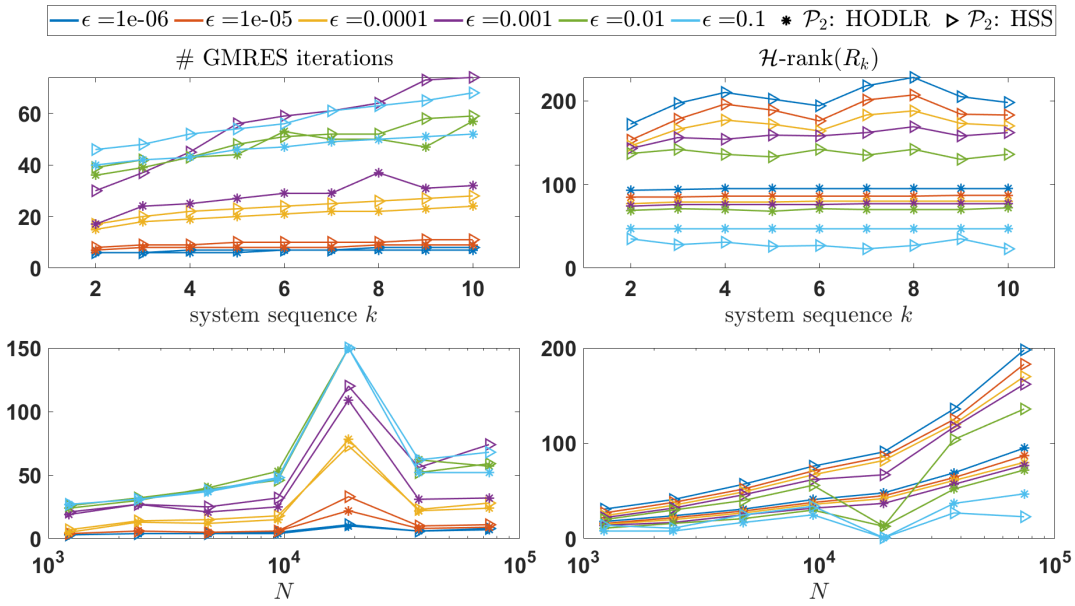

Figure 8: Number of iterations of the preconditioned GMRES (left) and the hierarchical rank of the matrix R_k (right) plotted against the system sequence (top) and system size (bottom) for the map $\mathcal{P}^{(2)}$ for both HODLR and HSS formats and various values of ϵ. Here we fixed $\beta = 256$ and $N = 73984$ for the top row and $k = 10$ for the bottom row.

However, even in this better comparison, the map $\mathcal{P}^{(1)}$ is favorable to its counterpart, in spite of the higher hierarchical rank. This is, perhaps, to be expected due to the lack of error accumulation compared $\mathcal{P}^{(2)}$. Since the number of nonzero columns for $E_{1,k}$ and $E_{k-1,k}$ is roughly the same, there is no direct counterweight to this benefit – the somewhat higher hierarchical ranks of $\mathcal{P}^{(1)}$ are well compensated for.

Comparing the HODLR and HSS formats, we see that the hierarchical rank *as well as* the number of GMRES iterations for the HSS format tends to be higher for both $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}$. However, recalling the complexities in (20), we see that this does not necessarily make the HSS format less efficient as the extra term of $\log^2(N)$ in the HODLR complexity can offset these differences. In fact, for $N \gtrapprox 5000$, the HSS format runtimes[15] of the results plotted on Figures 7–8 are comparable or even significantly lower than if using the HODLR format, as N increases.

Last but not least, we see that with growing N both the number of iterations as well as the hierarchical ranks grow. This is not surprising as incomplete factorization preconditioners, such as P_1, are known to exhibit this behavior, i.e., we cannot hope for anything better in terms of GMRES iterations in general. Importantly, looking at the hierarchical rank growth, we are, unfortunately, still on the same N^2-like trajectory for almost all values of ϵ, regardless of the format and map used[16]. In order to address this issue, we look for further approximations with the goal of decreasing the

[15]These need to be taken with the same caveats as above with respect to performance of the `hm-toolbox` and MATLAB. Also, the $\mathcal{H}$-FAINV approach would likely change this as the scaling of the application of the preconditioner with respect to the hierarchical rank would become linear compared the currently quadratic one.

[16]We note that the problem with $N = 18769$ was in some sense exceptional as the system matrices were consistently easier to approximate (resulting in a lower hierarchical rank) but resulted in a worse preconditioner (hence higher number of iterations).

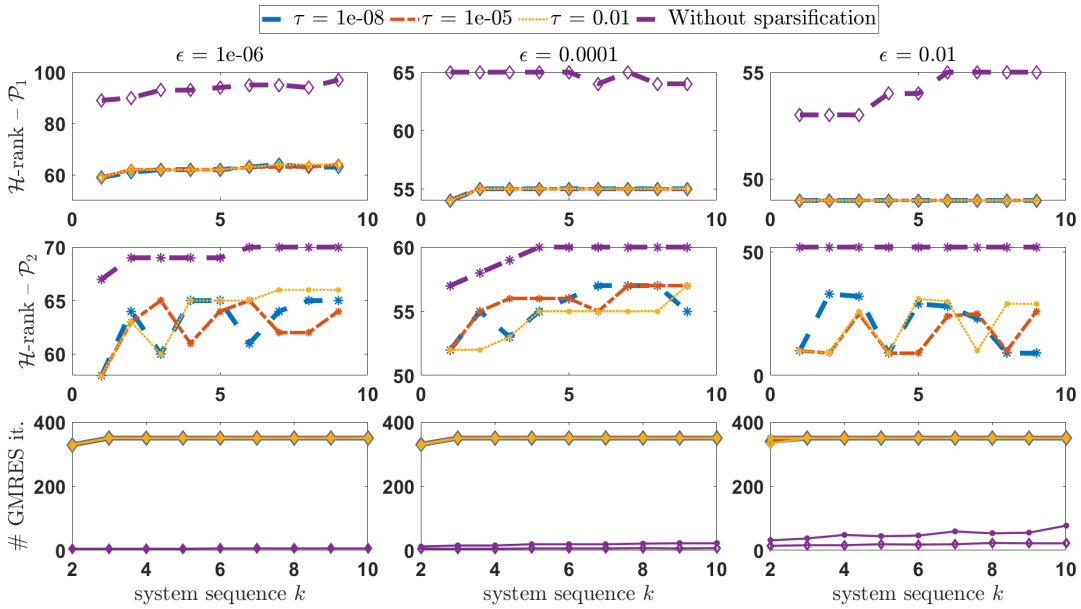

Figure 9: We show the hierarchical ranks of the resulting matrices R_k for the HODLR format when sparsifying the matrix $E_{1,k}$ ($E_{k-1,k}$) for the map $\mathcal{P}^{(1)}$ ($\mathcal{P}^{(2)}$) in the first row (second row) and the number of GMRES iterations (capped at 350) with the sparsified preconditioner compared to no sparsification. We fixed $\beta = 256$ and $N = 37249$.

hierarchical rank as well as the cost of construction of the preconditioner. For the SAM approach, the authors explored a similar direction by prescribing a *fixed sparsity pattern* of the preconditioner (and hence putting the complexity of the application of the preconditioner under the user control) and it turned out that the GMRES iteration count did not suffer significantly even from restrictive sparsity patterns (compared to the exact solution). We consider a similar approach in the following two sections – first *structural* sparsification, where some non-zero entries are dropped, and then *data* sparsification, where we truncate the hierarchical ranks.

Structural sparsification In our model problem the update matrix E (standing for either $E_{1,k}$ or $E_{k-1,k}$ depending on the chosen map) has only relatively few non-zero columns, each of which has only very few non-zero entries, as is the case in *many* applications. Having E structurally sparse, the idea of further sparsification seems at first natural in order to reduce the amount of information that needs to be captured by the hierarchical format approximation. In fact, there are two options – we can either sparsify the update matrix E itself or the matrix P_1E before we calculate its hierarchical approximation[17] – and those are the only options as once we calculate $\mathcal{H}(P_1E)$, there is no possibility for dropping non-zero elements of that matrix. We investigate the efficiency of both of the options numerically by dropping all entries that are smaller in magnitude than a fixed tolerance τ and show representative examples of our experience in Figure 9 (sparsifying the matrix E) and in Figure 10 (sparsifying the matrix P_1E).

[17]Notably, the later option would be harder to implement in the case of matrix-free assembly of the hierarchical format mentioned in Section 3. However, as we will see, there doesn't seem to be a reason to be concerned about this issue.

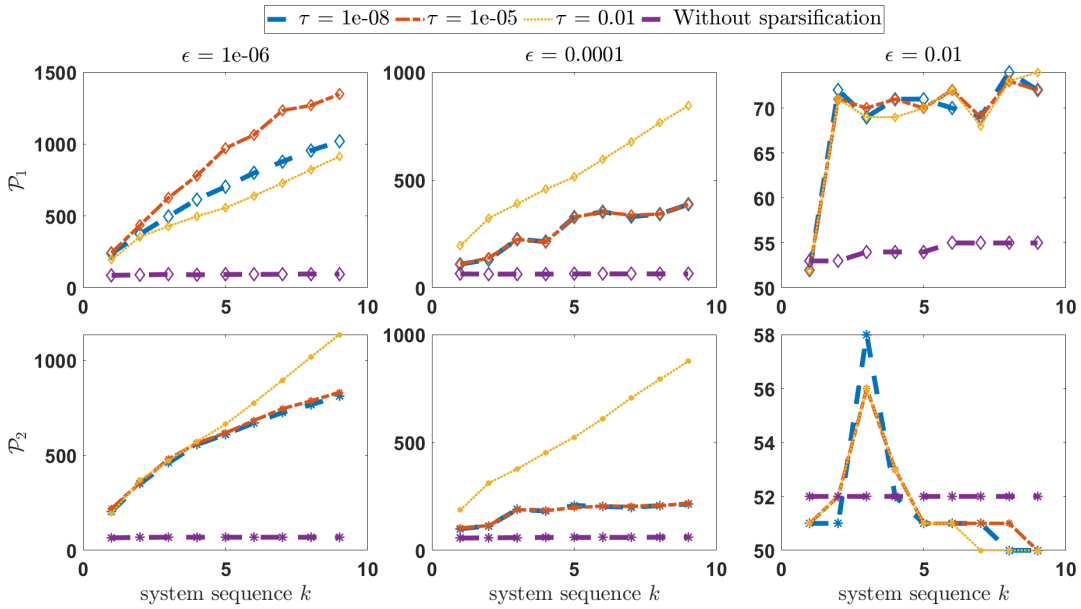

Figure 10: The hierarchical ranks of the resulting matrices R_k for the HODLR format when sparsifying the matrix $P_1E_{1,k}$ or $P_1E_{k-1,k}$. We fixed $\beta = 256$ and $N = 37249$.

First, we see that there is an important distinction between sparsifying E and P_1E – the former in fact does decrease the hierarchical rank (sometimes even significantly) while the latter does the opposite, it increases the hierarchical rank, and quite noticeably. This is a key difference that can be heuristically understood on the following example: consider the matrices

$$M = \begin{bmatrix} 10 & 5 & 2.5 \\ 1 & 0.5 & 0.25 \\ 0.1 & 0.05 & 0.025 \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0.5 & 0.25 \end{bmatrix} \quad \text{and} \quad M_{\text{sp}} = \begin{bmatrix} 10 & 5 & 2.5 \\ 1 & 0.5 & 0 \\ 0.1 & 0 & 0 \end{bmatrix},$$

where M_{sp} is obtained by sparsifying M with the drop tolerance $\tau = 1/3$. When we sparsify P_1E, the analogue of this occurs blockwise and *destroys* the low-rank structure that is naturally present due to the ellipticity of the underlying differential operator: the approximability in general stems from large blocks of the matrix being almost constant with only few outliers and hence dropping entries can disrupt these almost constant blocks and even a relatively small drop tolerance can make the hierarchical rank explode, see [4] but also [22, Chapters 1–4] and the references therein. Notice that this problem does not appear when we sparsify *before* we apply P_1, i.e., before we let the discretized differential operator act on the update matrix – quite on the contrary. In the case of sparsifying E, our initial intuition of decreasing the amount of information to approximate is a correct one – the non-zero columns of P_1E are linear combinations of columns of P_1 with coefficients stored in the columns of E and hence the sparsification of E moderates the number of the terms in these linear combinations. Assuming P_1 mimics the well-approximability in a hierarchical format of A_1^{-1}, having fewer of these columns combined leads to lower hierarchical rank.

However, we see that the information decrease has a *drastic* effect on the quality of the preconditioner in terms of GMRES iterations – we capped the maximal number of GMRES iterations at 350 and only the second system converged, taking about 330 iterations to reach the 10^{-10} relative

residual. Compared to the non-sparsified preconditioners and keeping (22) in mind, we see that the structural sparsification – either of E or P_1E – is unlikely to improve the overall efficiency of our preconditioners, based on the shown data as well as our general experience. We note that these observations remained true when varying other relevant parameters, such as the format (for brevity we do not show the analogous plots for the HSS format), accuracy ϵ, the drop tolerance τ and the size of the system N (with the caveat that for small systems, these effects are significantly damped).

Data sparsification As the structural sparsification did not yield the desired effect, we turn our attention to *data-sparsification* – truncation of the hierarchical rank of the considered matrices. This direction is the proper analogue of the fixed (structural) sparsity pattern in the SAM approach mentioned above – we replace the structural sparsity in both the map and in its further restriction by data-sparsity. The truncation for the HODLR format then consists of blockwise truncation, following the blocking of the format. For the HSS format, things are a bit more involved as the format itself is more delicate – following the notation of the `hm-toolbox` in [21] we calculated the SVD of the *core matrices* $S_{ij}^{(\ell)}$ stored in the `B12,B21` attributes of the HSS class and truncated it (as well as the other corresponding matrices in the format in order to reap the computational efficiency benefits of the truncation).

Similarly to the analysis in Section 3, we have only one option for hierarchical rank truncation to $r_{\max}$ for the map $\mathcal{P}_1$ – the matrix $R_k^{(1)}$ – but we have two options for the map $\mathcal{P}_2$ – the matrix $\mathcal{H}(P_1E_{k-1,k})$ or the matrix $R_k^{(2)}$. Clearly, only the second option guarantees the rank to be lower than $r_{\max}$ as the formatted sum $R_{k-1}^{(2)} \oplus \text{ranktrunc}\left(\mathcal{H}(P_1E_{k-1,k})\right)$ can increase the hierarchical rank – in fact for this option we can bound the hierarchical rank is by $2^k r_{\max}$ (and since the complexities scale with the square[18] of the rank, it can be up to 4^k times as expensive). In practice we have never observed anything close to this but we have observed that the hierarchical rank does somewhat increase as k grows and sometimes can even outgrow the hierarchical rank without any truncation – for the second option, that is. We illustrate this as well as the performance of of the resulting preconditioners in Figure 11.

We see that the number of GMRES iterations compared to the non-truncated preconditioner increases and this increase can be quite mild (plot in the (1,1) position) or very pronounced (plot in the (2,2) position) depending on the way the truncation is carried out and on the map. On one hand, we have observed that the map $\mathcal{P}_1$ is quite stable with respect to the hierarchical rank truncation, and decreasing the hierarchical rank 2-5 times still gives convergence in a comparable, say double, number of iterations. Recalling (22), this corresponds to a significant improvement in efficiency of the truncated preconditioner. On the other hand, we observed that the map $\mathcal{P}_2$ is more sensitive to the rank truncation and especially so if the truncation is applied to $\mathcal{H}(P_1E)$ rather than to $R_k^{(2)}$. Also, we observe that the higher the accuracy (i.e., the smaller the ϵ), the more diverse can these differences be – partly also because higher accuracy translates naturally to higher hierarchical ranks, which in turn leaves more space for highlighting the effects.

An explanation or at least a deeper insight into the interaction of the data-sparsification and the preconditioner maps are left as an open direction for future research.

[18] As mentioned above, this can be relaxed to linear instead of square by adding the $\mathcal{H}$-FAINV step.

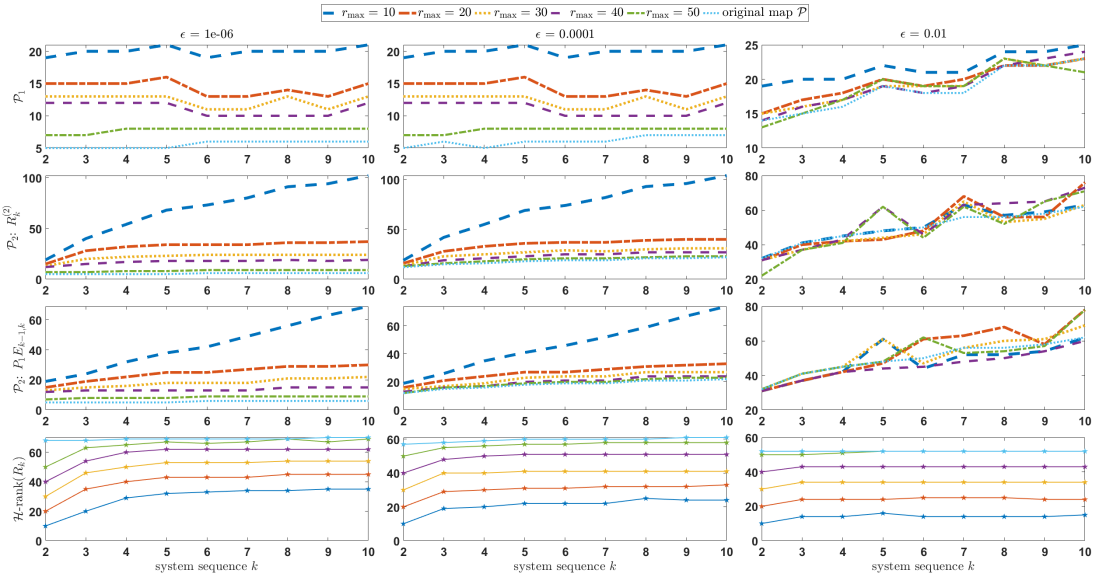

Figure 11: The number of GMRES iterations with fixed hierarchical ranks for $\mathcal{P}^{(1)}$ (first row) and $\mathcal{P}^{(2)}$ (second row) and for $\mathcal{P}^{(2)}$ when truncating the hierarchical rank of only $P_1 E_{k-1,k}$ (third row). The resulting hierarchical rank of $R_k^{(2)}$ when truncating only $P_1 E_{k-1,k}$ as well as that of the hierarchical rank of $R_k^{(2)}$ without truncation (original map) is shown below (fourth row). We show the results only for the HODLR format and fixed $\beta = 256$ and $N = 37249$.

5 Conclusion and future work

We have proposed and studied – both analytically and experimentally – a new type of preconditioner maps for sequences of systems of linear algebraic equations, using data-sparsity or, to be more precise, using the hierarchical matrix formats HODLR and HSS. The theoretical results give a complementary approach to analysis of preconditioner maps compared to [7] and give a direct insight into which quantities can ensure a solid GMRES convergence. We illustrated the need for preconditioner maps on a model example and pursued further avenues to make our maps more efficient so that application of the preconditioner is manageable. Further analysis as well as analysis for more particular settings is of clear interest and remains open for future research.

References

[1] K. Ahuja, B. K. Clark, E. de Sturler, D. M. Ceperley, and J. Kim. Improved scaling for quantum Monte Carlo on insulators. *SIAM Journal on Scientific Computing*, 33(4):1837–1859, 2011.

[2] J. Ballani and D. Kressner. *Matrices with hierarchical low-rank structures*, chapter Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, pages 161–209. Springer, 2016.

[3] M. Bebendorf and T. Fischer. On the purely algebraic data-sparse approximation of the inverse

and the triangular factors of sparse matrices. *Numerical Linear Algebra with Applications*, 18 (1):105–122, 2011.

[4] M. Bebendorf and W. Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numerische Mathematik*, 95(1):1–28, 2003.

[5] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[6] S. Börm, N. Albrecht, Ch. Börst, S. Christophersen, and J. Lorenzen. H2Lib, 2023. URL `http://www.h2lib.org/`.

[7] A. Carr, E. de Sturler, and S. Gugercin. Preconditioning parametrized linear systems. *SIAM Journal on Scientific Computing*, 43(3):A2242–A2267, 2021.

[8] A. K. Carr, E. de Sturler, and M. Embree. Analysis of GMRES for low-rank and small-norm perturbations of the identity matrix. volume 22, page e202200267. Wiley Online Library, 2023.

[9] M. Crouzeix and A. Greenbaum. Spectral sets: numerical range and beyond. *SIAM Journal on Matrix Analysis and Applications*, 40(3):1087–1101, 2019.

[10] M. Embree. How descriptive are GMRES convergence bounds? 2022.

[11] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation. *Communications on Pure and Applied Mathematics*, 64(5):697–735, 2011.

[12] P. Ghysels, X. S. Li, Y. Liu, and L. Claus. STRUMPACK, 2023. URL `https://portal.nersc.gov/project/sparse/strumpack/`.

[13] L. Grasedyck, R. Kriemann, and S. Le Borne. Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems. *Computing and Visualization in Science*, 11(4-6):273–291, 2008.

[14] L. Grasedyck, R. Kriemann, and S. Le Borne. Domain decomposition based-LU preconditioning. *Numerische Mathematik*, 112(4):565–600, 2009.

[15] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[16] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*, volume 49. Springer Berlin, Heidelberg, 2015.

[17] N. J. Higham. The Matrix Computation Toolbox for MATLAB (version 1.0). Technical report, 2002.

[18] R. Kriemann and S. Le Borne. $\mathcal{H}$-FAINV: Hierarchically factored approximate inverse preconditioners. *Computing and Visualization in Science*, 17(3):135–150, 2015.

[19] J. Liesen and Z. Strakoš. *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, Oxford, 2013.

[20] P. G. Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1251–1274, 2011.

[21] S. Massei, L. Robol, and D. Kressner. hm-toolbox: MATLAB software for HODLR and HSS matrices. *SIAM Journal on Scientific Computing*, 42(2):43–68, 2020.

[22] M. Outrata. *Schwarz methods, Schur complements, preconditioning and numerical linear algebra*. PhD thesis, University of Geneva, Math Department, 2022.

[23] J. A. Sifuentes, M. Embree, and R. B. Morgan. GMRES convergence for perturbed coefficient matrices, with application to approximate deflation preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1066–1088, 2013.

[24] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer. A survey of subspace recycling iterative methods. *GAMM-Mitteilungen*, 43(4):e202000016, 2020.

[25] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behaviour of Non-Normal Matrices and Operators*. Princeton University Press, Princeton, New Jersey, 2005.

[26] T. G. Wright. Eigtool. http://www.comlab.ox.ac.uk/pseudospectra/eigtool/, 2002.

[27] T. G. Wright and L. N. Trefethen. Large-scale computation of pseudospectra using ARPACK and `eigs`. *SIAM Journal on Scientific Computing*, 23(2):591–605, 2001.

[28] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

[29] X. Ye, J. Xia, and L. Ying. Analytical low-rank compression via proxy point selection. *SIAM Journal on Matrix Analysis and Applications*, 41(3):1059–1085, 2020.